Diplomarbeit

zur Erlangung des akademischen Grades
Diplom-Wirtschaftsinformatiker (FH)

an der Technischen Fachhochschule Wildau

Design und Implementation eines Newsletter-Versandtools zur Integration in das OpenSource-CMS *ww*Edit

Reg.-Nr. I 143/07

Technische Fachhochschule Wildau
Fachbereich Betriebswirtschaftslehre/Wirtschaftsinformatik

Studiengang Wirtschaftsinformatik

Alexander Schulze
Matrikelnummer: 01 277 0523

Tag der Ausgabe	17. September 2007
Tag der Abgabe:	13. Dezember 2007
1. Betreuer:	Prof. Dr. Ulrike Tippe
2. Betreuer:	Prof. Dr. Christian Müller
Themenstellender Betrieb:	wegewerk GmbH, Linienstr. 126, 10115 Berlin

Technische Fachhochschule Wildau
Fachbereich Betriebswirtschaft/Wirtschaftsinformatik

Diplomarbeit

		Reg.-Nr.:	*I 143/07*
für **Herrn/Frau**	*Alexander Schulze*	Matrikel-Nr.:	*01 277 0523*
Studiengang:	*Wirtschaftsinformatik*	SG:	*I 2/01*
Vertiefungsrichtung:			
Betreuende Lehrkraft:	*Prof. Dr. Tippe*		
Ausgabe der Arbeit:	*17. Sept. 2007*	Abgabe der Arbeit:	*16. Dez. 2007*

Thema der Arbeit:

*Design und Implementation eines Newsletter-Versandtools zur Integration in das OpenSource-CMS ww**Edit***

Inhalt der Arbeit:

- *Beschreibung ww**Edit** CMS (verwendete Technologien, Schnittstellen)*
- *Beschreibung des Ursprungszustandes und Erörterung der Notwendigkeit der Software-Neuentwicklung*
- *Erstellung eines Anforderungskataloges für Funktionalitäten*
- *Entwurf eines Datenbankschemas*
- *Implementation der Funktionalitäten, Eingabemasken und Ausgabeformate*
- *Besonderer Fokus: HTML-Newsletter, Konvertierung HTML zu Text, (Bild-)Dateianhänge, persönliche Anrede, mehrere Verteilerlisten, Import- und Exportfunktionen, Double-Opt-In*

Konsultationen: *mind. 2 Pflichtkonsultationen*

Wildau, den 02. Okt. 2007

.....	.....	.....
Student	Lehrkraft	Vorsitzender Prüfungsausschuss

Selbstständigkeitserklärung

Ich erkläre hiermit, dass ich die vorliegende Diplomarbeit selbstständig angefertigt und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe. Alle wörtlich oder sinngemäß übernommenen Textstellen sind gekennzeichnet.

Berlin, 13. Dezember 2007

Alexander Schulze

Inhaltsverzeichnis

1 Glossar der Abkürzungen und Akronyme

ACL:	Access Control List
AJAX:	Asynchronous JavaScript and XML
API:	Application Programming Interface
ASCII:	American Standard Code for Information Interchange
CMS:	Content Management System
CDATA:	Character Data
CO:	Content Object
CRLF:	Carriage Return, Line Feed
CSS:	Cascading Stylesheets
CSV:	Comma Separated Values
DDV:	Deutscher Direktmarketingverband e.V.
DNS:	Domain Name System
ER:	Entity Relationship
GIF:	Graphics Interchange Format
GNU:	GNU is not Unix
GPL:	GNU General Public License
IETF	Internet Engineering Task Force
JPEG:	Joint Photographic Experts Group
LAMP:	Linux, Apache, MySQL, PHP
LDAP:	Lightweight Directory Access Protocol
MD5:	Message Digest 5
MIME:	Multipurpose Internet Mail Extensions
MSDN:	Microsoft Developer Network
MTA:	Mail Transport Agent
MX:	Mail Exchange
PEAR:	PHP Extension and Application Repository
PHP:	PHP Hypertext Preprocessor
RFC:	Request For Comment
RPC:	Remote Procedure Call
RSS:	Really Simple Syndication
SMTP:	Simple Mail Transfer Protocol
SQL:	Structured Query Language
SSI:	Server Side Includes
UWG:	Gesetz gegen den unlauteren Wettbewerb
XHTML:	Extensible Hypertext Markup Language
XML:	Extensible Markup Language

2 Einführung

> „*Als **Newsletter** (engl. für Mitteilungsblatt, Verteilernachricht oder Infobrief) wird ein (meist elektronisches) Rundschreiben bezeichnet.*“[1]

Die meisten Unternehmen, Organisationen und Verbände verfügen heutzutage über einen Internetauftritt, durch den sie sich präsentieren und Informationen auf einfache Weise einem großen Publikum zugänglich machen können. Diese Methode der Informationsverbreitung setzt allerdings voraus, dass die Zielgruppe selbst die bereitgestellten Informationen bezieht, indem sie die Internetseite aufruft.

Doch wie erfährt ein Kunde oder Interessent überhaupt davon, dass es Neuigkeiten auf der Seite gibt? Und wie kann eine Organisation oder ein Unternehmen erreichen, dass die Webseite nicht nur einmalig aufgerufen wird und wieder in Vergessenheit gerät, sondern Besucher regelmäßig auf die Seite geführt werden, um sich über Neuigkeiten zu informieren? Hier gibt es mehrere mögliche Lösungen:

Eine Variante besteht in der besonders in den letzten Jahren populär gewordenen Methode, redaktionelle Inhalte oder Nachrichten über einen sogenannten *Newsfeed* via RSS[2], Atom[3] oder einem ähnlichen Format zu verbreiten. Ein Interessent kann sich so aus beliebigen abonnierten Quellen seine Informationen zusammenstellen. Die Feeds sind dank XML-Format maschinenlesbar, wodurch sich die Möglichkeit bietet, sie auch anderweitig zu verarbeiten, z.B. um sie auf einer anderen Webseite zu veröffentlichen (Syndikation). Der Nachteil besteht allerdings darin, dass hier viel Initiative vom Interessenten gefordert wird. So ist für die komfortable Verwendung durch einen Endnutzer oft die Installation eines entsprechenden Programms (Newsreader) erforderlich. Zudem ist es auf diese Weise für den Betreiber unmöglich Aussagen darüber zu treffen, wie groß der Kreis derer ist, die die bereitgestellten Informationen konsumieren. Für Webseiten mit einem hohen Aktualitätsgehalt, beispielsweise Portale von Zeitungen und

1 [WIKIPEDIA]
2 [RSS]
3 [IETF]

Magazinen, oder auch Weblogs und Internetforen, ist diese Variante dennoch zu empfehlen, da sie der Erwartungshaltung eines Benutzers entgegen kommt, schnellstmöglich über neue Entwicklungen oder Beiträge informiert zu werden.

Die zweite Möglichkeit besteht darin, redaktionell zusammengestellte Nachrichten und Informationen in Form eines Newsletters per E-Mail zu versenden. Auch gelegentlichen Internetnutzern mit geringen technischen Kenntnissen ist dieses Medium geläufig, da ein großer Teil privater und geschäftlicher Kommunikation mittlerweile über E-Mail abgewickelt wird. Es wird vom Endnutzer keine Umgewöhnung erwartet. Zudem erlaubt der Versand an einen definierten Benutzerkreis Rückschlüsse darüber, wer sich für die Informationen interessiert und welche Reichweite damit erzielt wird. Diese Art der Informationsverbreitung eignet sich besonders für Webseiten, bei denen Änderungen in größeren Abständen erfolgen oder für Unternehmen, die in regelmäßigen Abständen Informationen an Kunden oder Pressekontakte verteilen möchten.

Die Zielsetzung der vorliegenden Diplomarbeit liegt nun darin, für das Content Management System (CMS) *ww***Edit**[4] eine Newsletter-Versandsoftware (folgend unter dem Namen *ww***News**) zu konzipieren und zu implementieren. Nach der Betrachtung einiger Grundlagen werden dazu zunächst das System und die verwendeten Technologien beschrieben um einen Überblick über die technischen Gegebenheiten gewinnen zu können. Anschließend werden der Ursprungszustand analysiert und die Gründe erläutert, aus denen die Neuentwicklung angestrebt wurde.

Im weiteren Verlauf möchte der Autor definieren, welche Anforderungen die neu zu entwickelnde Software erfüllen soll und an einigen Beispielen den Weg beschreiben, wie diese Ansprüche umgesetzt wurden.

2.1 Rechtliche Grundlagen

Bevor die Anforderungen an eine Newsletter-Versandsoftware festgelegt werden, sollen die rechtlichen Grundlagen erörtert werden, welche Betreiber von Mailverteilern betreffen. Technisch unterscheiden sich Newsletter zwar kaum von anderen Massenmailings, wie sie beispielsweise benutzt werden, um Kettenbriefe, Werbung und andere zumeist unerwünschte E-Mails („Spam“)

4 [WEGEWERK]

an eine große Zielgruppe zu verteilen. Allerdings kann sich ein Betreiber technisch und juristisch absichern, um seinen seriösen Newsletter von der Flut unerwünschter elektronischer Post abzuheben.

Rechtlich gelten in Deutschland für Newsletterverteiler die Vorgaben des Gesetzes gegen den unlauteren Wettbewerb (UWG), im Speziellen § 7 „Unzumutbare Belästigungen“. Darin heißt es[5]:

> *(1) Unlauter im Sinne von § 3 handelt, wer einen Marktteilnehmer in unzumutbarer Weise belästigt.*
> *(2) Eine unzumutbare Belästigung ist insbesondere anzunehmen*
> *1. bei einer Werbung, obwohl erkennbar ist, dass der Empfänger diese Werbung nicht wünscht;*
> *2. [...]*
> *3. bei einer Werbung unter Verwendung von [...] elektronischer Post, ohne dass eine Einwilligung der Adressaten vorliegt;*
> *4. bei einer Werbung mit Nachrichten, bei der die Identität des Absenders, in dessen Auftrag die Nachricht übermittelt wird, verschleiert oder verheimlicht wird oder bei der keine gültige Adresse vorhanden ist, an die der Empfänger eine Aufforderung zur Einstellung solcher Nachrichten richten kann, ohne dass hierfür andere als die Übermittlungskosten nach den Basistarifen entstehen.*
> *(3) Abweichend von Absatz 2 Nr. 3 ist eine unzumutbare Belästigung bei einer Werbung unter Verwendung elektronischer Post nicht anzunehmen, wenn*
> *1. ein Unternehmer im Zusammenhang mit dem Verkauf einer Ware oder Dienstleistung von dem Kunden dessen elektronische Postadresse erhalten hat,*
> *2. der Unternehmer die Adresse zur Direktwerbung für eigene ähnliche Waren oder Dienstleistungen verwendet,*
> *3. der Kunde der Verwendung nicht widersprochen hat und*
> *4. der Kunde bei Erhebung der Adresse und bei jeder Verwendung klar und deutlich darauf hingewiesen wird, dass er der Verwendung jederzeit widersprechen kann, ohne dass hierfür andere als die Übermittlungskosten nach den Basistarifen entstehen.*

Daraus lassen sich drei wesentliche Vorgaben ableiten, die ein Newsletter erfüllen muss:

1. Der Empfänger muss dem Versand ausdrücklich zugestimmt haben.
2. Der Versender muss seine Identität deutlich zu erkennen geben.
3. Dem Newsletter ist der Hinweis zuzufügen, wo der Empfänger sein Abonnement jederzeit kündigen kann.

5 [UWG]

Weitere Ergänzungen empfiehlt der Deutsche Direktmarketingverband e.V. (DDV)[6], dessen Mitglieder sich in einem Ehrenkodex[7] zur Einhaltung weiterer Richtlinien insbesondere in Bezug auf Daten- und Verbraucherschutz verpflichten.

So schlägt der DDV zur Sicherstellung der Einwilligung eines Absenders das Verfahren des *Double-Opt-Ins* vor. Dies besagt, dass der Benutzer nach der Eintragung einer E-Mailadresse in ein Registrierungsformular eine Bestätigungsmail erhält. In dieser Mail wird ihm noch einmal erklärt, wofür er sich anmeldet, zudem enthält sie einen Bestätigungslink. Erst durch Besuchen dieses Links wird das Abonnement wirksam. Auf diese Weise kann zum Beispiel sichergestellt werden, dass kein Fremder einen Newsletter für eine Mailadresse abonniert, obwohl der eigentliche Inhaber der Adresse die Zustellung der Newsletter gar nicht wünscht.

Aus Gründen des Verbraucherschutzes schlägt der DDV auch das Führen einer Sperrliste vor, in welche Adressen von Empfängern eingetragen werden, die der Zustellung weiterer E-Mails ausdrücklich widersprochen haben. Vor jedem Versand von werblichen Mails ist der Adressbestand gegen diese Liste abzugleichen.

Weiterhin wird empfohlen, die Verteilerliste von fehlerhaften E-Mail-Adressen zu bereinigen. Wenn der Versand an eine Adresse zu Fehlermeldungen führt, die darauf schließen lassen, dass die Adresse auch in Zukunft nicht mehr korrekt beliefert werden kann, so sollte diese Adresse von der Verteilerliste entfernt werden, um die Qualität des Adressbestandes zu wahren und unnötigen Mailverkehr zu vermeiden. Zudem stellt diese Maßnahme einen wirksamen Schutz davor dar, als Spamverteiler bewertet zu werden, da große Mailprovider wie GMX unerwünschte E-Mails unter anderem auch nach der Anzahl gescheiterter Zustellungsversuche von einem versendenden Server an Adressen in ihren Domains klassifizieren[8]. Daher ist es erstrebenswert, fehlerhafte Adressen schnellstmöglich von den eigenen Verteilern zu entfernen, damit der Newsletter nicht unter Spamverdacht gerät.

6 [DDV]
7 [DDV]
8 [GMX]

2.2 *ww*Edit CMS

Wie der Titel der vorliegenden Arbeit bereits besagt, soll das zu entwickelnde Newsletter-Tool in die Software *ww***Edit** integriert werden. Daher soll zu Beginn ein kurzer Abriss darüber erfolgen, worum es sich bei *ww***Edit** handelt, welche Funktionen das System bietet, wo es zum Einsatz kommt und welche Technologien verwendet werden.

*ww***Edit** ist ein Content Management System, also eine Software, die es Benutzern ohne besondere technische Kenntnisse über die zugrunde liegenden Technologien (z.B. HTML) mit einfachen Mitteln ermöglichen soll, Inhalte strukturiert im Internet zu veröffentlichen. Entwickelt wird das System von der Berliner Agentur *wegewerk*[9], bei der der Autor während der Erarbeitung dieser Diplomarbeit angestellt ist.

Kunden, die das System für ihre Internetauftritte verwenden, kommen aus mittelständischen und Großunternehmen, Stiftungen, Verbänden, Gewerkschaften und Politik. *wegewerk* begleitet den Kunden entsprechend seiner Bedürfnisse und Wünsche von der Konzeption der Inhalte über Gestaltung, Umsetzung, Installation und Konfiguration bis hin zur redaktionellen Pflege des Systems.

Der Schwerpunkt bei der Entwicklung des Systems liegt aufgrund der Verwendung in öffentlichen Institutionen einerseits auf Barrierefreiheit der damit erstellten Webseiten und andererseits auf möglichst einfacher Bedienbarkeit und selbsterklärenden Eingabeschnittstellen. Das System wird stetig weiterentwickelt und beherrscht mittlerweile nicht nur die bloße Bereitstellung von redaktionellen Inhalten, sondern auch interaktive Elemente wie Foren und Weblogs, Abstimmungen und Umfragen, Unterschriftenlisten und einige weitere Funktionen, um Kampagnen oder auch einfache Social Networks damit umsetzen zu können.

*ww***Edit** baut zum großen Teil auf die unter der Bezeichnung LAMP zusammengefassten Technologien auf: Im Produktionsbetrieb läuft es in der Regel auf Linux-Servern mit der Distribution Debian GNU/Linux[10], als Webserver-Software wird Apache[11] benutzt. Die eingesetzte

9 [WEGEWERK]
10 [DEBIAN]
11 [APACHE]

Programmiersprache ist PHP[12], als Datenbanksystem kommt MySQL[13] zum Einsatz. Zur Authentifizierung wird der Verzeichnisdienst OpenLDAP[14] verwendet, in welchem Benutzerkonten von Redakteuren oder auch angemeldeten Webseitennutzern abgelegt werden. Abhängig vom gewünschten Funktionsumfang ist die Installation weiterer Software nötig, beispielsweise um Bilder zuzuschneiden, Videos zu konvertieren, PDF-Dateien zu generieren oder Archive zu entpacken.

Bei der Programmierung des Systems wird auf einige weitere Technologien zurückgegriffen. Zur Trennung zwischen dem eigentlichen Programmcode und dem Darstellungs-Markup (XHTML) wird die Smarty-Template-Engine[15] benutzt. So ist es möglich, dass Designvorlagen auch ohne PHP-Kenntnisse in XHTML und CSS umgesetzt werden können, solange die verfügbaren Platzhalter bekannt sind und man mit der einfach gehaltenen Smarty-Syntax vertraut ist. Außerdem kommen einige Klassen des PHP Extension and Application Repositorys[16] (PEAR) zum Einsatz. Insbesondere hervorzuheben sind hier die Pakete `PEAR::DB_QueryTool` als Datenbank-Abstraktion[17] sowie `PEAR::Mail`, welches zum Versand von E-Mails für die vorliegende Arbeit verwendet wird.

*ww***Edit** wird als Open-Source-Software unter der GNU General Public License (GPL)[18] vertrieben. Dies ermöglicht einerseits die Verwendung und Einbindung anderer unter der GPL veröffentlichter Software in das CMS (wie beispielsweise des Datenbank-Managementsystems MySQL oder der JavaScript-Bibliothek jQuery[19]). Andererseits garantiert es Kunden Investitionssicherheit, da die Software in ihren Quellen verfügbar ist und unabhängig vom Anbieter weiter bearbeitet werden darf, solange Änderungen gekennzeichnet sind und ebenfalls unter der GPL oder einer kompatiblen Lizenz vertrieben werden.

12 [PHP]
13 [MYSQL]
14 [OPENLDAP]
15 [PHP]
16 [PHP]
17 Durch konsequente Verwendung der Abstraktionsklassen kann erreicht werden, dass das System nicht ausschließlich auf der verwendeten MySQL-Datenbank lauffähig ist, sondern theoretisch auch mit anderen relationalen Datenbanksystemen wie z.B. Oracle, PostgreSQL oder SQLite betrieben werden kann, solange sie die gleichen SQL-Befehle unterstützen.
18 [GNU]
19 [JQUERY]

2.3 Der Ursprungszustand von *ww*News

Vor der Integration in das CMS bestand bereits eine Newslettersoftware unter dem Namen *ww***News**. Zur Verdeutlichung der Notwendigkeit einer Neuprogrammierung soll im Folgenden die Vorgängerversion beschrieben werden.

Diese bestand aus einem Webservice, also einer unabhängigen alleinstehenden Software, die an beliebigen Stellen in vorhandene dynamisch generierte Seiten z.B. durch PHP oder Server Side Includes (SSI) eingefügt werden konnte und über einen eigenen Redaktionsbereich verfügte. Jedem Kunden wurden eindeutige Zugangsdaten (Benutzername, Passwort) sowie ein Identifikationscode (ID) zugewiesen, mit dem sich die Kundenwebsite beim Tool identifiziert.

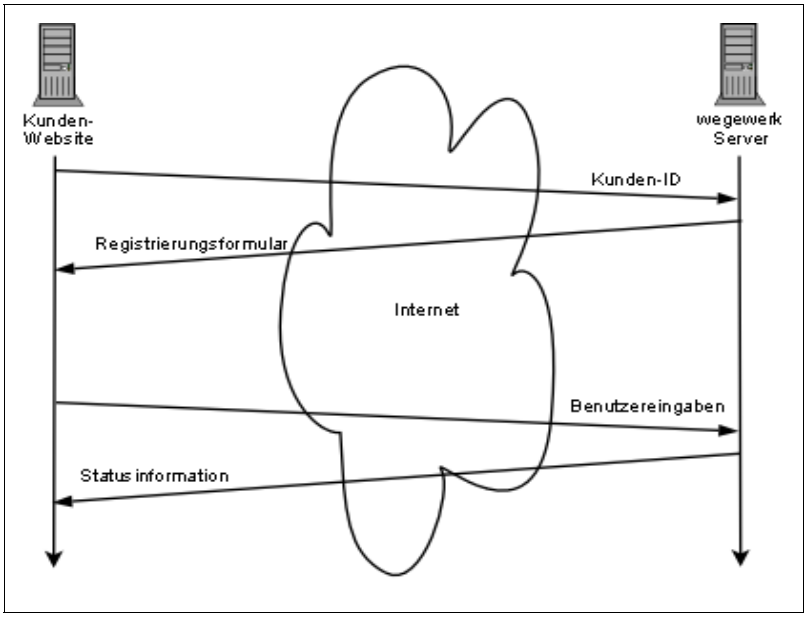


Abbildung 1: Prozessablauf in der Vorgängerversion

Mit Hilfe dieser ID erfolgte eine HTTP-Abfrage auf ein PHP-Skript, welches daraufhin ein Newsletter-Registrierungsformular als HTML-Codeblock erzeugte und zurückgab. Dieser Block konnte an entsprechender Stelle in PHP-Dateien eingebunden werden. Die Auswertung der Benutzereingaben in diesem Formular erfolgte ebenfalls durch das externe Skript, welches entsprechende Status- oder Fehlercodes zurücklieferte (vgl. Abbildung 1).

Über den via Benutzername und Passwort zugänglichen Redaktionsbereich des Tools konnte eine Verteilerliste pro Kundenkonto verwaltet werden. Es war vorgesehen eine beliebige Anzahl von Text-Newslettern anlegen zu können, die dann an diese Liste gesendet werden konnten. Ein Versand von HTML-Newslettern war nicht vorgesehen, die Formatierungsmöglichkeiten waren dadurch entsprechend eingeschränkt. Ebenso konnten keine Dateianhänge beigefügt werden.

Der Versand selbst erfolgte, indem der mittels PHP zusammengestellte Newsletter-Rohdatensatz an ein Perl-Skript übergeben wurde, welches daraus die E-Mails zusammenstellte und sie an die entsprechenden Empfänger verschickte. Dies war problematisch, da es aufgrund unzureichender Programmierung und der Vielzahl an involvierten Skripten in unterschiedlichen Programmiersprachen oft zu Schwierigkeiten mit der Konvertierung zwischen Zeichensätzen kam. Die Umlaute und Sonderzeichen in manchen E-Mails waren also beispielsweise nicht lesbar oder der Versand versagte komplett den Dienst. Zudem war es einem Kunden nicht möglich, mehrere separate Verteilerlisten mit unabhängigem Adressbestand zu verwalten ohne sich ein neues Benutzerkonto einrichten zu lassen. Dies führte zunehmend zu Verärgerung bei den Kunden, die das System nutzten. Es musste also eine neue Lösung gefunden werden.

Da *wegewerk* zu diesem Zeitpunkt keine Kunden betreute, die ausschließlich das Newsletter-Tool einsetzten, sondern alle Newsletter-Kunden auch über eine *ww***Edit**-Installation verfügten, lag es nahe, eine integrierte Lösung zu schaffen. Vorteilhaft dabei erschien, dass das CMS bereits über einige Funktionalität verfügte, um die Neuentwicklung zu vereinfachen, wie zum Beispiel eine Fehlerbehandlung von Formularen (Pflichtfeld- und Validitätsprüfungen) oder entsprechende Programmierschnittstellen (API), um auf einfache Weise die Formulareingaben auszuwerten oder Eingabemasken und Statistiken im Redaktionsbereich erstellen zu können.

3 Anforderungen

Die primären Anforderungen an die neu zu entwickelnde Software bestehen natürlich in der Behebung der Probleme der bestehenden Lösung bei zumindest gleichem Funktionsumfang. So soll es also möglich sein, Textnewsletter mit dem System zu verschicken, ohne dass der Versand aufgrund von unbekannten Zeichen abbricht oder die E-Mails fehlerhaft übermittelt werden. Zusätzlich wurde aber noch eine Reihe von Wünschen formuliert, die den Funktionsumfang der bestehenden Lösung überschreiten.

3.1 Integration in *ww*Edit CMS

Das neue Tool soll so ins CMS integriert werden, dass Redakteure ohne erneute Anmeldung mit ihren Zugangsdaten alle für sie wichtigen Einstellungen des Newsletters bearbeiten können. Ebenso soll das Anlegen eines Registrierungsformulars sowie der Versand des Newsletters direkt im CMS ermöglicht werden. Es soll kein Bruch in der Optik des Redaktionsbereiches entstehen und bekannte Arbeitsschritte wie das Anlegen von Texten und Bildern müssen auch auf den Newsletter anwendbar sein.

3.2 Mehrere Verteilerlisten

Dazu gehört, dass es über das CMS möglich sein soll, eine beliebige Anzahl von Verteilerlisten mit voneinander unabhängigem Adressbestand zu verwalten. Beispielsweise könnte ein Kunde auf diese Weise einen Verteiler für interessante Informationen für die Besucher seiner Webseite betreiben, einen für Mitteilungen an Pressekontakte und einen weiteren zur Kommunikation mit Abteilungsleitern innerhalb des Kundenunternehmens. Ein Newsletter soll an einen oder auch gleichzeitig an mehrere dieser Verteiler geschickt werden können. Beim gleichzeitigen Versand an mehrere Listen soll allerdings berücksichtigt werden, dass pro gestartetem Versandvorgang nur eine E-Mail pro Adresse gesendet wird, auch wenn beispielsweise ein Empfänger auf mehreren der Verteilerlisten erscheint.

Das CMS *ww***Edit** beherrscht Multi-Site-Management. Dies bedeutet, dass es möglich ist mehrere voneinander getrennte Kundenseiten in der gleichen Installation zu betreiben, ohne dass Redakteure die Seiten anderer Kunden im Redaktionsbereich sehen oder bearbeiten können. In

Bezug auf die Verwaltung mehrerer Verteilerlisten bedeutet dies, dass es auch möglich sein soll, die Listen mit einer Rechtesteuerung durch *access control lists* (ACLs) zu versehen, sodass Verteiler nur von denjenigen benutzt oder eingesehen werden können, die auch die entsprechenden Rechte dafür besitzen. Es wäre natürlich fatal, wenn Kunde A einen Newsletter an die Empfängerliste von Kunde B versenden kann, sei es auch nur unabsichtlich.

3.3 HTML-Newsletter mit Bildern

Eine weitere wesentliche Verbesserung zur alten *ww***News**-Software ist der Versand von HTML-Newslettern. Diese bieten eine größere Vielfalt an Formatierungsmöglichkeiten wie zum Beispiel Fett- und Kursivschrift, sowie wesentlich einfachere Methoden der Textstrukturierung mit unterschiedlich gestalteten Überschriften, Aufzählungslisten, Stichpunkten etc. Es soll ebenfalls möglich sein Bilder einzufügen, um die E-Mails optisch ansprechender gestalten zu können. Auf diese Weise können Newsletter auch analog zum Design der Website entworfen werden, sodass ein Wiedererkennungswert zwischen dem Internetauftritt eines Kunden und dessen Informationsmails besteht.

3.4 Persönliche Anrede

Um den Kontakt zwischen dem Versender und Abonnenten des Newsletters persönlicher zu gestalten ist es angedacht, in den Newsletter eine persönliche Anrede einfließen zu lassen. Dazu ist es nötig, im Bestellformular weitere Daten wie Vor- und Nachnamen, Geschlecht sowie ggf. einen akademischen Titel abzufragen. Mit diesen zusätzlichen Daten lassen sich die Informationsmails dann mit einer formellen Anrede (z. B. „Sehr geehrter Herr Dr. Schmidt“) oder auch einer familiären Anrede (z. B. „Liebe Anna“) einleiten.

Durch zusätzlich abgefragte Daten ist es dem Newsletterbetreiber auch möglich, einen Überblick darüber zu bekommen, wer sich für seine Informationen interessiert. Für einen Presseverteiler wäre es beispielsweise interessant zu erfahren, welche Publikationen mit dem Newsletter erreicht werden, indem bei der Anmeldung ein weiteres Formularfeld für eine solche Angabe eingefügt wird.

3.5 Automatisches Rückläufer-Management

Zwar wird durch die Adressbestätigung beim Eintragen in den Verteiler (Double Opt-In) bereits geprüft, ob eine Empfängeradresse gültig ist. Dennoch kann es passieren, dass die Zustellung in Einzelfällen scheitert, beispielsweise weil eine Mailbox seit geraumer Zeit nicht abgerufen wurde und daher überläuft oder weil der Empfänger sein Mailkonto gekündigt hat. In diesen und anderen Fällen antworten Mailserver mit einer Fehlermeldung, welche Aufschluss über Art und Umfang des Zustellungsproblems gibt.

Nun ist es insbesondere bei umfangreichen Verteilerlisten ziemlich mühsam, diese zu pflegen und die Fehlermeldungen manuell zu bearbeiten. Technisch unbedarfte Benutzer können zudem oft die Fehlermeldungen nicht korrekt deuten, ob es sich dabei um einen temporären Fehler (wie zum Beispiel Greylisting[20]) handelt oder ob weitere Zustellungsversuche an diese Adresse zwecklos sind und die Adresse daher aus dem Verteiler entfernt werden sollte. Daher ist es wünschenswert, entsprechend der eingangs erwähnten Vorgaben des DDV die Rückläufer automatisiert zu verarbeiten.

3.6 Sperrliste

Durch das verwendete Double-Opt-In Verfahren ist bereits sichergestellt, dass der Empfänger eines Newsletters dem Empfang zumindest an einem bestimmten Zeitpunkt zugestimmt hat. Trotzdem kann es vorkommen, dass Empfänger ihre Einwilligung zurückziehen möchten und unter Umständen sogar mit rechtlichen Schritten drohen, sollten sie nochmals eine E-Mail von diesem Newsletteranbieter erhalten.

20 Greylisting ist eine wirksame Variante des Schutzes vor unerwünschten E-Mails (Spam). Dabei sendet der empfangende Mailserver für jede Mail von einem bis dato unbekannten Absender zuerst eine SMTP-Fehlermeldung, dass die Zustellung vorerst abgelehnt wurde und später erneut versucht werden soll. Korrekt arbeitende Mailserver erkennen diese Meldung und versuchen nach einigen Minuten erneut die E-Mail zu senden. In diesem Fall ist der Absender dem Empfangs-Server bereits bekannt und die Mail wird nun zugestellt. Spam hingegen wird oft von einfachen Dialup-Internetzugängen oder von Würmern und Trojanern versendet, die keinen zweiten Versandversuch unternehmen.

Um sich somit juristisch abzusichern wird wie auch vom DDV empfohlen die Einrichtung einer Sperrliste (Blacklist) angestrebt. Das System darf an keinen der Empfänger, die auf dieser Liste eingetragen sind, eine E-Mail verschicken, weder durch den Versand von Newslettern, noch durch Bestätigungsmails.

3.7 Import- und Export von Adressdaten

Viele Kunden, die sich für *ww***Edit** entscheiden, verfügen bereits von vorherigen Internetauftritten über Adressdaten, die sie gern weiter verwenden möchten. Weiterhin könnte die Anforderung bestehen, dass die gesammelten Informationen über Newsletterempfänger anderweitig aufbereitet werden sollen. Für einen Presseverteiler wäre es beispielsweise interessant auszuwerten, Mitarbeiter welcher Publikationen sich für den Newsletter eingetragen haben und welche Reichweite eine Informationsmail an diesen Verteiler somit hat. Ein anderer vorstellbarer Fall ist, dass die Newsletterabonnenten zusätzlich in einer externen Software verwaltet werden sollen.

Daher liegt es nahe, für die Empfängerlisten auch eine Import- sowie Exportschnittstelle zur Verfügung zu stellen. Dabei sollen übliche Formate verwendet werden. Die einfachste Möglichkeit wäre also der Import oder Export einer CSV-Datei, in der die Adressdaten einfach kommasepariert aufgelistet werden.

3.8 Statistiken

Um den Überblick über die versendeten Newsletter zu behalten, ist es empfehlenswert, einige Statistiken generieren zu können. Interessante Kenngrößen wären hier beispielsweise wann Newsletter versendet wurden, an wieviele Empfänger E-Mails zugestellt wurden und wieviel Zeit der Versandvorgang in Anspruch nahm.

4 Grundlagen

Damit sind die Anforderungen abgesteckt. Bevor nun mit der Implementation begonnen wird, sollen aber einige Grundlagen besprochen werden, die für die Umsetzung von Bedeutung sind.

Zum besseren Verständnis des Systems soll zunächst darauf eingegangen werden, wie *ww***Edit** seine Inhalte verwaltet, da basierend darauf die Entwicklung des Newsletter-Tools erfolgt. Anschließend soll der Aufbau einer E-Mail näher untersucht werden.

4.1 Die Struktur der Inhalte in *ww*Edit

*ww***Edit** strukturiert seine Inhalte hierarchisch. Auf oberster Ebene steht ein Seitenbaum, von dem es mehrere in einer CMS-Installation geben kann. In der Regel gibt es für jede verwendete Sprache (Deutsch, Englisch, u. s. w.) einen eigenen Baum sowie einige Systembäume für sprachunabhängige Seiten wie beispielsweise HTTP-Fehlerseiten oder ausschließlich vom System verwaltete Seiten, welche nicht öffentlich sind und auch keine redaktionellen Inhalte bereitstellen (Vorlagen, Papierkorb).

In den Seitenbäumen werden vom Benutzer oder vom System Seiten angelegt. Verglichen mit einem Dateisystem kann man sich eine Seite als ein Verzeichnis vorstellen, welches weitere Unterverzeichnisse haben darf. Es gibt verschiedene Seitentypen, welche abhängig von ihrem Inhalt mit besonderen Funktionen und Eigenschaften ausgestattet sind. Jede einzelne Seite ist im Redaktionsbereich mit verschiedenen Reitern ausgestattet, über welche die Eigenschaften (Attribute) bearbeitet oder Funktionen der Seite bedient werden können. Für die vorliegende Arbeit wird ein eigener Seitentyp „Newsletter“ angelegt, welcher typische Funktionen und Eigenschaften eines Newsletters zur Verfügung stellt.

Auf einer einzelnen Seite wiederum werden Inhalte in Form von Inhaltsobjekten (*content objects,* im Folgenden abgekürzt mit *COs*) angelegt. Dabei stehen einem Redakteur verschiedene Inhaltsbereiche (*areas*) auf einer Seite zur Verfügung, in welchen die COs angelegt werden können. Auf einer einfachen Seite kann es beispielsweise einen Seitenkopf und zwei Inhaltsspalten geben, die separat mit Inhaltsobjekten befüllt werden können. Dabei steht dem Redakteur eine Reihe von verschiedenen Objekten zur Verfügung, zum Beispiel einfache Textblöcke mit Überschrift und Bild, oder auch Formulare, Anreißer, Seitenlisten etc.

Zur Veranschaulichung der Zusammenhänge soll folgender Screenshot des Redaktionsbereiches dienen, in dem links der Seitenbaum, und rechts eine Seite mit einem Kopfbereich und drei Spalten dargestellt sind. In diesen vier Inhaltsbereichen befinden sich mehrere Inhaltsobjekte.

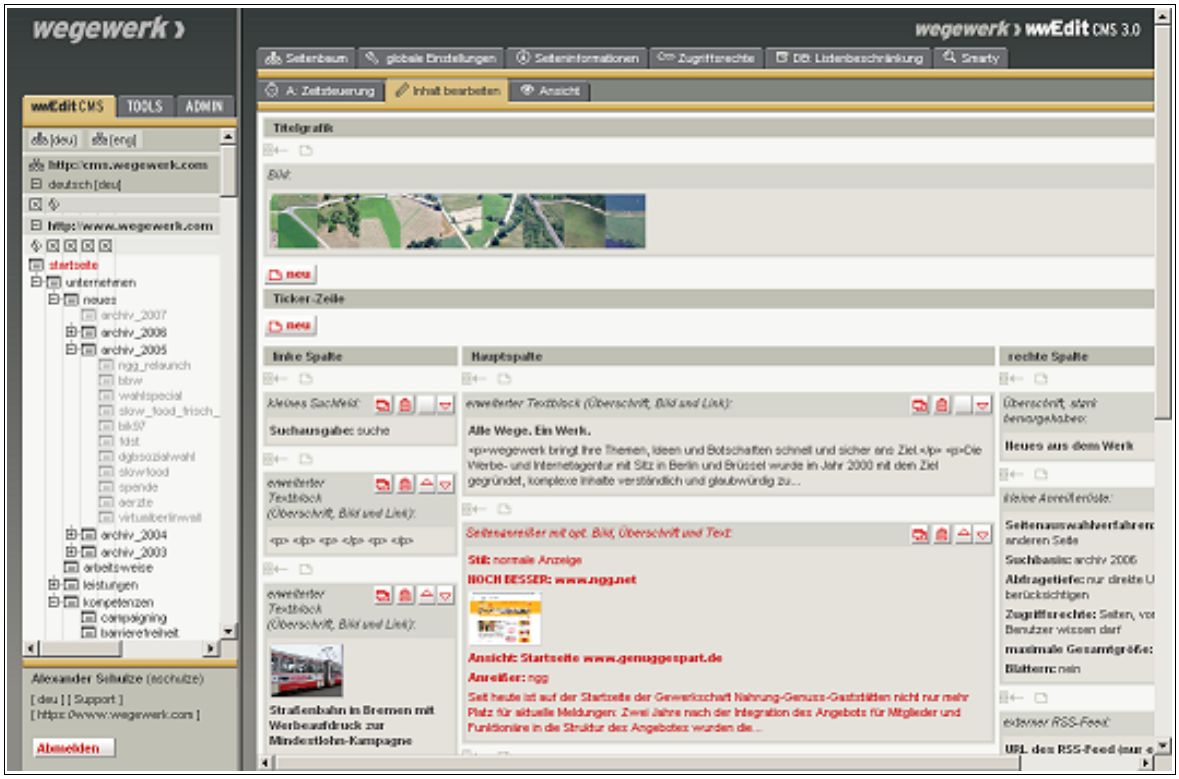


*Abbildung 2: Darstellung der Struktur in ww**Edit***

4.2 Formatvorgaben für E-Mails

Da der Versand elektronischer Post das zentrale Thema der vorliegenden Arbeit ist, soll an dieser Stelle soll auf das Format von E-Mails eingegangen werden. Die verwendete Klasse `PEAR::MAIL` übernimmt zwar einen Teil der Formatierung, allerdings lassen sich Fehler leichter untersuchen, wenn der zugrunde liegende Aufbau eine E-Mail und die damit verbundenen Einschränkungen bekannt sind. Im Folgenden soll allerdings nur auf die wichtigsten Punkte eingegangen werden, die für die *ww***News**-Entwicklung von Bedeutung sind.

Das Format einfacher Text-E-Mails wird in RFC 2822[21] definiert. Demnach besteht eine Mail aus einem Header und einem davon durch eine Leerzeile (als Zeilenumbruch gilt grundsätzlich die Kombination von *Carriage Return* und *Line Feed,* kurz *CRLF*) abgetrennten Body, der die eigentliche Nachricht enthält. Eine Zeile der Nachricht darf dabei nicht länger als 1000 Bytes (einschließlich CRLF) sein, empfohlen wird eine Zeilenlänge von nicht mehr als 80 Bytes.

Eine erste Beschränkung wird dadurch gebildet, dass in der Nachricht nur Zeichen des einfachen 7bit-ASCII-Zeichensatzes enthalten sein dürfen. Deutsche Umlaute beispielsweise sind nicht erlaubt und müssen entsprechend codiert werden, ebenso verhält es sich mit Dateianhängen. Auf diesen Vorgang wird weiter unten noch näher eingegangen.

Der Header der E-Mail enthält Meta-Informationen zur Nachricht und unterliegt besonderen Formatierungsbestimmungen. Eine Header-Information enthält einen Bezeichner gefolgt von einem Doppelpunkt, nach dem der Feldinhalt, die eigentliche Information, folgt. Um die Zeilenlängenbegrenzung einhalten zu können, darf die Information auf mehrere Zeilen aufgeteilt werden. Dies geschieht durch Einfügen eines Zeilenumbruchs direkt gefolgt von einem Whitespace-Zeichen, also einem Leerzeichen oder einem Tabstopp. Es gibt einige vorgegebene Header-Felder, die in einer E-Mail auftauchen sollten. Zwingend ist zwar nur eine Angabe über den Absender der Nachricht (*From*) und deren Absendezeitpunkt (*Date*), sinnvollerweise sollten aber weitere Informationen angegeben werden. Dazu gehören der Empfänger der Nachricht (*To*), eine mögliche Adresse für Rückläufer (*Return-Path*) und Antworten (*Reply-To*) sowie der Betreff (*Subject*). Die Reihenfolge der Informationen im Header ist nicht vorgeschrieben.

Neben den vorgegebenen Header-Feldern können weitere Feldbezeichner definiert werden. Üblicherweise sind diese zumeist nicht standardisierten Header an einem vorangestellten X zu erkennen, z.B. *X-Mailer* für eine Angabe über das zum Mailversand verwendete Mailprogramm.

21 [IETF]

Adressangaben für die Felder From, To, Reply-To und Return-Path folgen entweder einer einfachen Syntax, bei der nur eine E-Mail-Adresse genannt wird (`alschu@wi-bw.tfh-wildau.de`) oder einer Langform, bei der der Name des Mailboxinhabers vorangestellt und die Mailadresse in spitzen Klammern beigefügt wird (`Alexander Schulze <alschu@wi-bw.tfh-wildau.de>`). Die Angabe einer E-Mail-Adresse als Name in der Form `abc@def.gh <abc@def.gh>` ist nicht zulässig und führt bei einigen Mailbetreibern zur Zurückweisung der E-Mail.

Eine einfache syntaktisch korrekte E-Mail sieht nach den gegebenen Formatbeschreibungen wie folgt aus:

```
From: Max Mustermann <max.mustermann@host1.de>
To: Erika Musterfrau <e.musterfrau@host2.de>
Date: Sun, 11 Nov 2007
      17:41:23 +0100
X-Mailer: wwEdit 3.0
Subject: Dies ist ein Test

Hallo Welt.
```

Abbildung 3: Syntax einer einfachen E-Mail

Wie oben bereits erwähnt, dürfen in E-Mails nach RFC 2822 nur Zeichen des US-ASCII Zeichensatzes auftreten. Um diese Beschränkung aufzuheben und beispielsweise auch Dateianhänge an E-Mails zu ermöglichen, wird dieser Standard durch RFC 2045[22] erweitert. Dieser RFC führt zwei weitere Header-Felder ein: *Mime-Version* und *Content-Type*. Ersterer dient lediglich der Information, dass die vorliegende Nachricht die Erweiterungen dieses Standards benutzt und wird aktuell immer auf den Wert „1.0“ gesetzt. Das *Content-Type*-Feld enthält Informationen darüber, welcher Medientyp folgt und gegebenenfalls welcher Zeichensatz verwendet wird.

Der Medientyp wird über den mittlerweile nicht nur für E-Mail gebräuchlichen sogenannten MIME-Type angegeben. Hier stehen die Basistypen *text*, *image*, *audio*, *video* und *application* zur Verfügung, wobei nur die ersten beiden im Rahmen dieser Arbeit eine Rolle spielen werden. Eine

22 [IETF]

Besonderheit bildet der ebenfalls zum Einsatz kommende *multipart*-Basistyp, über den E-Mails mit Anhängen und mehreren verschiedenen Repräsentationen der gleichen Informationen ermöglicht werden.

Jeder Basistyp kann durch einen Subtyp genauer spezifiziert werden. So bedeutet die Angabe *text/plain* zum Beispiel, dass der folgende Text ohne besondere Formatierungen zu betrachten ist, *text/html* aber beispielsweise, dass der folgende Text das Format einer HTML-Nachricht enthält.

Die Zeichensatzangabe wird vom MIME-Type durch ein Semikolon abgetrennt und mit „`charset=`“ eingeleitet. Im deutschen Sprachraum ist die Angabe „`iso-8859-1`“[23] gebräuchlich, welche die üblichen Sonderzeichen im westeuropäischen Raum (so auch die deutschen Umlaute) einschließt. Auch die Angabe eines internationalen Zeichensatzes wie „`utf-8`“ wäre gültig, ist aber nur bedingt zu empfehlen. Microsoft Outlook, eines der am weitesten verbreiteten Mailprogramme, stellt UTF-8-codierte Textmails in seinen Standardeinstellungen beispielsweise in einer proportionalen Schrift dar, ISO-8859-codierte Mails hingegen in einer dicktengleichen Schrift. Letzteres kann aber aus Formatierungsgründen die gewünschte Darstellungsform sein, um beispielsweise tabellarische Daten zu strukturieren.

Für den besonderen *multipart*-Basistyp werden zwei Subtypen eine Rolle im Verlauf dieser Arbeit spielen. Dabei handelt es sich einerseits um den Typ *multipart/alternative,* welcher signalisiert, dass eine E-Mail mehrere verschiedene Darstellungsformen des gleichen Inhalts anbietet. So kann beispielsweise für einen HTML-Newsletter zusätzlich noch eine Version als reiner unformatierter Text bereitgestellt werden, sodass auch Mailprogramme, die keine HTML-Mails darstellen können, eine lesbare Ausgabe erzeugen können.

Der zweite Subtyp ist *multipart/related*, welcher verwendet wird, um einen aus verschiedenen zusammengehörigen Daten aufgebauten Teil des Newsletters zu kennzeichnen. Dies wird bei HTML-Newslettern mit eingebundenen Grafiken der Fall sein, da der Text selbst als HTML vorliegt, aber einige Grafiken im GIF- und JPEG-Format eingefügt werden, die mit dem HTML-Part semantisch verknüpft sind.

23 [WIKIPEDIA]

Da *multipart*-Nachrichten eine E-Mail in mehrere Abschnitte aufteilen, für die jeweils eine eigene Typangabe nötig sein kann, ist das Format hier etwas anders als bei den anderen Typen. So wird für einen *multipart*-Typ kein Zeichensatz angegeben, sondern stattdessen ein Begrenzungscode aufgeführt. Diese *Boundary* ist eine beliebige Zeichenkette, an welcher ein Mailprogramm erkennt, dass hier ein neuer Abschnitt der Mail beginnt. Innerhalb der Mail ist die Boundary an zwei führenden Minus-Zeichen zu erkennen. Nach einer solchen Begrenzung folgt ein eigener Header für den folgenden Abschnitt, in welchem ein eigener MIME-Typ spezifiziert werden kann. Nach der Auflistung aller Abschnitte folgt erneut die Boundary, der diesmal zusätzlich zu den vorangestellten Minuszeichen zwei weitere angehangen sind.

In Abbildung 4 folgt eine einfach aufgebaute Beispielmail mit Text- und HTML-Abschnitt, bei der die Boundarys farblich hervorgehoben sind:

```
From: Max Mustermann <max.mustermann@host1.de>
To: Erika Musterfrau <e.musterfrau@host2.de>
Date: Sun, 11 Nov 2007 17:41:23 +0100
X-Mailer: wwEdit 3.0
Subject: Dies ist ein Test
Mime-Version: 1.0
Content-Type: multipart/alternative;
    boundary="hier beginnt der naechste teil"

This is a multi-part message in MIME format.

--hier beginnt der naechste teil
Content-Type: text/plain; charset="iso-8859-1"
Content-Transfer-Encoding: 8bit

Hallo Welt.

--hier beginnt der naechste teil
Content-Type: text/html; charset="iso-8859-1"
Content-Transfer-Encoding: 8bit

<html>
  <head>
    <title>Dies ist ein Test</title>
  </head>
  <body>
    <h1>Hallo Welt</h1>
  </body>
</html>

--hier beginnt der naechste teil--
```

Abbildung 4: Syntax einer E-Mail mit mehreren Versandformaten

Der einleitende Text („`This is a multi-part message...`“) dient dabei lediglich als Information für Benutzer von Mailprogrammen, die diese Darstellungsform nicht unterstützen.

Eine weitere Komplexitätsstufe stellen E-Mails mit eingebundenen Grafiken dar. Bei gleichzeitiger Bereitstellung eines reinen Textformates ist hier die Schachtelung von *multipart*-Abschnitten notwendig. Die Grafiken selbst werden dabei (wie andere Binäranhänge auch) Base64[24]-codiert. Zur Referenzierung innerhalb der HTML-Version wird eine Content-ID angegeben. Zudem wird in einem weiteren Header-Feld *Content-Disposition* angegeben, ob es sich um einen echten Anhang handelt (*attachment*) oder um eine Datei, die innerhalb der Mail dargestellt werden soll (*inline*).

Ein Beispiel, diesmal mit unterschiedlicher farblicher Hervorhebung der verschiedenen Boundarys und der Content-ID der eingebundenen Grafik befindet sich in Abbildung 5 auf der folgenden Seite.

24 Base64 ist ein Algorithmus zur Codierung von 8-bit-Binärdaten. Das Ausgabeformat kommt mit einem 6-bit-Zeichensatz aus, der nur aus den Zeichen A-Z, a-z, 0-9, + und / besteht. So ist gewährleistet, dass die codierten Informationen über alle US-ASCII-fähigen Systeme transportiert werden können. Der Nachteil besteht allerdings darin, dass das Ausgabeformat aufgrund der Konvertierung von 8 zu 6 bit um ein Drittel größer ist als die ursprüngliche Datei.

```
From: Max Mustermann <max.mustermann@host1.de>
To: Erika Musterfrau <e.musterfrau@host2.de>
Date: Sun, 11 Nov 2007
      17:41:23 +0100
X-Mailer: wwEdit 3.0
Subject: Dies ist ein Test
Mime-Version: 1.0
Content-Type: multipart/alternative;
    boundary="hier beginnt der naechste teil"

This is a multi-part message in MIME format.

--hier beginnt der naechste teil
Content-Type: text/plain; charset="iso-8859-1"
Content-Transfer-Encoding: 8bit

Hallo Welt.

--hier beginnt der naechste teil
Content-Type: multipart/related;
    boundary="subabschnitt html"

--subabschnitt html
Content-Type: text/html; charset="iso-8859-1"
Content-Transfer-Encoding: 8bit

<html>
  <head>
    <title>Dies ist ein Test</title>
  </head>
  <body>
    <h1>Hallo Welt</h1>
    <img src="cid:grafiknr1">
  </body>
</html>

--subabschnitt html
Content-Type: image/gif; name="logo.gif"
Content-Transfer-Encoding: base64
Content-ID: <grafiknr1>
Content-Disposition: inline; filename="logo.gif"

R0lGODlhLgAuALMAAAgIfCUlfAUFhC4ukE1Nl1NTpmFhmXBwro2Nt6Kiu5OTxq6
uz76+4MXF2tPT5v39/iwAAAAALgAuAAAE/lDIR6ut7gTJAVFXeHGCKCJkKgWLaU
4u5QxqLRBObJW6s6WDgrAwAHACOZ3y4SMBDslL4geILkXNo1VEkBCursWzsUA0l
jRBCxw6HNgyIwBumdEpBwni/nDzFRIDfANrdIASDHQOVXwMHCBwDQF8TEYCb5GC
lFO3dAuaWAsGHgYEBQaFC4GeAkGQFggFBwstDg0NCFAURpNwBQADNAYhDgsKBwg
LWw9GoGADBwoIpjgNOcoPDRoGxQrGCglGb8sxNANdAAYDDg4IOQwO4UiEFkY59E
sJRH3ABJgICZicwYapwqkHDBKQE7HgwJNiM5Is6HVhjwVZeAr1KHApRy4L7wFDa
HxQQAGIAiNjEAC1IBGGkBTIhGBx5sCZK5JyOCjA0Bi2inMoWARjwNsCmM8EFUup
A1NBNgHe3ISzh9YdfHdu8bz66k4CA0MZ/NSxjtKDPUQU6Fx44YtZBg0YEKiS4Jw
1HcPMnrU0JQAAAAECILXQYKtZFEa8mUnEjpwfvY0lWAogi5wDAkPNFiOBIJmLA2
L1UhDDgRDbowhFMyGBldg4plf9XYJ9lgKDzGZNZYih64ECw3rFLJjaZioCZ5oDE
IA3C0OGLbTpcAQXAO6BcHlVuwi8gTuAwXB4KEHBATPbJTCWyO5KicQSdjb1cogA
ADs=
--subabschnitt html--
--hier beginnt der naechste teil--
```

Abbildung 5: Syntax einer E-Mail mit mehreren Formaten und Anhang

5 Entwurf eines Datenbankschemas

Nachdem nun die Grundlagen, auf denen diese Arbeit aufbaut, erörtert wurden, soll im folgenden Kapitel beschrieben werden, wie eine Datenbank zur Abbildung der gegebenen Anforderungen entworfen wurde. Zur Veranschaulichung der Zusammenhänge dient das ER-Diagramm in Chen-Notation aus Abbildung 6.

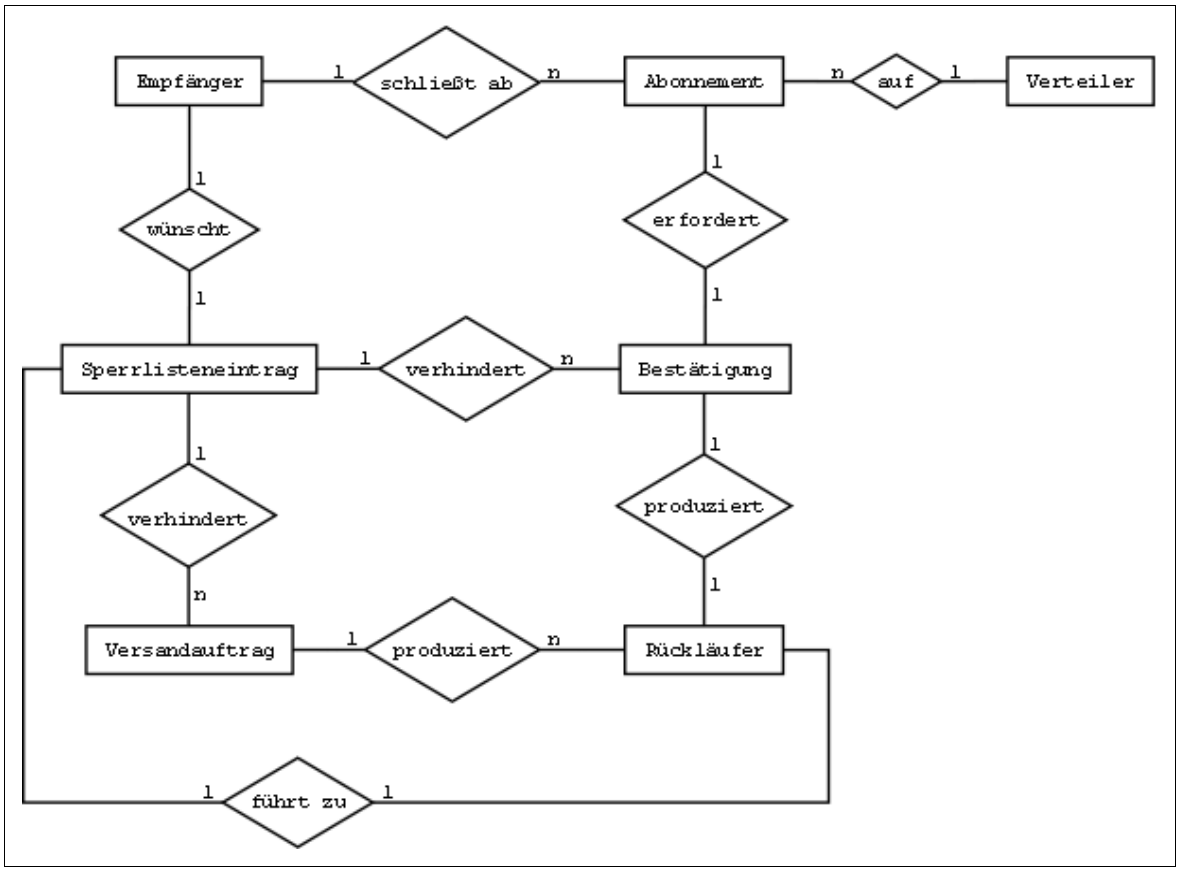


Abbildung 6: ER-Diagramm zur Veranschaulichung der Beziehungen

Aus dem in Kapitel 3 erarbeiteten Anforderungskatalog wird deutlich, welche Daten für den Versand von Newslettern verwaltet werden müssen: es gibt Newsletterempfänger (*recipients*) und Verteiler (*channels*). Beide stehen in einer m:n-Beziehung zueinander, da ein Verteiler eine beliebige Anzahl Empfänger beinhalten können soll, aber ein Empfänger auch gleichzeitig auf mehreren Verteilern vertreten sein kann. Um dieses Verhältnis auf Datenbankebene abzubilden wird eine verknüpfende Entität eingefügt: Abonnements (*subscriptions*).

Ein Abonnement erfordert eine Bestätigung (*opt*) durch den Empfänger, die allerdings nur dann möglich ist, wenn die Empfänger-Adresse nicht auf der Sperrliste (*blacklist*) steht. Ein Rückläufer (*bounce*) nach dem Versand der Bestätigung kann wiederum dazu führen, dass bei entsprechendem Schweregrad ein Sperrlisteneintrag angelegt wird. Ebenso kann ein Versandauftrag (*job*) durch gesperrte Adressen verhindert werden oder selbst Sperrlisteneinträge auslösen, wenn er Rückläufer produziert.

Die zu verwaltenden Entitäten lassen sich nun in Datenbanktabellen übertragen. Empfänger, Abonnements, Verteiler und Bestätigungen können in ihren Beziehungen so übernommen werden, wie es das ER-Modell beschreibt. Die Bestätigungs-Tabelle soll allerdings so allgemein formuliert werden, dass sie auch für weitere Funktionalitäten verwendet werden kann, die ein Double-Opt-In-Verfahren erfordern.

Die zu versendenden Newsletter-Mails werden in einer Job-Tabelle verwaltet. Die Beziehungen der Rückläufertabelle lassen sich in 2 Fremdschlüsseln auf Jobs und Bestätigungen abbilden. Die Sperrlistentabelle hingegen ist gänzlich unabhängig, die blockierende Beziehung vor dem Versand von Bestätigungs- oder Newsletter-Mails muss in der Programmlogik implementiert werden. Einträge auf der Sperrliste können optional auch auf einigen Verteilerlisten vorhanden sein, sind aber nicht davon abhängig. Es können also auch Mailadressen auf der Sperrliste stehen, die in keinem Verteiler auftauchen.

Damit sind die unterschiedlichen Arten von zu verwaltenden Funktionseinheiten abgesteckt. Im Folgenden wird im Detail geschildert, welche Daten in den einzelnen Tabellen gespeichert werden sollen.

5.1 cms_newsletter_recp

Die Empfängertabelle beinhaltet die meisten Informationen. Neben der E-Mailadresse (*addr*) sind für die persönliche Anrede zum Beispiel der volle Name (*firstname*, *surname*) mitsamt Anrede (*salutation*) und optionalen akademischen Titeln (*degree*) interessant. Außerdem besteht die Möglichkeit, eine Firma, Institution oder Organisation anzugeben (*institution*) sowie eine Position (*position*) innerhalb dieser. Die letztgenannten beiden Daten haben keinen Einfluss auf eine Anrede oder den Newsletterversand sondern sollen nur der Information des Newsletterbetreibers dienen. Für einen Presseverteiler ist es beispielsweise wissenswert, welche Publikationen ein Abonnent vertritt, für eine Rundmail innerhalb des Kundenunternehmens auch, welche Abteilungen bedient werden und welche Positionen die Empfänger dort innehaben.

Weiterhin ist ein Datum hinterlegt, welches besagt, wann diese Adresse zuletzt bestätigt wurde (*last_verify*). Dieses wird erstmalig bei erfolgreichem Double-Opt-In gesetzt, aber auch bei jedem weiteren Opt-In, wenn beispielsweise eine Eintragung auf einen weiteren Verteiler erfolgt. Dieses Datum dient ebenfalls nur der Information und hat keinen Einfluss auf den Versand des Newsletters.

Für das Rückläufer-Management wird ein Zähler hinterlegt (*bounce_no*), der je nach Anzahl und Schweregrad der Rückläufer (siehe Kapitel 7.1) inkrementiert wird. Ab einem konfigurierbaren Wert dieses Zählers erfolgt beim nächsten Newsletterversand kein Zustellungsversuch mehr an diese Adresse.

Für Statistikzwecke werden einige Daten in dieser Tabelle redundant vorgehalten. Die E-Mail-Adresse wird noch einmal in ihre Bestandteile Benutzername (*addr_local*) und Domain (*addr_dom*) zerlegt, wobei letzteres nochmal nach Top-Level-Domain (*addr_dom_l1*) und Second-Level-Domain (*addr_dom_l2*) aufgespaltet wird. Diese Informationen könnten auch alle im Nachhinein berechnet werden, sobald sie benötigt werden. Zur Vereinfachung von Abfragen für Statistiken (zum Beispiel von welchen Domains auffällig viele Rückläufer eingetroffen sind) werden sie hier aber bereits vorgehalten, da dies auch keine Daten sind, die Änderungen unterliegen.

Zusammgengefasst beinhaltet die Empfänger-Tabelle also folgende Informationen mit den angegebenen MySQL-Datentypen:

Tabelle 1: Struktur der Datenbanktabelle `cms_newsletter_recp`

Feld	Typ	Standardwert
id	INT (11)	*auto_increment*
addr	VARCHAR (255)	-
salutation	ENUM ('m', 'f')	*NULL*
degree	TEXT	*NULL*
firstname	TEXT	*NULL*
surname	TEXT	*NULL*
position	TEXT	*NULL*
institution	TEXT	*NULL*
last_verify	TIMESTAMP	0000-00-00 00:00:00
bounce_no	INT (4)	0
addr_local	VARCHAR (64)	-
addr_dom	VARCHAR (64)	-
addr_dom_l1	VARCHAR (16)	-
addr_dom_l2	VARCHAR (64)	-

5.2 cms_newsletter_channel

Im Vergleich dazu kommt die Verteilertabelle mit wenigen Daten aus. Zur Umsetzung der in Kapitel 3.2 aufgestellten Anforderungen sind eigentlich neben der obligatorischen ID nur der Verteilername (*name*) und eine optionale Auflistung von Rechteinhabern (*acl*) interessant, die diesen Verteiler verwalten dürfen.

Die Auflistung potenziell mehrerer Rechteinhaber (Benutzer, Benutzergruppen) in der ACL-Spalte widerspricht dem Normalisierungsprozess, sodass hier streng genommen zwei weitere Tabellen eingeführt werden müssten, die die Rechteinhaber beinhalten und die m:n-Beziehung von Verteilern zu Rechteinhabern abbilden. Allerdings handelt es sich hier um eine Spalte, die lediglich der Prüfung dient, ob ein beliebiges Recht aus der Rechtekombination eines Nutzers vorhanden ist.

Dafür sieht *ww***Edit** eine Schnittstelle vor, die mittels regulärer Ausdrücke diese Spalte hinreichend prüfbar macht. Als Beispiel sei die Definition in Abbildung 7 aus dem Formulargenerator des CMS genannt.

```
rnl_channel=check[@sql:channel@label@SELECT c.id  \
            AS channel, c.name AS label \
            FROM cms_newsletter_channel c \
            LEFT JOIN cms_newsletter_subscr s \
            WHERE ##USERPRINCIPALS:acl## \
            GROUP BY c.id ORDER BY label]
```

*Abbildung 7: Konfigurationseinstellung des ww**Edit**-Formulargenerators*

Hier erfolgt eine Abfrage auf die Verteiler, auf die der aktuell angemeldete Benutzer Bearbeitungsrechte besitzt, um diese dann als Checkboxen auswählbar zu machen. Ohne auf die genaue Syntax des Formulargenerators eingehen zu müssen ist erkenntlich, dass dabei eine Datenbankabfrage zusammengesetzt wird. Der unterstrichene Abschnitt dient als zu ersetzender Platzhalter für die Rechteprüfung. Dieser Abschnitt wird durch eine Kombination von Ausdrücken ersetzt, die die angegebene Spalte (hier *acl*) auf das Vorhandensein eines dem angemeldeten Benutzer verliehenen Rechtes prüft oder darauf, ob diese Spalte leer ist und somit alle Nutzer Verfügungsrechte besitzen. Eine fertige SQL-Abfrage könnte dann zum Beispiel so aussehen:

```
SELECT c.id  AS channel, c.name AS label
FROM cms_newsletter_channel c
LEFT JOIN cms_newsletter_subscr s ON s.channel=c.id
WHERE acl REGEXP "^(.+,)*u:aschulze(,.+)*$" OR
      acl REGEXP "^(.+,)*g:support(,.+)*$" OR
      acl IS NULL
GROUP BY c.id
ORDER BY label;
```

Abbildung 8: SQL-Abfrage auf Verteiler mit Rechteprüfung

Zugriff hätte also der User „aschulze“ oder ein Mitglied der Gruppe „support“. Leere ACL-Spalten bedeuten, dass keinerlei Rechtebeschränkungen bestehen, sodass jeder auf einen solchen Verteiler Bearbeitungsrechte hätte. Eine Prüfung dieser Art kann nicht nur auf der Newsletter-Verteilertabelle erfolgen, sondern in gleicher Weise in beliebigen vom CMS verwalteten Tabellen. Diese Abfragen erfolgen nur im Administrationsbereich und sind somit nicht zeitkritisch.

Insgesamt sieht die Verteilertabelle folgendermaßen aus:

Tabelle 2: Struktur der Datenbanktabelle `cms_newsletter_channel`

Feld	Typ	Standardwert
id	INT (11)	*auto_increment*
name	VARCHAR (255)	-
acl	TEXT	*NULL*

5.3 cms_newsletter_subscr

Die Abonnement-Tabelle enthält ebenfalls nur wenige Daten, da sie hauptsächlich die Empfänger- und die Verteilertabelle verknüpft. Daher sind nur zwei IDs als Fremdschlüssel nötig sowie ein Status-Flag, mit welchem ersichtlich ist, ob diese Newsletterbestellung bereits bestätigt wurde (erfolgreicher Double-Opt-In). Als weitere Information ist hier auch hinterlegt, welches Format der bestellte Newsletter haben soll. Verfügbar ist hier entweder die Textform oder HTML.

Tabelle 3: Struktur der Datenbanktabelle `cms_newsletter_subscr`

Feld	Typ	Standardwert
id	INT (11)	*auto_increment*
status	ENUM ('new', 'confirmed')	new
recp_id	INT (11)	-
channel_id	INT (11)	-
htmltext	ENUM ('html', 'text')	html

5.4 cms_opt_inout

Die Tabelle zur Verwaltung der Bestätigungsinformationen ist etwas komplexer. Hier wird neben der obligatorischen E-Mail-Adresse (*addr*) auch eine Information darüber benötigt, wofür eine Bestätigungsmail versendet wurde (*action*). Im Rahmen dieser Arbeit wird zwar nur der Versand von Newslettern behandelt, Bestätigungsmails sind aber auch für einige andere Funktionen im CMS von Bedeutung, zum Beispiel um eine E-Card zu versenden, das Anlegen

eines Benutzeraccounts zu bestätigen etc. Von dieser Information abhängig ist auch die Angabe eines Fremdschlüssels auf einen Datensatz, auf den sich die Bestätigungsinformation bezieht. Im Falle des Newsletterabonnements bezieht sich dieser Schlüssel auf einen Datensatz in der Tabelle `cms_newsletter_subscr`. Zur Identifikation eines Bestätigungslinks ist weiterhin ein Schlüssel zu hinterlegen (*jobkey*), dessen Funktion genauer im Kapitel 6.2 erläutert wird.

Die weiteren in dieser Tabelle abgelegten Daten beziehen sich darauf, wie das CMS mit angeklickten Bestätigungslinks umgehen soll. So wird beispielsweise noch der Zeitpunkt der Generierung des Datensatzes gespeichert (*timestamp*) sowie eine Information darüber, bis wann der Datensatz gültig ist (*valid_until*). Außerdem werden einige Seiten-IDs aus dem CMS gespeichert, namentlich die Seite, von der aus der Datensatz angelegt wurde (in der Regel die Seite mit dem Newsletter-Registrierungsformular, *page_src*), die Seite, auf welche verwiesen wird, wenn der Bestätigungslink abgelaufen ist (*page_expire*) und die Seite, auf die weitergeleitet wird, wenn der Bestätigungsvorgang erfolgreich abgeschlossen wurde (*page_success*). Für andere Bestätigungsvorgänge, die einen Abbruch der Bestätigung erlauben, ist auch ein Feld für eine Seite vorgesehen, auf die verwiesen wird, wenn man seine Bestätigung zurückgezogen hat (*page_fail*).

Tabelle 4: Struktur der Datenbanktabelle `cms_opt_inout`

Feld	Typ	Standardwert
id	INT (11)	*auto_increment*
action	VARCHAR (50)	-
addr	VARCHAR (255)	-
record_id	INT (11)	-
jobkey	CHAR (32)	-
timestamp	TIMESTAMP	*CURRENT_TIMESTAMP*
valid_until	TIMESTAMP	*NULL*
page_src	INT (11)	-
page_expire	INT (11)	-
page_success	INT (11)	-
page_fail	INT (11)	-

5.5 cms_newsletter_bounce

Sehr einfach gehalten ist die Tabelle, in der die Informationen für das in Kapitel 3.5 geforderte Rückläufermanagement verwaltet werden. Um Missverständnissen vorzubeugen sei an dieser Stelle erwähnt, dass hier nicht die Rückläufer selbst gezählt werden (dies passiert wie erwähnt in der Tabelle `cms_newsletter_recp`), sondern nur die nötigen Informationen vorgehalten werden, die die Auswertung eines Rückläufers ermöglichen.

Dazu gehört als Fremdschlüssel die ID eines Datensatzes. Abhängig davon, welche Art von Mail einen Rückläufer produziert hat, ob z.B. ein versendeter Newsletter oder eine Bestätigungsmail, bezieht sich dieser Fremdschlüssel aber wahlweise auf die `cms_newsletter_recp` oder die Tabelle `cms_opt_inout`. Um diese Unterscheidung ermöglichen zu können, gibt es für beide Fälle jeweils ein eigenes Fremdschlüsselfeld *recp_id* beziehungsweise *opt_id*. Abhängig davon, welches der beiden Felder einen Wert enthält, kann der betreffende Datensatz in den verknüpften Tabellen identifiziert werden.

Als weiteres Feld ist ein Bounce-Schlüssel vorzuhalten, über welchen der Rückläufer identifiziert wird. Eine nähere Erklärung der Rückläuferbehandlung folgt im Kapitel 7.1. Schließlich wird noch ein Zeitstempel vermerkt, um den Zeitpunkt des Anlegens des Datensatzes festhalten zu können. Dies ermöglicht eine regelmäßige Bereinigung der Tabelle, da man nach bestimmten Zeiträumen[25] davon ausgehen kann, dass eine E-Mail tatsächlich zugestellt wurde und kein Rückläufer mehr zu erwarten ist.

25 Erfahrungen haben gezeigt, dass Mailserver gescheiterte Zustellungsversuche in regelmäßigen Abständen wiederholen. Die Dauer dieser Wiederholungen variiert, in der Regel wurde der Versand nach 3 Tagen als endgültig fehlgeschlagen betrachtet. Eine Bereinigung der Tabelle von Einträgen, die älter als eine Woche sind, kann also als unproblematisch angesehen werden.

Zusammengefasst enthält die Tabelle `cms_newsletter_bounce` also folgende Daten:

Tabelle 5: Struktur der Datenbanktabelle `cms_newsletter_bounce`

Feld	Typ	Standardwert
id	INT (11)	*auto_increment*
opt_id	INT (11)	*NULL*
recp_id	INT (11)	*NULL*
bounce_key	CHAR (32)	-
time_create	TIMESTAMP	*CURRENT_TIMESTAMP*

5.6 cms_newsletter_job

Die Job-Tabelle verwaltet Daten über zu versendende Newsletter. Dazu gehören drei Zeitstempel: der Zeitpunkt der Erstellung des Versandauftrags (*create_time*), der Zeitpunkt, zu dem der Vorgang angestoßen wurde (*start_time*) sowie der Zeitpunkt der letzten Aktion (*last_action*).

Da der Versandvorgang stückchenweise erfolgt, also nicht alle Mails gleichzeitig an den Mail-Dienst gesendet werden sondern in Paketen zu einer definierten Anzahl Mails pro Sekunde, werden hier auch Informationen über den Status des Versandvorgangs abgelegt. Dazu gehören zwei Zähler über die Gesamtzahl an zu versendenden Mails (*mail_total*) sowie die Anzahl der bereits versendeten Mails (*mail_count*). Außerdem gibt es ein Status-Flag, aus welchem man lesen kann, ob der Job noch aussteht (`0`), bereits gestartet wurde und sich somit in der Verarbeitung befindet (`1`), erfolgreich abgeschlossen (`2`) oder manuell abgebrochen wurde (`3`). Ein fehlerhafter Abbruch des Versands lässt sich hier nicht erkennen, für die Umsetzung wird aber von einem Fehler ausgegangen, wenn der Versandstatus den Wert `1` ausweist, die letzte Aktion aber bereits länger als fünf Minuten zurückliegt.

Als weitere statistische Informationen wird hier die Seiten-ID (*pageid*) und der Betreff (*subject*) des Newsletters angegeben sowie die Session-ID des Benutzers, der den Versandvorgang gestartet hat. Abschließend enthält die Tabelle natürlich auch den Newsletter selbst als serialisiertes PHP-Array im Feld *data*.

Tabelle 6: Struktur der Datenbanktabelle `cms_newsletter_job`

Feld	Typ	Standardwert
id	INT (11)	*auto_increment*
pageid	INT (11)	-
session_id	CHAR (32)	*NULL*
subject	VARCHAR (255)	-
create_time	TIMESTAMP	*CURRENT_TIMESTAMP*
start_time	TIMESTAMP	0000-00-00 00:00:00
last_action	TIMESTAMP	0000-00-00 00:00:00
status	TINYINT (4)	0
mail_total	INT (7)	0
mail_count	INT (7)	0
data	LONGBLOB	-

5.7 cms_newsletter_blacklist

Die letzte benötigte Tabelle ist wieder recht einfach aufgebaut. Für eine Sperrliste gemäß Kapitel 3.6 sind nur zwei Informationen entscheidend: die zu sperrende E-Mail-Adresse (*addr*) sowie eine Information darüber, ob die Adresse gesperrt wurde, weil Mails an diese Adresse Rückläufer verursacht haben oder weil sie manuell gesperrt wurde (*bounced*).

Tabelle 7: Struktur der Datenbanktabelle `cms_newsletter_blacklist`

Feld	Typ	Standardwert
id	INT (11)	*auto_increment*
addr	VARCHAR (255)	-
bounced	TINYINT (1)	0

5.8 Verknüpfungen zwischen den Tabellen

Zur Verdeutlichung der Abhängigkeiten und semantischen Verknüpfungen zwischen den einzelnen Datenbanktabellen dient das Datenbank-Diagramm in Abbildung 9 auf der folgenden Seite.

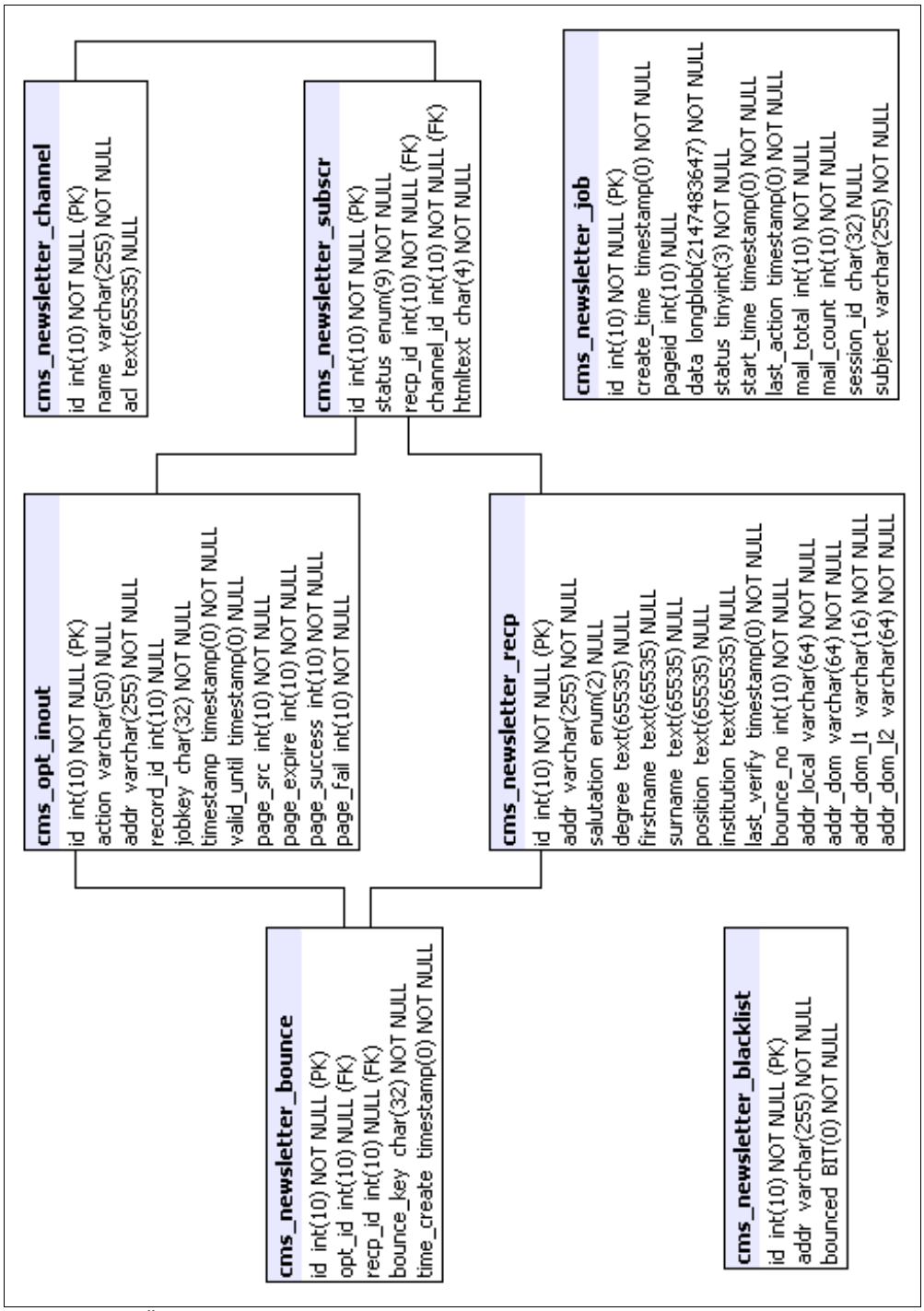


Abbildung 9: Übersicht der Tabellenverknüpfungen

6 Implementation der Basisfunktionen

Im Folgenden soll näher darauf eingegangen werden, wie die einzelnen Anforderungen aus Kapitel 3 praktisch umgesetzt wurden. Zunächst sollen in diesem Kapitel die Grundfunktionen einer Newslettersoftware implementiert werden. Dazu gehören das Abonnieren, Abbestellen, Anlegen und Versenden eines Newsletters.

6.1 Verwaltungsfunktionen

Allgemeine Funktionen, die keinen konkreten Newsletter betreffen sondern für den Registrierungsprozess auf Verteilern zuständig sind oder Empfänger verwalten, wurden in der Bibliothek `htdocs/management/include/nl_functions.php` zusammengefasst. Es wurde darauf geachtet, die Funktionen möglichst atomar und universal auszulegen, im Folgenden sollen die wichtigsten kurz in ihrer Funktion beschrieben werden.

`checkChannelACLs($channel_ids)`

Den Beginn macht eine Funktion zur Rechteprüfung von Verteilern. Aus einem übergebenen Array mit beliebigen Verteiler-IDs werden diejenigen zurückgegeben, auf die der eingeloggte Redakteur Verfügungsrechte hat. Anzumerken ist an dieser Stelle, dass der Administrator selbstverständlich alle Rechte besitzt und daher für diesen die Liste wie gehabt zurückgegeben wird. In den anderen Fällen wird aus der Datenbank eine Liste aller Verteiler bezogen, auf die der Redakteur Rechte besitzt und anschließend die Schnittmenge dieser Liste mit den übergebenen Verteilern zurückgegeben.

`getNLRecipients($which, $channels)`

Um einen Überblick über die Abonnements zu erhalten ist es erforderlich, eine Liste von Empfängern zusammenzustellen. Hierfür ist die Funktion `getNLRecipients()` zuständig, der zwei optionale Parameter übergeben werden können. Mit dem ersten Parameter wird ermittelt, welchen Status die Empfänger haben sollen, d. h. ob sie bereits bestätigte Empfänger sind (der Standardwert `confirmed`), ob ihre Bestätigung noch aussteht (`opts`) oder ob sie auf der Sperrliste vermerkt sind (alle anderen Werte). Der zweite Parameter ist ein Array, der Verteilerlisten-IDs

beinhalten kann. So können beispielsweise alle bestätigten Empfänger des Verteilers A ausgelesen werden, oder alle noch ausstehenden Bestätigungen für Verteiler B. Für die systemweit gültige und verteilerunabhängige Sperrliste wird der zweite Parameter ignoriert.

addToBlackList($addr, $bounced)

Es gibt zwei Gründe, aus denen eine Adresse auf der Sperrliste landen kann. Einerseits kann ein Redakteur sie manuell gesperrt haben, da der Besitzer der Adresse darum gebeten hat, keine weiteren E-Mails mehr zugestellt zu bekommen. Der zweite und häufigere Fall ist der, dass eine Mailadresse aufgrund schwerer Fehlermeldungen beim Zustellen von Newslettern als inaktiv gekennzeichnet wurde. Zwar kann bereits beim Versand eines Newsletters festgelegt werden, bis zu welchem Schweregrad an bisher aufgetretenen Fehlern eine Empfängeradresse noch als aktiv und damit relevant für den aktuellen Versand eingestuft werden kann. Damit allerdings ein Redakteur in der Empfängerliste ebenfalls einen Überblick darüber bekommt, welche Adressen Fehler produzieren, werden die Adressen auch automatisch auf die Sperrliste gesetzt.

Die Funktion nimmt zwei Parameter entgegen, wobei nur der erste Parameter, die zu sperrende E-Mailadresse, obligatorisch ist. Der zweite Parameter gibt Auskunft über den Schweregrad der Fehlermeldung bei einem Rückläufer. Beim Einfügen in die Sperrlistentabelle wird geprüft, ob bereits ein Datensatz existiert, andernfalls wird ein neuer Eintrag erzeugt.

addNLRecipient($addr, $additionalData)

Mit Hilfe dieser Funktion wird ein neuer Empfängerdatensatz angelegt. Als Parameter wird obligatorisch eine E-Mail-Adresse entgegengenommen sowie optional ein Array, in dem persönliche Daten wie Anrede, Name oder Institution aufgeführt sind.

Zuerst wird sichergestellt, dass die übergebene Zeichenkette ein gültiges Format für eine E-Mail-Adresse besitzt. Schlägt diese Prüfung fehl, wird der Bool'sche Wert *false* zurückgegeben. Handelt es sich hingegen um eine syntaktisch valide Adresse, so wird zunächst geprüft, ob bereits ein Eintrag mit dieser Mailadrresse in der Empfängertabelle vorliegt. Wenn dem so ist, beschränkt sich die Funktion darauf gegebenenfalls fehlende persönliche Daten zu ergänzen. Dies ist insbesondere dann interessant, wenn andere Tools, bei denen persönliche Daten abgefragt werden, Co-

Registrierungen für den Newsletter anbieten, d. h. eine gleichzeitige Newsletterregistrierung per Checkbox aktiviert werden kann. So können sich die Daten in der Empfängertabelle aus verschiedenen Datenquellen gegenseitig ergänzen.

Existierte noch kein Datensatz mit der übergebenen Mailadresse, so wird ein neuer angelegt. Die Rückgabe der Funktion ist die ID des angelegten oder bereits vorhandenen Eintrags in der Empfängertabelle.

`addNLSubscriptions($addr_ids, $channel_ids, $status, $version)`

Nachdem Adressen in die Empfängertabelle eingetragen wurden, muss auch eine Möglichkeit bestehen, für diese Datensätze Abonnements einzurichten. Diesen Zweck erfüllt `addNLSubscriptions()`. Die ersten beiden Parameter sind dabei selbsterklärend, sie enthalten Arrays mit jeweils einer oder mehreren IDs von Empfänger- bzw. Verteilerdatensätzen. Der dritte Parameter bezeichnet den Status des Abonnements, wobei die beiden Möglichkeiten „*new*“ für eine Neueintragung, die noch durch den Double-Opt-In-Prozess bestätigt werden muss, sowie „*confirmed*“ für eine abgeschlossene Eintragung zur Verfügung stehen. Wird dieser Parameter nicht übergeben, so wird von „new“ als Wert ausgegangen. Der letzte Parameter schließlich bezeichnet das Format der gewünschten Newsletter, wobei hier „*html*“ (Default) sowie „*text*“ möglich sind.

`importAddrs($addrlist, &$failedAddrs)`

Diese Funktion dient dem einfachen Befüllen der Empfängertabelle mit Adressen, persönliche Angaben werden dabei nicht berücksichtigt. Anhand typischer Trennzeichen wird diese zerlegt und jeder Bestandteil an die Funktion `addNLRecipient()` zum Test auf gültiges Format und gegebenenfalls Eintrag in die Datenbank übergeben. Der zweite als Referenz übergebene Parameter wird entsprechend der Rückgabe von `addNLRecipient()` befüllt mit den für die Funktion unverständlichen Zeichenketten. Die IDs der syntaktisch validen, in die Datenbank eingetragenen Adressen werden als Array zurückgegeben.

`createChannelIfNotExists($channelname)`

Um neue Verteiler anzulegen wurde diese Funktion entworfen. Die Benamung rührt daher, dass bei der Eingabe eines Namens für einen neuen Verteiler nicht davon ausgegangen werden kann, dass noch kein Verteiler dieses Namens existiert. Es kann beispielsweise passieren, dass einem

Redakteur die Existenz eines bestimmten Verteilers nicht bekannt ist, da er keine Verfügungsrechte für diesen hat. Nun wäre es natürlich nicht erwünscht, dass derjenige dennoch den Verteiler mit Adressen befüllt oder gar überschreibt. Daher prüft diese Funktion auf Existenz eines Verteilers und legt andernfalls einen neuen an.

`getChannelsFromForm($formdata)`

Bei dieser Funktion handelt es sich um eine Hilfsfunktion, die keine Aufgaben in der Datenbank übernimmt. Sie dient stattdessen der Auswertung eines Teils von Formulardaten. Im Redaktionsbereich gibt es mehrere Eingabemasken (insbesondere für den Import von Adressen), in denen mit Hilfe von Checkboxen eine Auswahl von Verteilern getroffen wird. Zudem kann ein freies Textfeld zur Eingabe eines Namens für einen neu anzulegenden Verteiler hinzukommen. Der Parameter $formdata enthält nun den entsprechenden Ausschnitt der Formulareingaben, in aller Regel aus dem `$_POST`-Array, der für Verteiler zuständig ist. Erwartet werden als Arrayschlüssel Channel-IDs, die den Feldwert „*on*“ besitzen, wenn dieser Channel im Formular ausgewählt wurde. Das Freitextfeld hat den Schlüssel „`_new_`“ und enthält bei Bedarf eine kommaseparierte Liste eines oder mehrerer Verteilernamen. Namen in dieser Liste werden an die Funktion `createChannelIfNotExists()` übergeben, die gegebenenfalls Verteiler anlegt und deren ID zurückgibt.

Auf die übrigen Funktionen zum Einholen oder Umbenennen von verfügbaren Verteilern, Entfernen von Einträgen aus der Empfänger- oder Sperrliste, Löschen von Verteilern, Abonnements oder Empfängern etc. soll an dieser Stelle aufgrund ihrer Trivialität nicht näher eingegangen werden. Nähere Informationen dazu können den Kommentaren im Quellcode (siehe Beilage) entnommen werden. Eine Beschreibung der `applyOpt()`-Funktion folgt in Kapitel 6.3.

6.2 Registrierung eines Newsletters

Der Ablauf einer Newsletterregistrierung kann anhand des Diagramms in Abbildung 10 nachvollzogen werden. Ein Webseitenbenutzer füllt das Registrierungsformular aus, welches bei ausreichenden Angaben ein inaktives Abonnement auf einen Verteiler anlegt und einen Bestätigungslink versendet. Wird der Link bestätigt, so wird das Abonnement aktiviert, andernfalls erfolgt keine weitere Aktion.

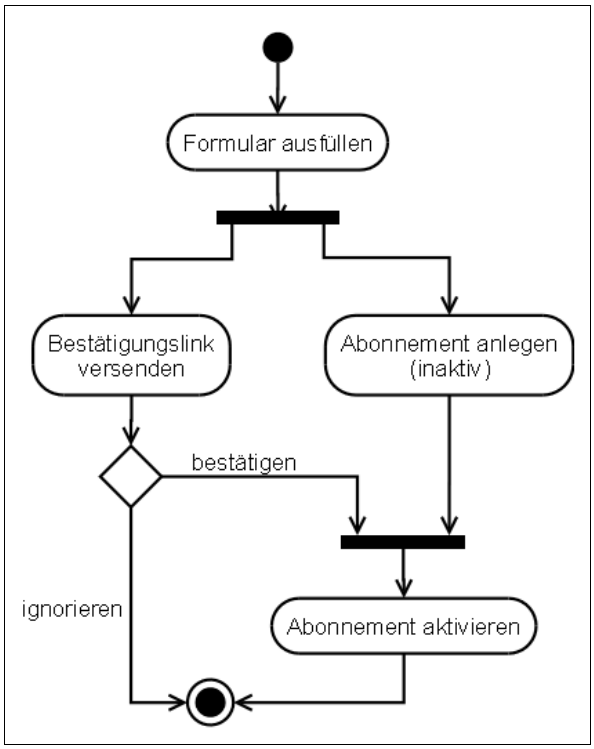


Abbildung 10: Ablauf einer Registrierung

Als Registrierungsformular für Newsletter wurde ein eigenes Inhaltsobjekt eingerichtet[26], mit dem eines von mehreren möglichen Formularen[27] eingebunden werden kann, je nach dem wieviele und welche persönlichen Informationen vom Abonnenten gewünscht werden. Die einfachste Form enthält lediglich ein Eingabefeld für eine E-Mailadresse, die umfangreiche Form weitere Felder für Anrede, vollen Namen, Firma/Organisation sowie Position. Zudem kann im Formular auch

26 vgl. `htdocs/conf/pt_exttext/cobjs_newsletter.ini`, Part `[wwe3_form_reg_newsletter]`
27 vgl. `htdocs/conf/universalform/unif_newsletter.ini`

abgefragt werden, ob ein HTML- oder eine Textversion gewünscht ist. Diese Formulare werden vom *ww***Edit**-eigenen Formulargenerator aus der Definition erzeugt. Auf diese Weise können schnell weitere Formulararten hinzugefügt werden, falls zusätzliche Informationen gewünscht werden oder die Kombination der anzugebenden Daten noch nicht in dieser Form verfügbar ist.

Zur Veranschaulichung folgt eine Formulardefinition für ein Registrierungsformular, in dem neben der Mailadresse auch Anrede und Name abgefragt werden (Abbildung 11), sowie ein Screenshot des generierten Formulars (Abbildung 12). Wie man sehen kann, wird über das gleiche Formular sowohl das Abonnieren eines Newsletters, als auch die Kündigung eines Abonnements abgewickelt.

Neben der Auswahl des Formulars muss ein Redakteur weitere Informationen im Inhaltsobjekt angeben. Dazu gehören Texte für die ausgelösten Bestätigungsmails, Weiterleitungsseiten für die jeweilige Reaktion auf den Opt-In, Fehlertexte (falls sich beispielsweise jemand für den Newsletter registrieren möchte, dessen Mailadresse auf der Sperrliste steht), der Zeitraum, den ein Bestätigungslink gültig ist sowie natürlich die Verteiler, die mit diesem Formular abonniert werden können.

```
[reg_newsletter_personalized]

rnl_salutation=select[,sal_mrs,sal_mr];\
               frontend_required??rnl_action!=rnl_unsubscribe;\
               folded??rnl_action==rnl_unsubscribe

   rnl_prename=label;auto_bobby;maxlength=20;\
               frontend_required??rnl_action!=rnl_unsubscribe;\
               folded??rnl_action==rnl_unsubscribe

   rnl_surname=label;auto_bobby;maxlength=20;\
               frontend_required??rnl_action!=rnl_unsubscribe;\
               folded??rnl_action==rnl_unsubscribe

      rnl_mail=label;constr=email_syntax,email_dns;\
               frontend_required;bobby_preset=mail_from

    rnl_action=radio[rnl_subscribe,rnl_unsubscribe];\
               hard_default=rnl_subscribe;no_br;\
               no_field_title;frontend_required
```

Abbildung 11: Formulardefinition des Registrierungsformulars

Abbildung 12: Ausgabe eines Registrierungsformulars

Die Auswertung von Formularen übernehmen in *ww***Edit** Formularprozessoren, die mit den Inhaltsobjekten für Formulare verknüpft sind. Allgemeine Funktionen wie die Prüfung auf korrekte Eingaben und Pflichtfelder müssen dabei allerdings nicht für jedes Formular separat implementiert werden, dies wird bereits von einer allgemeinen Routine übernommen, die Benutzereingaben mit

den in der Formulardefinition genannten Einschränkungen vergleicht und bei Bedarf entsprechende Fehlermeldungen generiert. Der Prozessor wird erst dann aufgerufen, wenn die Eingabedaten diesen Prüfschritt passiert haben und als korrekt bewertet wurden.

Aufgabe des Registrierungsformular-Prozessors (`w3fp_reg_newsletter.php`[28]) ist es nun, die geprüften Benutzereingaben entgegen zu nehmen und zu verarbeiten, im Regelfall also einen Double-Opt-In oder -Out auszulösen. Zunächst erfolgt allerdings ein Abgleich der angegebenen E-Mailadresse mit der Sperrliste. Außerdem wird geprüft, ob bereits eine Bestätigungsanfrage für die angegebene Adresse aussteht. In diesem Falle darf ebenfalls keine weitere Mail zugestellt werden und der Nutzer wird darüber informiert, dass seine bestehenden Bestätigungen zuerst bearbeitet werden müssen. Dieser Schritt ist nötig um zu verhindern, dass jemand beliebig viele Bestätigungsmails an eine fremde Adresse auslösen kann. Nur wenn beide Abgleiche ein negatives Ergebnis liefern, darf also eine Bestätigungsmail versendet werden.

Diese E-Mail setzt sich aus den redaktionell eingegebenen Texten des Inhaltsobjektes, sowie einem Bestätigungslink zusammen. Der Link enthält einen eindeutigen Schlüssel, der als *Jobkey* in der `cms_opt_inout`-Tabelle aufgeführt wird. Der Einfachheit halber wird hier ein MD5-Hash verwendet, in den die Benutzereingaben, die aktuelle Uhrzeit und ein Salz einfließen. Dadurch ist einerseits die Kollisionswahrscheinlichkeit äußerst gering, andererseits können die Schlüssel anderer Bestätigungsanfragen aber nicht einfach aus einem bestehenden Link geschlussfolgert werden. Ein generierter Link könnte beispielsweise so aussehen:

```
http://www.domain.com/nl-a4ed3dc61b45c934666bec9a312e7cb0/
```

Abbildung 13: Beispiel eines Bestätigungslinks zum Newsletterabonnement

Der Präfix „`nl-`“ dient der Kennzeichnung dieser Anfrage als Bestätigung eines Mail-Links. Mittels einer *RewriteRule*[29] werden URLs mit diesem Präfix, dem eine 32stellige alphanumerische Zeichenkette folgt, an ein gesondertes Skript übergeben, welches für die Behandlung von Bestätigungsanfragen verantwortlich ist.

28 der volle Pfad lautet: `htdocs/classes/pagetypes/processors/w3fp_reg_newsletter.php`

29 Dies ist eine Funktion des Apache-Moduls `mod_rewrite`, welches unter anderem ermöglicht, vorgegebene URLs auf ein bestimmtes Skript weiterzuleiten.

Abhängig von der gewünschten Aktion – Abonnement oder Kündigung – werden Datensätze in der Datenbank angelegt. Soll ein Newsletter neu registriert werden, so wird ein Benutzereintrag in der Empfängertabelle angelegt, in dem die angegebenen persönlichen Daten und die Mailadresse vermerkt sind. Dieser Eintrag wird über die Subscription-Tabelle mit den im CO spezifizierten Verteilern verknüpft, wobei diese Einträge über das Status-Feld noch als inaktiv gekennzeichnet werden. Der eigentliche Bestätigungsdatensatz in der Tabelle `cms_opt_inout` bekommt für das *action*-Feld (vgl. Kapitel 5.4) den Wert „`rnl_subscribe`“ zugewiesen und verweist in seinem Fremdschlüssel (*record_id*) auf den Eintrag in der Subscription-Tabelle. Werden gleichzeitig mehrere Verteiler abonniert, so werden mehrere gleichlautende Opt-Datensätze angelegt, die sich nur im jeweiligen Fremdschlüssel unterscheiden.

Andernfalls, wenn ein Benutzer sein Abonnement beenden möchte, werden aus der Datenbank die bestehenden Abonnements auf die Verteiler, für die das Inhaltsobjekt zuständig ist, zusammengetragen. Die IDs dieser Datensätze bilden dann die Fremdschlüssel für die angelegten Opt-Datensätze, die stattdessen das *action*-Attribut „`rnl_unsubscribe`“ erhalten.

Abschließend wird die generierte Bestätigungsmail an den Maildienst übergeben und dem Benutzer eine Meldung ausgegeben, die ihn über den weiteren Ablauf informiert. Der Fall, dass eine Bestätigungsmail nicht zugestellt werden kann, ist ebenfalls berücksichtigt, wird aber in Kapitel 7.1 beim Thema Rückläufer näher behandelt.

6.3 Bestätigung einer Registrierung/Kündigung

Wenn der Benutzer eine gültige Mailadresse angegeben hat, sollte innerhalb kurzer Zeit die Bestätigungs-Mail in seinem Postfach eintreffen. Diese kann nun entweder bestätigt werden, um das Abonnement abzuschließen, oder ignoriert werden, um keine weiteren E-Mails mehr vom System zu empfangen.

Im Bestätigungsfall ruft der Nutzer die in der Mail genannte URL mit dem generierten Opt-Schlüssel auf. Dieser wird im Skript `htdocs/nlink.php` verarbeitet, welches dediziert für Bestätigungsanfragen jeglicher Art zuständig ist. Anhand des Jobkeys wird der entsprechende Datensatz aus der Datenbank bezogen, welcher dann anhand des *action*-Feldes an die passende

Routine in der `applyOpt()`-Funktion in der `nl_functions.php` übergeben werden kann. Für den Newsletter spielen hier nur die beiden Routinen für *rnl_subscribe* und *rnl_unsubscribe* eine Rolle, es sind aber Routinen für andere Bestätigungsanfragen möglich.

Die Abarbeitung der Anfrage selbst ist recht simpel. Für ein bestätigtes Abonnement wird lediglich das Verifizierungsdatum in der Empfängertabelle aktualisiert und das Statusfeld in der Subscription-Tabelle auf „*confirmed*“ gesetzt, bei Kündigung eines Abonnements wird der Eintrag stattdessen aus letzterer Tabelle entfernt. Abschließend wird der Bestätigungsdatensatz selbst gelöscht und der Benutzer auf die im Registrierungsformularobjekt eingetragene Erfolgsseite geleitet.

6.4 Der Seitentyp „Newsletter“

Wie angesprochen wurde für Newsletter ein eigener Seitentyp definiert[30]. Von einer normalen Inhaltsseite unterscheidet sich ein Newsletter unter anderem dadurch, dass er nur einen Inhaltsbereich zur redaktionellen Bearbeitung zur Verfügung stellt, statt Kopfbereich und mehreren Inhaltsspalten steht in der Regel nur die Haupt-Inhaltsspalte zur Verfügung. Hier können einige einfache COs angelegt werden, die Auswahl hier ist ebenfalls beschränkt[31]. So wäre es wenig sinnvoll, einen Newsletter mit einem Formular zu versenden, zumal beachtet werden sollte, dass alle Inhalte des Newsletters auch eine einfache Textrepräsentation haben sollten. Die Auswahl beschränkt sich hier also auf ein automatisches Inhaltsobjekt für die persönliche Anrede im Newsletter (siehe Kapitel 6.6), einfache Textblöcke mit Überschrift, Bild und Link sowie bei Bedarf auch Seitenanreißern und -listen.

Zudem benötigt ein Newsletter spezifische Einstellungen, wie zum Beispiel Informationen über Namen und Mailadresse des Absenders, den gewünschten Verteiler, an den geschickt werden soll und Adressen für Rückläufer (*Return-Path*) und Antworten (*Reply-To*). Diese Eigenschaften werden in einem eigenen Attributreiter „Newsletter“ zusammengefasst[32].

30 vgl. `htdocs/conf/templates/newsletter.ini`
31 vgl. `htdocs/conf/pt_exttext/newsletter.ini`
32 vgl. `htdocs/conf/attributes.ini`

Eine Newsletterseite stellt außerdem einige Funktionen bereit, die explizit mit diesem Seitentyp verknüpft sind. Ein Redakteur bekommt in jedem Fall die Reiter „Versand“ um einen Newsletter zu verschicken (siehe Kapitel 6.8) und „Empfänger“, um die Liste der Adressaten einzusehen und zu bearbeiten (Kapitel 7.2). Bei Bedarf können noch die Reiter „Versandmanagement“ (Kapitel 7.5) und „Fehlzustellungsstatistik“[33] aktiviert werden, über welche Informationen und Statistiken den Versand betreffend angeboten werden.

6.5 Umwandlung einer CMS-Seite in die Newsletterformate

Prinzipiell kann die Newsletterseite als normale CMS-Seite verstanden werden, die auch über das Web aufgerufen werden kann. Dabei verhält sich die Seite wie jede andere Seite, zeigt also auch den Kopfbereich und Navigationen an. Dies kann genutzt werden, um beispielsweise ein Newsletterarchiv auf der Website anzubieten, über welches ältere Newsletter angesehen werden können oder auch um einem HTML-Newsletter einen Link hinzuzufügen, wo der Newsletter korrekt angezeigt werden kann, wenn die Darstellung im verwendeten Mailclient nicht wie gewünscht aussieht.

Für den Versand sind zwei Formate gewünscht: die einspaltige HTML-Ansicht und eine Textrepräsentation mit sehr einfacher Formatierung. Um eine Seite auf unterschiedliche Arten zu rendern, bietet das CMS eine Formatsteuerung an. Durch die konsequente Trennung von Inhalten und Markup mit Hilfe der Smarty-Engine ist es so möglich, die gleiche Seite mit unterschiedlichen Templates darstellen zu lassen.

Die Erzeugung einer CMS-Seite in HTML-Newsletterform ist dabei recht einfach. Geändert werden muss lediglich die Auswertung der Inhaltsbereiche, die sich hier auf einen einzelnen Bereich beschränkt, sowie der HTML-Kopfbereich. Hier ist es beispielsweise nicht gewünscht, Stylesheets oder JavaScripte einzubinden oder suchmaschinenrelevante Metainformationen einzubinden, die für Mailclients keinerlei Bedeutung haben. Der CSS-Abschnitt wird stattdessen

33 Die Funktionen dieses Reiters wurden nicht vom Autor umgesetzt, weswegen die Fehlzustellungsstatistik im Rahmen dieser Arbeit zwar erwähnt, aber nicht näher beschrieben wird.

direkt im Template aufgeführt und nur die wichtigsten Informationen wie Seitentitel und Zeichensatz in den Kopf übernommen. Zur Erzeugung der eigentlichen Inhaltsobjekte werden allerdings die gleichen Templates verwendet wie für die Webansicht.

Etwas komplexer ist die Umwandlung der Inhalte in ein Format für Textnewsletter. Hierfür mussten komplett eigene Templates im Textformat entwickelt werden, die aber eine sehr einfache XML-artige Auszeichnung erlauben, um beispielsweise Überschriften von normalem Text semantisch absetzen zu können. Folgendes Beispiel zeigt ein Template für einen einfachen Textabschnitt mit Überschrift und Link:

```
{txtmailfilter}{if $cc.subject.fieldvalue ne ""
  }<h2>{$cc.subject.fieldvalue|trim}</h2>
{/if}{if $cc.richtext.fieldvalue
    }{$cc.richtext.fieldvalue|trim
}
{/if}{if $cc.url.fieldvalue ne ""
    }<more href="{$cc.url.fieldvalue|link}" title="{if
$cc.urllabel.fieldvalue
              }{$cc.urllabel.fieldvalue
           }{elseif $cc.url.fieldvalue|linktarget:1 eq "intern"
             || $cc.url.fieldvalue|linktarget:1 eq "cms_intern"
              }{$cc.url.fieldvalue|page:"_name"
           }{else
              }extern{/if
           }" />
{/if}{/txtmailfilter}
```

Abbildung 14: Smarty-Template für ein Textobjekt

Der Übersichtlichkeit im Template abträglich ist der Umstand, dass Textnewsletter im Unterschied zu HTML wesentlich whitespace-sensibler sind, jedes Leerzeichen und jeder Zeilenumbruch an falscher Stelle sich also auf die Formatierung in der Ausgabe auswirkt. Erkenntlich ist allerdings die Kennzeichnung der Überschrift mittels `<h2>`-Tags sowie ein spezielles Linkformat `<more>`. Zu beachten ist auch, dass innerhalb der Variable `$cc.richtext.fieldvalue` HTML-Code enthalten sein kann, der noch umgewandelt werden muss.

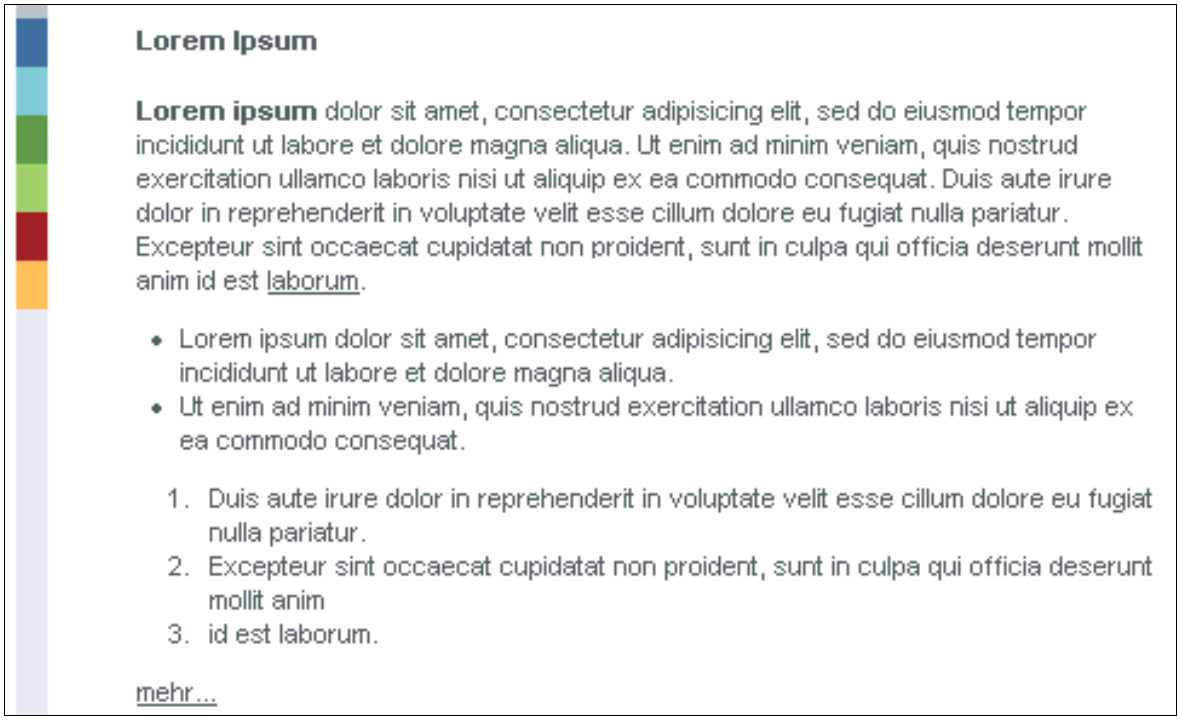
Diese Umwandlung erfolgt über einen Smarty-Block, der hier mit `txtmailfilter` bezeichnet ist. Aufgabe dieses Filters ist es, vorhandenes HTML oder HTML-ähnliche Elemente so zu formatieren, dass eine lesbare Textrepräsentation entsteht. Überschriften verschiedener Ebenen beispielsweise sollen unterschiedlich unterstrichen werden, Absätze eingerückt werden, Aufzählungslisten lesbar formatiert werden und Hyperlinks entsprechend umgewandelt werden,

dass auch in der Textform sowohl die Linkbeschriftung als auch die referenzierte URL erkenntlich ist. Natürlich gibt es für die Textumwandlung Grenzen, so ist es beispielsweise mit vertretbarem Aufwand nicht zu realisieren, auch HTML-Tabellen in eine Textform zu bringen, die auf gesetzter Maximalbreite von 78 Zeichen noch gut lesbar ist.

Die Funktionen des Filters sind in der Datei `smarty_mail.php`[34] aufgeführt. Der `txtmailfilter` bedient sich der von PHP bereitgestellten XML-Funktionen. Es wird ein XML-Parser initialisiert, der dann die Formatierung der einzelnen Elemente an Handler-Funktionen delegiert. So gibt es eine Funktion zur Bearbeitung öffnender und eine für schließende Tags, eine für die Zeichendaten (CDATA) zwischen Tags und einige Hilfsfunktionen, beispielsweise zur Umwandlung und Formatierung von Links. Durch diese Funktionen ist die Formatierung von Textmails strikt vorgegeben. Es ist beispielsweise nicht möglich, Überschriften anders zu unterstreichen als es in der jeweiligen Funktion definiert ist. Dies kann bei Bedarf aber mit geringem Aufwand konfigurierbar gemacht werden. Ebenso ist der Filter allgemein genug ausgelegt, um ebenso mit geringem Aufwand in einen generellen HTML-zu-Text-Konvertierer umgewandelt werden zu können.

Die zwei Screenshots in Abbildung 15 und 16 auf der folgenden Seite sollen die Arbeit des Filters veranschaulichen, indem die HTML- der Textausgabe gegenübergestellt wird.

34 voller Pfad: `htdocs/classes/smarty_ext/smarty_mail.php`



Lorem Ipsum

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

- Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.
- Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.

1. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur.
2. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim
3. id est laborum.

mehr...

Abbildung 15: Darstellung eines Textblocks in einem HTML-Newsletter

```
Lorem Ipsum
===========

*Lorem ipsum* dolor sit amet, consectetur adipisicing elit, sed do eiusmod
tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam,
quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo
consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse
cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non
proident, sunt in culpa qui officia deserunt mollit anim id est laborum
[LINK: http://www.wwedit.de].

   - Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod
     tempor incididunt ut labore et dolore magna aliqua.
   - Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi
     ut aliquip ex ea commodo consequat.


  1. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum
     dolore eu fugiat nulla pariatur.
  2. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia
     deserunt mollit anim
  3. id est laborum.

mehr... [LINK: http://www.php.net]
```

Abbildung 16: Darstellung eines Textblocks in einem Text-Newsletter

6.6 Personalisierte Newsletter

Nachdem nun die Grundlage gegeben ist Newsletter sowohl in Text- als auch in HTML-Form zu generieren, soll an dieser Stelle auf eine weitere der gesetzten Anforderungen eingegangen werden: die Personalisierung von Newslettern durch eine Anrede (Kapitel 3.4). Dies ist vergleichbar mit der Serienbrieffunktion vieler Textbearbeitungsprogramme. Dabei werden an bestimmten Stellen Platzhalter eingesetzt, die bei der Ausgabe (beim Drucken bzw. beim Newsletter während des Versandvorgangs) durch Informationen aus einem bestehenden Datenbestand ersetzt werden.

Für den Newsletter ist vorerst nur eine persönliche Anrede gewünscht, daher ist es nicht erforderlich, an beliebiger Stelle im Text Platzhalter definieren zu können. Die Anrede selbst erfolgt in der Regel zu Beginn, daher wurde die Umsetzung dieser Anforderung in Form eines eigenen Inhaltsobjektes beschlossen[35]. Das Objekt besteht aus zwei Teilen: der Anrede selbst, für die ein Redakteur festlegen kann, ob sie formell („Sehr geehrter Herr Dr. Schmidt“) oder informell („Liebe Susanne“) erfolgen soll, und einem frei zu gestaltenden Textabschnitt, in dem einleitende Informationen für den Newsletter untergebracht werden können.

Abgesehen von der Auswahl formell oder informell hat der Redakteur keinen Einfluss auf den Platzhalter. Dies hat den Nachteil, dass keine eigene Form der Anrede zusammengestellt werden kann (z.B. „Liebe Frau Maier“ statt „Liebe Susanne“). Andererseits ist es so ohne Kenntnis der Zeichenketten für verschiedene Platzhalter möglich, eine Anrede in üblicher Form einzurichten. Bei Bedarf können auch weitere Platzhalter implementiert werden, die dem Redakteur mehr Freiheiten in der Personalisierung seiner Newsletter geben, beispielsweise um auch inmitten eines Textabschnittes eine direkte Anrede einrichten zu können.

Die zwei vorgegebenen Platzhalter `@@@ANREDE@@@` beziehungsweise `@@@ANREDE_INFML@@@` werden beim Versand des Newsletters durch die angegebenen persönlichen Daten der Empfänger ersetzt. Es ist möglich, einen Newsletter an eine gemischte Empfängerliste zu schicken, in der sowohl Datensätze mit als auch ohne zusätzliche persönliche Daten vorhanden sind. Wenn keine Daten zur Anrede vorgefunden werden, wird der Platzhalter durch eine neutrale Anrede ersetzt, im Falle der formellen Anrede durch „Sehr geehrte Damen und Herren“ und im Falle der informellen

35 vgl. Abschnitt `[salutation]` in `htdocs/conf/pt_exttext/cobjs_newsletter.ini`

Anrede durch „Liebe Leserinnen und Leser“. Die Anredeformeln können nicht durch Redakteure geändert werden, allerdings ist es möglich, auf Kundenwunsch eine andere Anredeformel in der Internationalisierung des CMS festzulegen.

6.7 Funktionen eines Newsletters

Für Funktionen, die das Zusammenstellen und Vorbereiten eines Newsletters auf den Versandvorgang betreffen, wurde eine eigene Klasse definiert (`newsletter.php`[36]), die mit der ID einer zu versendenden CMS-Seite initialisiert wird. Auf deren wichtigste Methoden soll in diesem Kapitel eingegangen werden.

Für Methodennamen gilt hier die Konvention, dass Methoden, die mit einem Unterstrich beginnen, nur innerhalb der Klasse selbst aufgerufen werden sollten (bezogen auf den konsequenteren Objektansatz von PHP5 oder Java also als *private* zu betrachten sind), wohingegen die übrigen Methoden auch von außerhalb der Klasse aufgerufen werden können (*public*).

`render($version, $iconv, $area)`

Mittels dieser Methode wird der Rendervorgang veranlasst, an dessen Ende der Newsletter in HTML- oder Textform steht. Der erste Parameter entscheidet, welche der beiden Formatversionen gewünscht ist, der zweite darüber, ob eine Zeichensatzkonvertierung erfolgen soll. Der dritte Parameter schließlich bestimmt den zu versendenden Inhaltsbereich, also in der Regel die Hauptspalte.[37]

Der größte Teil der Methode besteht aus dem Sammeln der benötigten Daten für eine Seite. Dazu gehören zum Beispiel, unter welcher Domain sich die zu versendende Seite befindet, um seiteninterne Links entsprechend darauf verweisen lassen zu können. Die sogenannte Domaingroup wiederum legt fest, in welchem Design der Newsletter in seiner HTML-Form dargestellt werden

36 voller Pfad: `htdocs/classes/newsletter.php`

37 Zum aktuellen Zeitpunkt verfügt diese Methode über 3 weitere Parameter, die aber keine Bedeutung für den Versand eines Newsletters haben. Sie wurden nachträglich eingefügt, um weitere Funktionalitäten, wie beispielsweise das Versenden von Grußkarten oder Weiterempfehlungen mit Hilfe der Newsletterklasse zu ermöglichen.

soll. Nachdem alle wichtigen Informationen verfügbar sind und in ein Smarty-Objekt übergeben wurden, wird die `internalDisplay()`-Methode der Page-Klasse selbst damit beauftragt, die Seite unter den gegebenen Bedingungen im gewünschten Format zurückzugeben.

Die so generierte Rohform wird allerdings noch überarbeitet. So werden zum Beispiel noch einige Ersetzungen vorgenommen[38] und aus Kompatibilitätsgründen spezielle Sonderzeichen in Zeichen umgewandelt, die im normalen ASCII-Zeichensatz vorhanden sind (zum Beispiel typografische durch gerade Anführungszeichen). Für HTML-Newsletter wird zudem die Methode `_cleanUpHTML()` aufgerufen, die den generierten Quellcode weitgehend komprimiert, indem HTML-Kommentare und überflüssiger Whitespace entfernt werden. Abschließend erfolgt bei Bedarf eine Zeichensatzkonvertierung.

`_attachImages(&$content, &$images)`

Wie in Kapitel 3.3 gefordert, sollen in HTML-Newsletter Bilder eingefügt werden können. Der Parameter `$content` enthält die Rückgabe aus der `render()`-Methode, also eine als Newsletter gerenderte HTML-Seite. Die hier aufgrund ihrer Einfachheit nicht näher betrachtete Methode `_imageFilter()` macht eingebundene Grafiken anhand von Pfadangaben in *src*-Attributen ausfindig und übergibt sie über den Parameter `$images` der Methode `_attachImages()`, um die entsprechenden Dateien an die Mail anzuhängen.

Zum Anhängen der Grafiken werden die Dateien binär in eine Variable geladen und Base64-codiert. Außerdem werden die Header-Angaben zu den Anhängen vorbereitet, wie es in Kapitel 4.2 definiert wurde. Anschließend werden alle Vorkommen des Bildpfades im HTML-Code durch die vergebene Content-ID ersetzt. Um den (wenn auch unwahrscheinlichen) Fall zu verhindern, dass die gleiche Bilddatei mehrfach angehängt wird, wenn sie mehrfach eingebunden wurde, erfolgt eine Duplikatprüfung.

Die Anhänge selbst werden in der Klassenvariable `$attachments` gesammelt und in der folgend beschriebenen Methode `_makeContent()` mitsamt der gerenderten Versionen formatiert.

38 Diese Ersetzungen sind für den Newsletter selbst auch nicht von Bedeutung und wurden im Zusammenhang mit anderen Funktionen implementiert.

_makeContent()

Nachdem alle gewünschten Newsletterversionen zusammengestellt wurden und die Anhänge erzeugt wurden, müssen diese Daten entsprechend der Vorgaben aus Kapitel 4.2 in einen E-Mail-Body zusammengeführt werden. Diese Aufgabe übernimmt `_makeContent()`.

Hier wird geprüft, welche Versandformate gesendet werden, ob beispielsweise ausschließlich eine Textmail gewünscht ist oder eine HTML-Mail mit angehängten Bildern und einer Textrepräsentation. Basierend darauf wird der letztendlich zu versendende Mail-Body mitsamt der zu generierenden Boundarys und der eventuell erforderlichen Schachtelung zusammengesetzt. Die Rückgabe ist ein assoziatives Array, welches unter dem Schlüssel „`_body`“ den gesamten Mail-Body als String enthält und zusätzlich weitere Schlüssel mit Header-Inhalten wie beispielsweise der *Content-Type*-Angabe haben kann.

prepare($versions, $recipients, $headers, $options)

Die öffentliche Methode `prepare()` koordiniert die Zusammenstellung des Newsletters bis hin zur Speicherung des Versandjobs in der Datenbank. Dabei bedient sie sich der vorangehend beschriebenen klasseninternen Methoden.

Übergeben werden die gewünschten Versandformate („`text`“ und/oder „`html`“) als Array und die Liste der Adressaten, ebenfalls als Liste von Datensätzen. Über den Parameter `$headers` werden einzubindende Kopfdaten übergeben, dazu sollten Adresse und Name des Absenders ebenso wie Rückläufer- und Antwortadresse gehören, die außerhalb der Newsletterklasse zusammengestellt werden. Der letzte Parameter ermöglicht eine Steuerung des eigentlichen Versandvorganges über Parameter. Momentan werden hier die Optionen „`chunk_size`“ und „`chunk_pause`“ verstanden, die Auskunft darüber geben, in welchen Abständen wieviele Nachrichten am Stück an den Mail Transport Agent (MTA) übergeben werden. Über eine Angabe von 5 für `chunk_size` und 1 für `chunk_pause` kann man so festlegen, dass nach der Übergabe von fünf Mails an den MTA eine einsekündige Pause erfolgt bevor die nächsten fünf Mails abgesetzt werden.

6.8 Versand eines Newsletters

Nachdem die Funktionalität der Newsletterklasse weitgehend beschrieben ist, soll nun darauf eingegangen werden, wie ein Newsletter versendet wird, wie der Prozessablauf aussieht und wie die letztendliche Übergabe an den MTA erfolgt.

Nachdem ein Newsletter mit entsprechenden Texten und gegebenenfalls Bildern erstellt wurde, wird der Versandvorgang über den Reiter „Versand“ im Redaktionsbereich gestartet (`page_send_newsletter.php`[39]). Hier bietet sich dem Redakteur die Eingabemaske aus Abbildung 17. Über die ersten beiden Checkboxen kann hier vor dem Versand überprüft werden, wie der gerenderte Newsletter aussieht. Gerade für die Textansicht ist dies durchaus zu empfehlen, da man auf diese aufgrund der automatischen Konvertierung von HTML zu Text nur begrenzt Einfluss beim Einstellen der Inhalte hat.

Anschließend erfolgt die Auswahl der Versandformate. Im darauf folgenden Textfeld können E-Mail-Adressen für einen Testnewsletter angegeben werden, dieses Feld wird mit der angegebenen Adresse des eingeloggten Benutzers vorausgefüllt. Abschließend kann ausgewählt werden, ob zunächst ein Testnewsletter an die soeben eingetragenen Empfänger gesendet oder der finale Newsletterversand eingeleitet werden soll. Vor dem endgültigen Versand ist allerdings immer der Versand einer Testmail zu empfehlen um zu prüfen, ob die Darstellung in den verschiedenen E-Mail-Clients[40] wie gewünscht ist.

39 voller Pfad: `htdocs/management/page_send_newsletter.php`, das zugehörige Template ist `htdocs/management/templates/backend/page_send_newsletter.tpl.html`

40 Die Erfahrung hat gezeigt, dass die Darstellung von HTML-Mails in den Clients Mozilla-Thunderbird vergleichbar ist mit der Darstellung im Browser Mozilla-Firefox, da die gleiche Rendering-Engine (Gecko) verwendet wird. Ebenso ist die Ausgabe von Microsoft Outlook bis Version 2003 vergleichbar mit der Darstellung im Microsoft Internet Explorer der jeweiligen auf dem System installierten Version. Leider wurde diese Konstanz mit Microsoft Outlook 2007 gebrochen, da die Darstellung von HTML-Mails hier nicht mehr auf der Rendering-Engine des Internet Explorers basiert sondern auf der von Microsoft Word. Ebenfalls als problematisch hat sich die Darstellung in Lotus Notes erwiesen, da die CSS-Unterstützung des darin enthaltenen Mailprogramms nur sehr wenige Formatierungen unterstützt.

Abbildung 17: Eingabemaske zum Newsletterversand

Nach Bestätigung des Versands über den Button wird das Formular abgesendet. Die Seite öffnet sich erneut und ein oder mehrere Popups (`nl_progress.php`[41]) erscheinen auf dem Bildschirm, die Auskunft über den Status des Versands des jeweiligen Newsletterformates geben (Abbildung 18). Daraus wird ersichtlich, wieviele Newsletter bereits gesendet wurden, wie groß das Mailvolumen insgesamt ist, wie lange der Versand bereits dauert und ein geschätzter Wert, wie lange er noch andauern wird. Außerdem kann der Versand über das Popup auch abgebrochen werden. Abgesehen davon hat es aber keinerlei Einfluss auf den Versandvorgang, es dient lediglich der Information und kann auch geschlossen werden, ohne dass der Versand abbricht.

41 voller Pfad: `htdocs/management/nl_progress.php`

Das Popup aktualisiert in kurzen Abständen seine Anzeige. Dafür erfolgt im Sekundentakt ein RPC-Aufruf mittels JavaScript an den Server, welcher die wichtigsten Informationen aus der Datenbank lädt und formatiert zurückgibt. Die Rückgabe wird an den entsprechenden Stellen eingesetzt. Dieses Verfahren der teilweisen Ersetzung von Seiteninhalten wird gemeinhin mit dem Begriff „Asynchronous JavaScript and XML“ (AJAX)[42] beschrieben, wenngleich in diesem Fall kein XML als Transportformat verwendet wird sondern lediglich eine Auflistung von Daten mit einem Trennzeichen zwischen den Datenfeldern. Aus Performanzgründen wurde für das Popup auch auf die Benutzung von Templates sowie die Einbindung der kompletten *ww***Edit**-Funktionalitäten verzichtet.

Abbildung 18: Popup zum Versandstatus

Soweit zu dem, was ein Redakteur vom Versandvorgang sieht, aber was spielt sich im Hintergrund ab?

42 [WIKIPEDIA]

Nach dem Absenden des Versandformulars wird die Newsletterklasse mit der aktuellen Seiten-ID initialisiert. Zunächst wird die Empfängerliste zusammengestellt durch Aufruf der Methode `getMyRecipients()`. Hier wird nicht die in der `nl_functions.php` definierte Methode `getNLRecipients()` verwendet, da für die Bestimmung der Empfänger des Newsletters nicht nur das Abonnement auf die gewählten Verteiler entscheidend ist, sondern auch der Schweregrad der bisherigen Rückläufer sowie eventuell eingegebene Test-Adressen. Als Rückgabe reicht hier auch eine Liste von E-Mail-Adressen statt der gesamten Empfängerdatensätze mitsamt persönlicher Daten und Verteilerabonnements.

Nach dem Zusammentragen der Empfänger werden einige Header-Daten wie die Absender-, Rückläufer und Antwortadressen formatiert und ebenso die Versandoptionen festgelegt, die im Abschnitt über die `prepare()`-Methode erläutert wurden. Diese Daten werden anschließend an `prepare()` übergeben, um den Newsletter zusammenzustellen und entsprechende Jobs in der Datenbank anzulegen. Wenn dies alles ohne Fehler ablief, kann der Newsletterversand per Aufruf der `send()`-Methode eingeleitet werden. Diese entkoppelt den eigentlichen Versand vom aktuellen Prozess (der erneuten Generierung des Versandformulars) und gibt die betreffenden Job-IDs zurück, sodass für diese Versandaufträge ein Statuspopup geöffnet werden kann.

Die Prozessentkoppelung erfolgt unterschiedlich je nach auf dem Webserver verwendetem Betriebssystem. Aktuell sind zwei Varianten für Linux- und Windowssysteme implementiert[43]. Unter Linux kann ein Prozess detached werden, indem alle seine Ausgaben in Dateien oder das NULL-Device geleitet und der Prozess als ganzes durch Anhängen eines kaufmännischen Und (&) in den Hintergrund geschoben werden. Für Windows wird eine Instanz des Windows Scripting Hosts benötigt, die den Prozess anschließend in einem eigenen (minimierten) Fenster startet.

43 Zwar wird *ww***Edit** produktiv nur auf Linux-Servern betrieben, das Entwicklungssystem setzt allerdings auf Windows XP auf. Daher kommt bei der Entwicklung die Windows-Variante zum Einsatz.

Im Wesentlichen erfolgt ein Newsletterversand über folgende Befehlsfolge:

```
$nl = new Newsletter ( $pageid );

foreach ( $versions as $version )
    $recipients [ $version ] =
        $nl -> getMyRecipients ( $send_opt,
                                 $version,
                                 $test_addrs );

/* [...] Generierung $headers und $options [...] */

$jobs_ok = $nl -> prepare ( $versions,
                            $recipients,
                            $headers,
                            $options );

if ( $jobs_ok )
        $job_ids = $nl -> send();
```

Abbildung 19: Code-Abschnitt zum Versand eines Newsletters

6.9 Der Massenmailer

Der abgekoppelte Versandvorgang erfolgt über ein eigenes Skript, die `bin/massmail.php`. Neben dem Versand erfolgt auch erst hier die eigentliche Personalisierung. Das Skript verfügt über ein detailliertes Logging, einzelne Programmschritte werden ausführlich protokolliert und in ein Logfile geschrieben, um Fehler beim Versand nachvollziehen zu können.

Die `massmail.php` ist ein Kommandozeilenskript, welches mindestens einen Parameter benötigt. Als Parameter werden IDs von Job-Datensätzen in der Datenbank angenommen. Es können mehrere Aufträge hintereinander erfolgen durch Übergabe mehrerer Parameter, bisher wird aber nur jeweils ein Parameter übergeben. Für den gleichzeitigen Versand von HTML- und Textversion eines Newsletters beispielsweise werden zwei voneinander unabhängige Instanzen des Massmailers aufgerufen.

Zunächst wird ein Job-Datensatz aus der Datenbank geladen und dessen Status geprüft. Neben neu angelegten Aufträgen können auch Fälle auftreten, dass ein Versandauftrag manuell abgebrochen wurde oder durch einen Fehler nicht korrekt beendet werden konnte. In letzterem Fall wird ein Zeitvergleich zwischen der letzten Aktion, die in der Datenbank vermerkt ist, und der aktuellen Uhrzeit angestellt. Nur wenn die letzte Aktion bereits fünf Minuten zurückliegt, der

Status des Auftrages aber nicht „abgeschlossen“ lautet, wird davon ausgegangen, dass der Versand fehlerhaft abgebrochen ist. Sowohl im Falle des fehlerhaften wie auch des beabsichtigten manuellen Abbruchs wird die Empfängerliste um die Anzahl der bereits versendeten Mails gekürzt, damit die jeweiligen Empfänger den Newsletter nicht doppelt zugestellt bekommen.

Sollte bis hierhin kein Abbruch erfolgt sein, da beispielsweise keine Job-ID übergeben wurde, der entsprechende Versandauftrag in der Datenbank nicht gefunden werden konnte, der Auftrag schon erfüllt ist oder bei möglicherweise fehlerhaftem Abbruch die erforderliche Zeitspanne von fünf Minuten noch nicht verstrichen ist, so wird jetzt der Startzeitpunkt in der Datenbank vermerkt und mit dem Versand begonnen.

Die Empfänger werden nun einzeln abgearbeitet. Zunächst wird in der Empfängertabelle der Datenbank nach einem Eintrag mit der entsprechenden E-Mail-Adresse gesucht. Wurde ein Datensatz gefunden, so wird das „To“-Headerfeld auf diesen angepasst und gegebenenfalls der volle Name aufgenommen. Die Adressangabe des Empfängers lautet anschließend beispielsweise auf „`Dr. Werner Maier <w.maier@hostname.de>`“. Außerdem wird im Body-Abschnitt der Mail nach den Platzhaltern für eine persönliche Anrede gesucht. Wenn ein Platzhalter gefunden wurde, so wird aus den persönlichen Daten eine entsprechende Anredeformel generiert und der Platzhalter ersetzt. Falls kein Empfängerdatensatz gefunden wurde[44] oder keine persönlichen Daten zur angegebenen Adresse verfügbar sind, so wird ein eventuell gefundener Platzhalter durch eine neutrale Anredeformel ersetzt.

Nachdem die Daten vorbereitet wurden, werden Empfängeradresse, Header und Mailbody an ein `PEAR::MAIL`-Objekt übergeben, welches daraus eine E-Mail generiert und über SMTP versendet. Der Status des Versands wird in der Datenbank aktualisiert. Außerdem wird an dieser Stelle geprüft, ob der Versand über das Statuspopup abgebrochen wurde. Ist dies nicht der Fall wird nach einer eventuellen Pause (falls die Newsletter paketweise verschickt werden sollen) mit der nächsten Empfängeradresse fortgefahren, bis der Versandauftrag abgearbeitet ist.

44 Dies passiert im Falle des Newsletters dann, wenn ein Testversand an eine E-Mail-Adresse erfolgt, die keinen Verteiler abonniert hat.

7 Erweiterte Funktionalitäten

Das Abonnieren, Erstellen und Versenden von Newslettern ist damit implementiert, die Basisfunktionalität für *ww***News** also gegeben. In diesem Kapitel soll nun auf die erweiterten Funktionen eingegangen werden, die für den Newsletter nicht von essentieller Wichtigkeit sind, das Tool aber in seinem Funktionsumfang abrunden.

7.1 Rückläuferbehandlung

Ein Kernfeature der neuen *ww***News**-Software ist eine automatisierte Behandlung von Rückläufern. Um dies zu ermöglichen, muss eine Voraussetzung erfüllt werden: Jede ausgehende E-Mail muss mit einer Kennzeichnung versehen werden, die diese Mail für das System eindeutig identifizierbar macht. Hierfür ist ein Bounce-Key zuständig, der sowohl beim Versenden einer Bestätigungsmail als auch beim Versand eines Newsletters für jede einzelne E-Mail generiert wird. Dieser Schlüssel wird in der E-Mail im Header-Feld `X-wwNews-Bounce-Key` abgelegt und ebenso in der Datenbanktabelle `cms_newsletter_bounce` vermerkt. Die Auswertung der Rückläufer wird vom Skript `bin/bouncehandler.php` übernommen.

Allerdings müssen zuvor einige Vorkehrungen getroffen werden, damit die Rückläufer überhaupt an dieses Skript geleitet werden. Diese Einstellungen können nicht im CMS selbst vorgenommen, sondern hierfür muss die Serverkonfiguration angepasst werden. Zum einen muss der Server, auf dem das CMS läuft, E-Mail empfangen können. Dazu wird in der Regel ein sogenannter MX-Record im DNS[45] benötigt sowie ein eingerichteter Mailserver, der aus dem Internet auf dem SMTP-Port 25 erreichbar sein muss. Weiterhin muss ein Mailkonto auf dem Server bestehen, welches so konfiguriert ist, dass eingehende Mails nicht an eine Mailbox zugestellt sondern an das Rückläufer-Skript übergeben werden. Auf den zur Verwendung kommenden Debian-Systemen ist daher ein Exim-Dienst[46] installiert und ein sogenannter Pipe-Transport konfiguriert, um eingehende E-Mails an Skripte übergeben (*pipen*) zu können.

45 Der MX Resource Record einer Domain gibt an, an welchen Host E-Mails an eine Domain zugestellt werden sollen. Für jede Domain kann es mehrere solcher Einträge geben, die sich auch auf physikalisch unabhängige Server beziehen können. Für die beschriebene Konfiguration der Rückläuferverwaltung von *ww***News** muss der MX-Eintrag allerdings auf den Server verweisen, auf dem das CMS installiert ist.

46 [EXIM]

Die E-Mail-Adressen, an welche Rückläufer gesendet werden sollen, sind in der Regel nach der Konvention formatiert, dass sich der User-Abschnitt der Adresse (der Part vor dem @) aus dem Begriff „bounce“ und einem Kürzel zusammensetzt, welches die jeweilige CMS-Installation auf dem Server kenntlich macht. Ein Beispiel wäre `bounce-test@example.com`.

Um diese Adresse korrekt weiterzuleiten ist ein Eintrag in der Konfigurationsdatei `/etc/aliases` nötig. Für die angegebene Adresse würde der Eintrag wie folgt lauten:

```
bounce-test: "| /srv/test/bin/bouncehandler.php"
```

Abbildung 20: Konfigurationseintrag in der /etc/aliases

Dies besagt, dass der Inhalt eingehender Mails an Adressen mit dem User-Abschnitt „`bounce-test`“ komplett an das Skript `/srv/test/bin/bouncehandler.php` übergeben werden soll. Dieses Skript kann die E-Mail dann aus der Standardeingabe (STDIN) lesen und verarbeiten.

Um nun zu zeigen, wie der Bouncehandler einen Rückläufer auswertet, sei an dieser Stelle ein Beispiel gegeben, wie eine gewöhnliche Rückläufermail aussieht (Abbildung 21). Zu Beginn der E-Mail stehen die Header des Mailer Daemons, der die Mail abgewiesen hat. Dort befinden sich allerdings keine Daten, die für die Auswertung eine Rolle spielen. Im Body-Abschnitt der Mail hingegen befindet sich in aller Regel eine Fehlermeldung, die den Grund für den Rückläufer ausweist. Diese Meldung besteht unter anderem aus einem SMTP-Fehlercode und einer kurzen Beschreibung des Fehlers. Hier wird auch die E-Mailadresse angeführt, an welche der Zustellungsvorgang gescheitert ist. Dies ist aber nicht besonders aussagekräftig für die Auswertung. Oftmals sind Mailkonten so konfiguriert, dass sie eingehende Mails sofort an ein anderes Konto weiterleiten, beispielsweise um Post von vielen verschiedenen Mailaccounts an einer zentralen Stelle abholen zu können. Wenn nun das Zielkonto der Weiterleitung nicht mehr existiert, kann der dortige Mailer Daemon nicht die ursprüngliche Zieladresse bemängeln, sondern nur die Zieladresse nach der Weiterleitung. Auch aus diesem Grunde wurde der Bounce-Key eingeführt, um ausgehende Mails unabhängig von deren letztendlicher Zieladresse identifizieren zu können.

Auf die Fehlerbeschreibung folgt für gewöhnlich ein Zitat der ursprünglichen Mail. Oftmals wird die komplette Mail zitiert, manchmal nur eine bestimmte Anzahl Bytes vom Beginn der Mail. Zumindest sollten aber die ursprünglichen Header-Angaben komplett aufgelistet werden, dort kann der vergebene Bounce-Schlüssel gefunden werden, der den Rückläufer identifiziert.

Zu erwähnen ist, dass zwar die meisten Rückläufer diesem Format folgen, es aber einige Mailer Daemons gibt, die andere Formate verwenden. Ebenso ist der SMTP-Statuscode der Mail nicht besonders aussagekräftig und zudem schwierig aus den übrigen Informationen heraus zu filtern. Für die Auswertung wurde daher der Weg gewählt, die Textbeschreibung des aufgetretenen Fehlers auf bekannte typische Fehlermeldungen[47] hin zu untersuchen.

```
From: Mail Delivery System <Mailer-Daemon@otherhost.com>
To: bounce-test@hostname.de
Subject: Mail delivery failed: returning message to sender
Date: Wed, 05 Dec 2007 12:25:17 +0100

This message was created automatically by mail delivery
software.

A message that you sent could not be delivered to one or more
of its recipients. This is a permanent error. The following
address(es) failed:

  name@otherhost.com
    SMTP error from remote mailer after RCPT TO:
    <name@otherhost.com>: host mail.otherhost.com
    [1.2.3.4]: 550 #5.1.0 Address rejected name@otherhost.com

------ This is a copy of the message, including all the
headers. ------

Return-path: <bounce-test@hostname.de>
From: wwEdit CMS <no-reply@wwedit.de>
To: name@otherhost.com
Subject: Newsletter
X-wwNews-Bounce-Key: 8fa28e626534843636b16e1c060604ac
Content-Type: text/plain; charset="iso-8859-15"
Content-Transfer-Encoding: 8bit
X-Mailer: wwEdit CMS Version 3.1.1
Date: Wed, 05 Dec 2007 12:25:16 +0100

[...]
```

Abbildung 21: Beispiel einer Rückläufer-Mail

47 Die auftretenden Fehlermeldungen sowie die übliche Syntax der Rückläuferantworten wurden aus mehreren hundert echten Rückläufern eines Versands vor der Entwicklung der Rückläuferverwaltung bestimmt. Auf diese Weise ist sichergestellt, dass ein breites Spektrum an Mailservern angesprochen wurde, die verschiedene Formate für deren Rückläufer verwenden.

Für die Auswertung sind also nur die Fehlermeldung und der Bounce-Schlüssel von Bedeutung. Die betreffenden Zeilen in der Beispielabbildung 21 wurden farblich gekennzeichnet.

Aus dieser Beschreibung erschließt sich, wie ein Rückläufer am schnellsten verarbeitet werden kann. Der Header-Block wird komplett übergangen, erst ab der ersten Leerzeile beginnt die Suche nach Fehlertexten. Außerdem wird nach dem Auffinden des Bounce-Schlüssels das Parsing abgebrochen, da ab dieser Stelle alle nötigen Informationen bekannt sind.

Die Fehlermeldungen werden bei der Auswertung gewichtet nach Schweregrad. Diese Gewichtung unterlag zum großen Teil der Subjektivität des Autors, erlaubt aber eine Abstufung zwischen „harten“ Fehlern wie nicht existenten Zielhosts oder Mails an Benutzer, die auf dem Zielsystem nicht bekannt sind, und „weichen“, möglicherweise nur temporären Fehlern wie einer vollen Mailbox oder einer Abweisung, die aus fehlerhafter Konfiguration auf dem sendenden oder auch empfangenden Server resultieren können. Ebenso wird die temporäre Abweisung von Mails durch Greylisting berücksichtigt (vgl. Fußnote 20).

Die weitere Verfahrensweise hängt davon ab, ob ein Bounce-Schlüssel gefunden wurde (andernfalls endet die Auswertung an dieser Stelle, da die E-Mail nicht zugeordnet werden kann) und ob und welche Fehlermeldung gefunden wurde. Zunächst wird der Datensatz, auf den sich der Rückläufer bezieht, aus der Datenbank geladen. Es kann sich hier entweder um eine Opt-In-Anfrage handeln oder um einen Newsletterempfänger. Im Falle eines Opt-Ins wird der entsprechende Datensatz einfach aus der Opt-Tabelle und ebenso das damit verknüpfte Abonnement aus der Subscription-Tabelle entfernt. Zu diesem Zweck kann auf die Funktion `removeOpt()` aus der `nl_functions.php` (Kapitel 6.1) zurückgegriffen werden.

Handelt es sich hingegen um einen Rückläufer eines Newsletterversands, so wird der Bounce-Zähler des jeweiligen Empfängerdatensatzes um den Schweregrad der gefundenen Fehlermeldung inkrementiert und bei Überschreiten eines Grenzwerts von 3 die Adresse zusätzlich auf die Sperrliste gesetzt.

Wenn hingegen keine Fehlermeldung identifiziert werden konnte, so wird die E-Mail in einer Verzeichnisstruktur unter home/bounce/ abgelegt. Dies ermöglicht es dem Entwickler die dort gesicherten Mails nach Fehlermeldungen zu durchsuchen, um den Bouncehandler auf aktuellem

Stand zu halten. In aller Regel lassen sich dort weitere leicht abweichende Formulierungen zu den bereits identifizierten Meldungen finden, die dann an entsprechender Stelle im Skript nachgetragen werden können.

7.2 Empfängerverwaltung

Mit den bisher beschriebenen Eingabemasken ist es zwar möglich, einen Newsletter zu abonnieren und wieder zu kündigen. Dies ermöglicht es einem Betreiber allerdings noch nicht, die Verteilerliste einzusehen, eigene Listen in das System einzupflegen oder den vorhandenen Adressbestand zur anderweitigen Verwendung aus dem System zu exportieren. Die Voraussetzungen hierfür wurden zwar bereits durch die in Kapitel 6.1 beschriebenen Verwaltungsfunktionen erfüllt, es fehlt allerdings noch eine Schnittstelle für Redakteure, um diese Funktionen auch nutzen zu können.

Die Anzeige der Empfänger eines bestimmten Newsletters erfolgt über den Reiter „Empfänger“, der durch das Skript `page_nl_recipients.php`[48] verwaltet wird. Dieses Skript benötigt als Parameter eine Seiten-ID. Ist diese auf 1 gesetzt, so wird nicht davon ausgegangen, dass es sich hierbei um einen Newsletter handelt (die 1 ist der Wurzelknoten des CMS-Seitenbaums), sondern es wird eine allgemeine Empfängerverwaltung angezeigt. Nur in diesem Fall wird auch die systemweite Sperrliste angezeigt. In allen anderen Fällen generieren sich die angezeigten Empfängerlisten aus den Verteilern, an die die angegebene Seite als Newsletter versendet werden soll.

Beim Aufruf des Empfängerreiters erhält der Redakteur eine Ansicht wie in Abbildung 22, die mit Hilfe von Aufrufen der Funktion `getNLRecipients()` erstellt wird. Unter der Überschrift „Empfängerlisten“ können hier die Abonnenten mit Mailadressen und Namen, sowie gegebenenfalls Unternehmen/Organisation und Position eingesehen werden. Es ist ebenfalls ersichtlich, welche Verteiler abonniert wurden und durch farbliche Kennzeichnung auch, ob der abonnierte Newsletter in Text- oder HTML-Form gewünscht ist.

48 voller Pfad: `htdocs/management/page_nl_recipients.php`, das dazugehörige Template ist `htdocs/management/templates/backend/page_nl_recipients.tpl.html`

Durch Klick auf den Verteilernamen kann das Format für dieses Abonnement auch geändert werden. Ein Abonnement kann auch vom Redakteur durch Klick auf das rote X hinter einem Verteilernamen beendet werden, ohne dass dazu ein Opt-Out-Prozess erfolgen muss. Ein Redakteur kann zudem über die unter dem Titel „Aktionen“ gegebenen Links Adressen auf die Sperrliste setzen, ganze Empfängerdatensätze löschen sowie Bestätigungsanfragen stattgeben oder sie entfernen. All diese Funktionen bestehen im Wesentlichen aus Aufrufen der in Kapitel 6.1 beschrieben Verwaltungsfunktionen.

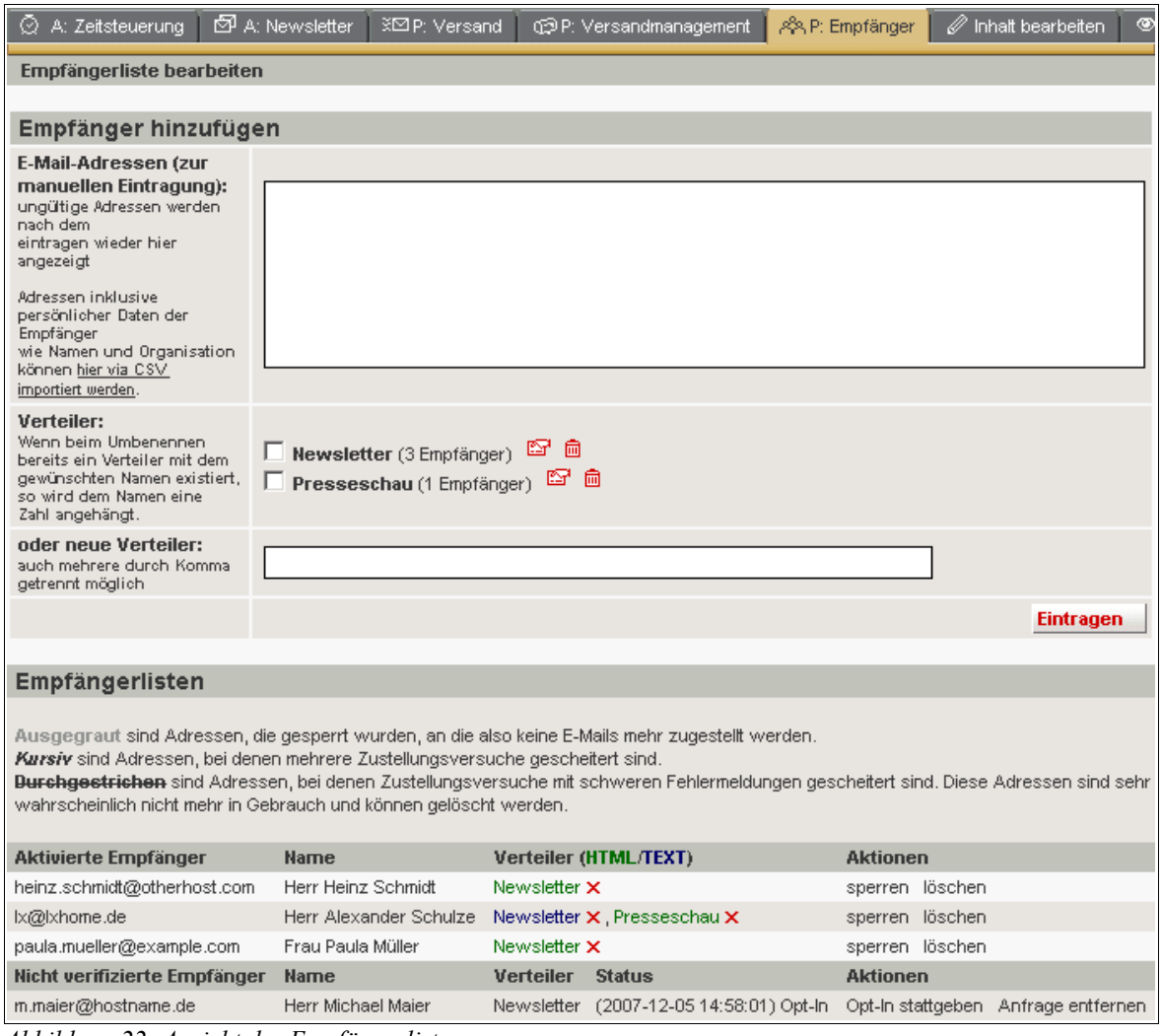


Abbildung 22: Ansicht der Empfängerliste

Über der Anzeige der Empfängerliste befindet sich ein einfaches Importformular. Darüber können Adressen ohne Angabe zusätzlicher persönlicher Daten importiert werden. Die Adressen werden dazu einfach in das Textfeld kopiert. Anschließend kann über die darunter aufgeführten

Checkboxen bestimmt werden, in welchen Verteiler die importierten Adressen eingetragen werden sollen. Alternativ oder ergänzend kann auch ein neuer Verteiler über das Freitextfeld darunter eingerichtet werden. Die Auswertung erfolgt über die Funktion `importAddrs()`.

Es können hier auch neue Verteiler eingerichtet werden, indem das Verteiler-Freitextfeld ausgefüllt wird, ohne Adressen im Importfeld einzutragen. Angelegte Verteiler erben dabei die Rechteeinstellungen der Newsletterseite oder sind für alle Benutzer zugänglich, wenn es sich um die allgemeine Empfängerverwaltung handelt (Seiten-ID 1).

Neben der Verteilerauswahl befinden sich bei jedem Verteiler zwei Icons. Mit ihnen ist es möglich, einen Verteiler umzubenennen oder mitsamt seiner Abonnements ganz zu löschen. Vor sämtlichen Aktionen, die Datensätze löschen (also Kündigungen von Abonnements, das Entfernen von Opt-Anfragen sowie Löschen von Empfängern oder Verteilern) erfolgt eine JavaScript-Sicherheitsabfrage. Es wurde bewusst an dieser Stelle keine Rücksicht auf deaktiviertes JavaScript genommen, da der Redaktionsbereich des CMS ohne JavaScript nicht zu bedienen ist und daher davon ausgegangen werden kann, dass diese Eingabemaske ohne aktiviertes JavaScript kaum zu erreichen ist.

7.3 Import mit persönlichen Daten

In der Beschreibung des Adressimport-Feldes in Abbildung 22 wird auf die Möglichkeit eines erweiterten Imports verwiesen, über den auch persönliche Daten wie Namen und Geschlecht der Empfänger in das System geladen werden können. Dies geschieht über die Eingabemaske in Abbildung 23, hinter der sich das Skript `csv_import.php`[49] verbirgt.

Als standardisiertes Importformat wurde CSV gewählt, da dieses Format auf allen Systemen verfügbar ist und keine spezifischen Formatierungen enthält. Eine CSV-Datei kann beispielsweise sehr einfach mit Hilfe von Tabellenkalkulationssoftware wie Microsoft Excel oder auch OpenOffice.org-Calc erstellt werden. Die Formatbeschreibung der Datei ist im Formular enthalten. Der CSV-Import bedient sich ebenfalls der beschriebenen Verwaltungsfunktionen.

49 voller Pfad: `htdocs/management/wwact/wwnews/csv_import.php`, das dazugehörige Template ist `htdocs/management/templates/backend/wwnews_nl_import.tpl.html`

Abbildung 23: Eingabemaske für den CSV-Import

7.4 Export einer Abonnentenliste

Analog dazu ist es möglich, die Abonnenten eines Verteilers in eine CSV-Datei exportieren zu lassen. Hierfür wurde das Skript `csv_export.php`[50] entwickelt.

Diese Eingabemaske (Abbildung 24) gestaltet sich allerdings etwas einfacher als für den Import. Eine umfängliche Formatbeschreibung entfällt. Stattdessen besteht lediglich eine Auswahl für einen zu exportierenden Verteiler sowie eine Auswahl, ob der Export für Microsoft Excel gedacht ist oder nicht. Es können nur Abonnenten eines einzelnen Verteilers exportiert werden. Dies soll verhindern, dass ein unbedarfter Redakteur mehrere Verteiler in eine Liste exportiert, einige Adressen ergänzt und dann wieder importiert, in der Erwartung, dass die Abonnements der bereits

50 voller Pfad: `htdocs/management/wwact/wwnews/csv_export.php`, das dazugehörige Template ist `htdocs/management/templates/backend/wwnews_nl_export.tpl.html`

bestehenden Adressen wie gehabt erhalten bleiben. Stattdessen würden aber *alle* Adressen der CSV-Liste auf *alle* angegebenen Verteiler abonniert werden, da im Export keine Information über die abonnierten Verteiler vorhanden ist.

Die Auswahlcheckbox für den Excel-Export veranlasst nur eine kleine Änderung: Statt eines Semikolons wird als Trennzeichen ein Komma gewählt. Excel öffnet CSV-Dateien zwar auch mit anderen Trennzeichen, kann aber nur bei Kommata die korrekte Spaltenzuordnung herstellen. Aus Gründen, die dem Autor nicht bekannt sind, wird beim Speichern als CSV aus der verwendeten Excel-Version dennoch ein Semikolon als Trennzeichen verwendet, sodass Excel gespeicherte CSV-Dateien nur noch über einen Import, aber nicht mehr durch einfachen Doppelklick korrekt öffnen kann.

Abbildung 24: Eingabemaske für den CSV-Export

Nach Auswahl eines Verteilers kann über den Button „Exportieren“ die Generierung der CSV-Datei veranlasst werden. Diese wird direkt als Download an den Browser gesendet, der dem Benutzer daraufhin die Auswahl zwischen Öffnen mit einem gewählten Programm oder dem Abspeichern auf der Festplatte bieten sollte.

7.5 Versandmanagement

Der Reiter „Versandmanagement“ (`page_nl_management.php`[51]), der in Abbildung 25 dargestellt ist, gibt einen Überblick über die Anzahl und den Status versendeter Newsletter. Wird eine Seiten-ID als Parameter übergeben, bezieht sich die Darstellung analog zum Empfänger-Reiter nur auf die aktuelle Seite, ansonsten wird eine CMS-weite Statistik angezeigt.

51 voller Pfad: `htdocs/management/page_nl_management.php`, das dazugehörige Template ist `htdocs/management/templates/backend/page_nl_management.tpl.html`

Angezeigt werden alle angelegten Versandaufträge, wie sie in der Job-Tabelle vorgefunden werden. Entsprechend des Status wird jeder Tabellenzeile ein farbiges Quadrat vorangestellt, dessen Bedeutung der Legende zu Beginn der Übersicht entnommen werden kann. In der Legende befindet sich zudem neben jedem Status ein grafischer Button, der einen Trichter darstellt. Dieser dient als Filter, um die jeweiligen Einträge aus der dargestellten Liste zu entfernen. Der Filter-Button neben „erfolgreich beendet“ in Abbildung 25 würde also dafür sorgen, dass alle grün gekennzeichneten Zeilen aus der Anzeige ausgeblendet werden und nur noch die orange und rot markierten Datensätze dargestellt werden.

Abbildung 25: Der Reiter "Versandmanagement"

Ist der Status eines Versandvorgangs „manuell abgebrochen“ oder „fehlerhaft abgebrochen“, so kann über einen Link „fortsetzen“ in der Aktionsspalte der Versand wieder aufgenommen werden. Hierfür wird eine Newsletterklasse initialisiert, doch anstatt einen Newsletter neu zusammenstellen zu lassen, wird lediglich die `send()`-Methode mit der ID des abgebrochenen Auftrags als Parameter aufgerufen.

8 Ausblick

Im letzten Kapitel dieser Diplomarbeit soll ein Ausblick gegeben werden, wie sich die entwickelte Software in das System integriert, wo Funktionen aus *ww***News** auch an anderer Stelle Verwendung finden können oder bereits verwendet werden und welche Verbesserungen noch implementiert werden können. Die folgend beschriebenen Funktionen wurden, sofern sie bereits im Produktionsbetrieb sind, allerdings zumeist nicht vom Autor umgesetzt, sondern nur während der Entwicklung unterstützend begleitet.

8.1 Nachhaltigkeit

Bei der Programmierung der beschriebenen Routinen wurde bedacht, dass diese möglichst nachhaltig ausgelegt sind, also entweder für ähnlich geartete Funktionen verwendet oder leicht an erweiterte Anforderungen angepasst werden können. So ist durch die Entwicklung des Double-Opt-In-Mechanismus eine Funktionalität entstanden, die mittlerweile auch an anderen Stellen im CMS zum Einsatz kommt. Bestätigungsmails können nicht nur für Newsletterabonnements versendet werden, sondern auch zur Überprüfung von Daten bei der Erstellung von Benutzerkonten, beim Unterschreiben auf einer digitalen Unterschriftenliste oder zum Bestätigen eines E-Card-Versands.

Die Versandfunktion selbst kommt ebenfalls noch an anderer Stelle zum Einsatz, im Speziellen beim Versand von Grußkarten und beim Weiterempfehlen einer Seite. In beiden Fällen wird aus einer bestehenden Webseite eine Newsletteransicht generiert, die anschließend von einem Besucher der Seite an einen Freund oder Bekannten gesendet werden kann. Ebenso ist der erstellte Textfilter, der bestehendes HTML in eine einfache Textdarstellungsform konvertiert, weiter verwendet worden, um Informationsmails für neue Beiträge in Foren, Blogs oder Gästebüchern zu formatieren.

8.2 Weiterentwicklung

Trotz Erfüllung der zu Beginn gestellten Anforderungen lassen sich an *ww***News** noch Verbesserungen und Ergänzungen vornehmen. Bereits in Kapitel 6.6 angesprochen wurde die Ergänzung um weitere Platzhalter für die Personalisierung, die an beliebiger Stelle im Newsletterinhalt platziert werden können. Ebenfalls bis dato unberücksichtigt ist ein

sprachunabhängiger Versand des Newsletters. Dabei muss auch die Internationalisierung überdacht werden: Während im Deutschen eine Anredeformel zum Beispiel auf „Dr. phil. Thomas Müller“ lautet, wird im englischsprachigen Raum der akademische Grad beispielsweise oftmals an den Namen angehängt („Thomas Miller, Ph. D.“). Dies muss bei der Anredeformel sowie bei der Zusammenstellung der Empfänger-Mailadresse berücksichtigt werden.

Da Verteilerlisten an unterschiedliche Zielgruppen adressiert sein können, zum Beispiel an Webseitenbesucher in einem Verteiler und Pressekontakte in einem anderen, ist perspektivisch auch angedacht, das Newsletterdesign nicht nur domaingruppen-, sondern verteilerspezifisch gestalten zu können. Hierbei handelt es sich allerdings um eine umfangreiche Änderung, da auch in der Datenbank Anpassungen vorgenommen werden müssen. Eine Möglichkeit bestünde darin, die Styleangaben, die im Newsletter eingebunden werden, mit in der Verteilertabelle aufzuführen. Wahrscheinlicher und flexibler ist allerdings die Erstellung weiterer Datenbanktabellen für Newsletterdesigns, die dann mit einem Verteiler verknüpft, oder aber individuell beim Versand ausgewählt werden können.

Für die Verteiler selbst ist in diesem Zuge ebenfalls die Entwicklung einer umfangreicheren Verwaltung geplant. Mit den in dieser Arbeit beschriebenen Funktionen können Verteiler nur angelegt, gelöscht und umbenannt werden. Verfügungsrechte beispielsweise werden aber fest vergeben und sind nachträglich nicht ohne direkten Datenbankzugriff änderbar. Es sollte eine Eingabemaske geschaffen werden, die besseren Zugriff auf die Einstellungen der Verteiler gibt. Hier kann ebenfalls die eben angesprochene Designauswahl untergebracht werden.

Für den Massenversand an drei- und mehrstellige Empfängerzahlen ist zwar aus Gründen des entstehenden Mailtraffics davon abzuraten, aber insbesondere für den Versand an einen kleinen Benutzerkreis könnte auch der Wunsch aufkommen, den Newslettern nicht nur Grafiken, sondern auch Dateien wie PDFs oder andere Dokumente anzuhängen. Hierfür müssten die Abläufe in der Methode `_attachImages()` der Newsletterklasse entsprechend verallgemeinert oder ausgelagert werden. Außerdem wird eine Eingabemaske benötigt, um Anhänge im CMS hochladen und mit einem Newsletter verknüpfen zu können. Dazu würde sich ein weiterer Attributreiter für Newsletterseiten anbieten.

Literaturverzeichnis

[WIKIPEDIA]

Wikipedia-Artikel: „Newsletter“
http://de.wikipedia.org/wiki/Newsletter
20. Oktober 2007

Wikipedia-Artikel: „ISO 8859-1“
http://de.wikipedia.org/wiki/ISO_8859-1
12. November 2007

Wikipedia-Artikel: „MX Resource Record“
http://de.wikipedia.org/wiki/MX_Resource_Record
5. Dezember 2007

Wikipedia-Artikel: „AJAX“
http://de.wikipedia.org/wiki/Ajax_(Programmierung)
6. Dezember 2007

[RSS]

RSS 2.0 Spezifikation
http://www.rssboard.org/rss-specification
22. Oktober 2007

[IETF]

RFC 2045: „Multipurpose Internet Mail Extensions“
http://www.ietf.org/rfc/rfc2045.txt
14. November 2007

RFC 2822: „Internet Message Format“
http://www.ietf.org/rfc/rfc2822.txt
12. November 2007

RFC 4287: „The Atom Syndication Format“
http://www.ietf.org/rfc/rfc4287.txt
22. Oktober 2007

[WEGEWERK]

wegewerk GmbH
http://www.wegewerk.com
20.Oktober 2007

Produktbeschreibung wwEdit CMS
http://www.wegewerk.com/produkte/wweditcms/
24. Oktober 2007

[UWG]

Gesetz gegen den unlauteren Wettbewerb
Bundesministerium der Justiz
Ausfertigungsdatum: 3. Juli 2004
http://www.gesetze-im-internet.de/uwg_2004/__7.html
13. Oktober 2007

[DDV]

Deutscher Direktmarketing Verband e.V.
http://www.ddv.de
28. Oktober 2007

Ehrenkodex eMail-Marketing des DDV (Stand: 24. Oktober 2003)
http://www.ddv.de/downloads/Service/Ehrenkodex_eMail-Marketing.pdf
28. Oktober 2007

[GMX]

GMX: FAQ-Eintrag „Spamschutz“
http://faq.gmx.net/optionen/email/antispam/4.html
12. November 2007

[DEBIAN]

Debian GNU/Linux
http://www.debian.org
24. Oktober 2007

[APACHE]

The Apache HTTP Server Project
http://httpd.apache.org
24. Oktober 2007

Apache Module mod_rewrite
http://httpd.apache.org/docs/2.0/mod/mod_rewrite.html
6. November 2007

[PHP]

PHP: Hypertext Preprocessor
http://www.php.net
24. Oktober 2007

Smarty Template Engine
http://smarty.php.net
24. Oktober 2007

PHP Extension and Application Repository
http://pear.php.net
24. Oktober 2007

PHP Dokumentation
http://de3.php.net/manual/de/index.php
25. September 2007

[MYSQL]

MySQL AB
http://www.mysql.com
24. Oktober 2007

[OPENLDAP]

OpenLDAP
http://www.openldap.org
24. Oktober 2007

[GNU]

GNU General Public License
http://www.gnu.de/documents/gpl-2.0.de.html
26. Oktober 2007

[JQUERY]

jQuery JavaScript Library
http://www.jquery.com
26. Oktober 2007

[MSDN]

Microsoft Developer Network
Artikel: „Word 2007 HTML and CSS Rendering Capabilities in Outlook 2007“
http://msdn2.microsoft.com/en-us/library/aa338201.aspx
4. Dezember 2007

[EXIM]

The Exim Home Page
http://www.exim.org/
5. Dezember 2007

Abbildungsverzeichnis

Tabellenverzeichnis

Beilagen

Der Diplomarbeit liegt eine CD bei. Darauf befinden sich neben dieser Arbeit als PDF auch die Dateien, auf die der Autor im Lauf der Implementationsbeschreibung Bezug genommen hat. Aus Übersichtlichkeitsgründen und zur Abgrenzung der eigenen Arbeit sind nur die Dateien enthalten, die vom Autor erarbeitet wurden und direkten Bezug zum Newslettertool haben.

Folgende Verzeichnisstruktur liegt auf der CD vor: