



FACHHOCHSCHULE BOCHUM
UNIVERSITY OF APPLIED SCIENCES



Konzept für eine Diplomarbeit

Implementierung einer
Web-basierten
Administrationsapplikation für
Linux-Hostingserver mittels
Ruby on Rails

WILLEM VAN KERKHOF, wvk@wvk-astro.com (PGP: 0xC31CB45F)

Bochum, 20. September 2007

Inhaltsverzeichnis

1	Ausgangssituation	1
2	Zielsetzung	3
3	Einsatz und Systemumgebung	6
4	Allgemeine/formale Anforderungen an die Software	7
5	Funktionale Anforderungen	8
6	Testszenarien	13

1 Ausgangssituation

1.1 Ist-Zustand

Immer mehr fortgeschrittene Privatanwender entscheiden sich zu einem eigenen Webserver um ihre Webpräsenz und die anderer zu Hosten. Viele versuchen sich als Hosting-Reseller, um sich ein Taschengeld mit der Bereitstellung von Webspeicher, e-Mail und anderen Serverdiensten an andere zu verdienen. Zu Beginn diese Arbeit steht ein funktionsfertig eingerichteter virtueller Hostingserver (“VServer”, “Rootserver”) eines großen Hostingunternehmens, welches fertig vorinstallierte Linuxserver mitsamt der weit verbreiteten webbasierten Administrationsoberfläche CONFIXX anbietet. Leider ist CONFIXX eine lizenzbehaftete Software (d.h. nicht frei und offen), wo, auch wenn diese fast vollständig in PHP geschrieben ist, eigene Änderungen und Erweiterungen Urheberrechtliche Konsequenzen haben könnten. Außerdem erlaubt CONFIXX nur eine bestimmte Konstellation von Serverprogrammen, welche in Tabelle 1.2 aufgeführt werden. Alternative Programme wie ISPConfig oder Plesk bieten eine umfangreichere Sammlung von Softwarekonfigurationen, im ersteren Fall jedoch ebenfalls nicht die vom Autor gewünschte und im zweiten Fall unter einer noch restriktiveren Lizenz wie Confixx.

Keines der genannten Serververwaltungsprogramme unterstützt die webbasierte Verwaltung von Ruby on Rails-Applikationen. Dies würde die Einrichtung separater VirtualHosts beinhalten, aber auch eine bequeme Schnittstelle zu Rake und Generate (z.B. zur Datenankpflege und für Scaffolding).

1.2 Unzulänglichkeiten der bestehenden Systemkonfiguration

Die in Tabelle 1.2 aufgeführte Softwarekonfiguration ist zwar für die meisten Anforderungen ausreichend, aber sobald z.B. echte Mailinglistenfunktionalität gefragt ist, treten erste Schwierigkeiten auf. Confixx bietet zwar eine

Server/Aufgabe	Software/Programm	Serverkonfiguration
Webserver	Apache HTTP Server	Datei: /etc/apache2/...
FTP	ProFTP	Datei: /etc/proftpd.conf
SMTP	Sendmail, Exim	Datei: /etc/aliases, /etc/exim4/conf.d/...
POP3/IMAP	Courier, Qmail	Datei: /etc/
SQL-Datenbank	MySQL, PostgreSQL	

Tabelle 1: Die von Confixx unterstützte Softwarekonfiguration

Server/Aufgabe	Software/Programm	Serverkonfiguration
Webserver	Apache HTTP Server	Datei: /etc/apache2/...
SFTP	OpenSSH (rSSH)	Datei: /etc/sshd.conf
SMTP	Postfix	Datenbank/Dateien
Antivir./-Spam	AMaViS/Spamassassin	Dateien: /etc/...
POP3/IMAP	Dovecot	Datenbank/Dateien
Mailinglisten	Mailman	Datenbank/Dateien
SQL-Datenbank	MySQL	

Tabelle 2: Die gewünschte Softwarekonfiguration

einfache Art, E-Mails an mehrere Empfänger zu verschicken, aber Software wie Mailman oder QMailadmin bieten außerdem umfangreiche Auskunft- und Administrationsmöglichkeiten via E-Mail. Diese mit Confixx zu verwalten ist aber nicht möglich, außerdem ist die Konfiguration des standardmäßig unter Debian verwendeten Mail-Daemons, Exim, zur Zusammenarbeit mit Mailman (und SmapAssassin zwecks Spammail-Bekämpfung und AmA-Vis o.ä. als Virenschanner), relativ aufwändig und fehleranfällig. Postfix ist in dieser Hinsicht wesentlich einfacher zu administrieren. Außerdem bietet Postfix die in dieser Arbeit als sehr wichtig betrachtete Möglichkeit, virtuelle Benutzeraccounts, Domains etc. direkt aus einer MySQL-Datenbank zu lesen. Dies macht das aufwändige Erstellen von Dateien wie `/etc/aliases` oder `/etc/domains` sowie `~/forward` faktisch überflüssig. In Confixx mussten diese Dateien alle paar Minuten von einem via Cronjob ausgeführten, compilierten Programm aus den in der Datenbank gespeicherten Daten neu erstellt werden. Ein Prozess, der einerseits zum verzögerten in Kraft treten der Änderungen führte, und andererseits manuelle Änderungen an Konfigurationsdateien alsbald rückgängig machte – eine Tatsache, die einem Debian-Administrator zuwiderläuft, da die für Softwareentwickler geltende Debian-Policy das automatische Überschreiben manuell vorgenommener Software-/Systemkonfigurationen verbietet.

1.3 Auslagerung der Systemkonfiguration in eine Datenbank

Wenn also Postfix die Möglichkeit bietet, die dynamische Konfiguration (d.i. die Konfiguration, die sich während des Betriebes häufig ändert) direkt aus einer Datenbank zu übernehmen, dann müssten das auch andere Programme können. In der Tat bieten sowohl Courier als auch Dovecot die Möglichkeit, die gesamte Benutzerverwaltung direkt über Datenbanktabellen zu Handhaben. Da aber Courier eine wesentlich umfangreichere und dadurch komplexere Software ist, wird im Rahmen dieser Arbeit bevorzugt Dovecot zum Einsatz kommen. Prinzipiell jedoch sollten beide Programme gleichermaßen verwendet werden können.

Nicht nur die Mailserver, sogar das Linux-System selbst unterstützt die komplette Benutzerverwaltung über MySQL: PAM macht es möglich, mit Hilfe von `pam_mysql`. PAM steht für "Pluggable Authentication Modules" und erlaubt es, die System-Zugriffskontrolle sehr fein abgestimmt über austauschbare und in Reihe schaltbare Module zu implementieren. `pam_mysql` funktioniert im Wesentlichen gleich wie das wichtigste Authentifizierungsmodul `pam_unix`, welches auf Grundlage der klassischerweise verwendeten Dateien `/etc/passwd` sowie `/etc/shadow` arbeitet, nur dass `pam_mysql` die angegebenen MySQL-Tabellen verwendet. Unter gewissen Voraussetzungen sind beide Methoden gleich sicher.

1.4 Nutzen

Sollte es also gelingen, weitestgehend alle zum Betrieb eines Mehrbenutzer-Hostingsystems nötigen Programmkonfigurationen ausschließlich über Datenbanktabellen zu gestalten, so würde sich ein Server-Administrationstool auf eine reine Datenbankapplikation beschränken und das fehleranfällige Generieren von Konfigurationsdateien fiele weg. Alle durch einen Benutzer oder Administrator durchgeführten Änderungen wären sofort im System sichtbar. Die Erstellung von Backups würde wesentlich vereinfacht, und die Wiederherstellung einer vorherigen Konfiguration nach einem versehentlichen großen Fehler wäre mit Hilfe der Transaktions-Logdateien des MySQL-Servers ohne großen Aufwand möglich. Die Lese- und Änderungs-Zugriffsverwaltung der Daten ist mittels einer Datenbank wesentlich einfacher und feiner abgestuft möglich, außerdem unterstützt MySQL Views, die für einen Benutzer nicht benötigte Information in den für ihn sichtbaren Tabellen ausblenden.

2 Zielsetzung

Ziel dieser Arbeit wird es sein, eine webbasierte Anwendung zu erstellen und zu dokumentieren, die mindestens die in Tabelle ?? aufgeführten Programme zu verwalten und administrieren in der Lage ist, sowie weiterhin die im zu erstellenden Pflichtenheft sowie die unten aufgeführten Anforderungen

erfüllt. Die Software soll unter der Verwendung moderner SW-Entwicklungs- und Programmier-Paradigmen erstellt werden und ausschließlich offene Standards verwenden.

Mit Einreichung des Berichtes in Form einer Diplomarbeit verfolgt der Autor das Ziel der Erlangung des akademischen Grades des Diplom-Ingenieurs (FH). Die Sprache der Diplomarbeit wird Deutsch sein, die primäre Sprache der Software/Dokumentation hingegen Englisch.

Der zeitliche Rahmen für die Umsetzung der genannten Ziele beträgt ab Fertigstellung des Pflichtenheftes vier Monate. Spätester Abschlussstermin soll der 15. Februar 2008 sein.

2.1 Bestandteile der Arbeit

- Erstellung eines detaillierten Pflichtenheftes
- Implementierung der Verwaltungsapplikation (inklusive Installationsroutinen und/oder -Anleitung), bestehend aus dem Kern-Modul zur Benutzerverwaltung sowie einigen Untermodulen (Plugins)
- Erstellen der API-Dokumentation
- Einrichten des Datenbankschemas für die Systemumgebung
- Test der selbst erstellten Software in der gegebenen Systemumgebung

2.2 Nicht-Bestandteile der Arbeit

- Installation und Einrichtung des Betriebssystems und der Serverumgebung mit Ausnahme des Datenbankschemas wird vorausgesetzt (ggf. kann auf HowTos und Dokumentation verwiesen werden)
- Gewährleistung der Systemsicherheit, Serververwaltung und -Betrieb findet nur im durch die Testszenarien beschriebenen Umfang statt.
- Die Installation der Software auf anderen Plattformen als eine Debian GNU/Linux-Distribution ist im Rahmen dieser Arbeit nicht vorgesehen.
- Nicht alle unten beschriebenen Module zählen als Bestandteil der Arbeit. Sie werden zwar im Endprodukt vorhanden sein, ihre Entwicklung geschieht aber außerhalb der vorgesehenen Arbeitszeit.

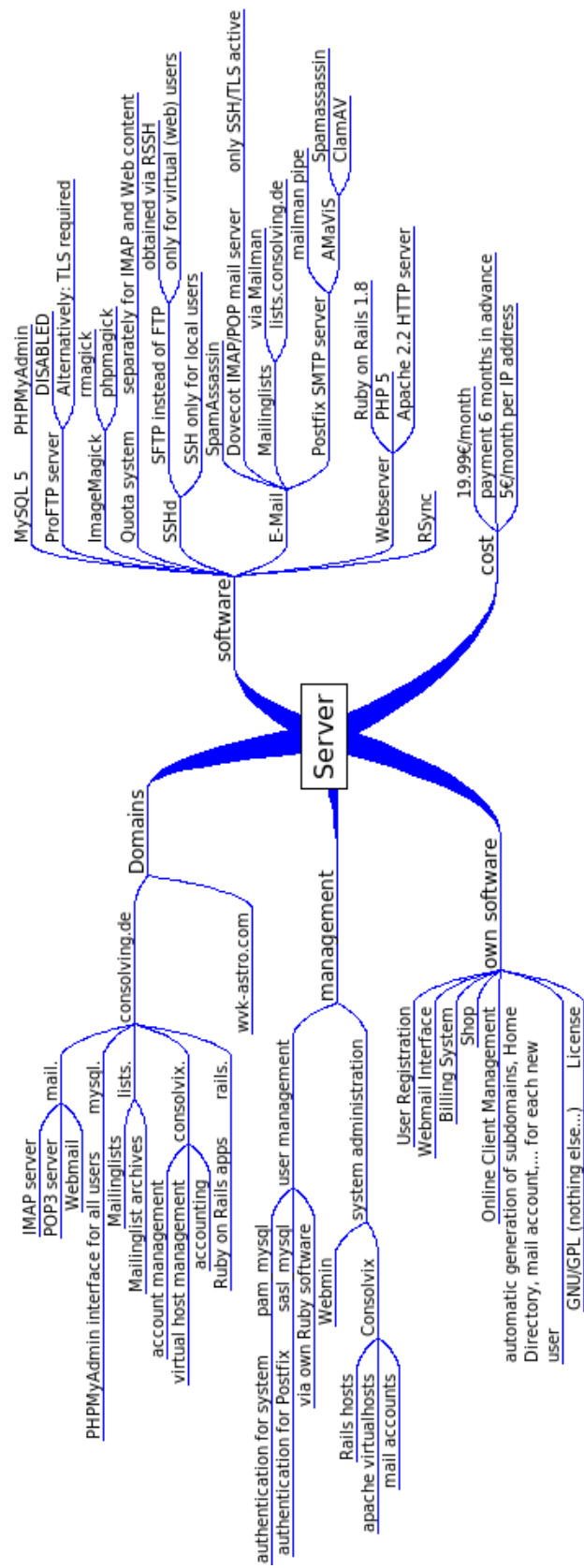


Abbildung 1: Überblick über das bestehende Server-Projekt, auf das die SW zugeschnitten werden soll.

2.3 Produktname

Die Software wird zu Test- und späteren Produktionszwecken für die in kleinem Umfang als Hostingunternehmen tätige fiktive Firma Consolving Network Solutions unter der Domain `consolving.de` betrieben werden. Der Slogan des Unternehmens lautet *“Consolving – WIR sind Internett.”*. Da die Gallier in *“Asterix und Obelix”* ebenfalls nette Kameraden waren und immer eine Lösung für auswegslose Situationen gefunden haben, soll die Software den Namen *“Consolvix”* tragen ;-).

2.4 Lizenz

CONSOLVIX soll eine Quelloffene, freie Alternative zu bestehenden, offenen und proprietären Programmen bilden. Mit Fertigstellung der Arbeit soll der gesamte Quelltext online veröffentlicht und unter der GNU General Public License weitergereicht und genutzt werden.

3 Einsatz und Systemumgebung

3.1 Zielsystem

Die Applikation soll serverseitig auf oben genannte Softwarekonfiguration zugeschnitten sein. Ein späterer Ausbau zur Unterstützung anderer Serverprogramme soll möglich sein, ist aber im Rahmen dieser Arbeit nicht vorgesehen. Clientseitig soll jede WWW-fähige Plattform unterstützt werden, wobei das Hauptaugenmerk jedoch vorwiegend auf klassische PC-Systeme mit grafischem Browser gerichtet sein soll. Die Bildschirmdarstellung soll jedoch so flexibel gestaltet werden, dass auch mobile Endgeräte Nutzen davon tragen sollen, solange diese mit HTTPS-Verbindungen und JavaScript/AJAX umzugehen wissen.

3.2 Datenbank

Wie oben beschrieben sollen möglichst alle nicht-statischen Konfigurationsoptionen für die verwendete Serversoftware in einer MySQL-Datenbank abgelegt werden. Dovecot, Postfix, Mailman (?) sowie PAM-Mysql und NSS-Mysql verfügen allesamt über die Möglichkeit, die Datenbankabfragen zur Abfrage der Datenbanken genau nach Wunsch zusammen zu stellen. Sinnvollerweise bekommt jedoch jedes Programm eine eigene View zugewiesen, mit genau den Informationen die es für seine Aufgabe benötigt. Die in den Views erhaltenen Informationen können sowohl Teilmengen von Benutzer- und Gruppentabellen (z.B. für die Benutzer-Authentifizierung) als auch Schnittmengen aus verschiedenen Tabellen enthalten (z.B. im Falle von E-Mailkonten). Jedes Programm erhält einen eigenen Datenbankbenutzer mit restriktiven Zugriffsrechten.

4 Allgemeine/formale Anforderungen an die Software

4.1 Beständigkeit manueller Konfigurationen

Neben der Serververwaltung mittels der zu erstellenden Software soll weiterhin uneingeschränkt die klassische Art der Systemverwaltung zur Verfügung stehen. Dies bedeutet, dass die Software auf keinen Fall bestehende manuelle Serverkonfigurationen überschreiben darf. Dies wird dadurch vereinfacht, dass alle betroffenen Systemkomponenten ihre Konfiguration direkt aus der zentralen Konfigurationsdatenbank beziehen und somit keine Änderungen an Konfigurationsdateien berücksichtigt werden müssen. Einige Ausnahmen werden sich im Falle von Apache und Software wie Spamassassin nicht vermeiden lassen; hierfür muss noch keine konkrete Lösungsmöglichkeit ausgearbeitet werden.

4.2 Modularer Aufbau

Die Applikation soll aus einem zentralen Kern und optionalen Modulen oder "Plugins" bestehen. Aufgabe des Kerns wird im wesentlichen die Benutzer- und Rechteverwaltung sein, während jedem weiteren Aufgabengebiet wie z.B. Domain-/VHost-Verwaltung, E-Mailkonten-Verwaltung oder Abrechnungswesen eigene Module zugeordnet werden. Welche Aufgaben genau implementiert werden sollen und welche dabei Bestandteil der Arbeit sein werden, wird später beschrieben. Benutzerrechte sollen pro Modul, Benutzer, Aufgabe und Kontext vergeben werden können.

4.3 Verwendung moderner Paradigmen

Die Implementierung soll vollständig Objektorientiert unter Verwendung gängiger Design-Pattern erfolgen. Aufbauend auf das Rails-Framework wird das MVC-Pattern zum Tragen kommen, welches Verwaltungslogik weitestgehend von der Präsentationslogik trennt. Sollte es sich als sinnvoll herausstellen, soll das REST-Architekturmodell zum Einsatz kommen, um nicht nur eine rein Webbasierte Oberfläche zu ermöglichen, sondern ebenso eine beim Hostinganbieter lokal laufende Desktopanwendung zur Serververwaltung. Konkret könnte es sich hierbei um eine in C++ oder Ruby geschriebene Anwendung handeln, die auf die Plattformunabhängiges Qt-Bibliotheken zurückgreift. Die Machbarkeit einer solchen Implementierung wurde vom Autor bereits erprobt.

4.4 Dokumentation

Sämtliche Klassen und Methoden sollen ausreichend dokumentiert werden, damit ein mit der Materie vertrauter SW-Entwickler/Programmierer zu einem

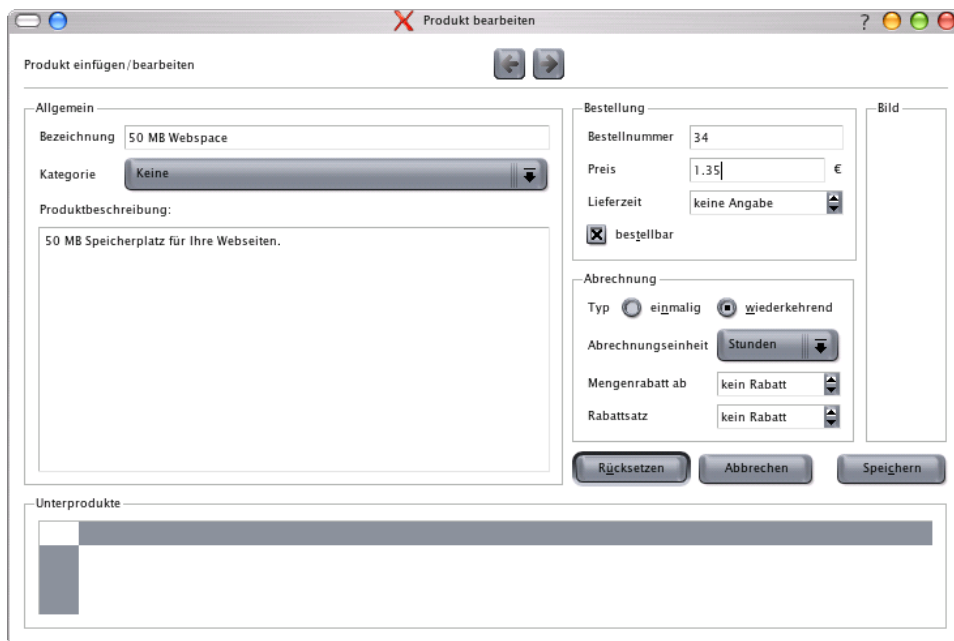


Abbildung 2: die verwendung von REST ermöglicht auch eine einfache Desktop-GUI-Implementierungen eines Clients z.B. mit Qt

späteren Zeitpunkt Veränderungen und Erweiterungen vornehmen kann. Besonderes Augenmerk soll auf die Modul-API entfallen, welche anderen Programmierern die Modulweise Erweiterung der Applikation ermöglicht werden soll.

5 Funktionale Anforderungen

5.1 Modul-Verwaltung

Dieser zentrale Modulbaustein ist Bestandteil der Arbeit und hat hohe Priorität.

- Module werden wie Plugins behandelt
- Eine eigene Oberfläche ermöglicht Modulweise Einstellungen
- Module können Konfigurationen in eine eigene Datenbanktabelle speichern
- Eine einheitliche Modul-API stellt eine Schnittstelle zur Rechteverwaltung etc. zur Verfügung.
- Jedes Modul stellt Konfigurationsoptionen über die API bereit, damit diese von einem Konfigurationsmodul einheitlich über eine Oberfläche

verwaltet werden können.

- ...

5.2 Benutzerverwaltung

Dieses zentrale Modul hat höchste Priorität und ist Bestandteil der Diplomarbeit.

- Unterscheidung zwischen Systembenutzern, Hosting-Benutzern und virtuellen Benutzern
- *Systembenutzer* sind z.B: root, postfix, www-data und andere Benutzer ohne Login-Shell, sowie einige wenige vollwertige normale Accounts wie vvk, die fest in den entsprechenden Systemdateien verdrahtet sind (/etc/passwd, /etc/shadow, /etc/group, ...).
- *Hosting-Benutzer* sind aus Systemsicht ebenfalls Systembenutzer, jedoch werden diese in
- Benutzer werden Resellern zugeordnet
- Reseller sind Hostingbenutzer
- Jeder Kunde bekommt 0–n Benutzer zugeordnet
- Jeder Benutzer kann sein eigenes Kennwort/persönliche Daten ändern, Reseller die ihrer Kunden, Kunden die aller ihnen direkt zugeordneter Benutzer
- Benutzer- und Resellerkonten können gesperrt werden
- Bei Sperrung eines Resellerkontos werden dessen Kundenkonten ebenfalls gesperrt
- Hostingbenutzer bekommen ein eigenes Home-Verzeichnis
- Kunden haben ein Benutzerkonto für das Online-Vertrags- und Rechnungswesen

5.3 Domainverwaltung/VirtualHosts

Dieses Modul ist Bestandteil der Diplomarbeit und hat hohe Priotität.

- Domains werden Hosting-Benutzern, nicht Kunden zugeordnet
- Domains können auf Unterverzeichnisse des Home-Verzeichnisses des Benutzers oder als Weiterleitung eingerichtet werden

- Domains können beliebig viele Unterdomains zugeordnet bekommen (⇒ Baumstruktur)
- Einer Domain/Subdomain entspricht ein Apache VirtualHost
- VirtualHosts werden (bis au Weiteres) in `/etc/apache2/sites-available/consolvix-virthosts` definiert
- Ein Parser für Apache2-Konfigurationsdateien liest jeweils die aktuellen Kongigdateien, nicht eventuell vorhandene Datenbankeinträge ein.

5.4 E-Mailverwaltung

Dieses Modul ist Bestandteil der Diplomarbeit und hat hohe Priotität.

- E-Mailkonten werden Hosting-Benutzern zugeordnet
- E-Mailkonten sind Unabhängig von Domains
- Einem E-Mailkonto können belibeig viele E-Mailadressen zugeordnet werden
- Jeder Domain/Subdomain können beliebig viele E-Mailadressen zugeordnet werden
- Quotas für E-Mailkonten sind möglich
- Autoresponder pro E-Mailadresse durch Benutzer zu/abschaltbar
- Spam- und Virencheck pro E-Mailadrese durch Benutzer zu/abschaltbar
- Webmailzugang erfolgt über Fremdprodukt oder eigenes Modul

5.5 Mailinglistenverwaltung

Dieses Modul ist Bestandteil der Diplomarbeit und hat mittlere Priorität.

- Sofern freigeschaltet, kann jeder Kunde beliebig viele Mailinglisten anlegen
- Als Mailinglistensoftware wird Mailman verwendet
- Das Anlegen und Löschen von Mailinglisten erfolgt über Consolvix
- Die Pflege von Mailinglisten erfolgt (bis auf weiteres) über Mailman's eigener Konfigurationsoberfläche mit an Consolvix angepasstem "Look&Feel"
- Die Möglichkeit der Mailinglistenverwaltung über MySQL mus snoch erörtert werden. Sollte dies möglich sein, so wird die Administration über ein Consolvix-Modul implementiert.

5.6 Shop-System

Dieses Modul ist nicht Bestandteil der Diplomarbeit, und hat niedrige Priorität.

- muss noch ausgearbeitet werden.

5.7 Kundenverwaltung/Rechnungswesen

Dieses Modul ist nicht Bestandteil der Diplomarbeit, aber hat hohe Priorität.

- Kunden müssen sich via Shop selbst registrieren können; hierdurch wird aber noch kein Zugang zum Kundenbereich möglich
- Sowohl Katalogpreise als auch Individualpreise für sämtliche angebotenen Leistungen müssen möglich sein
- Rechnungen sollen in auswählbaren Rechnungszeiträumen erstellt werden können
- Rechnungen werden per E-Mail an den Kunden gesandt
- Rechnungen werden pro Kunde, nicht pro Webhostingkonto erstellt
- Abgerechnet werden immer *alle* Hostingleistungen für den kommenden, alle weiteren Leistungen für den letzten Rechnungszeitraum.
- Die Fälligkeit von Rechnungen muss automatisch ermittelt und via Erinnerungsmail o.Ä. dem Hostingbetreiber bekanntgegeben werden.
- Rechnungen werden sowohl als Rohdaten als auch als PDF-Datei in der Datenbank abgelegt
- Nach Erstellen einer Rechnung können die abgerechneten Positionen/die Rechnung selbst nicht mehr modifiziert, allenfalls aber gelöscht und neu erstellt werden.
- Jeder Kunde kann eine beliebige Anzahl an Adressen anlegen. Eine Adresse ist die Default-Adresse
- Pro Rechnung wird eine Adresse explizit angegeben.

5.8 Ruby on Rails

PHP als serverseitige Programmiersprache hat sich mittlerweile als Standard etabliert, aber Ruby on Rails ist noch immer eine Seltenheit bei von Massenhostern angebotenen Serversystemen. Aus diesem Grunde wird eine zentrale Aufgabe der zu implementierenden Software die Verwaltung von Ruby on

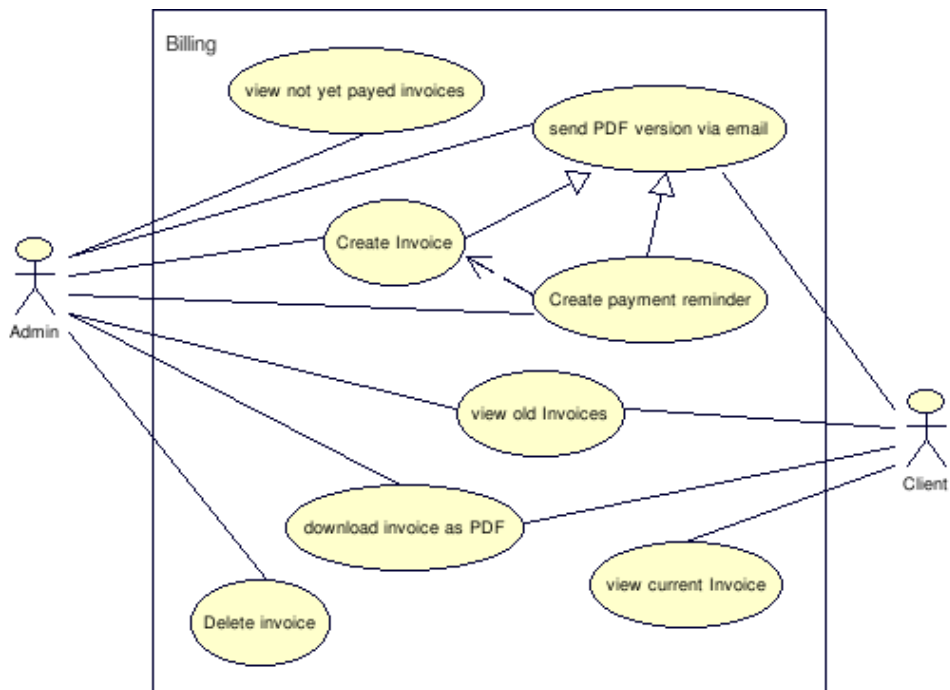


Abbildung 3: Use-Case Diagramm für das Rechnungswesen

Rails Applikationen sein. Im Wesentlichen würde diese Aufgabe im ersten Schritt darin bestehen, die von Rails zur Verfügung gestellten Skripte wie `rails <Verzeichnisname>` oder `script/generate ...` aufzurufen und die Datenbankkonfigurations- und `.htaccess`-Dateien zu erstellen sowie die entsprechenden Apache2-VirtualHosts zu verwalten. Noch ist nicht entschieden, ob die Rails-VirtualHosts in einem separaten Wurzelverzeichnis liegen sollen oder ob prinzipiell jeder Systembenutzer die Möglichkeit haben soll, in seinem eigenen Home-Verzeichnis beliebige Rails-Applikationen aufzusetzen. Benutzern soll es offen stehen, auch interagierende PHP- und Ruby-Applikationen zu hosten. Inwieweit die Kapselung der einzelnen Verzeichnisstrukturen daher sinnvoll ist, bedarf noch einer eingehenden Erörterung.

Dieses Modul ist Bestandteil der Diplomarbeit und hat hohe Priorität.

- Rails-VirtualHosts sind normale VirtualHosts mit einigen Zusatzangaben
-

5.9 Support-Ticket-System

Dieses Modul ist nicht Bestandteil der Diplomarbeit und hat niedrigste Priorität

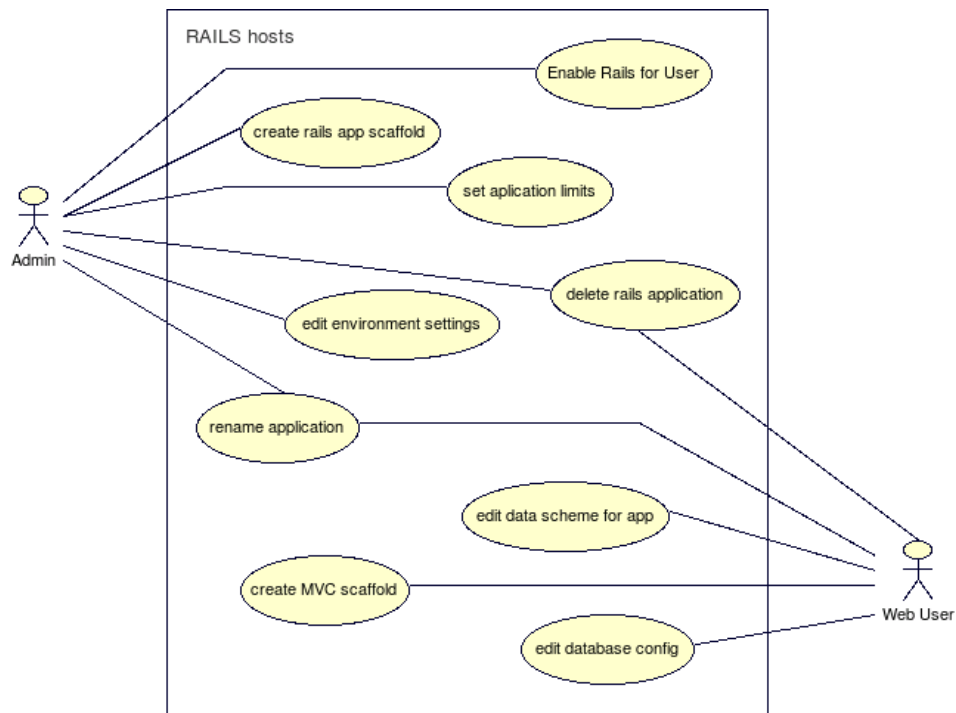


Abbildung 4: Use-Case Diagramm für die Ruby on Rails- und VirtualHost-Verwaltung

5.10 ...

Diese Liste ist noch unvollständig.

6 Testszzenarien

Eine Liste der modulweisen Testszzenarien kann und wird erst nach einer detaillierteren Anforderungsbeschreibung der Software erstellt werden.

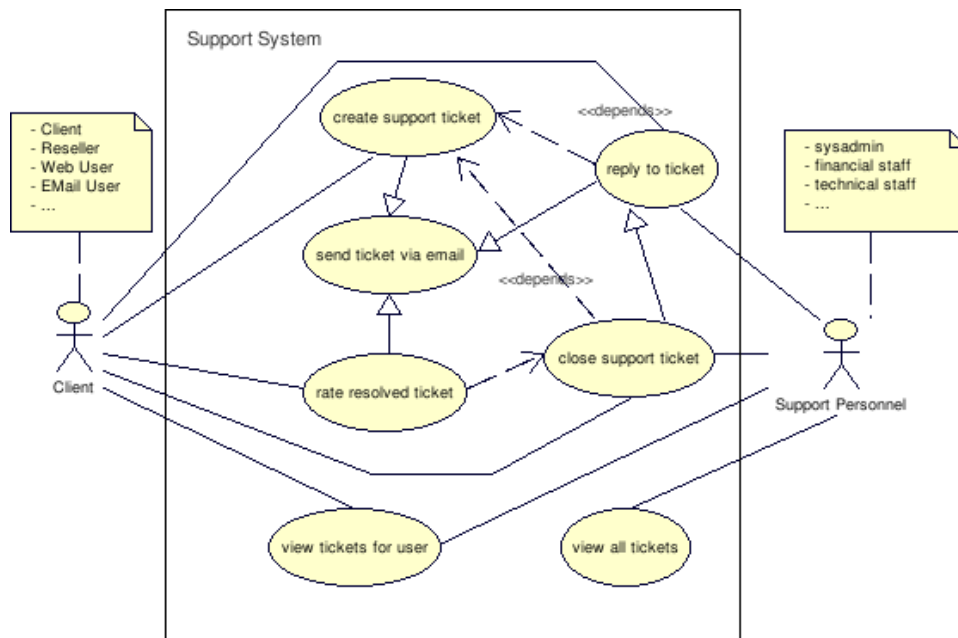


Abbildung 5: Use-Case Diagramm für das Support Ticket-System