Neugestaltung der Benutzerverwaltung der Fakultät für Informatik

Studienarbeit an der Fakultät für Informatik,
Universität Karlsruhe

Patrick von der Hagen und Christian Knierim

Betreut von:
Prof. Dr. P.C. Lockemann und J. Mülle, IPD
sowie
K. Scheibenberger und O. Hopp, ATIS

April 2001

Olaf Hopp

Inhaltsverzeichnis

Kapitel 1

Vorwort

Um an der Fakultät für Informatik Mailadressen zentral zu verwalten existiert eine zentrale Benutzerverwaltung, die von der ATIS administriert wird. Der Datenbestand dieser Benutzerverwatung - auch Useradm genannt - wird über zwei Benutzerschnittellen (ASCII und HTML) von sog. ATIS-Beauftragten der einzelnen Institute für ihren Bereich gepflegt. Zielsetzung dieser Studienarbeit war eine Neukonzeption der Benutzerverwaltung auf Basis eines SQL-Datenbank-Managementsystems.

Da an der Falkutät mehrere Auslieferungsrechner betrieben werden, ist die zentrale Aufgabe der Benutzerverwaltung die Zuordnung einer Adresse *y@ira.uka.de* zu einem Auslieferungsrechner. Weiterhin exisitierte ein Verzeichnisdienst namens PH-Server, der hauptsächlich von der Fakultätsverwaltung verwendet wurde. Bei der Neukonzeption der Benutzerverwaltung mußte somit darauf geachtet werden, daß die Anbindung des Mailsystems gewährleistet bleibt und zumindest eine Alternative zum Verzeichnisdienst zur Verfügung gestellt wird.

Die Aufgaben bei der Umstellung wurden auf Patrick von der Hagen und Christian Knierim aufgeteilt.

Patrick von der Hagen kümmerte sich um die Migration des alten Datenbestandes. Außerdem kümmerte er sich um die Anbindung des PP-Mailsystems und die Anbindung an die Datenbasis eines LDAP-Dienstes, der u.a. als Ersatz für den oben genannten PH-Server dient. Patrick von der Hagen ist somit Author der Kapitel 2, 3, 7 und 8.

Christian Knierim entwickelte die Benutzerschnittstellen der Benutzerverwaltung und verfasste somit die Kapitel 5 und 6.

Gemeinsam wurde die Struktur der Datenbasis entworfen. Kapitel 4 wurde anschließend ebenfalls von Christian Knierim verfasst.

An dieser Stelle möchten wir besonders den Betreuern J. Mülle, O. Hopp und K. Scheibenberger für deren Unterstützung während der Durchführung der Studienarbeit danken, die uns während der Studienarbeit mit Rat und Tat zur Seite standen.

Kapitel 2

Aufgabe des Useradm

An der Fakultät für Informatik steht der Namensraum "ira.uka.de" für Email-Adressen zur Verfügung. Diesen Namensraum müssen sich die angeschlossenen Institute teilen und, da Email-Adressen nur einmal vergeben können, gemeinsam verwalten.

Diese zentrale Verwaltung ist Aufgabe des "Useradm".

Des weiteren müssen Mailing-Listen, Mail-Aliase und Mail-Weiterleitungen mit in dieses System eingebunden werden. Deshalb ist die Bezeichnung "Useradm" leicht irreführend, denn der "Useradm" hat nicht primär die Aufgabe, User bzw. Anwender zu verwalten, sondern er soll in erster Linie die Mail-Adressen koordinieren.

Da jeder an der Fakultät für Informatik eingerichtete Unix-Account auch einer Email-Adresse entspricht, müssen alle Accounts im Useradm erfaßt und verwaltet werden. Da manche Institute NFS- bzw. NIS-Server zur Verwaltung der Accounts gemeinsam betreiben, bietet es sich an, zusätzlich zu den für das Mailsystem erforderlichen Informationen auch Daten für Unix-Accounts im Useradm zu erfassen und zu verwalten.

2.1 Alter Useradm

Der alte Useradm verfügte über eine textuelle und eine web-basierte Schnittstelle. Er basierte auf dem Bug-Tracking-Sytem "Clear-DDTS" in der Version 4.5.1 von Rational als Datenbank. Dieses System konnte zwar per SQL-Schnittstelle abgefragt werden, jedoch waren Änderungen nicht über diese Schnittstelle möglich. Statt dessen mußte zum Vornehmen von Änderungen ein proprietäres "Template"-System verwendet werden.

Damit war dieses System als Datenbank eigentlich ungeeignet und wurde nur aufgrund einer besonderen Eigenschaft gewählt: Alle Datensätze wurden in recht einfach aufgebauten ASCII-Dateien gespeichert.

Diese Eigenschaft war der zentrale Grund für die Verwendung von ClearDDTS als Back-End für den Useradm, hat sich jedoch vor allem in Hinblick auf die Geschwindigkeit als ausgesprochen nachteilig erwiesen, zumal das System auch auf jegliche Indizes verzichtet.

Der alte Useradm wurde ausschließlich mit Shell-Scripten implementiert. Zugriff war ausschließlich lokal möglich, es fehlten die inzwischen üblichen Datenbank-

Schnittstellen (JDBC, ODBC, Perl-DBI, Python, etc.). Erweiterungen der alten Struktur waren damit kaum sinnvoll möglich.

Zum Benutzer hin wurde der Useradm über eine textuelle und eine web-basierte Schnittstelle angeschlossen, die Anbindung des Mailinglisten-System der Fakultät für Informatik erfolgte separat. Aus dem Datenbestand des Useradm wurden einmal täglich Auslieferungs-Informationen für das angeschlossene Mail-system generiert und auf die beteiligten Mail-Server verteilt. Dabei wurden auch ein "PH-Server" als Adreßbuch gepflegt und einige Mailinglisten automatisch generiert.

Auch dieser Useradm wurde im Rahmen einer Studienarbeit eingeführt und hat 1998 seinen Vorgänger abgelöst, der auf einer Oracle-Datenbank basierte.

2.2 Gründe für die Neukonzeption

Seitdem der alte Useradm 1998 eingeführt wurde, wurden leider viele Probleme deutlich. Änderungen an den Benutzer-Schnittstellen waren gewünscht, hätten jedoch einen großen Aufwand erfordert. Die unbefriedigende Arbeitsgeschwindigkeit konnte nicht verbessert werden und aufgrund des Fehlens von internen Konsistenzbedingungen im Datenbank-System haben sich Inkonsistenzen der Daten ergeben.

Anlaß für die Neukonzeption war die Entscheidung der ATIS, das derzeit verwendete Mail-System "Isode PP" auf einigen zentralen Servern durch ein neues System zu ersetzen. Dies hätte massive Änderungen der bestehenden Anbindung von Useradm und Mailsystem zur Folge gehabt, so daß eine Neukonzeption unter den Gesichtspunkten "Wahrung der Anbindung an das PP-Mailsystem" und "Vorbereitung einer Anbindung an ein neues Mail-System" als sinnvollste Lösung erschien. Besonderer Wert wurde dabei auch auf Dokumentation und Erweiterbarkeit des Systems gelegt.

2.3 Neuer Useradm

Der neue Useradm sollte mit einer weiter entwickelten Programmier-Sprache programmiert werden, als das alte System, das vollständig mit Shell-Scripten (Unix-ksh) realisiert wurde. Die Wahl fiel auf Perl, da damit gleichermaßen das Web-Frontend, die textuelle Schnittstelle und die Anbindung an das zu Grunde liegende Mailsystem realisiert werden können. Außerdem wird diese Sprache von der ATIS auch für viele andere Aufgaben eingesetzt, was die Wartung und Erweiterung des neuen Useradm erleichtern wird.

Als Datenbank wurde Postgresql (zu Beginn der Studienarbeit in Version 7.0 verfügbar) gewählt. Gründe sind die freie Verfügbarkeit und die vergleichsweise umfangreiche Implementierung des SQL92-Standards. Insbesondere Trigger und Foreign-Keys haben die Implementierung deutlich erleichtert, da damit die Einhaltung wichtiger Konsistenz-Bedingungen von Datenbank-System überwacht wird. Außerdem existieren viele Programmierschnittstellen (z. B. ODBC, JDBC, Perl-DBI, Python, Php3, etc.) was in Hinblick auf spätere Erweiterungen ein wichtiges Kriterium war. Erfahrungen mit Postgresql waren bereits aus einem anderen Projekt vorhanden.

Der alte Useradm ist nur durch wenige Shell-Scripte direkt mit dem Mailsystem verbunden. Die Ausgabe dieser Scripte wird dann vom Mailsystem ausgewertet und weiter verarbeitet. Es bietet sich an, nur diese Scripte auszutauschen, so daß sie aus der neuen Datenbank möglichst die gleiche Ausgabe erzeugen, um den Rest des Mailsystems nicht verändern zu müssen. Dadurch wird der neue Useradm mit minimalem Aufwand eingebunden, während gleichzeitig auch die Wahrscheinlichkeit, neue Fehler einzuführen, minimiert wird.

Die Anbindung an die Mailing-Listen erfolgt analog zum Mailsystem über eine kleine, klar definierte Schnittstelle, die im Rahmen dieser Studienarbeit neu implementiert werden mußte. Neue Mailing-Listen müssen nicht per Frontend im Useradm angelegt werden, das Mailing-Listen-System (Petidomo) meldet und löscht neue Listen direkt in der Datenbank. Damit wird gewährleistet, dass die verwendeten Adressen der Mailing-Listen nicht mit den regulären Mail-Adressen kollidieren.

Statt den Useradm wie bisher an den PH-Server als Adreßbuch an zubinden, wurde universell eine Anbindung an LDAP angestrebt. Dies ermöglicht wesentlich mehr Mail-Clienten den Zugriff auf das Adreßbuch, als das bisher der Fall war.

Verwendet wird als frei verfügbare Software OpenLDAP, die Aktualisierung erfolgt regelmäßig Zeit-gesteuert per Perl-Script. Es wäre auch möglich, direkt C-Funktionen in die Postgresql-Datenbank einzubauen, jedoch sollte die Implementierung aus Wartungsgründen möglichst in nur einer Sprache realisiert werden.

Auch die Anbindung des neuen Mailsystems (Exim) an den Datenbestand des neuen Useradm kann mit Hilfe von LDAP besonders einfach gelöst werden und soll im Zuge der Studienarbeit vorbereitet werden.

Kapitel 3

Der alte Useradm

An dieser Stelle soll ein kurzer Überblick über den Datenbestand und die Datenstruktur im alten Useradm gegeben werden, da diese Daten den Rahmen für die Neukonzeption vorgegeben haben. Im Zuge der Neukonzeption sollte untersucht werden, welche Daten vorhanden sind, welche Daten nicht mehr benötigt werden und welche Daten in Zukunft ggf. neu/zusätzlich erfaßt werden müssen.

3.1 Tabellen des alten Useradm

3.1.1 Gemeinsamkeiten der Tabellen

Jede Tabelle, mit Ausnahme der Benutzer-Tabelle, enthält die folgenden Spalten: IDENTIFIER, ST, EINGETRAGEN_AM, EINGETRAGEN_VON, MOD, MOD_WER. Dabei handelt es sich um eine eindeutige ID als Schlüssel, den Status (aktiv oder ruhend), die Information wer wann den Datensatz angelegt und wer wann den Datensatz verändert hat.

Die ID ist nicht nur für die Tabelle, sondern für die gesamte Datenbank eindeutig. ClearDDTS speichert jeden einzelnen Datensatz in einer eigenen Datei ab. Benötigt man im alten Useradm einen Schlüssel, so wird einfach der eindeutige Datei-Name verwendet. Da sich die ID aus einer Zeichenkette "IRAtt" und einer fünfstelligen Zahl zusammensetzt, haben wir uns entschlossen, bei der Umstellung auf den neuen Useradm dieses Feld nicht direkt zu übernehmen sondern durch eine neue, numerische ID zu ersetzten. Dafür sind während der Umstellung zusätzliche Tabellen nötig, um eine Gegenüberstellung von alten und neuen IDs zu haben. Diese Tabelle können nach der Umstellung wieder gelöscht werden.

Das Status-Feld hat nahezu immer den Wert "aktiv". Da es kaum genutzt wird, entscheiden wir uns, es im neuen Useradm nicht mehr weiter zu führen. Die Felder EINGETRAGEN_AM, EINGETRAGEN_VON, MOD, MOD_WER können im neuen Useradm zu LETZTEAENDERUNGVON und LETZTEAENDERUNGAM zusammen gefasst werden.

Der alte Useradm umfasst die folgenden Tabellen, die nun näher beschrieben werden sollen:

- agent
- auslieferungsgruppe

- benutzer
- gebuehrengruppe
- gid_bereich
- institution
- mailadresse
- mailalias
- uid_bereich

3.1.2 Tabelle Benutzer

In der Tabelle "Benutzer" existieren die Felder ID, VORNAME, NACHNAME, TITEL. Zum Zeitpunkt der Umstellung umfaßt die Tabelle 2828 Einträge. Leider enthält das Feld TITEL nur Zeichenketten, z. B. "Professor" oder "Doktor", wobei diese Strings nicht einheitlich sind (z. B. "Dr." statt "Doktor"). Bei der Neukonzeption soll statt dessen eine eigene "Titel"-Tabelle eingefügt werden.

3.1.3 Tabelle Institution

Diese Tabelle hat die Felder NAME und BEMERKUNG, z. B. Name "I71" und Bemerkung "Telematik". Diese Tabelle definiert alle vorhandenen Institute, die vom Useradm verwaltet werden sollen und wird von den übrigen Tabellen über Fremdschlüssel referenziert.

3.1.4 Tabelle Auslieferungsgruppe

Die Tabelle "Auslieferungsgruppe" besteht aus den Feldern NAME, MASCHINE, ART, GEBUEHRENGRUPPE und BEMERKUNG. Die Emails an den Namensraum "ira.uka.de" sollen auf unterschiedliche Mailserver ausgeliefert werden. Um dies zu erreichen wurde das Konzept der Auslieferungsgruppe eingesetzt. Mehrere Email-Accounts werden einer Auslieferungsgruppe zugeordnet. Jede Auslieferungsgruppe hat einen Mailserver "Maschine" und gehört zu einer "Gebührengruppe". Über die Gebührengruppe wird eine Auslieferungsgruppe einem Institut zugeordnet, wobei ein Institut mehrere Auslieferungsgruppen haben kann.

3.1.5 Tabelle Gebuehrengruppe

Ursprünglich wurden von der ATIS Gebühren von den Instituten für den Betrieb des Mailsystems erhoben. Ein Institut kann mehrere Gebührengruppen einrichten um die Kosten nach den einzelnen Gruppen aufschlüsseln zu lassen. Die Zuordnung der Gebührengruppen zu den Instituten erfolgt in dieser Tabelle. Die dafür verwendeten Attribute sind NAME, INSTITUTION und BEMERKUNG.

Da inzwischen keine Volumen-abhängigen Gebühren mehr erhoben werden, ist die genaue Aufschlüsselung nach Gebührengruppen hinfällig geworden. Im neuen Useradm kann darauf verzichtet werden, wobei natürlich auch die Zuordnung von Auslieferungsgruppen und Instituten dementsprechend angepaßt werden muß.

3.1.6 Tabelle Agent

Besteht aus den Feldern: NAME, GEBUEHRENGRUPPE, BEMERKUNG. Da die Verwaltung des Useradm unter Leitung der ATIS dezentral in den einzelnen Instituten erfolgen soll, müssen Ansprechpartner in den Instituten vorhanden sein, die einen Teil der Account-Verwaltung übernehmen und bei Problemen kontaktiert werden können. Diese Ansprechpartner werden in dieser Liste aufgeführt und ihrem jeweiligen Institut zugeordnet.

3.1.7 Tabelle GID_Bereich

Einem Institut können mehrere (nicht notwendigerweise zusammenhängende) Bereiche zugewiesen werden, aus denen sie Unix-Group-IDs vergeben können. Dies ist zur Synchronisation der NIS- bzw. NFS-Server erforderlich. Die Zuordnung der Bereiche zu den Instituten erfolgt in dieser Tabelle. Die einzelnen Attribute sind INSTITUTION, GID_VON, GID_BIS, BEMERKUNG

3.1.8 Tabelle UID_Bereich

Diese Tabelle ist völlig analog zur Tabelle "GID_Bereich" aufgebaut. Die einzelnen Attribute sind INSTITUTION, UID_VON, UID_BIS, BEMERKUNG

3.1.9 Tabelle Mailadresse

Die Tabelle "Mailadresse" ist bei weitem die umfangreichste Tabelle, sowohl was die Zahl der Attribute als auch was die Zahl der Einträge betrifft. Im alten Useradm umfaßt sie mehr als 3500 Einträge.

Leider ist diese Tabelle im alten Useradm nur schlecht konzipiert. Grund ist die bereits erwähnte, unbefriedigende Verarbeitungsgeschwindigkeit der ClearDDTS-Datenbank, der man durch eine teilweise Denormalisierung der Tabellen begegnen wollte. Daher enthält diese Tabelle Einträge zu den Bereichen "Unix-Account", "Mail-Forward" und "Mailingliste", die wir bei der Neukonzeption des Useradm trennen wollen.

Die Tabelle besteht aus den folgenden Attributen: VERFALLSDATUM, ART, KLASSE, NAME, AUTHORISATION, DIENSTTELEFON, BEMERKUNG, MATRIKELNUMMER, BENUTZER, AUSLIEFERUNGSGRUPPE, USER_ID, GROUP_ID, FORWARD, LISTOWNER, LISTFILE, SU, CL, MO, AR, DI, DIEGESTINTER, EXTRAKT, LISTEN_EXTRAKT, IN.

Die Attribute DIENSTTELEFON und MATRIKELNUMMER beziehen sich nicht wirklich auf eine Email-Adresse sondern auf den zugehörigen Nutzer. Daher soll diese Attribute im neuen Useradm direkt einem Benutzer zugeordnet werden.

Zum Bereich "Unix-Account" zählt hier der NAME im Sinne von "Login" oder "Account", die USER-ID und die GROUP-ID, außerdem die AUSLIEFERUNGSGRUPPE als Angabe für das Mailsystem, auf welchem Auslieferungs-Server dieser Account zu finden ist. Das VERFALLSDATUM soll die Vergabe von Accounts "auf Zeit" ermöglichen. So können beispielsweise für Studien- oder Diplom-Arbeiten Accounts eingerichtet werden, bei denen der Betreuer bei Erreichen des Verfallsdatums informiert wird und den Account verlängern oder löschen kann.

AUTORISATION hat einen der Werte "extern, intern, nicht-autorisiert". Es war aus Kostengründen gewünscht, für einzelne Accounts festlegen zu können,

ob sie nach außen Emails verschicken dürfen ("extern"), ob sie nur innerhalb der Fakultät Emails verschicken dürfen ("intern") oder ob sie keinerlei Berechtigung zum Versenden von Email haben ("nicht-authorisiert"). Da keine Gebühren mehr erhoben werden, hat diese Unterscheidung ihren Sinn verloren und muss bei der Neukonzeption nicht mehr berücksichtigt werden.

Die KLASSE ordnet den Account einer Personengruppe zu. So gibt es Mitarbeiter, Professoren, Studenten, Diplomanden etc. Dies ist erforderlich, da automatisch Mailinglisten erzeugt werden sollen, mit denen z. B. alle Mitarbeiter eines Instituts oder alle Diplomanden der Fakultät erreicht werden können. Analog zum TITEL in der Benutzer-Tabelle ist es auch hier sinnvoll, eine eigene Tabelle "Klasse" anzulegen.

Das Feld IN gibt an, ob die automatische Aufnahme in die jeweiligen Mailinglisten gewünscht ist.

BENUTZER ist ein Fremdschlüssel und referenziert die ID des Benutzers in der Benutzer-Tabelle, der für diesen Account verantwortlich ist.

Das Attribut FORWARD hat keine Beziehung zu einem Unix-Account und soll deshalb einer anderen Relation zugeordnet werden. In diesem Feld kann eine Email-Adresse eingetragen werden, wenn alle Emails für NAME an eine andere Adresse weiter geleitet werden soll.

LISTOWNER und LISTFILE haben nur eine Bedeutung, wenn sich der Eintrag eine Mailingliste beschreibt. Dann kann man LISTOWNER den Verantwortlichen für diese Mailingliste entnehmen, LISTFILE enthält interne Informationen für das Mailinglisten-System. Auch diese Attribute sollen in eine eigene Relation verlegt werden.

Die Bedeutung der Felder SU, CL, MO, AR, DI und DIGESTINTER ist leider nicht dokumentiert und läßt sich aus dem Datenbestand nicht erschließen.

Die Felder EXTRAKT und LISTEN_EXTRAKT sind eine weitere Maßnahme zur Geschwindigkeits-Steigerung des alten Useradms. Sie enthalten die Informationen, die das Mail-System regelmäßig aus dem Useradm extrahieren muß, um diesen Vorgang zu beschleunigen. Für einen normalen Account den "Namen", die "Berechtigung", die "User-ID", die "Group-ID" und die Auslieferungsgruppe. Derartige Maßnahmen sollen beim neuen Useradm nicht ergriffen werden. Das genaue Format der Einträge in diesen Feldern wird im Kapitel über die Anbindung des alten Mailsystems erklärt.

3.1.10 Tabelle Mailalias

Die Tabelle "Mailalias" besteht aus den Attributen ALIAS, MAILADRESSE, BEMERKUNG und EXTRAKT. MAILADRESSE ist ein Fremdschlüssel auf eine Adresse in der Tabelle "Mailadresse", ALIAS bezeichnet den Alias, der dieser Adresse zugeordnet werden soll. Im EXTRAKT werden wie in MAILADRESSE wichtige Informationen für das Mailsystem zusammengefaßt, in diesem Fall Alias, Account, Berechtigung, User-ID, Group-ID uns Auslieferungsgruppe. Auch diese Tabelle soll überarbeitet werden. Es bietet sich an, ein Alias nur als einen Spezialfall eines Mail-Forwards zu betrachten, derart daß das Alias *Christian.Knierim* ein Mail-Forward auf *knierim@ira.uka.de* darstellt. Daher kann man im neuen Useradm eine Tabelle entwerfen, in der sowohl Aliase als auch Mail-Forwards verwaltet werden.

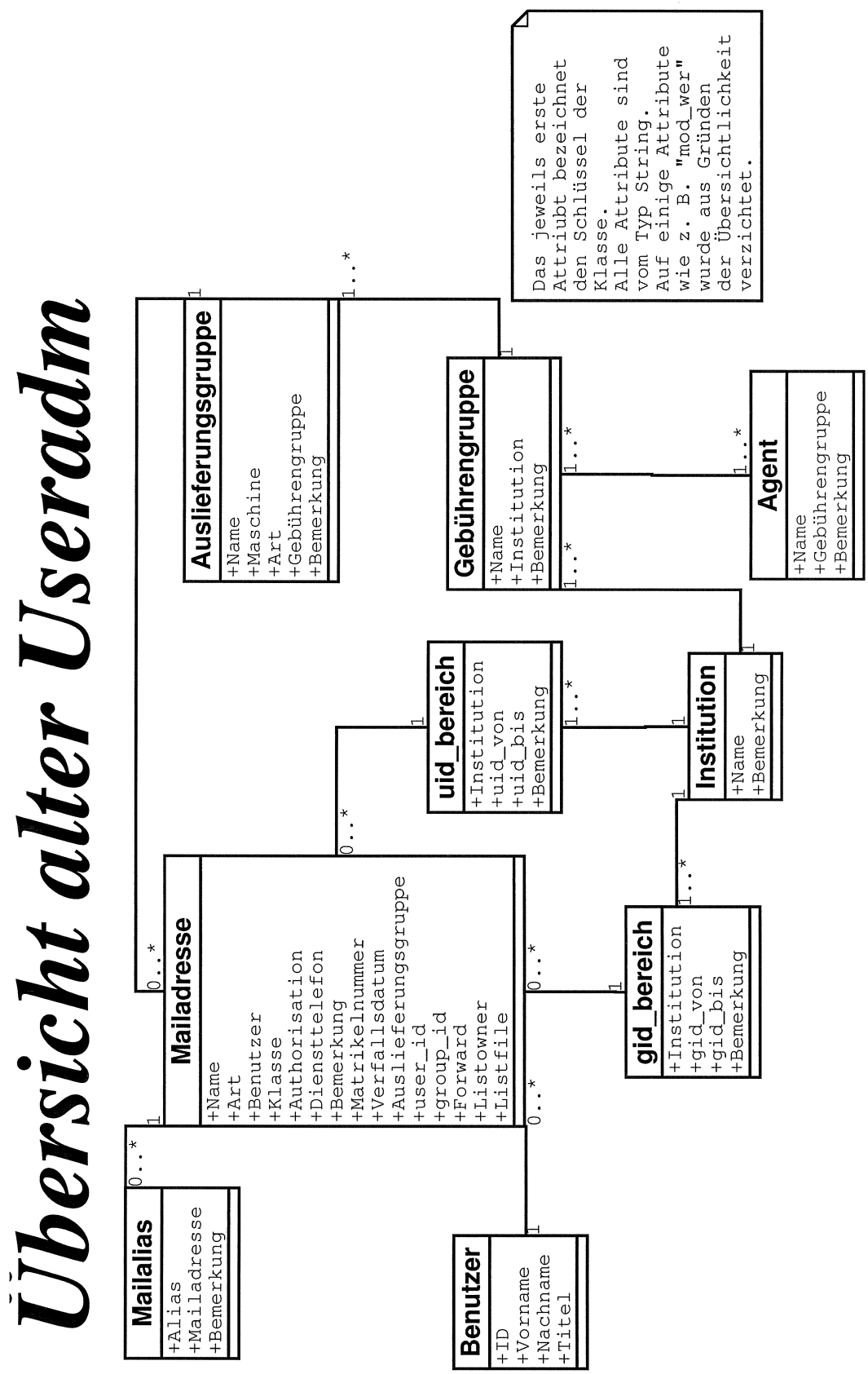

Abbildung 3.1: Tabellenstruktur des alten Useradm

Kapitel 4

Die Tabellenstruktur

In diesem Kapitel werden alle Tabellen des neuen Useradms beschrieben, sowie deren Bedeutung in den Front-Ends und der Mail-bzw. LDAP-Anbindung untersucht.
Als Vorbemerkung ist festzustellen, daß der Benutzer der Useradm-Frontends lediglich auf die Benutzer-, Unixaccount- und Mailaliastabellen sowie der Blacklist direkt schreibenden Zugriff hat.
Eine detaillierte graphische Übersicht der Tabellenstruktur ist im Anhang zu finden (Abb A.1). Hier soll zuerst die grobe Struktur der Tabellen m.H. eines UML-Klassendiagramms dargestellt werden. (vgl. Abb. 4.1[1])

4.1 Benutzer-Tabelle:

In der Benutzertabelle werden neben NAMEN, VORNAMEN und der TITEL-ID auch die MATRIKEL-Nummer - falls vorhanden - abgelegt. Die Matrikelnummer wird beispielsweise zur automatischen Verlängerung von Studenten-Accounts verwendet. NAME, VORNAME und eine vom Front-End zugewiesene Benutzer-ID stellen eine eindeutige Zuordnung des Benutzers zu den entsprechenden Accounts und Alias-Namen sicher.
Weiterhin kann in dieser Tabelle die Telefonnummer eines Benutzers eingetragen werden, damit diese im LDAP-Server publiziert werden kann. Als Standardwert wird für dieses Feld *false* eingesetzt.
Über die Felder LETZTEAENDERUNGVON und LETZTEAENDERUNGAM von kann zurückverfolgt werden, wer wann die letzte Änderung am Datensatz vorgenommen hat. Diese beiden Felder werden mit den Standardwerten des eingeloggten Benutzers und dem Zeitstempel der Einfügeoperation gesetzt.

4.2 Titel-Tabelle:

In dieser Tabelle werden lediglich ein Titelschlüssel und die ensprechende Titelbezeichnung abgelegt. Sinn und Zweck dieser Tabelle ist sicher zu stellen, daß

[1] Die unterstrichenen Attribute repäsentieren die Primärschlüssel (Ausnahme: Bei UID handelt es sich um den Sekundärschlüsasel der Relation UNIXACCOUNT). Die Pfeile auf die Relation VERGEBENEADRESSEN sollen die in der Datenbasis spezifizierten *rules* repräsentieren. Diese werden in Abschnitt 11 dieses Kapitels beschrieben.

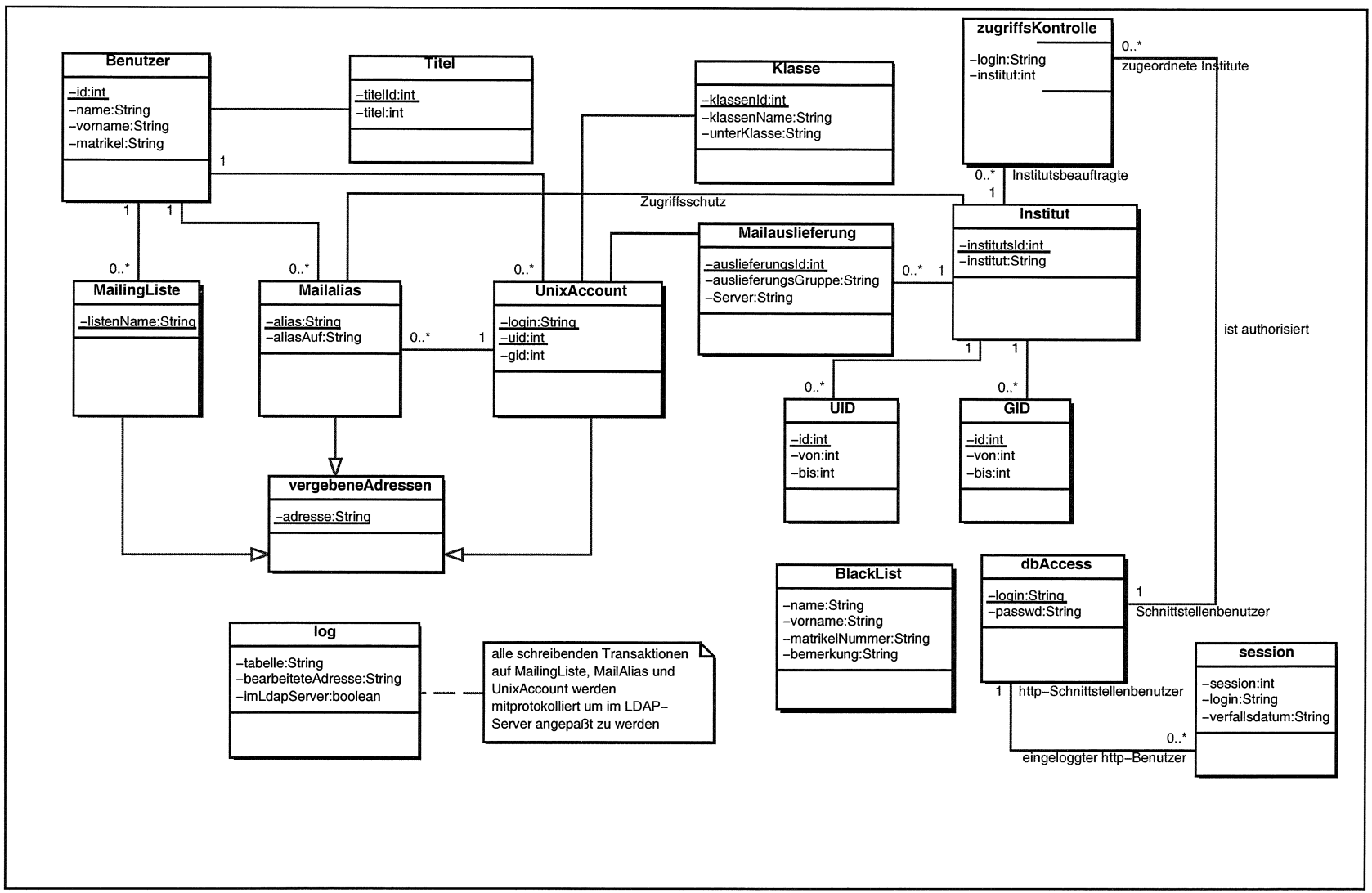

Abbildung 4.1: Tabellenstruktur der Useradm-Datenbasis

das Titel-Feld der Benutzer-Tabelle nur mit konsistenten Daten gefüllt wird und keine Titeleinträge wie MITARBEITER o.ä. im TITEL-Feld der Benutzertabelle landen. Erreicht wird dies mit Hilfe der von postgresql unterstützten Fremdschlüsselreferenz.

4.3 Unix-Account-Tabelle:

Hier werden alle relevanten Daten für die Existenz eines Unix-Accounts abgelegt. Typischerweise findet man hier neben der Unix-Id, dem Loginnamen und der Gruppen-Id, den für das Mailsystem relevanten Schüssel der Auslieferungsgruppe. Im Bereich Mailsystem gibt es in dieser Tabelle noch ein boolsches Feld für automatisch generierte Listen. Falls ein Benutzer nicht in die zu seinem Institut gehörenden autmatisch generierten Institutslisten aufgenommen werden möchte, kann man an dieser Stelle die Variable ININSTITUTSLISTEAUFNEHMEN auf *false* setzen.
Die Zuordnung zum Besitzer des Accounts wird über die in der Benutzertabelle abgelegten ID des Benutzerdatensatzes hergestellt. Dank Fremdschlüsselreferenzen innerhalb der Datenbank ist die Existenz eines Accounts ohne einen entsprechenden Benutzer ausgeschlossen. Es besteht die Möglichkeit einem Account ein VERFALLSDATUM sowie BEMERKUNGEN zuzordnen. Außerdem existiert ein KLASSE-Feld, das zur Zeit nur statistische Funktion für die Front-Ends besitzt und einen Schlüßel der in der u.g. Klassentabelle angegebenen Klasse referenziert.
Für den LDAP-Dienst existiert ein boolsches Feld, damit die Möglichkeit, einen Account nicht im LDAP-Server zu publizieren, gegeben ist.
LETZTEAENDERUNGAM und LETZTEAENDERUNGVON werden lediglich statistisch verwendet.

4.4 Klassen-Tabelle:

Unix-Benutzerkonten werden in sog. Klassen eingeteilt. Unter einer Klasse werden beispielsweise alle Professoren zusammengefasst. Jeder Klasse ist eine automatisch generierte Mailingliste zugeordnet, die dann sämtliche Mitglieder einer Klasse beeinhaltet. Neben der der KLASSENID und dem KLASSENNAMEN ist in dieser Version des Useradm die Verwendung von Unterklassen eingeführt worden. Diese Unterklassen sollen die oben genannten Klassen weiter verfeinern, indem z.B. der Klasse *Mitarbeiter* die Unterklassen *Sekretärin* und *techn. Angestellter* zugeordnet werden um aus diesen Informationen automatisch unterklassenbezogene Listen zu erzeugen.

4.5 Mailauslieferungs-Tabelle:

Diese Tabelle ist in erster Line für das Mailsystem relevant, da die Mailserver der verschiedenen Auslieferungsgruppen eingetragen sind.
Mailauslieferungsgruppen sind notwendig, da eine Mail-Adresse die Form *y@ira.uka.de* hat, somit ein Adreßraum für alle Institute existiert aber die Nachrichten an den Adressaten nicht an einen zentralen Mailserver ausgeliefert werden. Jedes Institut besitzt einen oder mehrere Mailserver, auf denen eine Nachricht ausgeliefert

wird. Gehört ein Benutzerkonto y einer Auslieferungsgruppe x an, kann das Mailsystem bzw. der Mailgateway für die Domäne *.ira.uka.de* m.H dieser Relation ermitteln, auf welchen Rechner Nachrichten für den Benutzernamen y ausgeliefert werden sollen.

Zusätzlich wird jeder Auslieferungsgruppe ein Institutsschlüssel zugeordnet, was innerhalb der UNIXACCOUNT-Tabelle die Zuordung von User- bzw. Group-ID-Bereichen für die Unixaccounts ermöglicht. Weiterhin wird die Auslieferungsgruppe als Zugriffsschutzmechanismus [2] für die Unix-Accounts innerhalb der Web- und ASCII-Schnittstelle verwendet. Hiermit ist es einem nicht-authorisiertem Administrator nicht möglich Accounts über die Front-Ends zu löschen oder zu modifizieren. Die AUSLIEFERUNGSID ist ebenfalls primärer Schlüssel für die AUSLIEFERUNGRUPPE in der Unixaccount-Tabelle.

4.6 Instituts-Tabelle:

Es existiert das Feld INSTITUTSID dem der Name des Instituts in dem Feld INSTITUT zugeordnet ist. Die INSTITUTSID ist u.a. primärer Schlüssel für das INSTITUT-Feld der Gid-Tabelle bzw der Uid-Tabelle und der Mailauslieferungstabelle.
In der Beauftragten-Tabelle existiert ebenfalls eine Fremdschlüsselreferenz auf die INSTITUTSID.
Aus Zugriffsschutzgründen existieren innerhalb der Mailalias-Tabbelle, sowie innerhalb von der Zugriffskontrolle-Tabelle Refernzen auf die INSTITUTSID.

4.7 Beauftragten-Tabelle:

In dieser Tabelle werden jedem Institut über die INSTITUTSID BEAUFTRAGTE zugewiesen und deren KONTAKTADRESSE gespeichert. Die Beauftragten sind zuständig für die jeweiligen Institute und müssen in der Benutzertabelle abgelegt sein. Sie werden über die ID des Benutzers zugeordnet und per Fremdschlüssel-Referenz vor Inkonsistenzen geschützt.

4.8 UID- und GID-Tabelle:

Jedem Institut sind eindeutige UID- bzw. GID Bereiche zugeordet, die nicht notwendigerweise zusammenhängend sind. UIDs oder auch User IDs bzw. GIDs (Group IDs) sind für das Anlegen eines Unix-Accounts notwendig, da jedes Unix-Betriebssystem eine eindeutige User ID und eine Group ID verlangt.

Über die Felder VON und BIS werden die Bereiche eingegrenzt und über das INSTITUT-Feld eindeutig dem entsprechendem Institut zugeordnet. Weiterhin hat jeder Uid - bzw Gid-Bereich eine eindeutige ID, die ohne weitere Bedeutung ist und lediglich die Identifikation eines ausgewählten Bereichs erleichtert.

[2] Die Administratoren werden für ganze Institute authorisiert

4.9 Mailalias-Tabelle:

Hier werden alle Daten, die für Mailforwards bzw. Alias-Namen relevant sind abgelegt. Für das Mailsystem ist neben dem ALIAS-Namen selbst noch die Zieladresse notwendig. Diese wird in dem Feld ALIASAUF abgelegt. Dieses Feld kann leider nicht über Fremdschlüsselreferenzen abgesichert werden, da die Zieladresse auch außerhalb der Domäne *.ira.uka.de* liegen kann. Beispielsweise kann ein Forward eingerichtet werden, der *knierim@ira.uka.de* auf *knierim@web.de* weiterleitet.

Zusätzlich gibt es noch ein ID-Feld, das per Fremdschlüsselreferenz auf die ID in der Benutzer-Tabelle die Zuordnung zu eienem existierendem Benutzerdatensatz herstellt und ein PUBLIZIEREINLDAP-Feld, in dem angegeben wird, ob das Alias in den LDAP-Server aufgenommen werden soll. Dieses PUBLIZIEREINLDAP-Feld ist analog zum gleichnamigen Feld in der UNIXACCOUNT-Relation mit dem Standardwert *false* belegt.

Um die Alias-Namen vor unbefugter Modifikation über die Front-Ends sicherzustellen, ist noch ein INSTITUTS-Feld erforderlich.

Für statistische Zwecke existieren die Felder LETZTEAENDERUNGAM und LETZTEAENDERUNGVON.

4.10 Mailinglisten-Tabelle:

Die relevanten Daten zur Verwaltung von Mailinglisten werden in dieser Tabelle abgelegt. Zu diesen Daten gehören neben dem LISTENNAMEN eine kurze BESCHREIBUNG der Liste und die KONTAKTADRESSE des Betreuers der Mailingliste. Zusätzlich wird der Liste noch ein Benutzer über die ID zugeordnet. Aus statistischen Gründen sind ebenfalls die Felder LETZTEAENDERUNGAM und LETZTEAENDERUNGVON vorhanden.

Auf Daten für den Zugriffsschutz kann hier verzichtet werden, da Modifikationen der Daten über die Web- und ASCII-Schnittstellen nicht vorgesehen sind.

4.11 VergebeneAdressen-Tabelle:

Diese Tabelle ist eine reine Status-Tabelle, die nur zur Sicherstellung eindeutiger Adressen (d.h. Account-,Alias- und Listennamen) innerhalb der Domäne *.ira.uka.de* verwendet wird. Aufgrund der Aufspaltung in Mailinglisten-, Alias- und Unix-Account-Tabellen ist es unmöglich über *unique*-Attribute innerhalb der einzelnen Tabellen eine Eindeutigkeit zu erzwingen. Da postgresql sog. *Rules* anbietet, wird die Eindeutigkeit über fogenden Trick realisiert:

In der VergebeneAdressen-Tabelle ist das Feld ADRESSE *primärer Schlüssel* und somit eindeutig.

Wird eine *update*-, *insert*- oder *delete*-Aktion auf eine der drei oben genannten Tabellen ausgeführt, wird aufgrund erzeugter *Rules* automatisch selbige Aktion mit der gleichlautenden ADRESSE in dieser Relation durchgeführt und die ADRESSART automatisch abgelegt. Möchte nun jemand beispielsweise einen Unix-Account mit dem Login knierim anlegen, obwohl bereits eine Liste mit dem Namen *knierim* existiert, wird dies von der Datenbank automatisch abgelehnt

und eine Fehlermeldung ausgeworfen. Wichtig ist die Realisierung des gegenseitigen Auschlusses vor allem für das Mailsystem, da es zu Problemen mit der Zustellung von Mail beim Eintreten dieset Situation kommen würde.

Sollten bei evtl. Erweiterungen weitere Account- oder andere Adreß-Tabellen für das Mailsystem angelegt werden, ist eine Ausweitung dieses Systems kein Problem, da nur die entsprechenden *Rules* eingefügt werden müssen.

Natürlich können ebenfalls reservierte Adressen wie root vor der Verwendung als Liste, fakultätsweitem Unixaccount oder Alias geschützt werden, indem sie in die VergebeneAdressen-Tabelle per Hand eingetragen werden.

4.12 Blacklist-Tabelle:

Seit dieser Version des Useradm gibt es eine schwarze Liste, auf die Benutzer gesetzt werden können, die aus verschiedenen Gründen (z.B. Hacking oder Betreiben illegaler Dienste) keinen Account oder Alias bekommen dürfen. Die Daten der Benutzer werden einschließlich einer BEMERKUNG und optional der MATRIKEL-Nummer in der Blacklist-Tabelle abgelegt. Es wird ebenfalls der Login des Admins im Feld EINGETRAGENVON inklusive dem Datum im Feld EINGETRAGENAM abgelegt. Neben der Statistik realisiert das EINGETRAGENVON-Feld den Zugriffsschutzmechanismus, da in den Useradm-Schnittstellen nur vom Admin erzeugte Datensätze auch wieder vom Admin gelöscht werden dürfen. Ist ein Benutzer in dieser Relation eingetragen, so werden diese Informationen verwendet, um die Administratoren zu warenen, falls sie einen für den eingetragenen Benutzer ein Alias oder Unix-Account bearbeiten wollen.

4.13 Log-Tabelle:

In der Log-Tabelle werden alle Veränderungen in den Unixaccounts sowie den Mailaliasdatenbeständen und den Mailinglistentabellen automatisch mitprotokolliert. Neben dem TABELLENNAMEN, der BEARBEITETEN ADRESSE und der AKTION, die entweder *update*, *insert* oder *delete* sein kann, wird in BEARBEITETVON der Front-End-Benutzer inklusive eines Zeitstempels gespeichert. Zusätzlich existiert ein Bemerkungsfeld, in das bei *updates* die alte Adresse abgelegt wird, und das Feld IMLDAPSERVER, das für die Replikation der Daten in den LDAP-Server erforderlich ist, da hiermit festgestellt werden kann [3], welche Daten im LDAP-Dienst noch nicht aktualisiert wurden.

4.14 Benutzeridentifikationstabellen:

Unter dieser Gruppe ist die dbaccess-Tabelle, in der LOGIN des Datenbankbenutzers, sein zugewiesenes verschlüsseltes Paßwort und das Datum des letzten Zugriffs gespeichert wird. Die Daten dieser Tabelle sind für die Authorisation des Benutzers an den Front-Ends erforderlich.

Da im CGI-Front-End ein eingeloggter Benutzer nach 10 Minuten aktionsloser Zeit automatisch abgemeldet wird und außerdem das Paßwort nach einer erfolgreichen Anmeldung durch eine eindeutige Sessionid zugeordnet wird,

[3] indem die Daten, die jünger als die letzte Aktualisierung sind, ausgewertet werden

müssen auch diese Daten in der Datenbank abgelegt werden. SESSION-Id, der LOGIN des angemeldeten Benutzers und das VERFALLSDATUM wird daher in der Tabelle Session abgelegt.

Abschließend sind noch Rechte für die Benutzer der Front-Ends zu vergeben. Die Schreibrechte auf die Daten des Useradm werden auf Institutsebene vergeben. In der Tabelle Zugriffskontrolle werden LOGIN des Datenbankbenutzers und die entsprechende INSTITUT-ID, aus der Institutstabelle für die entsprechenden Institute eintragen. Selbstverständlich existiert eine Fremdschlüsselreferenz vom INSTITUT-Feld auf die INSTITUTSID der der Institutstabelle.

Es existieren Fremdschlüsselreferenzen zwischen der dbaccess-Tabelle und der Unixaccount über das LOGIN-Feld und von der Sessiontabelle und der ZUGRIFFSKONTROLLE-Tabelle auf das LOGIN-Feld der dbaccess-Tabelle.

Kapitel 5

Architektur der Benutzerschnittstellen

5.1 Überblick:

Ziel bei der Implementierung der Benutzerschnittstellen war neben der Effizienz vor allem die Realisierung eines modularen Aufbaus um spätere Erweiterungen bzw. Anpassungen zu vereinfachen. Prinzipiell läßt sich der Aufbau der implementierten Front-Ends als Schichtenarchitektur mit 4 Ebenen gliedern:

- Die unterste Schicht bildet hierbei die Datenbasis selbt. Im folgenden wird diese als *Datenverwaltungsschicht* bezeichnet.

- Die zweite Schicht könnte man als *Vermittlungsschicht* zwischen Datenbasis und Applikation bezeichnen.

  Hier findet man die verschiedenen Perl-Module, die die Datenbankoperationen standardisieren und je nach mitgegebenen Parametern die Abfragekommandos ensprechend anpassen. Hier werden prinzipiell alle Abfrage-, Einfüge, Lösch- und Aktualisierungsoperationen generiert und ausgeführt. Die Ergebnisse werden dann über Hashes, Felder oder einfache Strings an die dritte Schicht weiter gegeben.

- Die *Darstellungs-* oder 3. Schicht besteht aus mehreren Modulen, die für die einzelnen Dienste des Useradm Zugriffsschutz realisieren und Formulare als Ausgabe liefern. Weiterhin werden hier die Ergebnisse der Funktionen aus der zweiten Schicht für die Ausgabe formatiert bzw. selektiert und bei Modifikationen bzw. der Neuanlage von Datensätzen die Konsistenz der Daten überprüft, um eine durch eventuelle Fehlbedienungen des Benutzers hervorgerufene Fehlermeldung der Datenbank nicht in einem Absturz der Applikation enden zu lassen.
  Die für die ASCII-Schnittstelle relevanten Module enden auf `ASCII.pm`, die für die Web-Schnittstelle relvanten Module enden auf `Panel.pm`.

- Als *Applikationsschicht* kann man die eigentlichen Useradm-Programme sehen. Diese Programme werden vorrangig zur Auswahl der einzelnen Unterfunktionen und zur Anmeldung an der Datenbank verwendet.

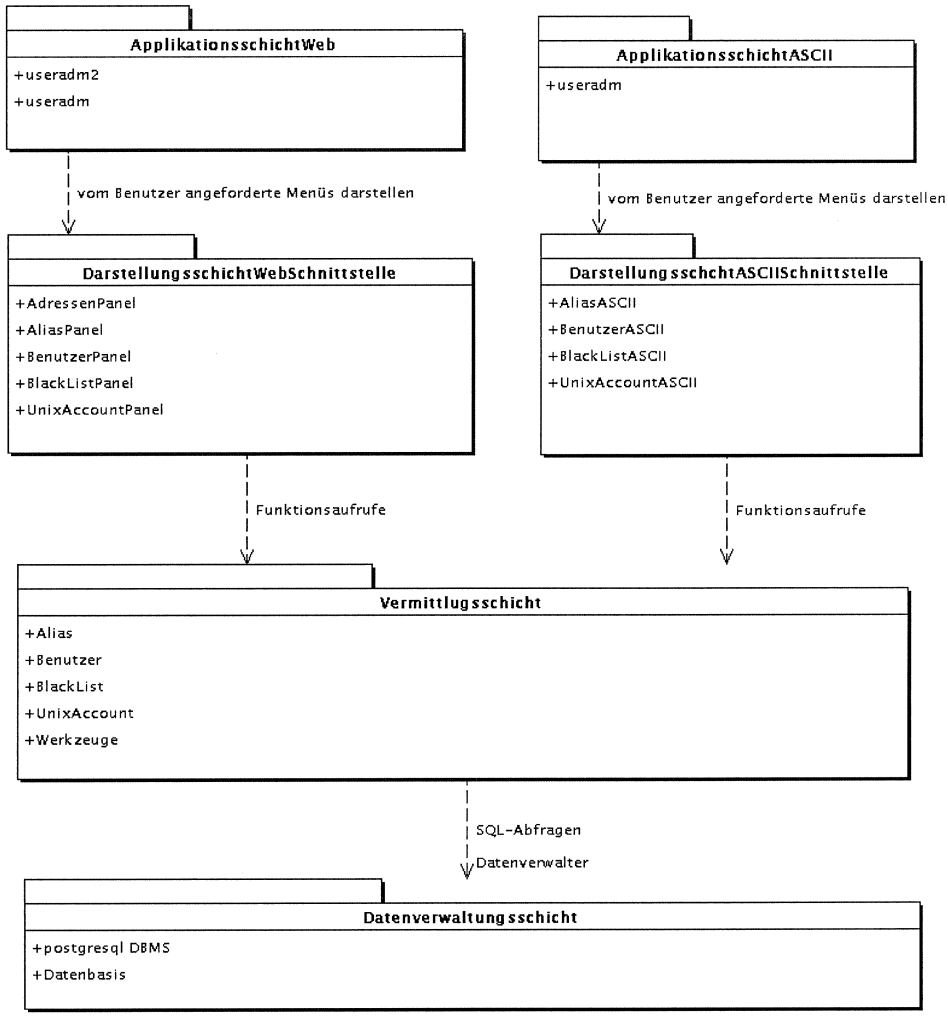

Abbildung 5.1: Systemarchitektur der Benutzerschnittstellen

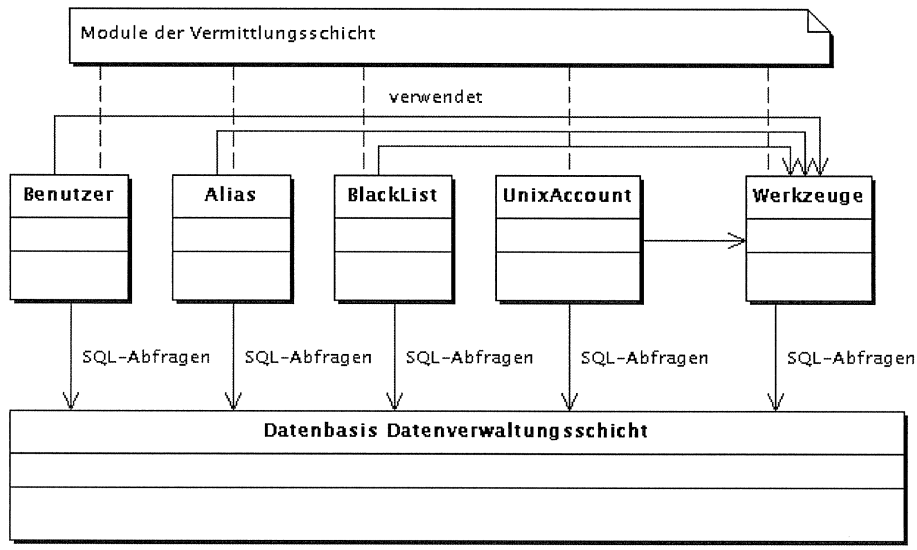

Abbildung 5.2: Modulstruktur der Vermittlungssschicht

Bei der Web-Schnittstelle ersetzt das Programm `useradm2.pl` außerdem die übliche Benutzerauthenifikation über `htaccess` [1]. Die ASCII-Schnittstelle `useradm.pl` wird dem Benutzer als Shell zugeteilt. Daher ist hier noch eine Funktion zum Ändern des Paßworts implementiert. Auf eine Authentifikation kann verzichtet werden, da die übliche Unix-Anmeldung ausreichend ist.

5.2 Dokumentation der Module:

Hier werden alle Module des Useradm dokumentiert. Besonders ausführlich werden die Module der zweiten Schicht beschrieben, da diese bei Neuentwicklungen oder Aktualisierungen der Benutzerschnittstelle vornehmlich verwendet werden können.

5.2.1 Module der Vermittlungsschicht:

Wie bereits oben erklärt wurde, bilden diese Module eine Funktionssammlung, die Kommandos auf die Datenbank abhänging der entsprechenden Funktion übergebenen Parameter formulieren und ausführen. Die Module sind nach den Tabellen benannt, auf welchen sie Aktionen ausführen.
Auf Zugriffsschutzmechanismen wurde an dieser Stelle verzichtet um die Wiederverwendbarkeit der geschriebenen Funktionen nicht einzuschränken. Diese Mechanismen sind nur für die Benutzerschnittstellen relevant und werden somit in der dritten Schicht implementiert.

[1] Standard-Authentifizierung am Web-Server

Eine Sonderrolle unter den Modulen nimmt `Werkzeuge.pm` ein. Hier sind Funktionen untergebracht, die von allen Modulen des Useradm bennötigt werden. Eine genauere Dokumentation des Moduls findet man weiter unten im Text.

Benutzer.pm

Dieses Modul beinhaltet folgende externe Schnittstellen:

- `sucheBenutzer($name, $vorname, $dbh):`

  Diese Funktion ermöglicht die Suche von Benutzern, die mit `$name` und `$vorname` anfangen. Beide Parameter können leer sein. Allerdings werden dann aus Speicherplatzgründen nur 100 Datensätze zurück geben. Notwendig ist diese Begrenzung, da das Ergebnis als Feld zurückgegeben wird und somit bei einer großen Anzahl von Datensätzen der Speicherbedarf enorm groß ist. Wird kein Datensatz gefunden, so wird der String `leer` als Ergebnis geliefert. Ansonsten werden die Datensätze in einem Feld zurückgegeben, die dann in der darüber liegenden Schicht verarbeitet werden müssen.

  Als Suchwerte können Teilworte der gesuchten Werte oder auch leere Strings als Belegung mitgegeben werden.

  Verwendung findet diese Funktion bei den Abfragemasken in denen der Benutzer des Useradm eine Person mit Namen und Vornamen sucht und bei der Neuanlage von Unixaccounts und Alias-Namen, da hier nur bei Bedarf ein neuer Benutzerdatensatz angelegt werden muß.

- `gibBenutzer($id, $dbh):`
  Hier wird der Eintrag der Benutzertabelle zu einer angegebenen `id` in einem Hash zurückgeliefert. Um bei der Ausgabe von Unixaccounts und Alias-Namen innerhalb der Benutzerschnittstellen die Verbindung zum Besitzer herstellen zu können, ist diese Funktion erforderlich. Wird kein Benutzer mit der angegebenen `id` gefunden, wird ein leeres Hash geliefert.

- `gibBenutzerId($name, $vorname, $dbh):`
  Diese Methode liefert bei Existenz eines Benutzers seine eindeutige `id`. Existiert kein Benutzer mit `$name`, `$vorname` wird -1 zuruückgegeben. Somit kann innerhalb der Benutzerschnittstellen leicht festgestellt werden, ob ein Benutzer exisitiert oder ob er neu angelegt werden muß. Relevant ist diese Funktionalität bei der Neueinrichtung von Accounts und Alias-Namen.

- `aendereBenutzer($id, $name, $vorname, $titel, $tel, $matrikel, $login, $dbh):`
  Hier ist der Name selbsterklärend. Die Variable `$id` bildet den Schlüsselwert, `$login` ist der Login des Schnittstellenbenutzers.
  `$name`, `$vorname`, `$login` und `$titel` müssen definierte/gültige (vgl. Tabellendokumentation im vorherigen Kapitel) Werte besitzen, die restlichen Werte sind optional. `aendereBenutzer` beinhaltet allerdings keine Funktionalität, die diese Bedinungen überprüft.

- `loescheBenutzer($id, $dbh)`: Hier wird der Benutzer mit der entsprechenden Id gelöscht. Weiterhin werden dank eingerichteter *Rules* innerhalb der Datenbank automatisch auch alle zugeordneten Unixaccounts und Alias-Namen gelöscht.

- `setzeBenutzer($name, $vorname, $titel, $tel, $matrikel, $login, $dbh)`:
  Hiermit werden die Benutzerdaten gesetzt. Name, Vorname, `$Titel` und Login müssen definiert sein. Die übrigen Variablen können undefinierte Werte besitzen. Diese Funktion gibt als Ergebnis die zugeordnete Id zurück.

  Der Unterschied zwischen dieser Funktion und AENDEREBENUTZER besteht darin, daß mit SETZEBENUTZER neue Datensätze in der Relation Benutzer anlegt, also *insert*-Operationen ausführt und AENDEREBENUTZER bestehende Datensätze anpaßt, also *update*-Operationen auf der Relation Benutzer ausführt.

- `gibBenutzerZumNamen($name, $vorname, $dbh)`:
  Diese Methode wird nur für den Menüpunkt *Benutzer verändern* innerhalb der ASCII-Schnittstelle verwendet, da - im Gegensatz zur Webschnittstelle - die Verwendung der Benutzer-Id als Benutzereingabe nicht praktikabel für den ASCII-Schnittstellen-Benutzer ist. Die vom Schnittstellen-Benutzer anzugebenden Parameter `$name` und `$vorname` sind deutlich praktikabler und können aufgrund dieser Methode direkt als Suchschlüssel fr einen Benutzerdatensatz verwendet werden.
  `gibBenutzerZumNamen()` gibt daher das gleiche Hash wie `gibBenutzer()` zurück.

UnixAccount.pm

In diesem Modul befinden sich ausschließlich Funktionen, die für das Anlegen neuer Unix-Benutzerkonten relevant sind. Neben Zugriffen auf die Unixaccount-Tabelle werden u.a. in den Funktionen dieses Moduls lesende Zugriffe auf die UID, GID, Institut und MailAuslieferung ausgeführt. Diese Zugriffe sind notwendig, da in der UnixAccount-Tabelle lediglich die Auslieferungsgruppe abgelegt ist und die UID und GID-Bereiche auf Institutsebene vergeben werden. Weiterhin ist noch ein Zugriff auf die Klassentabelle erforderlich, da jeder Unixaccount in eine Klasse eingeordnet wird.

- `gibKlassen($dbh)`:
  Gibt alle existierenden Klassen in einem Hash zurück, wobei das Schlüsselattribut des Hashes der KlassenId der Klassentabelle entspricht.

- `gibAuslieferung($dbh)`:
  Hier wird ein Hash mit allen Auslieferungsgruppen zurückgegeben. Schlüssel ist die Id der Auslieferungsgruppe. Der Wert des Hashs ist der entsprechende Name. Verwendung findet diese Funktion u.a. beim Suchen von Unixaccounts.

- `sucheAccounts($account, $datum, $klasse, $auslieferung, $dbh)`:
  Mit dieser Methode kann die Unixaccount-Tabelle nach bestimmten Krite-

rien durchsucht werden. `$account` kann hierbei ein Accountname, ein Teilwort eines Accounts oder leer sein. `$datum`, `$klasse` und `$auslieferung` sind ebenfalls optional, da die Funktion entsprechend der mitgegebenen Parameterwerte die Abfrage formt. Das Ergebnis wird als Feld zurückgegeben; allerdings ist - wie bei `sucheBenutzer()` - die maximale Anzahl der Datensätze auf 150 beschränkt. Grund hierfür ist die Reduzierung des Speicherbedarfs.

- `zeigeUnixAccountsZurId($id, $dbh)`:
  Hier werden alle Accounts als Feld zurückgegeben, die die gleiche Id, also den gleichen Besitzer haben. Verwendung innerhalb der Benutzerschnittstellen findet diese Funktion u.a. bei dem Modus *Benutzer suchen.*

- `gibAccount($uid, $dbh)`:
  Diese Funktion gibt den Accountdatensatz einer UID als Feld zurück. Verwendet wird diese Funktion bei der Web-Schnittstelle, da hier die UID u.a als Schlüsselparameter bei Links auf die Modi *Account verändern* und *Account löschen* verwendet wird.

- `gibGidBereich($institut, $dbh)`:
  Hier werden die nicht zusammenhängenden GroupId-Bereiche eines Institutes als Hash zurückgegeben. Schlüsselattribut des Hashes ist hier die Id der GID-Tabelle. Diese Funktion wird u.a. bei den Modi *Account verändern* und *Account neu anlegen* aufgerufen.

- `gibUidBereich($institut, $dbh)`:
  Analog zur obigen Funktion mit dem Unterschied, daß hier die UID-Bereiche zurückgegeben werden.

- `modifiziereAccount($adresse, $gid, $instlist, $klasse, $auslieferung ,$datum, $bemerkung, $dbBenutzer, $uid, $neueUid, $dbh)`:
  Methode zum Ändern eines Accounts. Schlüsselwert für die Änderung ist hierbei die Variable `$uid`. Die weiteren Variablen können neue Werte zugewiesen bekommen. Allerdings sind die Werte von `$adresse` und `$neueUid` zu überprüfen und die Klassen, Auslieferungsbelegungen entsprechend der Tabellenschlüssel zu wählen. `$dbBenutzer` ist der derzeit angemeldete Benutzer des Useradm. Der Wert dieser Variable wird somit in das Feld `letzeAenderungVon` in der Unixaccount-Tabelle geschrieben.

- `loescheAccount($accountName, $dbh)`:
  Die Aufgabe dieser Funktion ist selbsterklärend. Neben dem Account selbst werden dank der *rules* von postgresql allerdings auch alle zugehörigen Mailalias-Namen entfernt.

- `istDbBenutzer($account, $dbh)`:
  Hier wird nachgeschaut, ob der Account Zugriff auf den Useradm hat. Als Ergebnis wird *ja* oder *nein* geliefert. Diese Abfrage ist notwendig, da Benutzer-Accounts, die Zugriff auf die Schnittstellen haben weder über die Schnittstellen gelöscht noch deren Login-Name verändert werden darf.

- `gibAccountdaten($accountName, $dbh)`:
  Leistet gleiches wie `gibAccount()`. Einziger Unterschied ist der Schlüsselwert, da hier der Accountname verwendet wird. Diese Funktion ist notwendig um die Handhabung der ASCII-Schnittstelle zu vereinfachen. Würde man `gibAccount()` in der ASCII-Schnittstelle verwenden, so müßte man den durch den Benutzer eingegebenen Accountnamen in eine Id umrechnen.

- `gibInstitutZurAuslieferung($auslieferung, $dbh)`:
  Hier wird die InstitutsId zu einer Auslieferungsgruppe zurückgegeben. Neben der Benutzung dieser Funktion innerhalb des Moduls wird diese Funktion auch benötigt um innerhalb des Useradm Zugriffsrechte zu überprüfen.

- `gibAuslieferungZumInstitut($institut, $dbh)`:
  Diese Funktion übergibt alle zu einem Institut gehörenden Mailauslieferungsgruppen in einem Hash. Dieses Hash hat als Schlüssel die Id der Auslieferungsgruppe und als Wert deren Namen. Erforderlich ist diese Funktion um beim Neuanlegen und dem Verändern der Accounts sicher zu stellen, daß der Schnittstellenbenutzer nur die dem Institut zugeteilten Auslieferungsgruppen verwendet, da über die Institute auch der Zugriffsschutz realisiert wird.

- `berechneUid($uidBereich, $dbh)`:
  Mit Hilfe von `berechneUId()` wird innerhalb eines UID-Bereichs, dessen Id als Parameter der Funktion mitgegeben wird, die nächste freie UID berechnet und ausgegeben. Falls der Bereich voll sein sollte, wird *-1* als Ergebnis geliefert. Verwendung findet diese Methode bei der Neuanlanlage von Unixaccounts.

- `checkUid($uid, $auslieferung, $dbh)`:
  Diese Funktion existiert als Gegenstück zu `berechneUid()` . Sie überprüft, ob eine angegebene UID noch frei ist, bzw. ob diese in den Bereich des zur Auslieferungsgruppe gehörenden Instituts paßt. Ist `checkUid()` erfolgreich, wird *1* als Ergebnis geliefert, falls nicht, *-1*. `checkUid()` wird ausschließlich beim Modus *Account verändern* verwendet, da es hier möglich ist, eine UID per Hand zu verändern.

- `setzeAccount($id, $uid, $gid, $login, $klasse, $auslieferung, $instList, $datum, $bemerkung, $dbBenutzer, $dbh)`:
  Mittels dieser Methode wird ein Account im Useradm neu eingerichtet. `$id` verlangt als Wert die Id des dazugehörigen Besitzers, `$dbBenutzer` den Loginnamen der Person die den Account einrichtet. Die Variablen `$klasse` und `$auslieferung` bennötigen gültige Schlüssel. `$login` und `$uid` müssen zudem eindeutige Werte besitzen. Die restlichen Variablenbelegungen sind prinzipell beliebig.

Alias.pm

Dieses Modul beeinhaltet die folgenden Funktionen:

- `sucheAliasse($aliasName, $dbh)`:
  Diese Funktion liefert ein Feld mit allen Datensätzen, die entweder mit dem Wert von `$aliasName` anfangen oder ihn enthalten. Allerdings wird analog zu den Suchfunktionen der anderen Module nur eine begrenzte Anzahl an Datensätzen geliefert.

- `gibAliasZurId($id, $dbh)`:
  Alle Datensätze mit gleicher Id - also mit gleichem Besitzer - werden als Feld zurückgegeben. Diese Funktion wird für beide Schnittstellen verwendet, um bei Ausgabe der Benutzerdaten gleichzeitig die entsprechenden Aliasnamen anzeigen zu können.

- `gibAliasseZumAccount($login, $dbh)`:
  Leistet das gleiche wie `gibAliasseZurId()`. Einziger Unterschied ist hierbei, daß der Suchschlüssel gleich der Spalte *aliasAuf* in der Tabelle Mailalias sein mußMan kann diese Funktion allerdings nicht als vollwertigen Ersatz zu `gibAliasseZurId()` sehen, da eventuell angelegte Mailforwards, die ebenfalls in der Alias-Tabelle stehen, nicht gefunden werden.

- `gibAliasDaten($aliasName, $dbh)`:
  Rückgabedatum dieser Funktion ist ein Feld, in dem alle Daten abgelegt sind, die zu dem Alias mit dem Wert von `$aliasName` gehören. Innerhalb der Schnittstellen wird vor dem Verändern der Alias-Namen oder dem Löschvorgang auf diese Funktion zugegriffen, um vor dem eigentlichen Änderungs- bzw. Löschvorgang noch einmal die Daten anzeigen zu lassen.

- `modifiziereAlias($alterName, $institut $neuerName, $zielAdresse, $dbBenutzer, $dbh)`:
  Mit dieser Funktion ist es möglich, den Aliasnamen zu ändern und es einem anderen Institut zuzuorden, falls der Benutzer des Useradm dazu authorisiert ist. Da der Wert des Institutes bei den Alias-Namen für Zugriffsschutzmechanismen verwendet wird und der Aliasbesitzer bei mehreren Instituten einen Account haben kann, kann es erforderlich sein, daß die Institutszugehörigkeit und die Zieladresse angepaßt werden müssen.

- `loescheAlias($alias, $dbh)`:
  Löscht das entsprechende Alias. Allerdings ohne Zugriffsschutzmechanismen.

- `setzeAlias($id, $inst, $aliasName, $zielAdresse, $dbBenutzer, $dbh)`:
  Setzt ein Alias. Die `$id` muß den Wert einen gültigen Benutzerschlüssel haben. `$inst` verlangt als Wert den Schlüssel eines gültigen Institus. Die weiteren Parameter sind selbsterklärend.

Blacklist.pm

Hier sind alle Funktionen abgelegt, die es ermöglichen einen Benutzer auf die schwarze Liste zu setzen.
Folgende Schnittstellen hat BlackList.pm:

- `suenderAufnehmen($dbBenutzer, $name, $vorname, $matrikel, $bemerkung, $dbh)`:
  Mit dieser Funktion wird ein Benutzer vom `$dbBenutzer`, also dem Benutzer des Useradm, in die schwarze Liste eingetragen. Die Werte von `$name`,`$vorname` und `$bemerkung` müssen gültige Werte besitzen. Die Matrikelnummer kann allerdings auch leer sein.

- `suenderAusgeben($dbh)`:
  Gibt alle Datensätze der auf der schwarzen Liste stehenden Benutzer in einem Feld zurück. Verwendung findet diese Methode lediglich bei der Ausgabe der auf der Liste stehenden Benutzer.

- `suenderLoeschen($name, $vorname, $dbBenutzer, $dbh)`:
  Hier wird ein auf der schwarzen Liste stehender Benutzer gelöscht, falls er die Werte von `$name` und `$vorname` als Name hat und von dem Benutzer der Schnittstelle eigetragen wurde. Der Benutzername des Schnittstellenbenutzers wird `$dbBenutzer` mitgegeben.

- `benutzerAufSchwarzerListe($name, $vorname, $dbh)`:
  Hier wird überprüft, ob ein Benutzer auf der schwarzen Liste ist. Falls er darin eigetragen ist, wird sein Name zurückgegeben, falls nicht, wird ein undefinierter String geliefert. Bei den Schnittstellen wird diese Funktion verwendet um Warnmeldungen bei Änderungen von betreffenden Benutzerdaten oder der Neuanlage von Accounts oder Alias-Namen für auf der Liste stehenden Benutzer auszugeben.

Werkzeuge.pm

Wie bereits erwähnt nimmt dieses Modul eine Sonderstellung ein. Hier sind neben den LDAP-Methoden noch Funktionen, die in allen Modulen benötigt werden. Zudem werden hier Funktionen angeboten, die es ermöglichen den Wechsel der Datenbank oder des Datenbank-Benutzers an einer zentralen Stelle durchzuführen.
Folgende Funktionen sind hier zu finden:

- `fixString($string)`:
  Entfernt am Ende des Strings das return-Zeichen, wandelt Umlaute in Ae, Ue usw. um und gibt den modifizierten String wieder zurück.

- `gibDbString(), gibDbBenutzer()`:
  gibDbString() gibt den String zurück, in dem steht, wo die Datenbank zu finden ist. gibDbBenutzer liefert den Datenbankbenutzer. Da sämtliche Aktionen, die die Datenbank betreffen, über einen einzigen Benutzer laufen und die Zugriffsschutzmechanismen in den Schnittstellen sitzen, ist es sinnvoll auf diese Funktionen zuzugreifen, damit bei evtl. Umzug der Datenbank oder der Änderung des Benutzers nur hier die Modifikation vorgenommen werden müssen.

- `dbAccess($statement, $dbh)`:
  Ebenfalls eine Werkzeugmethode. Vereinfacht die Ausführung von sql-Kommandos, da man dank dieser Funktion auf permanentes Erzeugen

des Statement-Handles[2] verzichten kann.
Praktisch ist diese Funktion in erster Linie für Statements, die keine Ergebnisse liefern, also keine Abfragen sind.

- `gibDefaultButtons($login, $sessionId, $action)`:
  Diese Funktion wird ausschließlich für die Web-Schnittstelle verwendet. Als Ausgabe liefert diese Funktion drei *hidden*-Komponenten, die die Belegungen von $login, $sessionId und $action ausgeben. Notwendig sind diese Ausgaben, da bei jeder Aktion innerhalb der Web-Schnittstelle diese Parameter mitgegeben werden müssen. `$login` und `$sessionId` sind hierbei für die Authentifikation des Benutzers erforderlich.

- `adressSuche($adresse, $dbh)`:
  Gibt in einem Feld alle Adressen (Accounts, Mailinglisten und Alias-Namen) die mit dem Wert von `$adresse` anfangen in einem Feld zurück. Ein Adreßdatensatz enthält neben dem Adressennamen den Typ der Adresse und den LDAP-Status.

- `gibTitelHash($dbh)`:
  Gibt sämtliche Titel in einem Hash zurück, dessen Schlüssel einem Benutzer zugewiesen werden können. Schlüssel ist hierbei die *id* des Titels; Wert ist ist die Titelbezeichnung.

- `adressenStatus($adresse, $dbh)`:
  Falls ein Adressenname bereits vergeben ist, wird die Adresse wieder zurückgegeben. Falls sie noch frei ist, wird ein undefinierter String geliefert. Verwendung findet diese Funktion bei der Neuanlage oder dem Ändern von Accounts bzw. Alias-Namen.

- `holeLdapStatus($adresse, $dbh)`:
  Gibt den LDAP-Status der angegebenen Adresse. *0* bedeutet, daß die Adresse nicht im LDAP-Server veröffentlicht werden soll, *1* zeigt an, daß veröffentlicht werden soll.

- `setzeLdap($adresse, $status, $dbh)`:
  Methode zum Ändern des LDAP-Status einer Adresse. Die Belegung von `$status` entspricht der Rückgabe von `holeLdapStatus()`. Standardmässig wird die Veröffentlichung neu eigerrichteter Accounts und Alias-Namen auf *nicht veröffentlichen* gesetzt. Für Listen ist keine Aufnahme in den LDAP-Server vorgesehen.

- `gibInstitutsHash($dbh)`:
  Liefert ein Hash, in dem alle Institute sind. Schlüssel ist die *institutsId*, Wert der Name des Instituts.

- `gibAuslieferungsHashZumLogin($login, $dbh)` und `gibInstiutsHashZumLogin($login, $dbh)`:
  Gibt alle Auslieferungsgruppen bzw. Institute für die der Benutzer der Datenbank authorisiert ist als Hash zurück. Diese beiden Funktionen werden ausschließlich zur Realisierung des Zugriffsschutzes verwendet.

[2] Die Erzeugung eines Statement-Handles ist bei der Verwendung von Perl-DBI erforderlich.

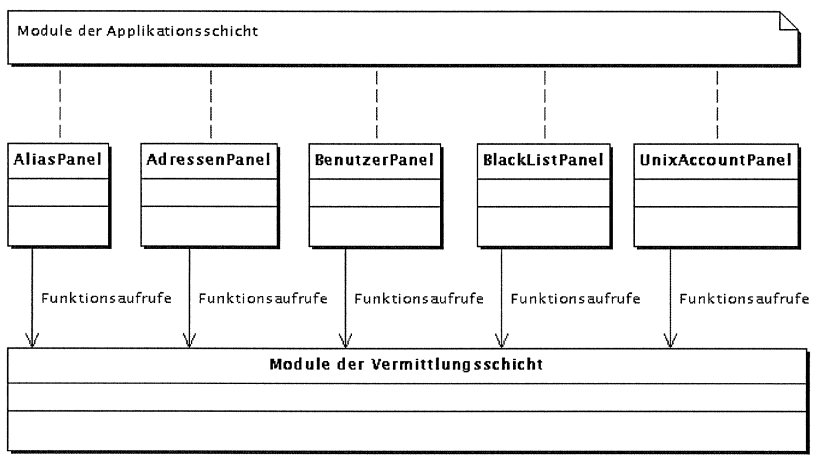

Abbildung 5.3: Module der Darstellungsschicht für die Webschnittstelle

- `institutsAnzahlZumLogin($login, $dbh)`:
  Liefert die Anzahl der Institute, für die ein Benutzer der Schnittstellen authorisiert wird. Notwendig ist diese Funktion, damit ein Benutzer der ASCII-Schnittstelle bei Berechtigungen für mehrere Institute das Institut, für das er Accounts oder Alias-Namen einrichten möchte, auswählen kann. Bei der Web-Schnittstelle ist diese Funktionalität nicht erforderlich, da dort einfach Hash-Referenzen einem *popup_menu* übergeben werden können.

5.2.2 Die Module der Darstellungsschicht:

Prinzipiell ist die Modulaufteilung so gewählt, daß für jeden Modus der Benutzerschnittstellen ein Modul existiert, welches die Ein-/Ausgabe-Masken beeinhaltet. Hier sind Zugriffsschutzmechanismen implementiert.

Module der Web-Schnittstelle:

Für die Web-Schnittstelle existieren die folgenden Module:

- `AdressenPanel.pm`:
  Hier wird lediglich eine Suchmaske mit Ergebnisanzeige generiert, die zur Adressensuche verwendet wird.
  Modulschnittstelle ist hierbei `adressSuchPanel($login, $sessionId, $dbh)`.

- `AliasPanel.pm`:
  Hier wird das Panel erzeugt, das die Unterfunktionen *Alias anlegen*, *Alias löschen* und *Alias suchen* beeinhaltet.
  Schnittstellen des Moduls sind die Funktionen `aliasPanel($login,`

`$sessionId, $dbh)` und `aliasseZurIdPanel($id, $dbh)`.
Das `aliasPanel()` ist hierbei als Schnittstelle für die Unterfunktionen im Alias-Bereich zu sehen. `aliasseZurIdPanel()` wird nur exportiert, damit bei der Benutzersuche - welche in `Benutzer.pm` implementiert ist - gleichzeitig zum Benutzer gehörende Alias-Namen ausgegeben werden können.

- `BenutzerPanel.pm`:
  Dieses Modul beeinhaltet sämtliche Funktionen, die die Benutzerschnittstellen zum Benutzer suchen, anlegen und löschen realisieren.
  Externe Schnittstellen sind hier `benutzerPanel($login, $sessionId, $dbh)`, `gibBenutzerDaten($id, $dbh)` und `benutzerAnlegePanel($name, $vorname, $dbh)`. Das `benutzerPanel()` beeinhaltet neben dem Löschen von Benutzern auch die Möglichkeit Benutzerdaten zu modifizieren und nach Benutzern zu suchen.
  `gibBenutzerDaten($id, $dbh)` ist exportiert, damit u.a. bei der Suche von UnixAccounts und Alias-Namen gleichzeitig der dazugehörige Benutzerdatensatz mit ausgegeben wird.
  Da die Neuanlage von Benutzern ausschließlich innerhalb der Neuanlage von Accounts und Alias-Namen möglich ist, wird die Funktion `benutzerAnlegePanel()` exportiert. Damit ist ein einheitliches Formular für die Neuanlage von Benutzern geschaffen.

- `BlackListPanel.pm`:
  Schnittstelle dieses Moduls ist die Funktion `blackListPanel($login, $sessionId, $dbh)`. Neben dem Ausgabeformular für Benutzer, die auf der schwarzen Liste stehen, werden Panele erzeugt, die es ermöglichen Benutzer auf die schwarze Liste zu setzen und sie wieder zu löschen.

- `UnixAccountPanel.pm`:
  `accountPanel($login, $sessionId, $dbh)` beinhaltet die folgenden Formularaufrufe: *Account neu anlegen*, *Account löschen* und *Account suchen.* Zusätzlich wird noch die Funktion `idAccountPanel($id, $dbh)` exportiert. Sie wird u.a. bei der Suche von Benutzern verwendet.

Weitere Funktionalität der Web-Schnittstelle kann relativ einfach durch weitere Panel-Module erreicht werden. Insofern sollten leichte Erweiterungen möglich sein.

Module der Darstellungsschicht für die ASCII-Schnittstelle:

Für die ASCII-Schnittstelle existieren mit Ausnahme des Adressen-Such-Panels äquivalente Module. Allerdings wurde hier auf Verwendung von Funktionen anderer Module dieser Schicht verzichtet. Aufgrund des begrenzten Platzes, den ein ASCII-Terminal bietet, mussten bei den Suchroutinen eigene Ausgaben geschrieben werden, die anstatt aller Parameter nur die jeweils wichtigsten Werte ausgeben.
Es existieren die folgenden Module:

- AliasAscii.pm
- BenutzerAscii.pm

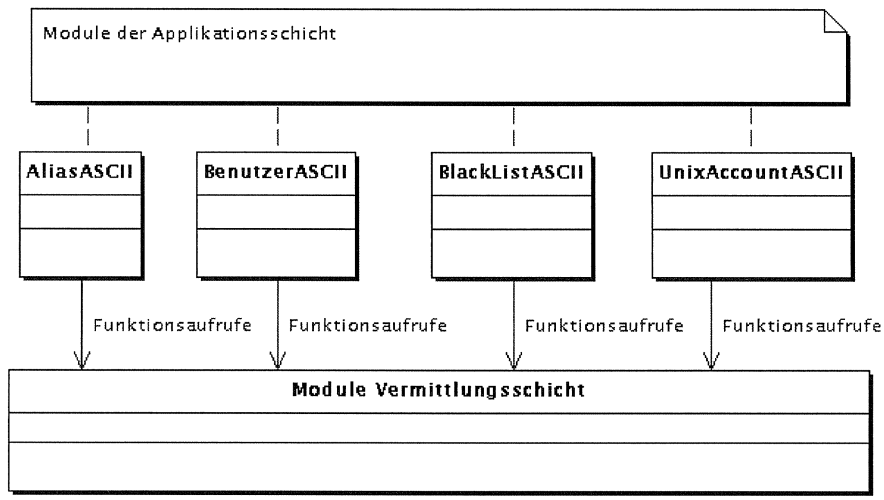

Abbildung 5.4: Module der Darstellungsschicht für die ASCII-Schnittstelle

- BlackListAscii.pm
- UnixAccountAscii.pm

Kapitel 6

Aufgaben der Benutzerschnittstellen:

In diesem Kapitel werden die Benutzerschnittstellen beschrieben. Diese Schnittstellen bilden die Applikationsschicht und bietet eine HTTP-Schnittstelle an, die durch das Programm useradm2.pl repräsentiert wird.

Außerdem wird eine weitere Applikation angeboten, die im ASCII-Terminal läuft und den gleichen Funktionsumfang wie die HTTP-Schnittstelle bietet. Diese Schnittstelle wird durch das Programm useradm.pl repräsentiert und ist über eine Terminal-Verbindung zu einem zentralen Rechner für die Institutsadministratoren erreichbar.

6.1 useradm2.pl

Dieses Programm ist die Schnittstelle für die Web-Schnittstelle. Neben der Auswahl der Modi *Benutzer*, *UnixAccount* usw., ist in useradm2.pl die Benutzerauthentifizierung implementiert. Hier wird bei angegebenem Benutzernamen und gültigem Passwort der Zugriff auf die Schnittstelle gestattet und dem Benutzer eine SessionId zugeordnet, die das ständige Übertragen des Paßwortes ersetzt. Diese SessionId wird in der Session-Tabelle mit dem Login des eingetragen. Weiterhin wird bei jeder Aktion die Gültigkeit der SessionId überprüft und ggf. verlängert, da diese nur 10 Minuten gültig ist.

6.1.1 useradm.pl

useradm.pl bietet lediglich ein Auswahlmenü, von dem aus die Unterfunktionen der ASCII-Schnittstelle aufgerufen werden können. Das Programm selbst bietet keinen Schutz vor nicht-authorisiertem Zugriff, da die Authorisation bereits über die Anmeldemechanismen des Unixrechners realisiert wird.

6.2 Benutzung der Web-Schnittstelle:

Hier wird eine kurze Einführung in die Benutzung der Web-Schnittstelle gegeben. Exemplarisch werden hier die Einrichtung, das Löschen und die Modifika-

tion eines Unixaccounts durchgespielt. Die Konzeption bei der Bearbeitung von Alias-Namen ist identisch.

6.2.1 Einrichtung eines Unixaccounts:

Nach der Anmeldung wähle man die Option *Unix-Accounts*. Danach die Option *Account neu anlegen*. Auf dieser Seite kann man neben dem Verfallsdatum, dem Loginnamen auch den Benutzer zuordnen. Sollte man für mehrere Institute authorisiert sein, muß auch noch ein Institut gewählt werden. Ist man mit allen Eingaben fertig, muß man auf *Account eintragen* klicken. Falls der Loginname bereits vergeben ist oder Felder falsch ausgefüllt wurden, wird eine Fehlermeldung ausgegeben. Falls die Daten korrekt sind, werden die Daten eines existierenden Benutzers angezeigt oder falls der Accountbesitzer noch keinen Eintrag im Useradm hat, der Administrator dazu aufgefordert, weitere, erforderliche Daten einzutragen. Diese Daten bestehen aus Titel und ggf. aus der Matrikelnummer.

Außerdem sind noch UID-Bereich, GID-Bereich und Auslieferungsgruppe zu spezifizieren. Sind die Daten entsprechend gewählt, muß zur endgültigen Eintragung des Accounts noch einmal ein entsprechender Button gedrückt werden.

6.2.2 Accounts verändern und löschen:

Bevor ein Account bearbeitet oder gelöscht werden kann, ist es notwendig ihn erst einmal zu Suchen. Zur Such-Funktion für Unixaccounts gelangt man über den Punkt *Unix-Accounts*. Nun gibt man in das Feld Benutzername einen Accountnamen ein und drückt auf *Abfrage starten*. Falls der Administrator Zugriff auf den eingegebenen Account hat, werden Unterhalb der Ausgabe des gefundenen Accounts die Optionen *Benutzername verändern* und *Benutzername löschen* angeboten.
Wählt man *Benutzername verändern*, kann man den Benutzernamen, UserId, Klasse, Auslieferungsgruppe und weitere Parameter anpassen. Bei Accounts, die Zugriff auf den Useradm haben, ist allerdings die Modifikation des Benutzernamens nicht möglich. In diesen Fällen sind die Datenbankadministratoren des Useradm zu kontaktieren. Weiterhin wird überprüft, ob die eingetragene UserId zum entsprechenden Institut gehört und ob sie noch frei ist.
Bevor man endgültig Accounts aus dem Useradm löscht, wird noch einmal der gesamte Datensatz mit allen zugehörigen Alias-Namen angezeigt. Ebenso werden die zu einem Account gehörendem Alias-Namen ebenfalls gelöscht. Möchte man nun endgültig einen Account löschen, so muß man dies noch einmal bestätigen.

6.2.3 Abschließende Bemerkungen:

Steht ein Benutzer auf der Schwarzen Liste, so kann er nur von der eintragenden Person wieder ausgetragen werden. Werden neue neue Accounts oder Alias-Namen für Benutzer angelegt, die auf der Schwarzen Liste stehen, wird eine Warnmeldung ausgegeben.

Weiterhin ist festzuhalten, daß es im Gegensatz zum alten Useradm nicht möglich ist, einen einzelnen Benutzerdatensatz anzulegen. Neue Benutzerda-

tensätze werden nur bei der Anlage von Alias-Namen, UnixAccounts und Mailweiterleitungen eingerichtet.

Der Zugriff auf die www-Schnittstelle ist nur innerhalb des Informatik-Netzes und Teilen des Universitätsnetzes der Universität möglich.

6.3 Benutzung der ASCII-Schnittstelle:

Die Konzeption der ASCII-Schnittstelle gleicht der Web-Schnittstelle. Allerdings müssen hier die Unterfunktionen direkt über das Eingangsmenü aufgerufen werden. Zusatzfunktion dieser Schnittstelle ist der Punkt *Passwort ändern* Dieses Paßwort ist das Zugangspaßwort für die ASCII-Schnittstelle und die Web-Schnittstelle.

Diese Schnittstelle ist für alle Benutzer interessant, die keinen WWW-Browser zur Verfügung haben, bzw. sich außerhalb des Uni-Netzes befinden und somit keinen Zugriff auf die www-Schnittstelle haben oder eine Schnittstelle im ASCII-Modus bevorzugen. Außerdem wird durch diese Schnittstelle ein schneller administrativer Zugriff gewährleistet.

Kapitel 7

LDAP-Anbindung der Postgresql-Datenbank

7.1 Motivation

Als Adress-Verzeichnis existiert bisher ein "PH-Server", der an der Fakultät kaum bekannt ist und nur wenig genutzt wird. Die Funktionalität sollte erhalten bleiben, ggf. mit einem anderen Verzeichnis-Dienst.

Gleichzeitig muß die Aufgabe gelöst werden, ein zukünftiges Mail-System wie Exim oder Postfix an den Datenbestand des Useradm zu koppeln. Zwar wäre für einige Mail-Systeme (z. B. Exim) eine direkte Anbindung an die Postgresql-Datenbank möglich, jedoch wurde von uns aus Gründen der Performance und Redundanz eine Anbindung über LDAP bevorzugt. Dies wird im folgenden genauer erklärt.

7.2 Bereitstellung von Adreß-Informationen für Mail-Clients

Der bestehende PH-Server als Adress-Verzeichnis (beschrieben in RFC 2378) kann zwar über eine eigene Abfrage-Sprache sehr einfach abgefragt werden, wird aber nur von wenigen Mail-Programmen unterstützt (beispielsweise von Eudora Mail). Die Möglichkeit, Adress-Informationen per LDAP zu beziehen, ist hingegen wesentlich weiter verbreitet (Outlook, Outlook Express, Netscape-Communicator, Eudora-Mail, Pegasus-Mail, etc.). Damit erscheint es nicht sinnvoll, den PH-Server als Verzeichnis-Dienst weiter zu betreiben und es bietet sich an, die Umstellung des Useradm zu nutzen, um den Wechsel von PH nach LDAP zu vollziehen.

Ziel ist die Veröffentlichung von Vorname, Nachname, Telefon-Nummer und Email-Adresse. Aus Gründen des Datenschutzes soll diese Veröffentlichung nur dann erfolgen, wenn der jeweilige Mitarbeiter dieser Veröffentlichung nicht widerspricht. Außerdem kann ein Mitarbeiter mehrere Mail-Adressen bzw. Mailaliase haben, die nicht alle gleichzeitig veröffentlicht werden sollten. Es wäre nicht sinnvoll, bei der Suche nach einem bestimmten Namen drei oder mehr Treffer für eine einzelne Person zu erhalten, die mehrere Email-Adressen an

der Fakultät besitzt. Daher ist für Unix-Accounts und für Mail-Aliase ein Feld PUBLIZIEREINLDAP vorgesehen, mit dem für jede einzelne Email-Adresse die Veröffentlichung gesteuert werden kann. Eine Veröffentlichung von Mailinglisten ist nicht vorgesehen.

Die Einträge im LDAP-Server verwenden die Email-Adresse als eindeutigen Schlüssel und haben die folgende Form: "cn=Email-Adresse, ou=People, dc=IRA, dc=UKA, dc=DE" mit den Attributen "vorname", "Nachname", "Telefon-Nummer" und "Email-Adresse".

Der Zugriff auf diese Informationen wurde mit Microsoft Outlook, Netscape Communicator und Eudora-Mail getestet, ohne dass dabei Probleme aufgetreten wären.

7.3 Anbindung an das Mailsystem

Diese Studienarbeit dient unter anderem der Vorbereitung zu Umstellung des Mailsystems. Geplant ist eine Migration vom bisher verwendeten PP-System zu Exim.

Dazu ist es erforderlich, das Mailsystem an den Datenbestand des Useradm an zubinden. Die Durchführung für das bisher verwendete PP-Mail-System wird an anderer Stelle beschrieben, an dieser Stelle soll nur erklärt werden, warum eine Anbindung über LDAP sinnvoll ist und wie sie durchgeführt werden kann.

Problem: Emails erreichen das Mail-Gateway in der Form *Ziel@ira.uka.de*. Jedoch soll die Email nicht auf dem Mailgateway ausgeliefert, sondern an den für das jeweilige Ziel zuständigen Auslieferungs-Server weiter geleitet werden. Über den Schlüssel "Ziel" muß also der Auslieferungs-Server ermittelt werden.

Beim bisherigen Mailsystem erfolgte diese Zuordnung über dbm-Dateien, die einmal täglich zentral erzeugt und auf die beteiligten Server kopiert wurden. Prinzipiell wäre dies auch mit dem neuen Mail-System möglich, nachteilig ist jedoch die seltene Aktualisierung. Da aber bereits ein LDAP-Server eingeplant ist und an Postgresql angebunden werden soll, um Adress-Informationen bereit zu stellen, eröffnet sich die Möglichkeit, die LDAP-Postgresql-Anbindung etwas zu erweitern und auch die für das Mailsystem nötigen Informationen über LDAP abzufragen.

7.3.1 Performance

An einem normalen Werktag werden vom Mail-System der ATIS über 6000 Emails verarbeitet. Diese passieren mindestens das zentrale Mailgateway und den zentralen Mailhost, wobei jeweils (mindestens) 2 LDAP-Lookups erforderlich werden. Damit kommt man auf mindestens 24000 LDAP-Lookups pro Tag. Im LDAP-Server sind die erforderlichen Daten im Vergleich mit der Datenbank denormalisiert hinterlegt. Email-Adresse und Auslieferungs-Server bilden eine Einheit, mit der Email-Adresse als Schlüssel. Bei einer direkten Abfrage in der Postgresql-Datenbank müßte man nachprüfen, ob sich die Email-Adresse um einen Account handelt und (falls ja) in der Auslieferungs-Tabelle den Auslieferungs-Server bestimmen. Falls es sich um keinen Account handelt, ist zu prüfen, ob die Email-Adresse ein Alias ist (Zugriff auf Tabelle "mailalias") oder eine Mailingliste (Zugriff auf Tabelle "mailingliste"). Falls es sich um ein

Mailalias handelt, müßte diese Prüfung mit der neuen Email-Adresse wiederholt werden.

Diese Zugriffs-Reihenfolge ist sinnvoll, da die Treffer-Wahrscheinlichkeit für die Tabelle "unixaccounts" am größten und die Treffer-Wahrscheinlichkeit für die Tabelle "mailinglisten" am kleinsten ist.

Es wäre auch möglich, über einen Zugriff auf die Tabelle "vergebeneAdressen" festzustellen, in welcher Tabelle die nötige Information vorhanden ist. Dann werden mindestens zwei, im wahrscheinlichsten Fall eines Unix-Accounts aber drei Zugriffe erforderlich.

Damit werden also mindestens zwei Datenbank-Zugriffe nötig, um das Ergebnis nur eines LDAP-Zugriffs zu erhalten. LDAP bietet zusätzlich die Möglichkeit der Replikation auf mehrere Server. Falls erforderlich kann man dann die LDAP-Anfragen der Mailserver auf unterschiedliche Server lenken, so dass sich die Last verteilt.

7.3.2 Redundanz

Eine zentrale Postgresql-Datenbank stellt einen Single-Point-of-Failure da. Weil das Mailsystem jedoch einen außerordentlich wichtigen, zentralen Dienst darstellt, ist eine maximale Verfügbarkeit das Ziel. So fungieren derzeit zwei Server als Mail-Gateway, um die Ausfall-Wahrscheinlichkeit zu senken.

Der Einsatz von LDAP löst das Verfügbarkeits-Problem auf einfache Weise. Durch Einsatz der Replikations-Mechanismen wird das LDAP-Verzeichnis auf anfangs drei, später möglicherweise auch mehr LDAP-Server repliziert.

7.4 Anbindung konkret

7.4.1 Struktur des LDAP-Verzeichnisses

Um die Objekte des LDAP-Verzeichnisses zu beschreiben, wird im folgenden das "ldif"-Format verwendet. Dieses Format ist sehr einfach gehalten und dient dazu, große Mengen an Daten in ein LDAP-Verzeichnis einzubringen. Die Einträge werden durch Leerzeilen getrennt und haben die Form "Attribut: Wert", wobei jeder Eintrag mit dem Attribut "dn" (=Distinguished Name') beginnt, der einen eindeutigen Schlüssel darstellen muß.

Als Wurzel für das LDAP-Verzeichnis wurde der DN "dc=ira, dc=uka, dc=de" gewählt. Darunter existieren die zwei Bereiche "ou=People, dc=ira, dc=uka, dc=de" (im folgenden kurz "People") und "ou=Mail, dc=ira, dc=uka, dc=de" (im folgenden kurz "Mail"). "ou" steht hierbei für "Organizational Unit".

Der Bereich "People" ist für die ganze Fakultät lesend zugreifbar. Er enthält das Adress-Verzeichnis der Informatik-Fakultät und hat einen DN der Form:

```
dn: cn=s_hagen ,ou=people,dc=ira,dc=uka,dc=de
cn: Patrick von-der-Hagen       Common Name = vollständiger Name
sn: von-der-Hagen               Surename = Nachname
givenName: Patrick              Vorname
telephoneNumber: unbekannt      Telefon-Nummer
mail: s_hagen@ira.uka.de        Email-Adresse
ou: people                      Organisations-Einheit
objectClass: top
```

```
objectClass: Person
```

Die Attribute "objectClass" bedürfen einer kurzen Erklärung. Für LDAP muß jedes Element eines Verzeichnisses über ein "Schema" spezifiziert sein. Da die vordefinierten Schemata für unsere Zwecke ausreichten, wurden keine neuen Spezifikationen erstellt. Die ObjectClass gibt an, welche Schema-Spezifikation verwendet werden soll. Die Angabe mehrerer Schema-Spezifikationen ist zulässig. Das Schema "Person" definiert die vergebenen Attribute, das Schema "top" ist obligatorisch für jeden Eintrag.

Der Bereich "Mail" besteht im wesentlichen nur aus "DN" und dem Attribut "mail". Die "objectClass: top" ist auch hier obligatorisch.

```
dn: cn=s_hagen,ou=mail,dc=ira,dc=uka,dc=de
objectclass: top
mail: s_hagen@studsun1.ira.uka.de
```

Erhält das Mailsystem eine Mail für *Email@ira.uka.de*, so wird die Domain-Angabe ignoriert und eine LDAP-Anfrage für "Email" formuliert. Der "dn" für den gesuchten Eintrag läßt sich also sofort konstruieren und erlaubt den Zugriff ohne jede Suchoperation. Intern erfolgt der Zugriff über eine Hash-Tabelle mit dem "dn" als Schlüssel. Die Mail kann dann vom Mail-System an den Auslieferungs-Server, im Beispiel "studsun1.ira.uka.de", weiterleiten. Der Mail-Bereich ist nur für die vorhandenen Mailserver lesbar.

7.4.2 Erstellen des LDAP-Verzeichnisses

Das LDAP-Verzeichnis kann mit dem oben angegebenen Format sehr einfach erstellt werden. Dazu wird eine Datei erstellt, die sämtliche einzufügende Daten im obigen Format enthält. Zusätzlich zu den Einträgen in den Bereichen "People" und "Mail" müssen auch das Wurzel-Objekt und die Bereiche selbst spezifiziert werden. Anschließend muß beim verwendeten OpenLDAP der Server-Prozeß gestoppt werden. Das Kommando "ldapadd" fügt dann die Daten ein und erstellt die gewünschten Indizes. Anschließend kann der Server wieder gestartet werden. Bei der Unterbrechung des Server-Betriebs verliert man leider die Möglichkeit, die Daten automatisch zu replizieren.

Verglichen mit dem Ansatz, alle Einträge im laufenden Betrieb einzufügen, ist dieses Verfahren um ein vielfaches schneller. In einem Test hat dieses Verfahren auf beim eingesetzten Datenbestand weniger als 15 Minuten gedauert, während ein Einfügen im laufenden Betrieb auf 10 und mehr Stunden gekommen ist und damit als unbrauchbar ausscheidet.

Um jetzt aber die Änderungen aus der Postgresql-Datenbank nach LDAP zu übernehmen ist das Verfahren, mit "ldapadd" ein komplettes Verzeichnis neu zu erstellen, nicht praktikabel, da sowohl die Ausfall-Zeit als auch der Verlust der automatischen Replikation unerwünscht sind. Dies kann nur bei der Einrichtung eines neuen LDAP-Servers praktiziert werden.

Damit wird es erforderlich, die Änderungen am Datenbestand festzustellen und nur diese Änderungen im laufenden Betrieb unter Ausnutzung der Replikations-Mechanismen in das LDAP-Verzeichnis einzufügen.

7.4.3 Logging von Änderungen und Übernahme

Angenehmerweise bietet Postgresql mit der Hilfe von "Triggern" ein geeignetes Werkzeug, um ein Logging von Änderungen zu realisieren. Während der Ausführung eines Triggers stehen bei einem *insert* oder *delete* jeweils der betroffene Datensatz, bei einem *update* der alte und der neue Datensatz zur Verfügung. Damit können wir erreichen, dass jede dieser Aktionen, wenn sie auf einen Account, ein Alias oder eine Mailingliste ausgeführt werden, von Postgresql in eine Tabelle "log" geschrieben wird, in der festgehalten wird, um welche Aktion *insert*, *update*, *delete*, um welche Tabelle "unixAccount", "mailalias" und "mailingliste" und um welche Email-Adresse ("login","alias", "listenname") es sich handelt. Bei einem Update werden sowohl die alte als auch die neue Adresse gespeichert, wobei diese natürlich auch identisch sein kann, wenn nur ein anderes Attribut verändert wurde. An einem boolschen Flag kann die LDAP-Übernahme dann erkennen, welche Einträge noch übernommen werden müssen und welche bereits bei einem vorherigen Durchlauf bearbeitet wurden. Alle Einträge erhalten einen Zeitstempel und die Verarbeitung erfolgt sortiert nach diesem Zeitstempel. Damit wird vermieden, dass eine falsche Abarbeitungs-Reihenfolge zu Inkonsistenzen führt.

Im laufenden Betrieb muß noch untersucht werden, ob die Einträge in die Logging-Tabelle zu zahlreich werden und dann zu Performance-Problemen führen. Sollte das der Fall sein, kann später darüber nachgedacht werden, z. B. nur die Logging-Daten der letzten drei Monate zu speichern und ältere Datensätze regelmäßig zu entfernen.

Kapitel 8

Anbindung des PP-Mailsystems

In der Fakultät für Informatik wird derzeit noch das Mailsystem "Isode PP" eingesetzt, ein Wechsel auf ein anderes Produkt ist allerdings mittelfristig geplant. Dennoch muß natürlich bis auf weiteres die Anbindung an "Isode PP" gewährleistet werden.

Für den Betrieb des Mailsystems müssen mehrere Dateien erstellt werden, beispielsweise eine Liste aller existierenden Accounts, eine Zuordnung existierender Aliase zu diesen Accounts, eine Zuordnung dieser Accounts zu Auslieferungsgruppen und eine Zuordnung von Auslieferungsgruppen zu Mailservern.

Aufgrund der sehr langsamen Geschwindigkeit von "ClearDDTS" hat man versucht, die Anzahl der Datenbank-Zugriffe zu minimieren und fragt deshalb die nötigen Informationen nur einmal aus der Datenbank ab, um sie in Text-Dateien zwischen zu speichern. Im weiteren Verlauf der Tabellen-Erstellung greift man nur auf diese Dateien und nicht mehr auf die Datenbank zu.

Es werden die Programme "ExtractLists", "ExtractAdresses" und "ExtractGroups" ausgeführt, die jeweils Datenbank-Abfragen durchführen und in einem vordefinierten Format speichern. Um die Tabellen-Erstellung für das Mailsystem insgesamt zu beschleunigen, enthalten die Tabellen MAILADRESSE und MAILALIAS die Felder "Extrakt" bzw. "Listen-Extrakt". Der Inhalt dieser Felder könnte durch eine einfache Konkatenation anderer Felder des gleichen Eintrags erzeugt werden, aber man hat es vorgezogen, diese Konkatenation nur einmalig beim Einbringen der Daten in den Useradm zu erzeugen, um bei der Tabellenerzeugung auf diesen Schritt verzichten zu können uns so einen Geschwindigkeits-Vorteil zu erzielen.

Das Gesamt-System zur Anbindung des Mailsystems ist umfangreich und leider nur schlecht dokumentiert. Es fragt den Useradm ab, bezieht Informationen aus dem Mailing-Listen-System, erzeugt eigenständig Mailinglisten, aktualisiert den PH-Server, erstellt die für das Mailsystem notwendigen Auslieferungs-Dateien und kopiert diese Dateien automatisch auf die eingebundenen Mailserver.

Da die Programme "ExtractLists", "ExtractAdresses" und "ExtractGroups" eine klar definierte Schnittstelle zwischen der Datenbank und dem System zur Anbindung des Mailsystems darstellen, reicht es aus, diese drei Programme,

unter Beibehaltung der alten Schnittstelle, neu zu implementieren.

Da auf das bisherige "Extrakt" verzichtet wird, führen diese Programme nicht mehr nur Datenbank-Abfragen durch, sondern müssen diese Daten noch im richtigen Format aufbereiten.

8.1 Anbindung konkret

8.1.1 ExtractGroups

Dieses Programm erzeugt eine Liste aller Auslieferungsgruppen mit dem zugehörigen Auslieferungs-Server.

Erklärung: Jedes Institut verfügt über eine oder mehrere Auslieferungsgruppen. Jede Auslieferungsgruppe verfügt über genau einen Mailserver. Damit kann man erreichen, dass die Mail innerhalb eines Instituts auf unterschiedliche Mailserver zugestellt wird, etwa um aus Sicherheitsgründen zu verhindern, dass ein Professor und ein Student auf dem gleichen Mail-Server Accounts haben

Format:

```
Auslieferungsgruppe:Server
```

Beispiel:

```
STUD:studsun1.ira.uka.de
```

8.1.2 ExtractAddresses

Ein Script "ExtractAddresses" wird vom Mail-Sytem bei der Generierung der Auslieferungs-Informationen aufgerufen, um eine Liste aller Email-Adressen zusammen mit den nötigen Auslieferungs-Informationen, z. B. der Auslieferungsgruppe, zu erhalten.

Leider existierte keine Definition, in welchem Format diese Informationen erwartet werden. Nach einer Analyse der Ausgabe des existierenden Scripts ließen sich drei verschiedene Fälle feststellen.

Da die Scripte nur Kleinbuchstaben ausgeben, sind im Folgenden Variablen in Großbuchstaben geschrieben.

- Account mit/ohne Alias
- Mailingliste
- Mail-Forward

Account mit/ohne Alias

Format:

```
ALIAS::ACCOUNT:e:UID:GID:u:GRUPPE
```

Beispiel:

```
::s_hagen:e:27097:2700:u:STUD
```

bzw.

```
patrick.von-der-hagen::s_hagen:e:27097:2700:u:STUD
```

Dabei ist die Angabe von ALIAS optional. "e" bezeichnet die Mailberechtigung "extern". Neben "extern" existierten im alten System noch "i" für "intern" und "n" für "nicht berechtigt". Da diese Unterscheidung im neuen Useradm nicht mehr getroffen wird, wird durchgehend "e" verwendet. Die Bedeutung von "u" ist leider nicht bekannt.

Mailingliste

Format:

```
::LISTENNAME:e:::f:*list,ANSPRECHPARTNER,file=lists/LISTENNAME,BESCHREIBUNG
```

Beispiel:

```
::atis-sw:e:::f:*list,fvd,file=lists/atis-sw,ATIS Software-Hiwis
```

In diesem Fall ist keine Angabe optional. Aus diesen Angaben erzeugt das Mailsystem implizit weitere Email-Adressen, nämlich "LISTENNAME-owner" als Alias auf den ANSPECHPARTNER und "LISTENNAME-request" um eine automatische Verwaltung der Mailinglisten zu ermöglichen. Im Punkt "Anbindung det Mailinglisten" wird darauf näher eingegangen.

Mail-Forward

Format:

```
::ADRESSE:e:::f:ZIEL
```

Beispiel:

```
::hagen:e:::f:patrick@vonderhagen.de
```

Als letzter Fall gibt "ExtractAdresses" noch sämtliche Mail-Forwards im beschriebenen Format aus.

8.1.3 ExtractLists

Format:

```
AUSLIEFERUNGSGRUPPE:KLASSE:ACCOUNT
```

Beispiel:

```
STUD:Student:s_hagen
```

Für jede Auslieferungsgruppe werden sämtliche Accounts mit ihrer zugeordneten Klasse ausgegeben, sofern einer Veröffentlichung in den automatisch erzeugten Mailinglisten zugestimmt wurde. Aus diesen Informationen werden dann institutsbezogene Mailinglisten automatisch generiert, z. B. wird die Mailingliste "atis-mitarbeiter" auf diese Weise erzeugt.

8.2 Anbindung der Mailinglisten

Derzeit wird "Petidomo" als Mailinglisten-Management-System an der Fakultät für Informatik eingesetzt. Dabei belegt jede Mailingliste drei Email-Adressen, "LISTENNAME", "LISTENNAME-owner" und "LISTENNAME-request". Natürlich dürfen diese Mailinglisten nur dann erzeugt werden, wenn diese drei Adressen noch nicht vergeben sind. Im Gegensatz zum alten Useradm registrieren wir alle drei Email-Adressen, dabei werden "LISTENNAME-owner" und "LISTENNAME-request" als Mail-Forwards gespeichert. Dazu mußten die Programme "createlist", "removelist" und "changeadmin" neu implementiert werden.

8.2.1 createlist

"createlist Listen-Name Verantwortlicher Beschreibung" hat die Aufgabe, die drei zu einer Mailingliste zugehörigen Email-Adressen in den Useradm einzutragen, eine Standard-Konfiguration für die neue Mailingliste anzulegen und den Ansprechpartner mit einer Email über diesen Vorgang zu informieren.

8.2.2 removelist

"removelist Mailingliste" entfernt die zugehörigen Einträge aus dem Useradm und aus dem Mailsystem. Darüber wird der Verantwortliche für diese Mailingliste per Email informiert.

8.2.3 changeadmin

"changeadmin Mailingliste Alter-Admin neuer-Admin" hat die Aufgabe, den Eintrag für den Verantwortlichen im Useradm zu verändern. Diese Änderung wird im Useradm ausgeführt, anschließend werden der alte und der neue Verantwortliche per Email über den Wechsel informiert.

Kapitel 9

Zusammenfassung und Ausblick

Hier soll noch einmal abschließend erörtert werden, welche Zielsetzungen im Rahmen dieser Studienarbeit geleistet wurden.

9.1 Zielsetzungen

Zielsetzung der Studienarbeit war die Neukonzeption der Benutzerverwaltung unter Gewährleistung der folgenden Dienste:

- Existenz einer http-Benutzerschittstelle
- Existenz einer äquivalenten ASCII-Schnittstelle
- Ersatz für den PH-Server mittels LDAP-Verzeichnisdienst herstellen
- Anbindung an das bestehende pp-Mailsystem
- Vorbereitung der Benutzerverwaltung für ein alternatives Mailsystem(Exim)

Weiterhin wurden folgende Zielsetzungen bezüglich Entwurf gegeben:

- Leichte Erweiterbarkeit des Systems
- Existenz einer Dokumentation

9.2 Ergebisse der Studienarbeit

9.2.1 Allgemeines

Wie in Kapitel 5 und 6 beschrieben, wurden HTTP- und ASCII-Schnittstellen für die neue Benutzerverwaltung implementiert. Hierbei wurde dank der Schichtenarchitektur eine Basis geschaffen um m.H. weiter unten liegender Schichten neue Applikationen zu gestalten bzw. leicht weitere Dienste in die Benutzerschnittstellen zu integrieren.

Aufgrund einer standardisierten Schnittstelle des Datenbasis- Verwaltungssystems ist es auch möglich komplett neue Applikationen in anderen Programmiersprachen (außer Perl) zu entwickeln. Außerdem soll diese Ausarbeitung für spätere Erweiterungen als Dokumentation verwendet werden können.

Wie in Kapitel 7 beschrieben Exisitiert nun ein LDAP-Dienst, der unter anderm die Funktionalität des PH-Servers ersetzt. Dieser LDAP-Dienst wird zukünftig als Verzeichnisdienst für das neue Mailsystem (Exim) verwendet werden und relatisiert somit eine Schnittstelle zwischen Datenbasis und Mailsystem.

9.2.2 Aktivierung des neuen Systems

Vor der Ablösung der alten Benutzerverwaltung wurden die neuen Benutzerschittstellen 4 Wochen im Testbetrieb für die ATIS-Beauftragten freigeschaltet. Hierbei wurde getestet, ob die Migration des Datenbestandes der alten Benutzerverwaltung funktioniert und die Benutzerschnittstellen von den ATIS-Beauftragten getestet.

Nachdem keine größeren Probleme im Testbetrieb auftauchten wurde das System umgestellt und läuft seit der Umstellung ohne nennenswerte Probleme.

9.3 Ausblick

In zukünftigen Arbeiten werden Applikationen entwickelt, die die Administation von kompletten Benutzergruppen realisieren. Beispielsweise können dann Unix-Benutzerkonten, dessen Benutzer noch an der Fakultät immatrikuliert ist, automatisch um ein Jahr verlängert werden.

Weiterhin bietet es sich an weitere Dienste in die (z.B. Verwaltung der ATIS-Laptop-Zugänge) Benutzerverwaltung zu integrieren oder die Funktionalität der Benutzerschnittstellen zu erweitern, damit abhängig von der Auslieferungsgruppe auf bestimmten Rechnern ein Benutzerkonto automatisch angelegt wird.

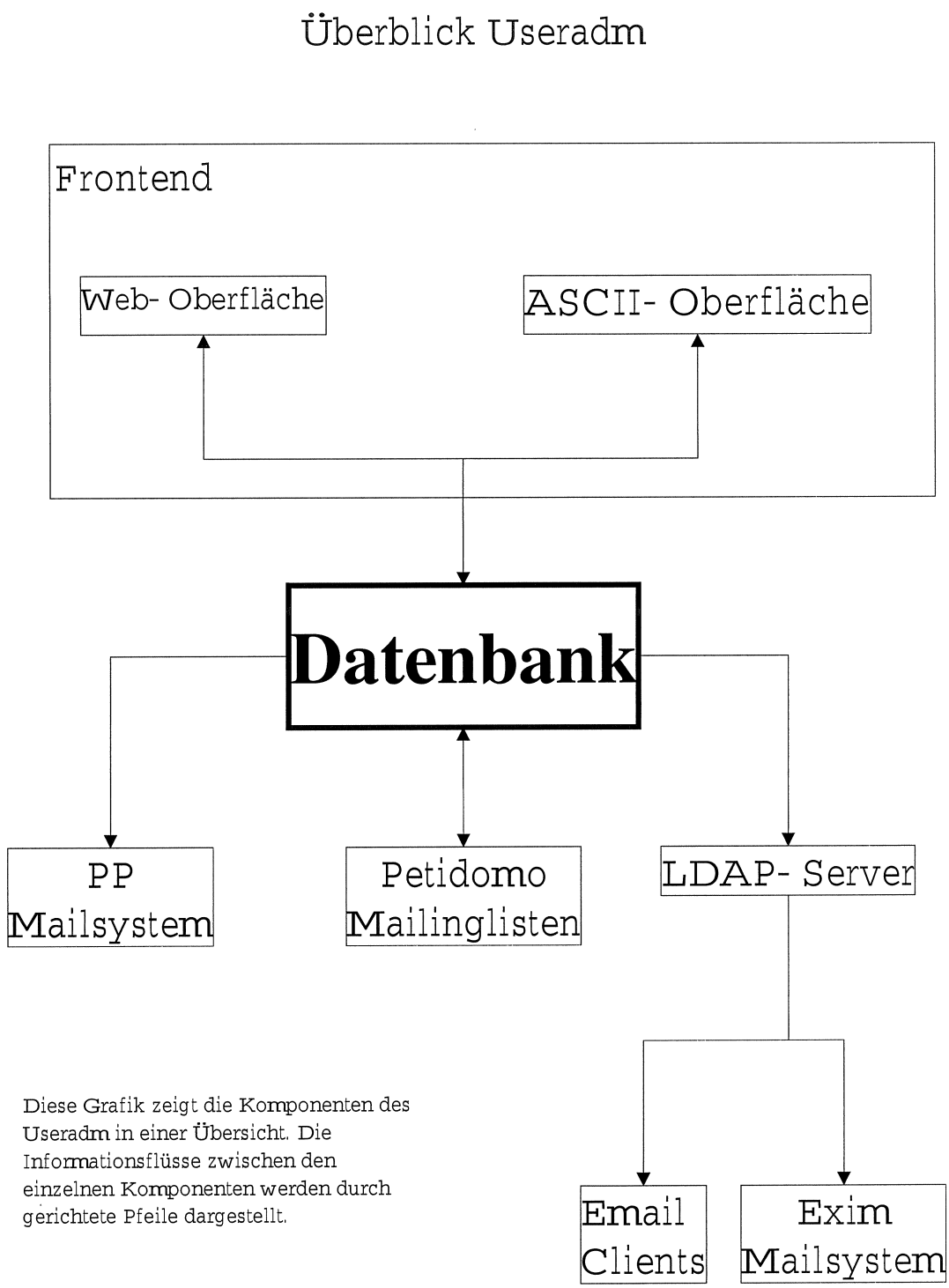

Abbildung 9.1: Der neue Useradm im Überblick

Anhang A

Tabllenstruktur des Gesamtsystems

Diese Grafik stellt die Tabellenstruktur mit sämtlichen in der Datenbasis vorkommenden existierenden Attributen dar. Die unterstrichenen Attribute repräsentieren die Primär- bzw. - falls vorhanden - Sekundärschlüssel der jeweiligen Relationen. Das zuerst unterstrichene Attribut einer Relation ist bildet den Primärschlüssel.

Auf die Darstellung der Fremdschlüsselreferenzen wurde aus Übersichtlichkeitsgründen verzichtet.

Die auf die VERGEBENEADRESSEN-Relation verweisenden Pfeile sollen die in der Datenbasis existierenden *Rules* darstellen, die für die Eindeutigkeit der ensprechenden Bezeichner *listenName*, *alias* und *login* in der gesamten Datenbasis sorgen.

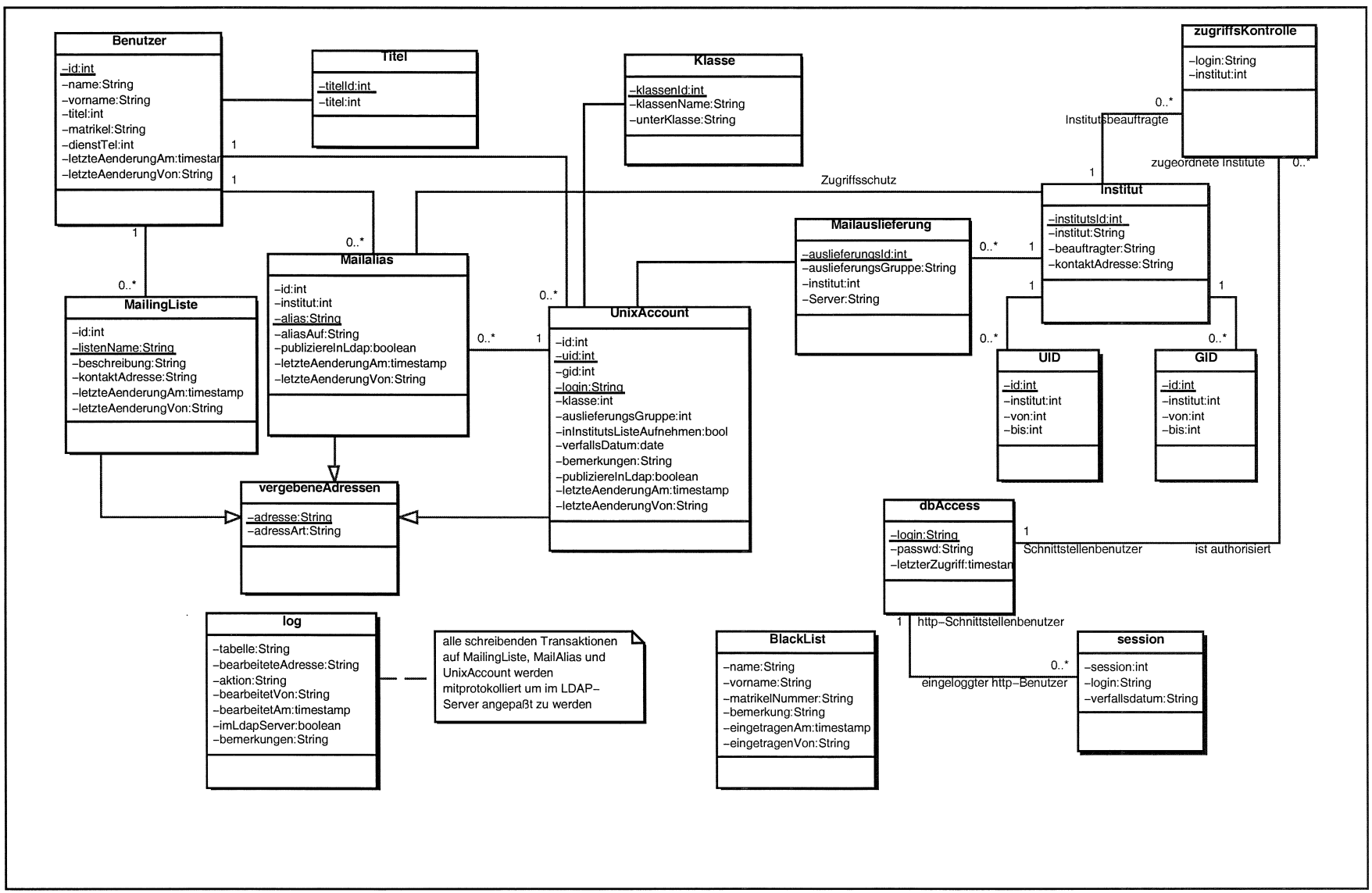

Abbildung A.1: Tabellenstruktur des Useradm

Anhang B

Beispiel für eine Exim-Konfiguration mit LDAP

Ein Ziel der Studienarbeit war es, einen Wechsel der Mailsystems der ATIS von PP zu Exim vorzubereiten. In diesem Anhang soll kurz gezeigt werden, wie Exim an die in Kapitel 5 beschriebene LDAP-Struktur angebunden werden kann.

Hier ein Auszug einer möglichen Exim-Konfiguration:

```
ldap_default_servers=irams1.ira.uka.de:irafs1.ira.uka.de

[...]

# LDAP-Director
ldap_aliases:
  driver = aliasfile
  file_transport = address_file
  pipe_transport = address_pipe
   search_type = ldap
  query = ldap:///cn=$local_part,ou=mail,dc=ira,dc=uka,dc=de?mail
  no_rewrite
```

Die Angabe der verfügbaren LDAP-Server muß in Main-Bereich der Konfigurations-Datei erfolgen. Diese Server werden bei einer Anfrage der Reihe nach angesprochen. Der angegebene LDAP-Direktor nimmt den User-Teil der Email-Adresse, z. B. *hagen* bei der Adresse *hagen@ira.uka.de*, stellt die Anfrage nach dem Attribut *mail* des LDAP-Eintrages "cn=hagen, ou=mail, dc=ira, dc=uka, dc=de" und erhält als Antwort die vorgegeben Auslieferungs-Adresse, z. B. *hagen@irams1.ira.uka.de*, an die die Email weiter geschickt wird.

Dieser Direktor muß am Anfang der DIRECTORS CONFIGURATION eingefügt werden. Damit werden die LDAP-Anfragen zuerst ausgeführt und erhalten somit Priorität gegenüber allen anderen Direktor-Einträgen. Damit wird eine Email auch dann an einen anderen Mail-Server ausgeliefert, wenn lokal ein gleichnamiger Account existiert.

Der Direktor wird übergangen, wenn die LDAP-Anfrage den eigenen Rechner als Auslieferungs-Server angibt und nur in dem Fall werden also die folgenden Einträge beachtet.

Mit dieser Konfiguration betreibt die ATIS derzeit mehrere Mailserver.

Literaturverzeichnis

[Dal98] Matthias Kalle Dalheimer. *Latex - kurz und gut.* O'Reilly Verlag, Köln, 1998.

[Loc00] Prof. Dr.-Ing P.C. Lockemann. *Kommunikation und Datenhaltung.* Institut für Programmstrukturen und Datenorganisation, Universität Karlsruhe, 2000. Vorlesungsskriptum zur entsprechenden Vorlesung.

[Mom00] Bruce Momjian. *PostgreSQL Introduction and Concepts.* Addison-Wesley, Dezember 2000.

[Rec00] Keith Reckdahl. Using imported graphics in latex 2e. Technical report, CTAN, 10 2000. Erhältlich unter ftp://ftp.dante.de/tex-archive/CTAN/info/epslatex.ps.

[Rob00] Arnold Robbins. *Unix In A Nutshell - Für SVR4 und Solaris 7.* O'Reilly Verlag, Köln, zweite edition, 2000.

[SL95] P.C. Lockemann S. Lang. *Datenbankeinsatz.* Springer Verlag, Heidelberg, 1995.

[SS92] Rudolf Potucek Stefan Schwarz. *Latex - Satzkunst statt DTP.* Chip Verlag, Würzburg, 1992.

[TC99] Nathan Torkington Tom Christiansen. *Perl Kochbuch.* O'Reilly Verlag, Köln, 1999.