INSTITUT FÜR INFORMATIK
DER LUDWIG–MAXIMILIANS–UNIVERSITÄT MÜNCHEN

Projektarbeit

Entwicklung einer grafischen Benutzeroberfläche für ein Middleware-unabhängiges Management virtueller Organisationen im D-Grid

Matthias Kager, Andrea Nutsi

INSTITUT FÜR INFORMATIK
DER LUDWIG–MAXIMILIANS–UNIVERSITÄT MÜNCHEN

Projektarbeit

Entwicklung einer grafischen Benutzeroberfläche für ein Middleware-unabhängiges Management virtueller Organisationen im D-Grid

Matthias Kager, Andrea Nutsi

Aufgabensteller:	Prof. Dr. D. Kranzlmüller
Betreuer:	Dr. Michael Schiffers
Abgabetermin:	9. Januar 2011

Hiermit versicheren wir, dass wir die vorliegende Projektarbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet haben.

München, den 9. Januar 2011

. . . . . .

(Unterschrift der Kandidaten)

Abstract

Im Laufe dieser Arbeit wird eine grafische Benutzerschnittstelle für den bereits in einer früheren Arbeit konzipierten VO-Layer erarbeitet. Die Anwendung soll ein Middleware-übergreifendes Management von Usern und Ressourcen in virtuellen Organisationen eines Grid ermöglichen. Dabei wird in dieser Arbeit das vorhandene Modell des VO-Layers teilweise angepasst und um die Konzepte der Change Requests und der WatchList erweitert. Dem Benutzer eines Grids wird die Möglichkeit gegeben, mittels eines Change Requests dem Administrator einen Änderungswunsch seiner Rechte mitzuteilen. Der Administrator wiederum kann einen User sperren, falls dieser sich auffällig verhält. Dabei wird der User auf die WatchList der entsprechenden VO gesetzt. Außerdem werden dem Administrator Statistiken über die VO-Struktur zur Verfügung gestellt. Nach der Analyse der Anforderungen und der Weiterentwicklung des Modells des VO-Layers wird dieses als Webanwendung prototypisch implementiert. Die GUI wird mit JSP und Ajax realisiert und läuft auf einem Tomcat Server mit Anbindung an eine MySQL Datenbank.

Inhaltsverzeichnis

1 Einleitung

Diese Arbeit basiert auf dem VO-Layer, welcher in einer früheren Diplomarbeit entwickelt wurde. Das Konzept des VO-Layers wird im Folgenden vorgestellt. Außerdem beschäftigt sich dieses Kapitel mit der Aufgabenstellung, bevor die Vorgehensweise näher erläutert wird. Abschließend folgt eine Definition der wichtigsten in dieser Arbeit verwendeten Begriffe.

1.1 Aufgabenstellung

In diesem Abschnitt wird sowohl der Hintergrund als auch das Ziel der vorliegenden Arbeit thematisiert.

1.1.1 Hintergrund der Arbeit

Diese Arbeit basiert auf der Diplomarbeit mit dem Titel 'Entwicklung einer einheitlichen Autorisierungs- und Authentifizierungsschnittstelle für heterogene Grids am Beispiel D-Grid' von Kirchler aus dem Jahr 2008:

> Im Rahmen dieser Diplomarbeit wird [...] eine Abstraktionsschicht eingeführt, welche die Authentifizierung und Autorisierung von der Grid-Middleware entkoppelt und in eine darüber liegende Schicht, den so genannten VO-Layer, integriert. Diese Schicht hat ihren Namen vom Konzept der virtuellen Organisation (VO), welches im Grid-Umfeld sehr verbreitet ist und zur Autorisierung der Benutzer verwendet wird. Der VO-Layer hat die Aufgabe die Authentifizierungs- und Autorisierungsanfragen der Benutzer entgegen zunehmen, zu prüfen und anschließend an die entsprechende Middleware-Technologie weiterzuleiten. Der VO-Layer fungiert somit als Proxy zwischen dem Benutzer und der Grid-Middleware, welche die Ressource bereitstellt. Damit die Autorisierung vereinheitlicht werden kann, wird in dieser Diplomarbeit eine generische VO-Struktur entwickelt, in der die verschiedenen VO-Konzepte der einzelnen D-Grid Communities abgebildet werden können. Dadurch kann jede Community ihr eigenes VO-Konzept behalten, die Autorisierung aber wird dadurch vereinheitlicht. [Kir08]

1.1.2 Ziel der Arbeit

Das Ziel dieser Arbeit ist die Entwicklung einer grafischen Benutzeroberfläche für den von Kirchler entwickelten VO-Layer, um die Benutzung dessen überhaupt zu ermöglichen und anwenderfreundlich zu gestalten. Dabei liegt der Fokus dieser Arbeit auf dem Management von VO-Strukturen. Dies umfasst das Erstellen, Modifizieren und Deaktivieren von Usern, VOs, FQANs, Ressourcen, Rollen, Gruppen und Capabilities. Zusätzlich wird als Ergänzung zur Arbeit von Kirchler eine WatchList eingeführt, mit welcher Benutzer aus VOs gesperrt werden können. Des Weiteren bekommen Benutzer die Möglichkeit mittels Change Requests Änderungswünsche an den Administrator zu übermitteln. Diesem wird

ermöglicht, den Change Request zu bearbeiten. Außerdem wird eine rudimentäre Sicherheitsarchitektur erarbeitet, damit die Anwendung vor unbefugten Zugriffen geschützt wird. Über die Aktivitäten im VO-Layer sollen Logs und Statistiken erzeugt werden, wobei für die Statistiken eine angemessene Darstellung erforderlich ist.

1.2 Vorgehensweise

Die Entwicklung der grafischen Benutzerschnittstelle erfolgt nach dem Konzept eines objektorientierten Ansatzes. Dabei wird die Arbeit von Kirchler als Ausgangspunkt verwendet und daraus nach der Analyse und dem Entwurf eine erweiterte Version des VO-Layers entwickelt. In der objektorientierten Analyse (Kapitel 2) werden die Anwendungsfälle erstellt und die daraus resultierenden Anforderungen aufgezeigt. Im Entwurf (siehe Kapitel 3) wird das statische sowie das dynamische Modell abgeleitet und der Objektentwurf angepasst. Außerdem wird das bereits vorhandene Datenbankmodell gemäß den Anforderungen ergänzt. Im Kapitel 4 wird der von Kirchler erarbeitete Prototyp nun zur Webanwendung weiterentwickelt. Abschließend wird aufgezeigt was mit dieser Arbeit erreicht wurde sowie welche Erweiterungen notwendig sind um dieses Konzept im D-Grid einsetzen zu können. Im letzten Kapitel wird die Arbeit nochmals kurz zusammengefasst.

Da diese Arbeit von zwei Personen durchgeführt wird, ist eine Aufgabenteilung notwendig. Diese wurde wie folgt festgelegt: Nachdem die Anforderungen und das Modell gemeinsam erarbeitet werden, wird der Benutzerbereich, die WatchList sowie die Change Requests von Andrea Nutsi implementiert. Das Management der Entities, die Statistik, die Authentifizierung sowie die Autorisierung wird von Matthias Kager umgesetzt.

1.3 Begriffsdefinition

In diesem Abschnitt wird kurz darauf eingegangen was unter bestimmten Begriffen in dieser Arbeit verstanden werden soll.

Zertifikat Als Zertifikat wird ein PKCS-12 User-Zertifikat angenommen, welches in einen Browser importiert werden kann und von einer vom D-Grid anerkannten CA signiert wurde.

Entity Mit Entities sind VOs, FQANs, Ressourcen, User, Group-, Role- sowie Capability-Type gemeint, also jene Typen die der Administrator direkt ändern und beeinflussen kann. Alle Entities haben ID, Name und Beschreibung, wobei die ID für die Datenbank eindeutig sein muss und aus diesem Grunde für den Anwender unveränderbar und nicht sichtbar ist.

Alle weiteren Fachbegriffe wurden in [Kir08] definiert und sollen an dieser Stelle nicht wiederholt werden.

2 Anforderungsanalyse

Im Laufe dieses Kapitels werden die Anforderungen an das zu entwickelnde System analysiert. Dazu werden die Anwendungsfälle des erweiterten Konzepts und der grafischen Benutzerschnittstelle des VO-Layers definiert um daraus die funktionalen und nicht-funktionalen Anforderungen abzuleiten.

2.1 Anwendungsfälle

In diesem Abschnitt werden die Anwendungsfälle erläutert, die die Weboberfläche des VO-Layers ermöglichen soll. Diese sind im Folgenden aufgelistet und werden auf den nächsten Seiten näher beschrieben:

- Authentifizierung und Autorisierung
- Benutzer beantragt VO-Beitritt, analog VO-Austritt sowie Hinzufügen und Entfernen von FQANs
- Zulassung eines neuen Zertifikats
- Registrierung eines neuen Benutzers
- VO anlegen, analog FQAN, RoleType, GroupType, CapabilityType und Ressource
- Daten einer VO ändern, analog RoleType, GroupType, CapabilityType, Benutzer und Ressource
- Inaktiv setzen eines GroupType, analog RoleType und CapabilityType
- Inaktiv setzen einer Ressource, analog User
- Inaktiv setzen eines FQAN
- Inaktiv setzen einer VO
- User auf WatchList setzen
- User von WatchList entfernen

Anwendungsfall	Authentifizierung und Autorisierung
Akteure	Benutzer, DB
Vorbedingung	Benutzer möchte auf das System zugreifen
Nachbedingung	Benutzer befindet sich auf der Startseite und seine Rolle ist dem System bekannt (User oder Admin)
Primärszenario	1. User sendet mit Hilfe seines Browsers sein Zertifikat an das System 2. System prüft, ob Zertifikat gültig ist 3. Hole Benutzer (mit seinen VOs und FQANs) aus DB 4. Identifiziere Benutzerrolle 5. Zeige Startseite des Benutzers an
Sekundärszenarien	• Zertifikat nicht gültig • Benutzer existiert nicht in DB (siehe Anwendungsfall Registrierung eines neuen Benutzers in Abbildung 2.4) • DB nicht erreichbar

Abbildung 2.1: Anwendungsfall Authentifizierung und Autorisierung

Anwendungsfall	Benutzer beantragt VO-Beitritt
Akteure	Benutzer, Admin, DB
Vorbedingung	Benutzer ist angemeldet
Nachbedingung	Benutzer erfolgreich einer VO beigetreten
Primärszenario	1. Benutzer wählt VO-ChangeRequest 2. Benutzer teilt dem System durch 'add' mit, dass er einer VO beitreten will 3. Benutzer wählt die VO der er beitreten möchte und schreibt eventuell einen Kommentar dazu 4. System sendet Antrag an DB 5. Antrag erscheint als Change Request auf der Übersichtsseite der offenen Change Requests des Admins der VO 6. Admin ruft die Detailansicht des Change Requests auf 7. Admin gibt Antrag statt und fügt eventuell einen Kommentar hinzu 8. System sendet die Änderungen an die DB, User wird der VO hinzugefügt und der Change Request wird als bearbeitet markiert 9. Dem Benutzer wird dieser Change Request auf seiner Startseite als bearbeitet angezeigt und er bestätigt dessen Kenntnisnahme 10. Der Change Request wird aus der DB gelöscht
Sekundärszenarien	• DB nicht erreichbar • Admin lehnt Antrag ab

Abbildung 2.2: Anwendungsfall Benutzer beantragt VO-Beitritt. Antrag auf Hinzufügen von FQANs sowie Austritt aus VO und Entfernen von FQANS erfolgt analog dazu. Im Falle von FQAN wird beim ersten Punkt ein FQAN-ChangeRequest ausgewählt und im dritten die VO und anschließend die FQAN gewählt. Will der User aus der VO austreten oder eine FQAN entfernen, wird dem System dies bei Schritt zwei durch 'remove' mitgeteilt.

Anwendungsfall	Zulassung eines neuen Zertifikats
Akteure	Benutzer, DB, Admin
Vorbedingung	Benutzer ist angemeldet, besitzt neues Zertifikat und ist Mitglied einer VO
Nachbedingung	Der Benutzer kann sich mit dem neuen (gültigen) Zertifikat am System anmelden
Primärszenario	1. Benutzer wählt Cert-ChangeRequest und sendet neues Zertifikat an das System, eventuell zusammen mit einem Kommentar 2. System prüft, ob Zertifikat gültig ist 3. System sendet Antrag an DB, dabei wird der Antrag einem Administrator zugeteilt der berechtigt ist diesem Benutzer zu verwalten 4. Antrag erscheint als Change Request auf der Übersichtsseite der offenen Change Requests des Administrators 5. Admin ruft die Detailansicht des Change Requests auf 6. Admin gibt Antrag statt 7. System sendet die Änderungen an die DB und der Change Request wird als bearbeitet markiert 8. Dem Benutzer wird dieser Change Request auf seiner Startseite als bearbeitet angezeigt und er bestätigt dessen Kenntnisnahme 9. Der Change Request wird aus der DB gelöscht
Sekundärszenarien	• Zertifikat nicht gültig oder von einer nicht anerkannten CA signiert • Zertifikat hat falsches Format • DB nicht erreichbar • Admin lehnt Antrag ab

Abbildung 2.3: Anwendungsfall Zulassung eines neuen Zertifikats

Anwendungsfall	Registrierung eines neuen Benutzers
Akteure	Benutzer, DB, Admin
Vorbedingung	Benutzer existiert noch nicht im System und er besitzt ein gültiges, vom System akzeptiertes Zertifikat
Nachbedingung	Benutzerdaten im System gespeichert
Primärszenario	1. Benutzer landet auf Begrüßungsseite und kann eine VO auswählen der er beitreten möchte, außerdem kann er einen Kommentar mitsenden 2. Benutzer schickt Daten ab und bekommt Namen und Emailadresse des Administrators der gewählten VO angezeigt. 3. System sendet Antrag an DB 4. Antrag erscheint als Change Request auf der Übersichtsseite der offenen Change Requests des Administrators 5. Admin ruft die Detailansicht des Change Requests auf 6. Admin gibt Antrag statt 7. System prüft, ob Zertifikat gültig ist 8. System sendet die Änderungen an die DB und der Change Request wird als bearbeitet markiert 9. Dem Benutzer wird dieser Change Request auf seiner Startseite als bearbeitet angezeigt und er bestätigt dessen Kenntnisnahme 10. Der Change Request wird aus der DB gelöscht
Sekundärszenarien	• Zertifikat nicht gültig • Zertifikat hat falsches Format • DB nicht erreichbar • Admin lehnt Antrag ab

Abbildung 2.4: Anwendungsfall Registrierung eines neuen Benutzers

Anwendungsfall	VO anlegen
Akteure	Admin, DB
Vorbedingung	Admin ist angemeldet
Nachbedingung	Eine neue VO wurde angelegt
Primärszenario	1. Admin gibt Daten (Name und Beschreibung) für die neue VO in das entsprechende Formular ein und sendet diese an das System 2. System prüft Daten auf Vollständigkeit 3. System sendet Daten an DB 4. System legt Standard-FQANs für diese VO in der DB an, Absender wird Admin dieser neuen VO
Sekundärszenarien	• Daten unvollständig • DB nicht erreichbar

Abbildung 2.5: Anwendungsfall VO anlegen. Analog FQAN, RoleType, GroupType, CapabilityType, Ressource anlegen. Im Falle von FQAN muss kein Name angegeben werden. Dafür muss die Kombination von RoleType, GroupType, CapabilityType sowie die zugehörige VO ausgewählt werden. Beim Anlegen einer Ressource wird der distinguished name, der Hostname, die Beschreibung der Ressource, eine Email-Adresse sowie die zugehörige Middleware eingetragen. Außerdem muss aus den VOs des Administrators eine VO ausgewählt werden in der die Ressource Mitglied sein soll.

Anwendungsfall	Daten einer VO ändern
Akteure	Admin, DB
Vorbedingung	Admin ist angemeldet und VO existiert
Nachbedingung	Daten der VO wurden geändert
Primärszenario	1. Admin wählt die VO deren Daten er ändern möchte 2. Admin ändert die Daten der VO im entsprechenden Formular und sendet diese an das System 3. System prüft Daten auf Vollständigkeit 4. System sendet Daten an DB
Sekundärszenarien	• Daten unvollständig • DB nicht erreichbar

Abbildung 2.6: Anwendungsfall Daten einer VO ändern. Analog Daten von RoleType, GroupType, CapabilityType, Benutzer und Ressourcen ändern.

Anwendungsfall	Inaktiv setzen eines GroupType
Akteure	Admin, DB
Vorbedingung	Admin ist angemeldet und GroupType ist aktiv
Nachbedingung	FQANs mit diesem GroupType sind inaktiv und die Zuordnungen von Benutzern und Ressourcen zu diesen FQANs sind entfernt
Primärszenario	1. Admin klickt bei entsprechendem GroupType auf 'deactivate' 2. System setzt in der DB den Status der FQANs, die den gewählten GroupType enthalten und einer VO zugehören die der Administrator verwalten darf, auf inaktiv 3. System entfernt in der DB die Zuordnungen von Usern und Ressourcen zu diesen FQANs
Sekundärszenarien	• DB nicht erreichbar • FQAN enthält eine Admin-Rolle

Abbildung 2.7: Anwendungsfall GroupType deaktivieren, analog inaktiv setzen von RoleType und CapabilityType

Anwendungsfall	Inaktiv setzen einer Ressource
Akteure	Admin, DB
Vorbedingung	Admin ist angemeldet und Ressource ist aktiv
Nachbedingung	Zuordnung der FQANs und VOs zu der Ressource sind entfernt
Primärszenario	1. Admin klickt bei entsprechendem Ressource auf 'deactivate' 2. System entfernt in der DB die Zuordnungen von FQANs und VOs des Administrators zu dieser Ressource
Sekundärszenarien	• DB nicht erreichbar

Abbildung 2.8: Anwendungsfall Ressource deaktivieren, analog inaktiv setzen von User

Anwendungsfall	Inaktiv setzen einer VO
Akteure	Admin, DB
Vorbedingung	Admin ist angemeldet und VO ist aktiv
Nachbedingung	VO ist inaktiv und kann nicht mehr verwendet werden
Primärszenario	1. Admin klickt bei entsprechendem VO auf 'deactivate' 2. System setzt in der DB den Status der VO auf inaktiv
Sekundärszenarien	• DB nicht erreichbar

Abbildung 2.9: Anwendungsfall VO deaktivieren

Anwendungsfall	Inaktiv setzen eines FQAN
Akteure	Admin, DB
Vorbedingung	Admin ist angemeldet und FQAN ist aktiv
Nachbedingung	FQAN ist nicht mehr aktiv und Zuordnungen von Usern und Ressourcen zu diesem FQAN wurden entfernt
Primärszenario	1. Admin klickt bei entsprechendem FQAN auf 'deactivate' 2. System setzt in der DB den Status des entsprechenden FQAN der VO auf inaktiv 3. System entfernt in der DB die Zuordnungen von Usern und Ressourcen zu diesem FQAN
Sekundärszenarien	• DB nicht erreichbar

Abbildung 2.10: Anwendungsfall FQAN deaktivieren

Anwendungsfall	User auf WatchList setzen
Akteure	Admin, DB
Vorbedingung	Admin ist angemeldet und Benutzer ist Mitglieder einer VO des Admins
Nachbedingung	Der Benutzer steht auf der WatchList einer VO des Admins
Primärszenario	1. Admin wählt einen Benutzer aus, der auf die WatchList soll 2. Admin wählt VO aus der der Benutzer ausgeschlossen werden soll, gibt eventuell einen Kommentar ein und sendet dies an das System 3. System sendet Daten an DB
Sekundärszenarien	• DB nicht erreichbar • Benutzer nicht Mitglied der gewählten VO • Benutzer ist Admin der gewählten VO: diese Rolle wird dem Benutzer entzogen

Abbildung 2.11: Anwendungsfall User auf WatchList setzen

Anwendungsfall	User von WatchList entfernen
Akteure	Admin, DB
Vorbedingung	Admin ist angemeldet und Benutzer steht auf WatchList bei einer VO des Admins
Nachbedingung	Der Benutzer steht in der WatchListHistory und nicht mehr auf der WatchList dieser VO
Primärszenario	1. Admin wählt 'remove' bei entsprechendem Datensatz 2. System sendet Daten an DB
Sekundärszenarien	• DB nicht erreichbar

Abbildung 2.12: Anwendungsfall User von WatchList entfernen

2.2 Funktionale Anforderungen

An dieser Stelle folgen nun die funktionalen Anforderungen an eine GUI, welche aus den Anwendungsfällen abgeleitet wurden. Grundsätzlich gibt es zwei Benutzerrollen die berücksichtigt werden müssen: die Administratoren und die normalen Benutzer, also jene ohne die Rolle `VO_Admin`. Beide müssen sich für die Anwendung authentifizieren. Die GUI muss außerdem die Möglichkeit bieten, die VO-Struktur mit ihren Entitäten zu visualisieren und einfache Änderungen ermöglichen. Für jede Tätigkeit muss der Benutzer jeweils autorisiert sein.

2.2.1 Authentifizierung und Autorisierung

Zur Authentifizierung an der Webanwendung wird ein Zertifikat des Benutzers benötigt. Der Benutzer wird nach erfolgreicher Authentifizierung auf die Startseite des VO-Layers geleitet. An dieser Stelle muss die Identität des Benutzers festgestellt werden, um zu wissen, ob es sich um einen Administrator oder einen normalen Benutzer handelt, da das Menü entsprechend anzupassen ist.

Ein Administrator darf im Management-Bereich nur jene VOs mit zugeordneten Usern, Ressourcen und FQANs verwalten, in denen er eine Rolle als Administrator einnimmt.

Im nächsten Abschnitt werden wir uns dem Benutzer-Bereich widmen.

2.2.2 Der Benutzer-Bereich

Zunächst wird jedem Benutzer nach erfolgreicher Anmeldung sein Status angezeigt. Er sieht in welchen VOs er Mitglied ist und welche FQANs er besitzt. Auch die Daten seines Zertifikates werden dem Anwender an dieser Stelle visualisiert.

Im Folgenden werden die Funktionalitäten, die einem Benutzer zur Verfügung stehen, näher erläutert.

Statusanzeige

Benutzer des VO-Layers bekommen von der GUI folgende Informationen angezeigt: Informationen aus ihrem Zertifikat, Informationen zu den VOs in denen sie involviert sind, sowie offene oder zu bestätigende Change Requests. Zu ersteren gehören die Bestandteile des distinguished name, die dem User in einer übersichtlichen Form einzeln angezeigt werden sollen sowie der Aussteller mit seinem DN und die Lebensdauer des Zertifikats. Des Weiteren soll dem User präsentiert werden in welchen Rollen und Gruppen er in den verschiedenen VOs ist. Steht der User auf der WatchList einer VO wird dies ebenfalls visualisiert. Dem Benutzer wird auch der Name des jeweiligen VO Admins sowie dessen Email-Adresse als Kontaktmöglichkeit angezeigt.

Außerdem soll ein Benutzer nach Anforderung die Ressourcen der VOs einsehen können bei denen er Mitglied ist. Die Anzeige sollte dabei auch die verwendete Middleware, den Hostname sowie die Emailadresse der Ressource enthalten.

Druckansicht

Die Benutzer haben die Möglichkeit sich die angezeigten Informationen auszudrucken. Dafür wird eine speziell aufbereitete Druckansicht zur Verfügung gestellt bei der überflüssige Informationen, wie beispielsweise das Menü, nicht mehr enthalten sind.

Change Requests

Weiterhin ist es von Bedeutung, dass ein Benutzer Change Requests an den Administrator schicken kann. Mittels eines Change Requests kann ein User den Beitritt zu einer VO oder Austritt aus einer VO beantragen. Außerdem können Change Requests Änderungswünsche, wie Hinzufügen oder Entfernen von FQANs oder die Registrierung eines neuen Zertifikats, beinhalten. Auch die Registrierung eines neuen Benutzers für den VO-Layer muss durch die grafische Oberfläche unterstützt werden.

2.2.3 Der Administrator-Bereich

Da jeder Administrator zugleich auch ein normaler Benutzer ist, beginnt auch er im Benutzer-Bereich. Außerdem stehen ihm eine Reihe von Management-Funktionen zur Verfügung, die im Admin-Bereich liegen und in diesem Abschnitt näher betrachtet werden.

Change Requests bearbeiten

Ein Administrator muss die vom Benutzer abgesetzten Change Requests bearbeiten, d.h. bestätigen oder ablehnen können. Bei dieser Aktion wird ihm die Möglichkeit geboten einen Kommentar hinzuzufügen. An dieser Stelle muss darauf geachtet werden, dass auch hier der Administrator für diese Aktion autorisiert ist, d.h. möchte ein Benutzer beispielsweise einer VO beitreten so muss der Administrator genau für diese VO Admin-Rechte haben um diesen Change Request zu bearbeiten. Andernfalls sollte er diesen Change Request nicht angezeigt bekommen.

VO verwalten

Einer VO können User, Ressourcen und SubVOs in beliebiger Anzahl zugeordnet sein. Außerdem können beliebig viele FQANs für eine VO erstellt werden, aber jedes FQAN kann nur einer VO zugeordnet sein. Der Administrator muss somit die Möglichkeit haben, SubVOs zu einer VO hinzuzufügen oder zu entfernen. Im Rahmen dieser Arbeit kann eine VO mit Namen und Beschreibung angelegt werden, wobei derjenige die Rolle `VO_Admin` erhält, der die VO erzeugt hat. Hierfür wird automatisch das notwendige FQAN erzeugt und dem Admin zugeordnet. Außerdem werden für diese neue VO Standard-FQANs erzeugt. Eine Liste dieser findet sich in [DGr09].

Ressourcen verwalten

Zusätzlich zu den bekannten Eigenschaften einer Entity besitzt eine Ressource einen eindeutigen Bezeichner (distinguished name), einem Hostnamen sowie eine Email-Adresse. Eine Ressource ist genau einer Middleware zugeordnet. Sie kann beliebig vielen VOs zugeteilt werden und in diesen eine beliebige Anzahl von FQANs besitzen.

User verwalten

Der Benutzer besitzt neben ID, Name und Beschreibung einen eindeutigen Bezeichner (distinguished name), eine Email-Adresse sowie einen öffentlichen Schlüssel. Ein User kann durch den Administrator einer VO hinzugefügt oder entfernt werden sowie FQANs erhalten oder entzogen bekommen. Hierbei ist darauf zu achten, dass die Rolle `VO_Admin` mindestens einmal pro VO erhalten bleiben muss. Sollte versucht werden den letzten Administrator zu entfernen so scheitert diese Aktion.

FQAN verwalten

Ein FQAN stellt eine Zugriffsberechtigung dar, welcher wiederum aus Rollen, Fähigkeiten und Gruppen besteht. Dies ist nötig, da verschiedene Middlewares verschiedene Mechanismen zur Autorisierung verwenden. Der Administrator muss in der Lage sein neue FQANs, Gruppen, Rollen und Fähigkeiten zu erzeugen. Gruppen, Rollen und Fähigkeiten können modifiziert werden, FQANs jedoch nicht. Wie bereits erwähnt, muss ein FQAN genau einer VO zugeordnet werden. Andererseits kann ein FQAN einer beliebigen Anzahl von Ressourcen und Benutzern zugeordnet werden.

Middleware und CA

Eine Möglichkeit, Middlewares oder CAs zu bearbeiten oder hinzuzufügen ist momentan im Rahmen der GUI des VO-Layers nicht vorgesehen. Bei der Einführung einer neuen Version einer Middleware müsste der VO-Layer angepasst werden. Da davon ausgegangen wird, dass sich CAs nicht regelmäßig ändern, wird auch auf diesen Punkt verzichtet. CAs werden einmal bei der Installation des VO-Layers festgelegt (vgl. Abschnitt 4.3).

Detailansicht von VO, Ressourcen, User und FQAN

Bei allen Übersichtsseiten des Admin-Bereichs gibt es die Möglichkeit Detailinformationen zur gewählten Entity abzurufen, welche aus Platzgründen nicht auf der Übersichtsseite auf-

geführt sind. Dabei können zu den in der Detailansicht angezeigten Entities wiederum die Details dargestellt werden.

WatchList

Falls ein Benutzer durch Fehlverhalten auffällig wird, muss es die Möglichkeit geben, diesen aus einer VO auszuschließen. Sein Name erscheint dann auf der WatchList. Dem User bleiben seine FQANs und die Mitgliedschaft in dieser VO erhalten. Eine Ausnahme bildet die Rolle `VO_Admin`, welche dem Benutzer entzogen wird. Allerdings kann ein Administrator einen Benutzer nur aus den VOs ausschließen in denen er auch die Rolle `VO_Admin` besitzt. Nach dem Ausschluss hat das ehemalige Mitglied weiterhin Zugriff auf den VO-Layer, kann aber für die VO aus der er ausgeschlossen wurde keine FQANs mehr beantragen. Für den Administrator besteht die Möglichkeit einen Benutzer wieder von der WatchList zu entfernen, was einen neuen Datensatz in der WatchListHistory zur Folge hat. Der User erhält in diesem Fall seine früheren FQANs zurück und ist wieder Mitglied der VO. War er vorher Administrator dieser VO so muss er diese Rolle nun neu beantragen.

Druckansicht

Die Druckansicht steht auch für den Administrator-Bereich zur Verfügung.

Statistiken

Dem Administrator stehen außerdem Statistikfunktionen zur Verfügung mit denen Veränderungen in der VO-Struktur, d.h. VO-Eintritt oder VO-Austritt eines Benutzers sowie Hinzufügen oder Entfernen von FQANs, statistisch erfasst und dargestellt werden. Es wird im Rahmen dieser Arbeit eine globale Statistik geben in die alle Administratoren Einsicht haben. Pro VO gibt es eine Statistik-Tabelle über die folgenden Zeiträume: letzte Woche / Monat / 6 Monate / Jahr. Zusätzlich soll auch noch die aktuelle Anzahl der VO-Mitglieder angezeigt werden. Die Statistik für FQANs erfolgt analog. Für diese Statistik soll es eine Export-Möglichkeit geben.

2.3 Nicht-Funktionale Anforderungen

Im Folgenden werden die von [Kir08] als nicht-funktionale Anforderungen bezeichneten Punkte aufgegriffen und in den Zusammenhang dieser Arbeit gestellt.

2.3.1 Sprache

Die Anwendung wird nur in englischer Sprache zur Verfügung gestellt. Eine Erweiterung auf mehrere Sprachen ist nicht vorgesehen und wird daher in dieser Arbeit vernachlässigt.

2.3.2 Benutzerfreundlichkeit

Im Laufe dieser Arbeit wird auf diesen Punkt sehr viel Wert gelegt. Benutzerfreundlichkeit wird unter anderem erreicht durch ein einheitliches Layout, intuitives Design sowie gut lesbare Schrift mit hohem Kontrast.

2.3.3 Zuverlässigkeit

Die Benutzer sollen stets über den Status ihrer Aktion informiert werden. Bei Fehlern wird eine angemessene Meldung angezeigt.

2.3.4 Sicherheit

Der Benutzer muss sich mit einem Grid-Zertifikat identifizieren. Nach dessen Überprüfung erhält er Zugang zur grafischen Oberfläche. Die Administratoren werden durch die Rolle `VO_Admin` als solche erkannt. Jeder sieht immer nur das, wofür er auch autorisiert wurde.

2.3.5 Leistung

Die Antwortzeiten werden durch die Trennung der Darstellung und Verarbeitungslogik vom Inhalt verbessert. Dabei wird die zu übertragende Menge an Informationen pro Anfrage verringert, wodurch die Ladezeit verbessert wird. Durch asynchrone Anfragen an den Server könnten diese weiter optimiert werden.

2.3.6 Skalierbarkeit

Die grafische Oberfläche muss so konzipiert werden, dass auch große Datenmengen gut verwaltet werden können. Dabei bietet es sich an, Details nur auf Anforderung anzuzeigen und dynamisch nachzuladen.

2.3.7 Portabilität

Der zugrunde liegende VO-Layer ist zur Zeit nur auf Linux lauffähig. Da es sich bei der zu entwickelnden GUI um eine Webanwendung handelt, benötigt man für die Verwendung nur einen Webbrowser und ist somit plattformunabhängig. Um eine korrekte Darstellung in möglichst vielen Browsern zu gewährleisten, muss auf standardkonformen Code geachtet werden.

3 Systementwurf

Nachdem im vorherigen Kapitel die Anforderungen an die Webanwendung definiert wurden, wird in diesem der daraus resultierende Systementwurf vorgestellt. Dazu werden im ersten Abschnitt die Subsysteme definiert und das daraus resultierende Gesamtsystem dargestellt. Im Anschluss wird im Objektentwurf das statische und dynamische Modell hergeleitet um im dritten Abschnitt das Datenbankschema daraus zu erzeugen.

3.1 Gesamtsystem und Subsysteme

Im Folgenden werden die Änderungen an den bereits vorhandenen Subsystemen sowie die neu hinzukommenden Subsysteme beschrieben. Die Abhängigkeiten der Subsystem im Gesamtsystem sind in Abbildung 3.1 dargestellt.

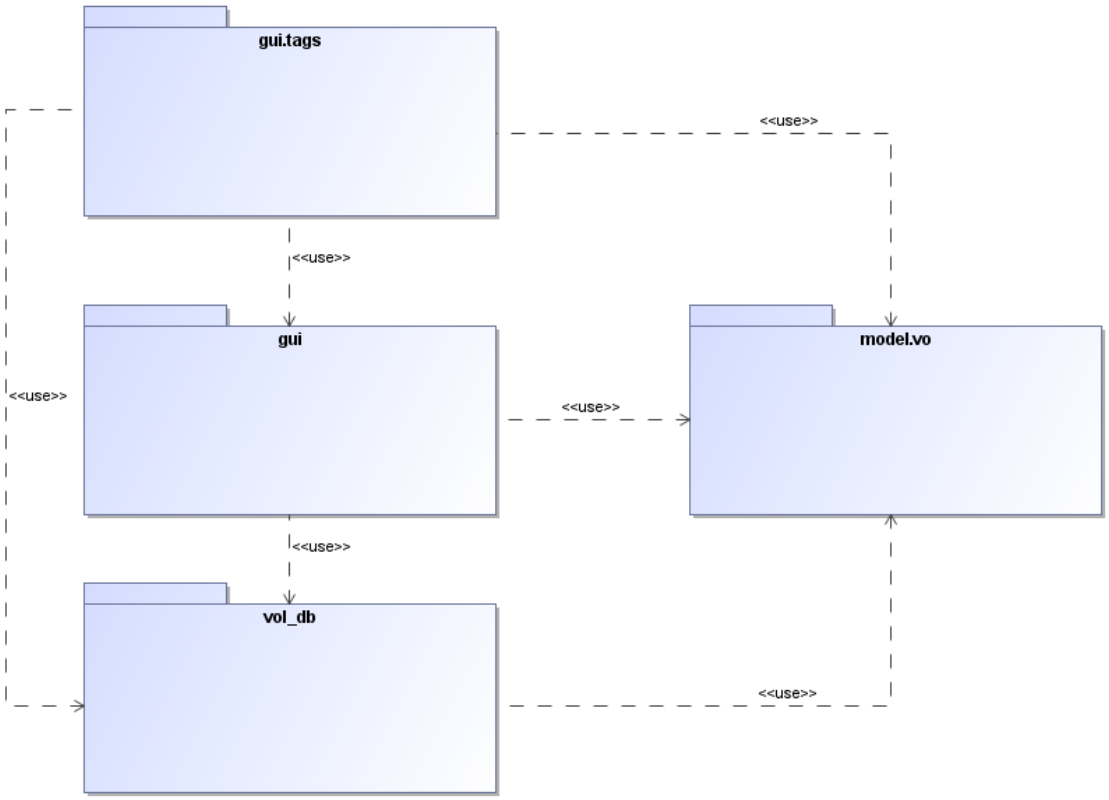

Abbildung 3.1: Abhängigkeiten der Subsysteme im Gesamtsystem

3.1.1 Subsystem gui

Das Subsystem `gui` beinhaltet alle Klassen, welche für die Darstellung sowie die Anwendungslogik der Webanwendung relevant sind. Dabei beinhaltet dieses Package die Klassen `ManageEntity` und `Login`. `ManageEntity` dient dem Auslesen der Request-Parameter und die daraus folgende Objektgenerierung. Die erzeugten Objekte werden von dieser Klasse anschließend validiert und an den `DataManager` weitergereicht. Die Klasse `Login` stellt Methoden bereit mit denen geprüft wird, ob sich der User bereits beim System registriert hat und welche Rechte er besitzt. In Abbildung 3.2 ist das Klassendiagramm des Subsystems dargestellt. Das im Folgenden beschriebene Subsystem `gui.tags` ist dabei ebenfalls in diese Abbildung integriert.

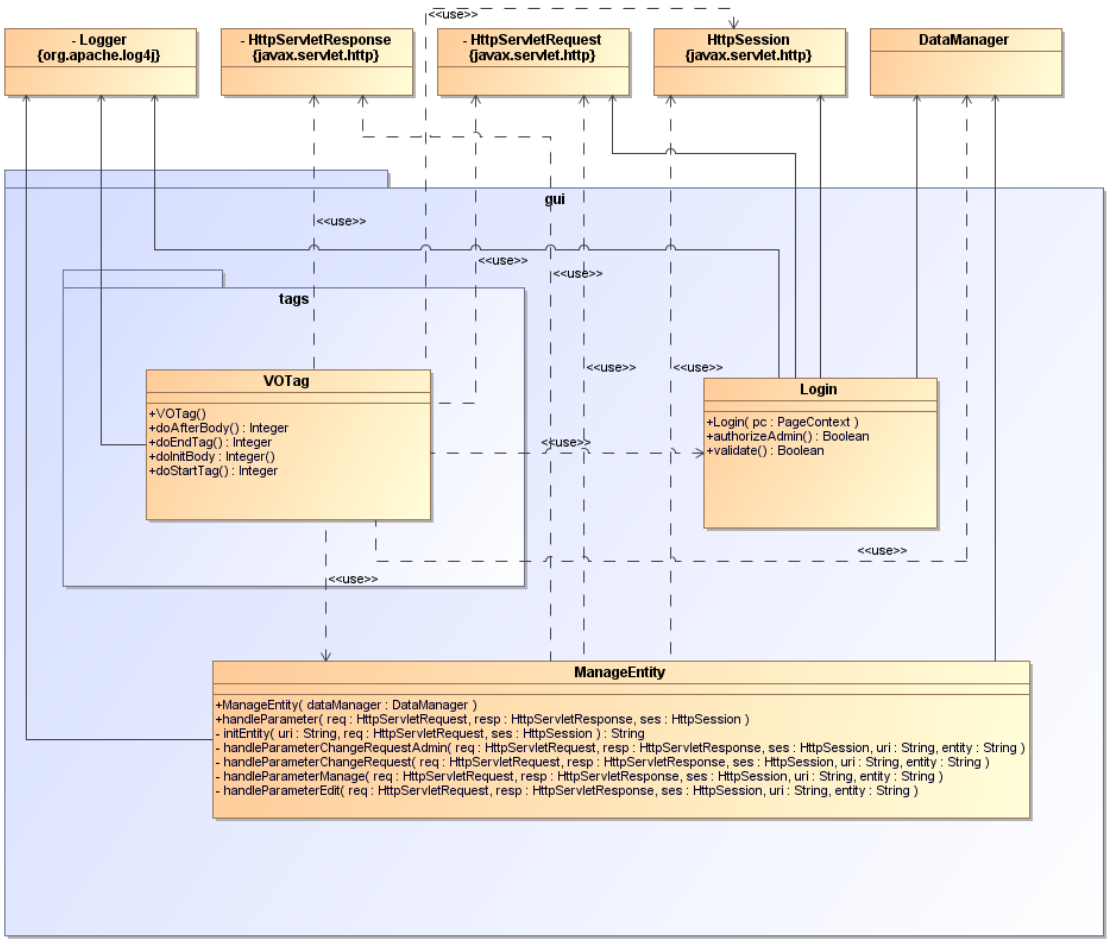

Abbildung 3.2: Klassendiagramm des Subsystems gui

Subsystem gui.tags

Die Klasse `VOTag` des Subsystems `gui.tags` bildet den Kern der Anwendung. Hierbei werden alle benötigten Objekte aus der Session gelesen. Falls sie noch nicht vorhanden sind, werden sie erstellt. Daraufhin werden die Klassen des Subsystems `gui` instanziiert. Da nur die Klasse

`VOTag` auf das Session- und Request-Objekt zugreifen kann, werden diese bei den folgenden Methodenaufrufen an die beiden Objekte der Klassen `Login` und `ManageEntity` übergeben. Außerdem wird an dieser Stelle entschieden, welche Menüpunkte dem User angezeigt werden.

3.1.2 Subsystem model.vo

Das im Objektentwurf zu erstellende statische Modell bildet zugleich das Subsystem `model.vo`. Diese Klassen werden von der Webanwendung als Java Beans verwendet um die eingegebenen und angezeigten Daten zu halten. Entsprechend werden bei diesen Klassen die nötigen Getter und Setter definiert.

3.1.3 Subsystem vol_db

Im Subsystem `vol_db` befindet sich die Logik zur Datenbankanbindung. Dieses bereits vorhandene Package muss teilweise angepasst werden, da es sich in der bestehenden Form für Webanwendungen nicht eignet. Aufgrund der Verwendung von Threads und dem Observer-Pattern wird das Ergebnis einer Anfrage an die Datenbank asynchron an die Anwendung zurückgegeben, wodurch dem Benutzer keine direkte Anzeige des Ergebnisses präsentiert werden kann. Daher wird der Thread nicht verwendet und entfernt. Außerdem wird im `DBManager` eine Methode `runQuery` definiert, die das Ergebnisobjekt als Rückgabewert liefert.

Wie in Abbildung 3.3 dargestellt, hält ein `DataManager` den `DBManager` welcher die Anfragen an die Klasse `VolDBConnection` weiterreicht. In der Abbildung sind der Einfachheit halber nur die Klassen aufgeführt die modifiziert wurden. Jeder Benutzer hält in der Session ein Objekt der Klasse `DataManager`, wodurch sichergestellt wird, dass jeder Benutzer eine eigene Datenbankverbindung verwendet, die unabhängig von anderen Usern agiert. Dies ist ebenfalls eine Änderung dieses Subsystems, da die Aufrufe des `DataManager` ursprünglich nur statisch erfolgten und der `DBManager` als Singleton konzipiert wurde. Dies führte dazu, dass jeder Benutzer mit derselben Verbindung arbeitete. Um die Anwendung vor SQL-Injection zu schützen, werden Datenbankabfragen mit PreparedStatements durchgeführt. Dies führt dazu, dass die SQL Queries vorkompiliert werden und bei der Abfrage nur noch die Werte eingesetzt werden. Eine Manipulation der eigentlichen Anfrage ist somit nicht mehr möglich.

3.2 Objektentwurf

In diesem Abschnitt wird das statische und dynamische Modell entwickelt und aus diesem entstandenen Modell der Objektentwurf abgeleitet.

3.2.1 Statisches Modell

Das statische Modell von [Kir08] wird um die Klassen für die WatchList sowie für die Change Requests erweitert. Außerdem werden noch weitere Ergänzungen getätigt, welche im Folgenden erläutert werden. Das resultierende Klassendiagramm des Modells kann Abbildung 3.4 entnommen werden.

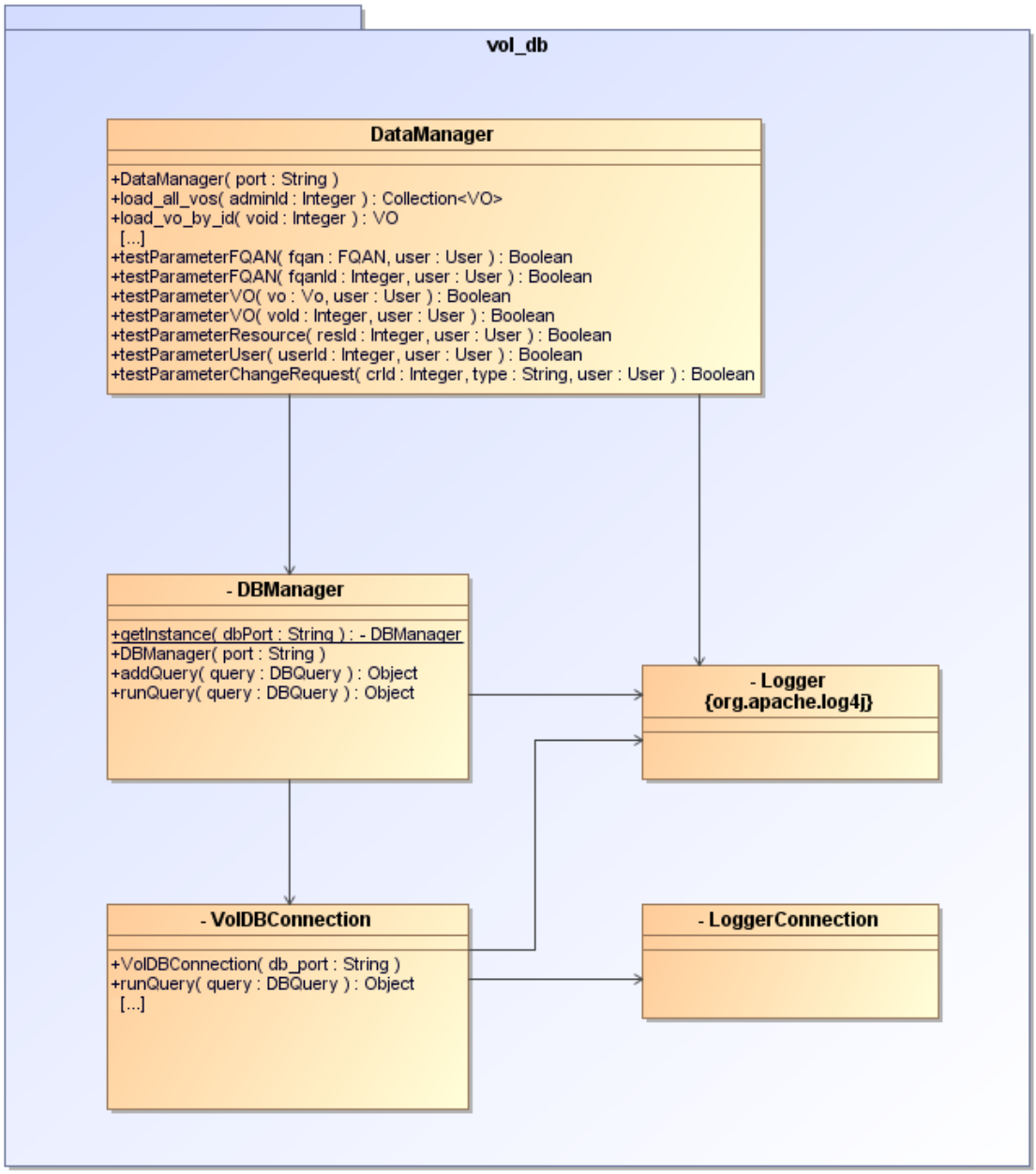

Abbildung 3.3: Klassendiagramm des Subsystems vol_db

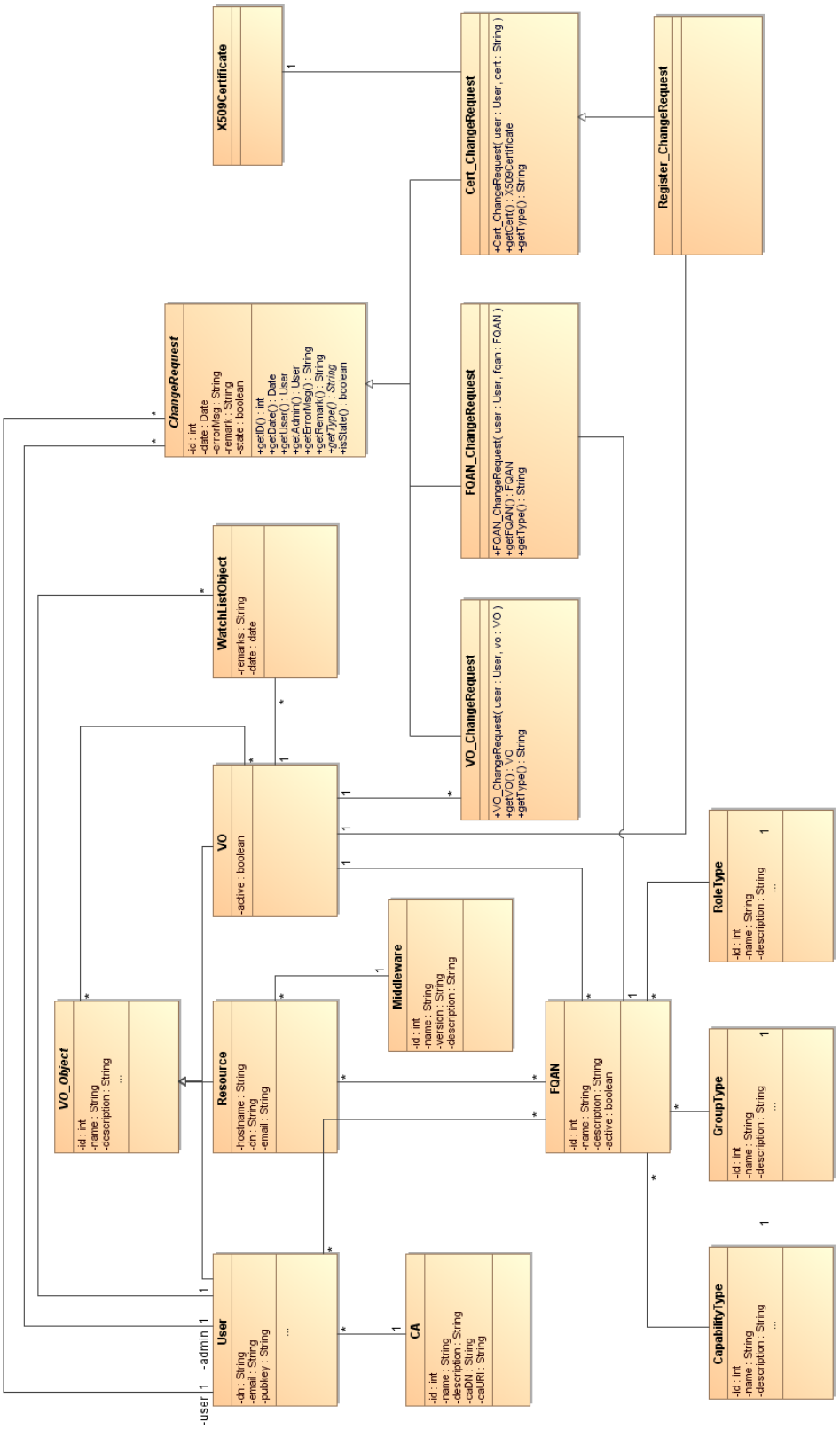

Abbildung 3.4: statisches Modell des VO-Layers

WatchList

Die Klasse `WatchListObject` repräsentiert einen Eintrag der WatchList und benötigt daher jeweils eine Assoziation zu User und eine zu VO. Zusätzlich erhält die Klasse Attribute für eine Bemerkung und das Datum an welchem ein Benutzer aus der VO ausgeschlossen wurde.

Change Request

Bei den Change Requests werden verschiedene Typen und dementsprechend verschiedene Klassen unterschieden. Mithilfe von Change Requests können Ein- und Austritte von VOs, das Entfernen oder Hinzufügen von FQANs oder die Zulassung eines neues Zertifikats beantragt werden. Daher wurde, wie Abbildung 3.4 entnommen werden kann, eine Oberklasse `ChangeRequest` mit den Attributen `id`, `date` für das Datum des Change Requests, `errorMsg` für eventuell auftretende Fehler, `remark` für Bemerkungen des Antragstellers und des Administrators und `state` für den Status, d.h. noch zu bearbeiten oder bearbeitet, eingeführt. Zusätzlich dazu besitzt die Klasse zwei Assoziationen der Klasse User mit den Assoziationsrollen user und admin. Die Rolle user entspricht dabei dem User der den Change Request abgesendet hat und die Rolle admin dem zugeordneten Administrator, welcher den Antrag zu bearbeiten hat. Bemerkungen werden aneinandergefügt und enthalten auch, ob und wann ein Antrag angenommen oder abgelehnt wurde. Der Status wird für die Anzeige des Antragstellers verwendet, der all seine abgesetzten Change Requests auf der Startseite angezeigt bekommt. Wurde ein Change Request bearbeitet, so muss der Benutzer die Kenntnisnahme bestätigen. Daraufhin wird der Change Request aus der Datenbank entfernt.

Die Information, ob ein User sich einer VO anschließen, oder von ihr austreten will, wird nicht gespeichert, da dies aus den vorhandenen Einträgen in der Datenbank leicht nachvollziehbar ist. Dasselbe gilt beim Entfernen und Hinzufügen von FQANs. Für die vier verschiedenen Change Request Arten gibt es die speziellen Unterklassen `VO_ChangeRequest`, `Cert_ChangeRequest` sowie `FQAN_ChangeRequest`. Für die Registrierung wird die Klasse `Register_ChangeRequest` als Unterklasse von `Cert_ChangeRequest` eingeführt. Die Subklassen beinhalten jeweils eine Assoziation zu den Klassen für die der User den Request gesendet hat, also `VO`, `FQAN` oder `X509Certificate`.

Deaktivierung von Entities

Entities können vom Administrator geändert, aber nicht aus der Datenbank gelöscht werden. Anstelle dessen können diese auf inaktiv gesetzt und auch wieder aktiviert werden. Daher bekommen die entsprechenden Klassen das zusätzliche Attribut `active` mit den zugehörigen Gettern und Settern. Allerdings ist eine Deaktivierung der Entitäten GroupType, RoleType und CapabilityType nicht vorgesehen. Die Deaktivierung anstelle des richtigen Löschens ist eine Änderung des bisherigen VO-Layers.

3.2.2 Dynamisches Modell

In diesem Unterkapitel werden die neu hinzukommenden Anwendungsfälle, falls nötig, mit Hilfe von Sequenzdiagrammen beschrieben. Bei ähnlichen Anwendungsfällen wird nur jeweils ein Anwendungsfall durch ein Sequenzdiagramm exemplarisch dargestellt. Als Schnittstelle zur Datenbank dient in diesem Modell der `DataManager`, der bereits im Prototyp von [Kir08] eingesetzt wird und im Subsystem `vol_db` zu finden ist.

Authentifizierung und Autorisierung

Zur Authentifizierung wird vom Benutzer ein Zertifikat verlangt, welches vom System geprüft und mit dem Datenbestand abgeglichen wird. Anschließend wird ermittelt, ob der User die Rolle `VO_ADMIN` besitzt.

Im ersten Schritt wird das Zertifikat in der Session gespeichert, damit dieses nicht bei jedem Aufruf neu geladen werden muss. Mit Hilfe der Daten des Zertifikats wird der entsprechende Datensatz des Users bei jeder Anfrage an die Anwendung aus der Datenbank gelesen, damit die Daten stets aktuell sind.

Falls das System in der Datenbank keinen passenden Eintrag findet, wird die Anfrage zur Registrierungsseite weitergeleitet. Dort wählt der User eine VO aus, der er beitreten möchte. Diese Information wird zusammen mit dem Zertifikat des Users als `Register_ChangeRequest` gespeichert und wird dem Administrator der gewählten VO angezeigt.

Der nun folgende Schritt wird nur ausgeführt falls der User bereits einen Eintrag in der Datenbank hat. Mithilfe des User-Objektes kann bestimmt werden, ob der Benutzer Adminrechte besitzt oder nicht. Sollte der User ohne Adminrechte auf eine Administrationsseite gelangen, wird er auf eine Fehlerseite weitergeleitet. Unerlaubte Aktionen des Administrators, beispielsweise Parametermanipulation, werden vom Datamanager behandelt. Der zeitliche Ablauf der Authentifizierung und Autorisierung ist in Abbildung 3.5 dargestellt.

Benutzer beantragt VO-Beitritt oder VO-Austritt

Die Abbildung 3.6 zeigt den zeitlichen Ablauf eines VO-Change Requests. Beim ersten Aufruf stellt der User einen Change Request mittels des entsprechenden Formulares des VO-Layers. Dabei sendet er an das System welcher VO er beitreten möchte. Beim VO-Austritt wird dem System entsprechend die VO mitgeteilt von der der User austreten möchte. Hier stehen dem Benutzer nur jene VOs zur Verfügung, in denen er noch nicht Mitglied ist bzw. in denen er Mitglied ist. Beim Auswählen einer VO wird der Name und die Email-Adresse des Administrators der VO geladen und dem Benutzer angezeigt. An dieser Stelle findet die Prüfung statt, ob sich der User auf der WatchList dieser VO befindet. Sollte dies der Fall sein, darf er sich weder zur VO Hinzufügen noch aus dieser Entfernen lassen. Dem Benutzer wird ein entsprechender Hinweis zu seinem Eintrag auf der WatchList angezeigt. Wird keine Warnung ausgegeben, so kann der Anwender den Antrag absenden und der Change Request wird vom `DataManager` gespeichert.

An den nächsten Schritten ist nur der Admin beteiligt. Er ruft die Liste der Change Requests auf und kann sich die Details zu jedem Change Request anzeigen lassen. An dieser Stelle hat er die Möglichkeit den Antrag zu akzeptieren oder abzulehnen. Die Aktion wird zusammen mit einem eventuellen Kommentar des Administrators zu den bisherigen Kommentaren des Change Requests hinzugefügt und wiederum durch den `DataManager` gespeichert.

Zum Abschluss muss der User bestätigen, dass er die Meldung zur Abarbeitung seines Change Requests gesehen hat. Erst dann kann der Request gelöscht werden.

Die verschiedenen Arten eines Change Requests unterscheiden sich nur anhand der Daten die der Benutzer eingibt, und der Aktionen die auf dem Datenbestand nach einer erfolgreichen Bestätigung durchgeführt werden. Diese Unterschiede werden im Folgenden näher erläutert.

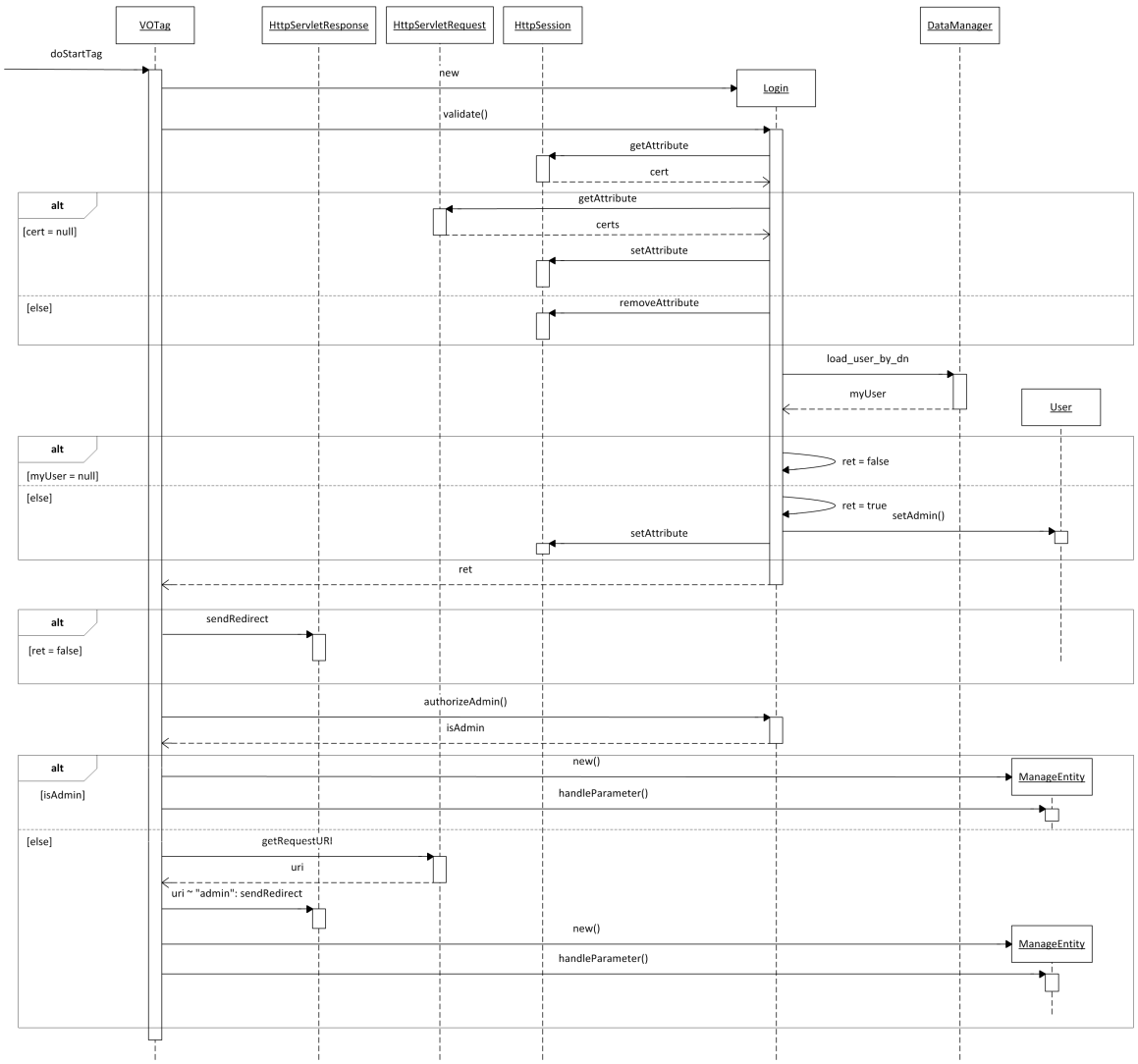

Abbildung 3.5: Zeitlicher Ablauf von Authentifizierung und Autorisierung

Benutzer beantragt Hinzufügen oder Entfernen von FQAN

Das Hinzufügen und Entfernen eines FQAN erfolgt analog zum VO-ChangeRequest, mit dem Unterschied, dass ein FQAN-ChangeRequest erzeugt wird. Dabei muss der Benutzer zusätzlich zur VO noch den FQAN auswählen, der hinzugefügt oder entfernt werden soll. Dem Benutzer stehen dabei immer nur jene VOs zur Auswahl in denen er Mitglied ist. Sobald eine VO gewählt wurde, wird in diesem Fall der Name und die Email-Adresse des Administrators der VO geholt und dem User angezeigt. An dieser Stelle findet auch bei dieser Art des Change Requests die Prüfung statt, ob sich der User auf der WatchList dieser VO befindet. Sollte dies der Fall sein, darf er keine Änderung seiner FQANs beantragen und bekommt wiederum einen Hinweis angezeigt. Steht der User nicht auf der WatchList, werden beim Entfernen die FQANs der gewählten VO geladen die der User bereits hat und beim Hinzufügen die FQANs der VO die dem User nicht zugeordnet sind. Die verbleibenden Schritte laufen analog zu Abbildung 3.6 ab.

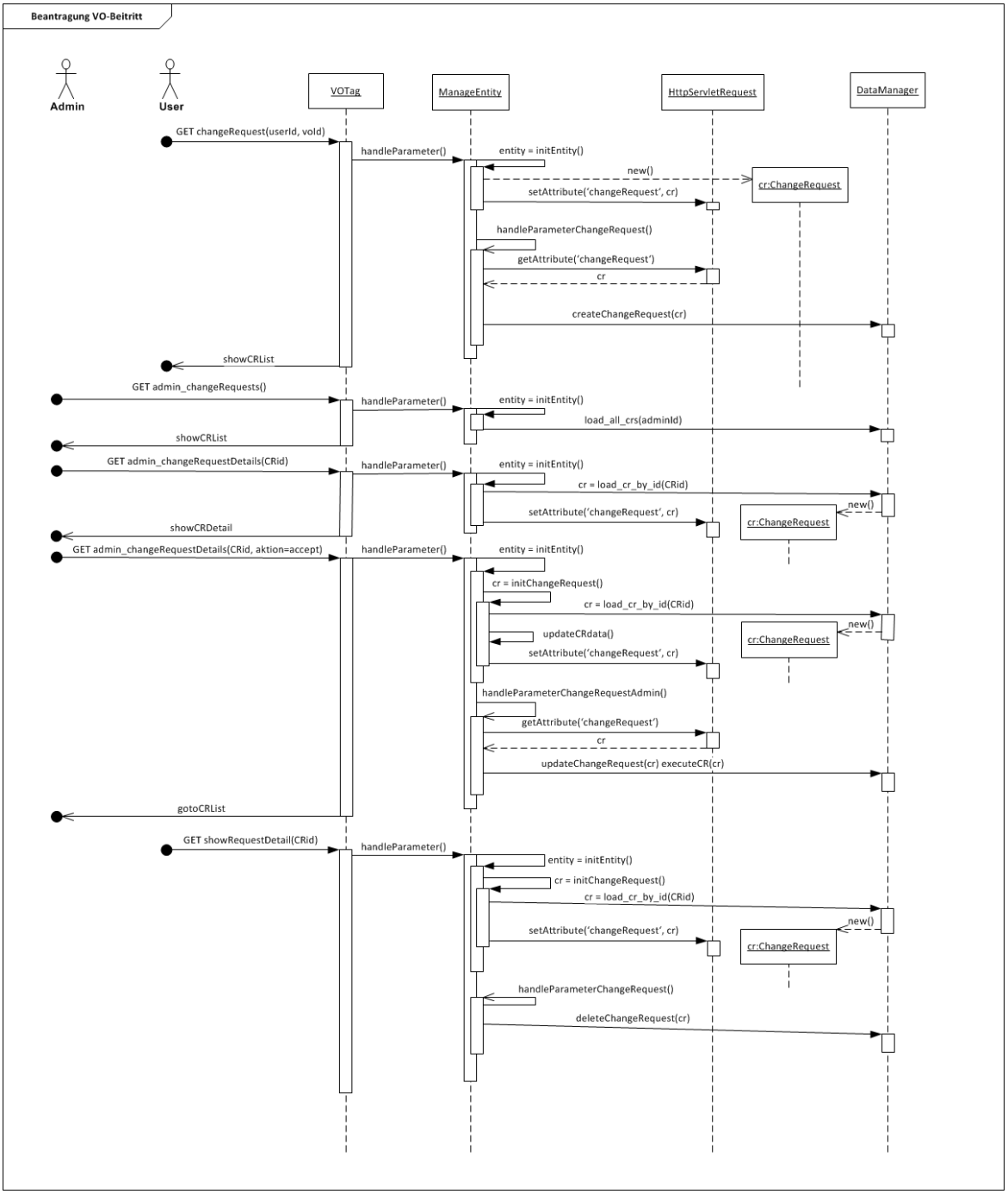

Abbildung 3.6: Zeitlicher Ablauf eines VO-Change Requests

Zulassung eines neuen Zertifikats

Die Zulassung eines neuen Zertifikats kann über einen Cert-Change Request oder einen Register-Change Request erfolgen. Die Art ist davon abhängig, ob der User bereits in der Datenbank gespeichert ist oder er sich zum ersten Mal für das System registrieren möchte. Die Vorgehensweise erfolgt auch in diesem Fall analog zu Abbildung 3.6, wobei die jeweilige Change Request-Klasse erzeugt wird. In beiden Fällen sendet der User ein Zertifikat an das System. Beim Register-Change Request wird das Zertifikat aus der Session verwendet, während es beim Cert-Change Request über das Formular für Change Requests hochgeladen werden muss. Das Zertifikat wird in beiden Fällen auf seine Validität getestet. Außerdem wird geprüft, ob das Zertifikat von einer vom D-Grid anerkannten CA stammt. Beim Register-Change Request muss der Benutzer eine VO auswählen der er beitreten möchte. Dies wird benötigt, um den Antrag einem Administrator zuteilen zu können.

Anlegen und Ändern von Entities

Das Anlegen und Ändern von Entities erfolgt bei allen Entities gleich. Daher zeigt Abbildung 3.7 ein Sequenzdiagramm mit dem Ablauf der Erzeugung sowie der Änderung einer Entity am Beispiel der VO. Analog gilt dies für RoleType, CapabilityType, GroupType und Resource. FQANs können nicht geändert werden. Deswegen gilt für diese nur der Fall der Erstellung. Außerdem können User nicht erstellt werden. Benutzer können sich nur selbst durch einen Register-Change Request am System registrieren. Beim Anlegen einer VO werden die Standard FQANs, definiert in [DGr09], für diese VO angelegt und die Rolle `VO_ADMIN` dem Erzeuger zugewiesen.

Deaktivierung von Entities

Der zeitliche Ablauf der Deaktivierung von Entities wird in Abbildung 3.8 dargestellt. Je nach Typ werden vom `DataManager` andere Aktionen veranlasst.

Bei der Deaktivierung einer VO wird nur die VO auf inaktiv gesetzt und kann somit nicht mehr verwendet werden, beispielsweise bei Change Requests. Die einzige noch mögliche Aktion ist die Reaktivierung der VO.

Wird ein User oder eine Ressource deaktiviert, werden die Zuordnungen der Entity zu den FQANs und zu den VOs entfernt. Dabei werden nur die VOs berücksichtigt in denen der aktuelle User die Administrator-Rolle besitzt. Daher erscheint der deaktivierte User oder die deaktivierte Ressource nicht mehr in der Liste der Benutzer bzw. Ressourcen, die der Administrator verwalten darf.

Bei der Deaktivierung eines FQAN werden die Assoziationen dieses FQAN zu den Usern und den Ressourcen entfernt. Die FQANs, die die Rolle `VO_ADMIN` enthalten, dürfen allerdings nicht deaktiviert werden.

Da die Typen Capability, Role und Group global für alle Administratoren sichtbar sind, werden bei deren Deaktivierung die FQANs deaktiviert die zu den VOs des Administrators gehören und die entsprechende Capability, Role oder Group beinhalten. Dies wiederum hat zur Folge, wie bereits vorher beschrieben, dass die entsprechenden Zuordnungen zu dieser FQAN gelöscht werden. Deaktivierte CapabilityTypes, RoleTypes und GroupTypes werden nicht als inaktiv angezeigt, da sich eine Deaktivierung eines Typs nur auf den Verwaltungsbereich eines Admins auswirkt und somit nur seine FQANs betreffen kann. Auch hier gilt, dass die Administrator-Rolle nicht deaktiviert werden darf.

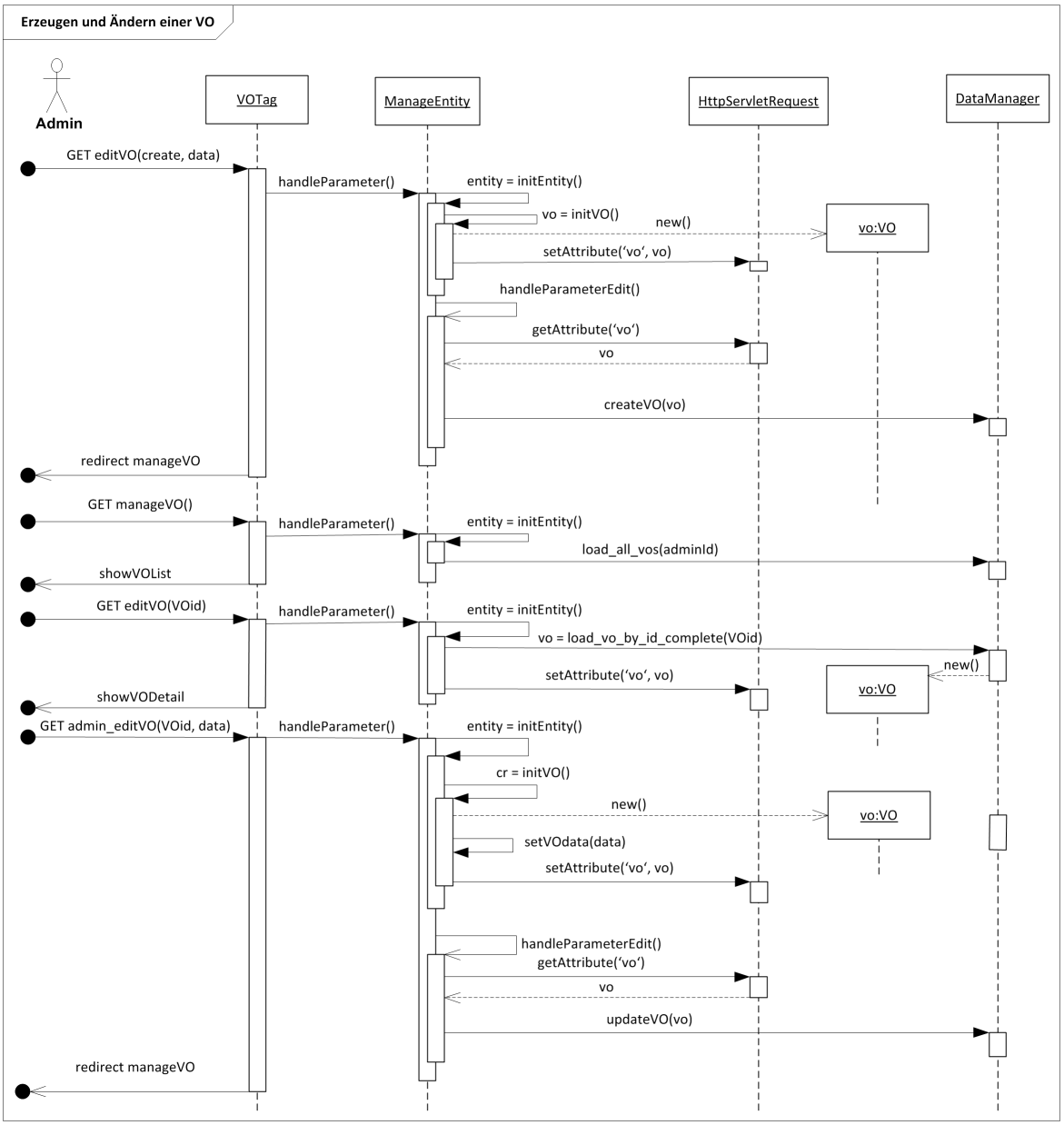

Abbildung 3.7: Zeitlicher Ablauf der Erzeugung sowie Änderung eine VO. Gilt analog für alle anderen Entities.

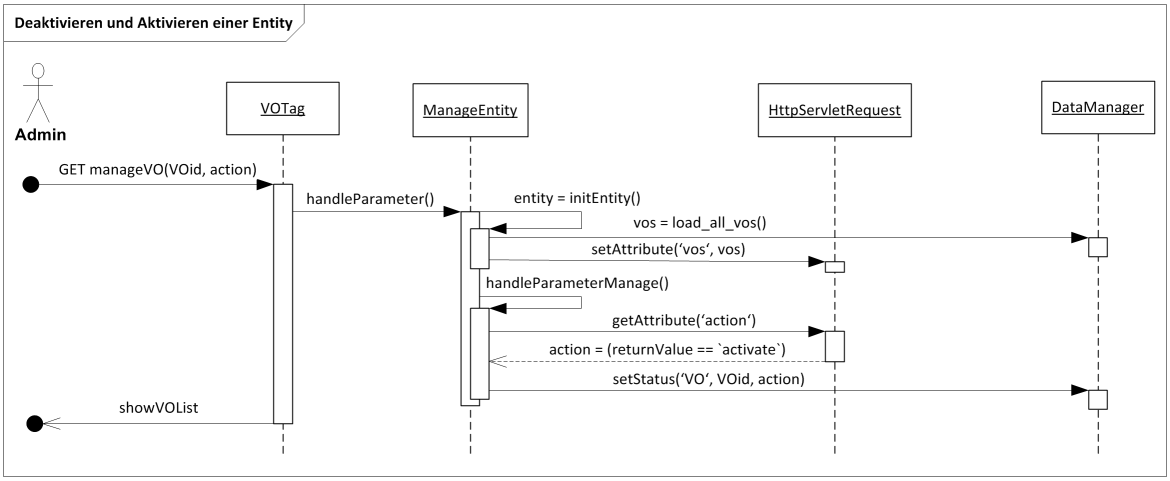

Abbildung 3.8: Zeitlicher Ablauf der Deaktivierung eine VO. Gilt analog für alle Entities.

WatchList

Ist ein User in einer VO auffällig geworden, kann der Administrator diesen auf die WatchList der VO setzen. Dazu wählt er im entsprechenden Formular den User aus. Das System lädt daraufhin die VOs dynamisch nach in denen der User Mitglied ist und in denen der Administrator zugleich auch die Rolle `VO_ADMIN` inne hat. Nachdem der Administrator die entsprechende VO gewählt hat, kann er noch einen Kommentar hinzufügen bevor er die Daten an das System sendet. Das System trägt daraufhin den User in die WatchList der gewählten VO ein. Sollte der gesperrte User in dieser VO Administratorrechte besitzen, so werden ihm diese entzogen.

Der Administrator sieht auf der Übersichtsseite der WatchList alle Einträge aus den VOs seines Verwaltungsbereiches. Dabei kann er bei allen WatchList-Einträgen Details anzeigen lassen, den Kommentar ändern oder den User von der WatchList entfernen.

Beim Entfernen des Users von der WatchList löscht der `DataManager` den Eintrag in der WatchList und trägt ihn in die Historie ein. Diese wird auf der Übersichtsseite der WatchList nach Anforderung dynamisch nachgeladen.

Der zeitliche Ablauf der beschriebenen Aktionen mit der WatchList sind in Abbildung 3.9 dargestellt.

3.3 Datenbankschema

Das bestehende Datenbankschema wird übernommen und durch die im Folgenden erläuterten Ergänzungen erweitert. Das komplette Schema ist in den Abbildungen 3.10 und 3.11 dargestellt.

3.3.1 WatchList und WatchListHistory

Die WatchList benötigt zwei Fremdschlüssel für die VO und den User, der aus der VO ausgeschlossen werden soll. Außerdem wird ein Feld `remarks` und ein Feld `date` für das Datum des Ausschlusses angelegt.

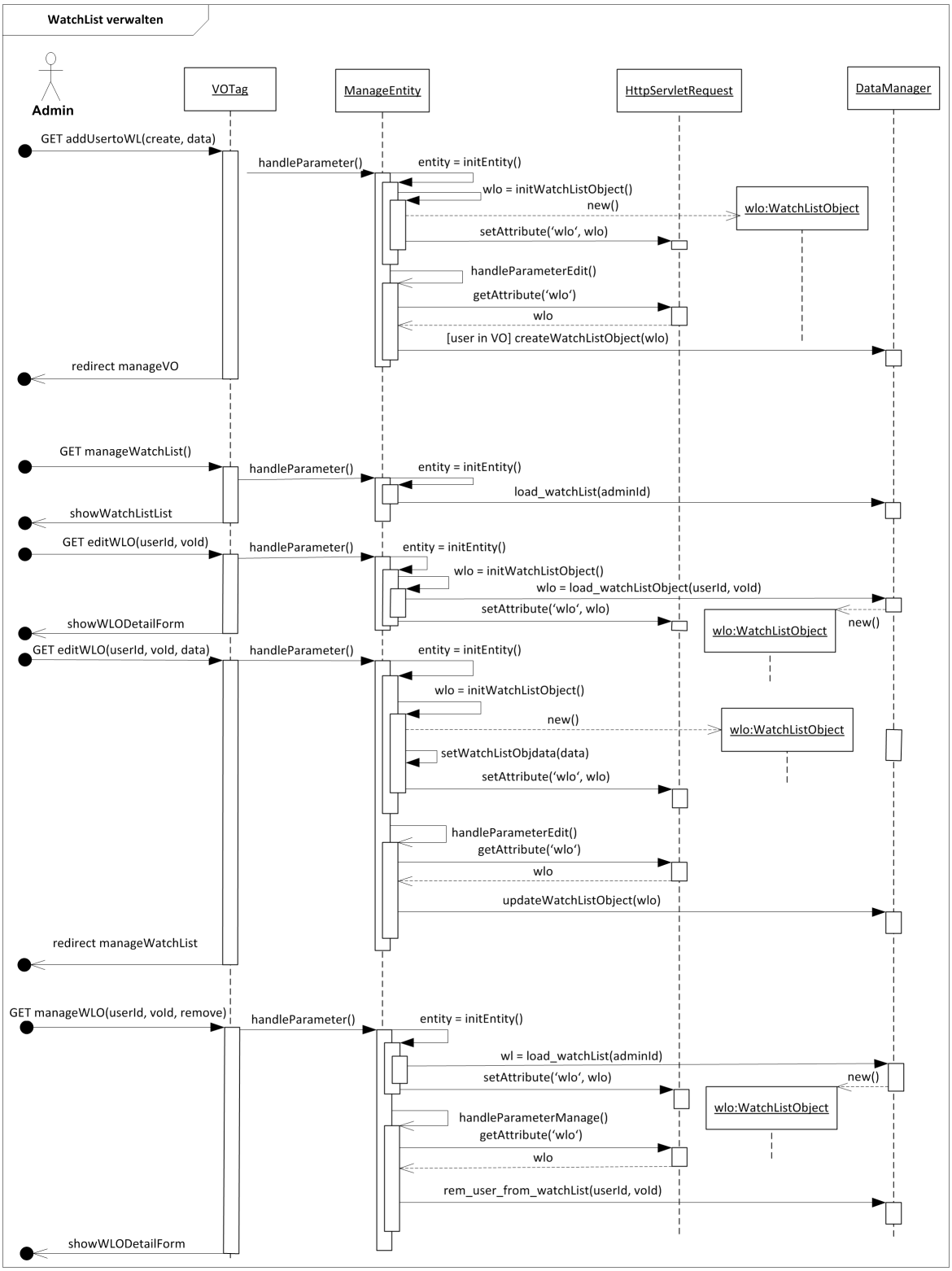

Abbildung 3.9: Zeitlicher Ablauf beim Hinzufügen und Entfernen eines Users von der Watch-List einer VO

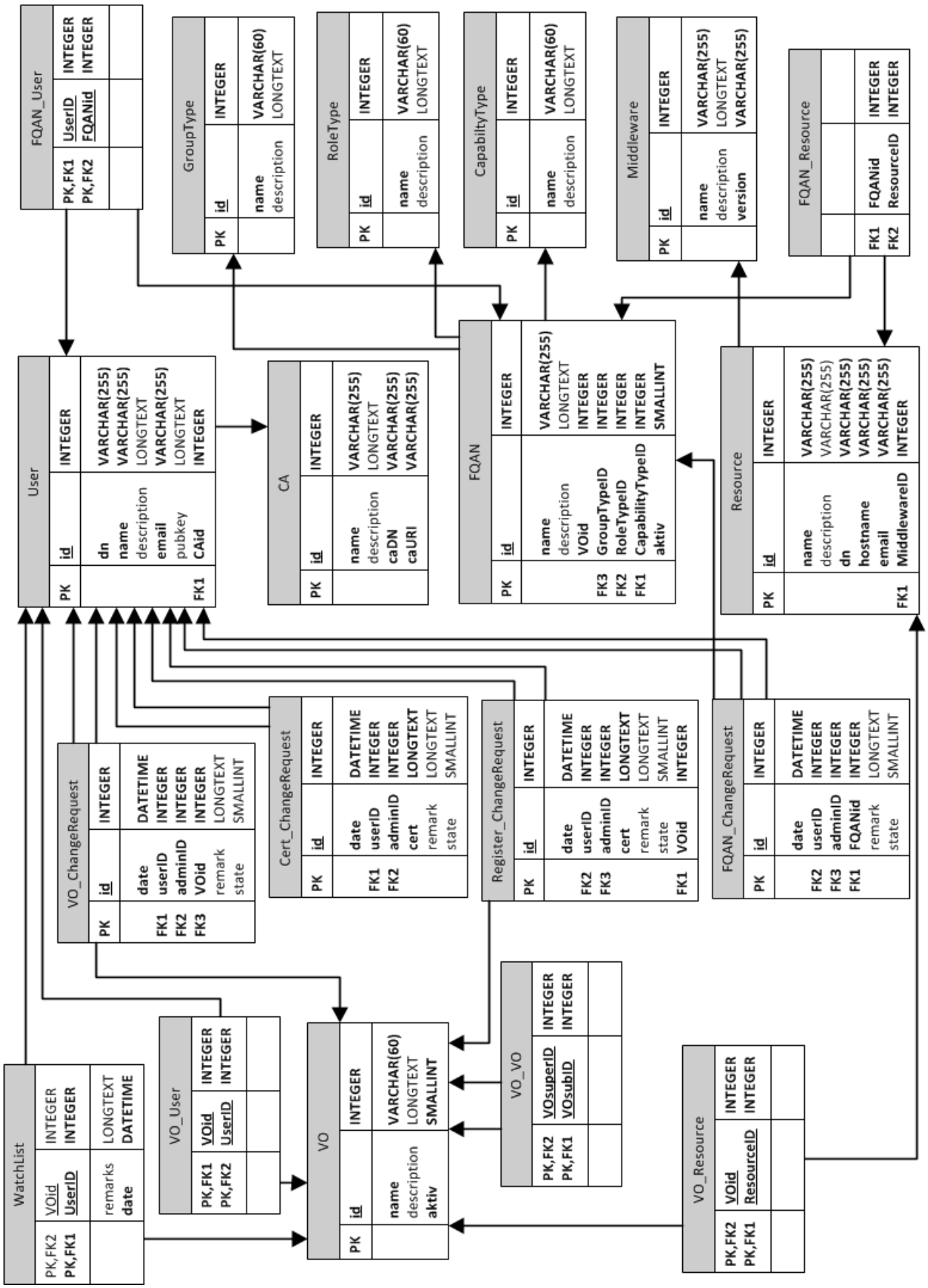

Abbildung 3.10: Datenbankschema

Die `WatchListHistory` Tabelle dient der Übersicht welcher Benutzer aus welcher VO aus welchem Grund ausgeschlossen wurde. Sie ist analog zur `WatchList` Tabelle aufgebaut, erhält aber zusätzlich noch eine `ID` zur eindeutigen Identifikation, da es theoretisch sein kann, dass ein Benutzer mehrfach aus einer VO ausgeschlossen und wieder zugelassen wird. Beim Entfernen eines Benutzers aus der WatchList besteht daher die Möglichkeit anzugeben aus welchem Grund das Löschen erfolgt. Das Datum wird dabei automatisch mit abgespeichert.

Wird ein neuer Datensatz erzeugt, so bleiben alle anderen Tabellen unberührt, d.h. wird ein User auf die WatchList gesetzt, so hat dies keine Auswirkungen auf seine Assoziationen. Eine Ausnahme hiervon bildet der Fall, wenn ein Administrator einer VO auf die WatchList gesetzt werden soll. Dies führt dazu, dass dem betroffenen User seine FQANs mit der Rolle `VO_ADMIN` in dieser VO entfernt werden.

3.3.2 Change Request

Für die vier Request-Typen `VO-ChangeRequest`, `FQAN-ChangeRequest`, `Cert-ChangeRequest` und `Register-ChangeRequest` wird jeweils eine Tabelle hinzugefügt, die das Datum, den betreffenden Benutzer sowie den zuständigen Administrator beinhaltet. Dabei sind die Felder `userId` und `adminId` jeweils Fremdschlüssel. Das Attribut `adminId` ist zwar redundant, wird allerdings aus Effizienzgründen mit in die Tabelle eingetragen, damit nicht bei jeder neuen Anzeige der Change Request Liste ein Join durchgeführt werden muss. Die Tabelle `VO-ChangeRequest` besitzt außerdem einen Fremdschlüssel zu VO, die `FQAN-ChangeRequests` entsprechend zu FQAN und das Zertifikat des `Cert-ChangeRequest` wird als String in die Tabelle gespeichert. Die Tabelle `Register-ChangeRequest` enthält neben einem String für das Zertifikat auch einen Fremdschlüssel zu VO.

3.3.3 Deaktivierung von Entities

Die Tabellen der Entities VO und FQAN werden um das Attribut `aktiv` erweitert. Ist der Wert 1, also true, ist die Entity aktiv und kann verwendet werden. Ansonsten ist es inaktiv.

3.3.4 Statistiken und Historie

Des Weiteren werden zwei eigene Statistik-Tabellen angelegt, in der VO Ein- und Austritte sowie das Hinzufügen und Entfernen von FQANs bei einem Benutzer erfasst werden. Zur effizienten Pflege der Statistiktabellen werden beim Hinzufügen und Entfernen von Datensätzen aus den Tabellen `VO_User` sowie `FQAN_User` Trigger eingesetzt, die die Statistiktabellen befüllen.

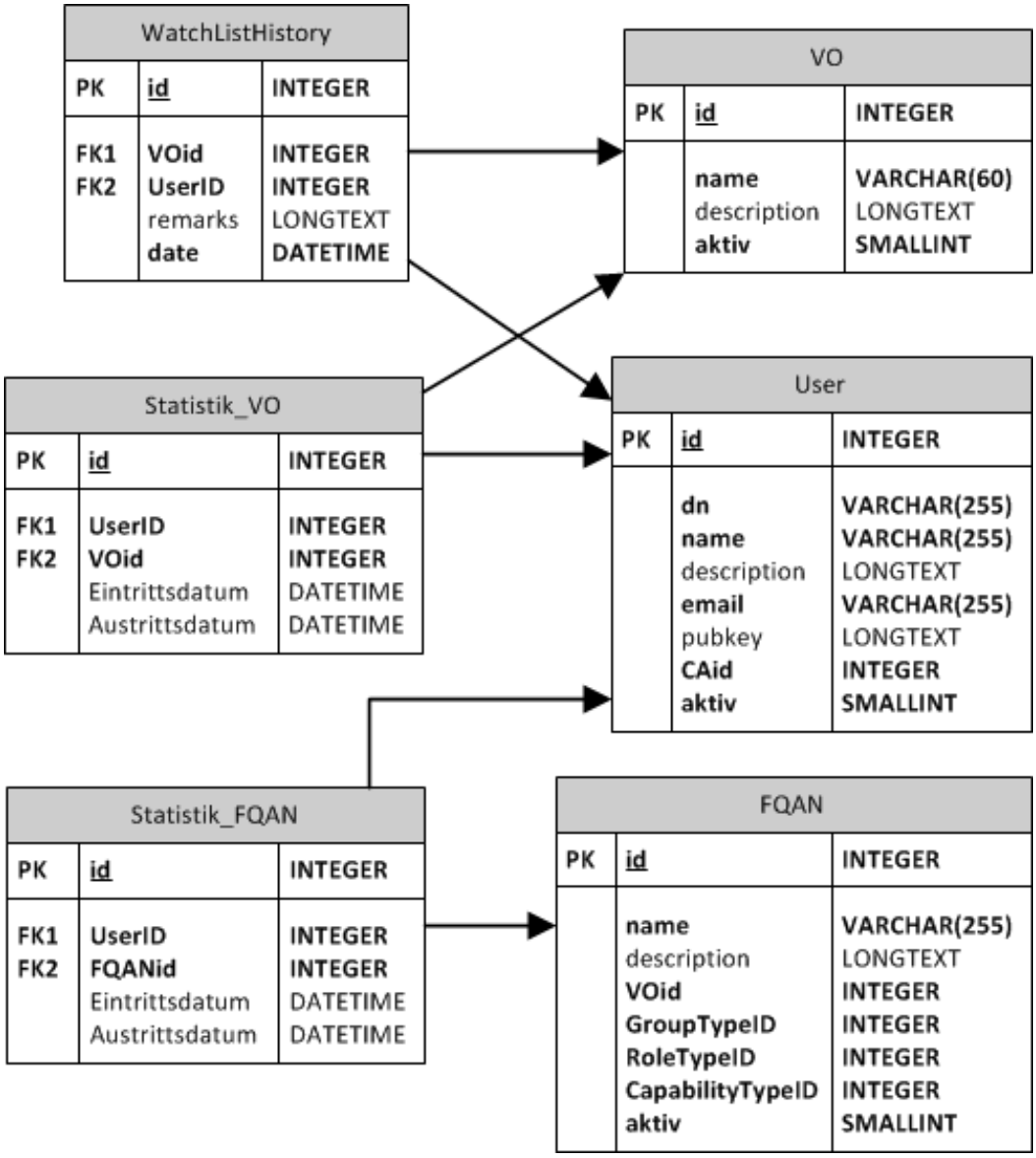

Abbildung 3.11: Datenbankschema der Statistik-Tabellen mit ihren Abhängigkeiten

4 Implementierung

Im vierten Kapitel dieser Projektarbeit wird das im vorangegangenen Kapitel entwickelte Model konkret implementiert. Außerdem wird die Systemumgebung der Webanwendung vorgestellt sowie auf das Deployment eingegangen.

4.1 Webanwendung

Im Folgenden werden die einzelnen Bestandteile der Webanwendung mit Screenshots und Codestücken vorgestellt.

4.1.1 Authentifizierung und Autorisierung

Für die Authentifizierung und Autorisierung wurde eine Klasse `VOTag` eingeführt, welche von der Oberklasse `TagSupport` sowie dem Interface `Tag` abgeleitet wird. Die Methoden der Klasse werden durch den entsprechenden Tag `volayer:voLoggedIn` aufgerufen, welcher näher im Abschnitt 4.3 beschrieben wird. Der Tag muss den gesamten Inhalt jeder jsp-Seite umschließen. Listing 4.1 zeigt, den prinzipiellen Aufbau einer solchen Seite. Dabei wird zuerst der DocType der Seite spezifiziert, damit der Browser weiß, wie der Inhalt der Seite interpretiert werden muss. Im Fall des VO-Layers wurden die Seiten in XHTML 1.1 geschrieben. In den nächsten Zeilen des Listings werden durch die taglib-Direktive angegeben, welche Tag-Bibliotheken auf dieser Seite verwendet werden. An dieser Stelle wird der volayer-Tag eingebunden, außerdem kommt auf den meisten Seiten die Tag-Bibliothek c sowie die Funktions-Bibliothek fn aus der JSTL (Java Standard Tag Library) zum Einsatz. Durch die page-Direktive wird angegeben, dass die Session verwendet werden soll und die Zeichenkodierung, hier utf-8, spezifiziert. Der eigentliche Inhalt der Seite wird an der Stelle des Kommentars im Listing in Zeile 9 platziert. Dadurch wird dem Benutzer der Inhalt der Seite nur nach erfolgreicher Authentifizierung und Autorisierung angezeigt.

```
1  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
2    "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
3  <%@ taglib uri="/volayer" prefix="volayer" %>
4  <%@ taglib uri="/WEB-INF/tld/c.tld"  prefix="c"%>
5  <%@ taglib uri="/WEB-INF/tld/fn.tld" prefix="fn"%>
6  <%@ page session="true"%>
7  <%@ page contentType="text/html; charset=utf-8" %>
8  <volayer:voLoggedIn>
9    <!-- Inhalt kommt hier rein -->
10 </volayer:voLoggedIn>
```

Listing 4.1: Einsatz des VO-Layer Tags in jeder jsp-Datei

Die relevanten Methoden der Klasse `VOTag` sind `doStartTag` sowie `doEndTag`, die vom Parser des Servers ausgeführt werden wenn dieser den Anfangs- oder den Ende-Tag erreicht. In `doStartTag` werden die benötigten Klassen instantiiert und in der Session zwischengespeichert. Anschließend wird der Benutzer authentifiziert und geprüft, ob er Adminrechte besitzt. Abhängig vom Ergebnis wird der entsprechende Header mit den passenden Menüeinträgen eingebunden und dem Benutzer angezeigt. Für normale Benutzer ist dies `WEB-INF\includes\header.jsp` und für Admins `WEB-INF\includes\header_admin.jsp`. In der Methode `doEndTag` wird der Footer aus `WEB-INF\includes\footer.jsp` in die Seite eingebunden. Um Konsistenz zu gewährleisten sind Header und Footer jeweils in eigene Dateien ausgelagert.

Um unberechtigten Zugriff zu verhindern, wird bereits in der Klasse `VOTag` geprüft, ob der Benutzer grundsätzlich die Zugangsberechtigung für die aktuelle Seite besitzt. Sollte ein normaler Benutzer versuchen, auf eine Administrationsseite zu gelangen, scheitert er bereits an dieser Stelle.

Versucht ein Administrator eine VO zu verwalten, für die er keine Rechte besitzt, wird dies vom `DataManager` verhindert. Der `DataManager` prüft bei jeder Anfrage, ob der angemeldete User die Berechtigung besitzt, auf die entsprechende Entity zuzugreifen.

4.1.2 Registrierung

Um den VO-Layer nutzen zu können ist neben einem gültigen und anerkannten Zertifikat eine Registrierung notwendig. Aus diesem Grunde wird der Benutzer beim erstmaligen Aufruf der Webanwendung auf die Registrierungsseite `login.jsp` geleitet (siehe Abbildung 4.1). Auf dieser muss der Anwender eine VO auswählen und hat die Möglichkeit einen Kommentar anzufügen. Somit ist die Registrierung für den VO-Layer gleichzeitig ein Beitritt zur gewählten VO. Nach dem Absenden des Formulars erscheint ein kurzer Hinweistext, welcher Abbildung 4.2 entnommen werden kann. Diese enthält den Namen und die Emailadresse des Administrators der die Registrierung bearbeiten wird, also derjenige der zugleich Administrator der gewählten VO ist.

4.1.3 Startseite

In Abbildung 4.3 sieht man den grundlegenden Aufbau der GUI sowie ein Beispiel für eine individuelle Startseite. Das Menü des VO-Layers wurde links platziert und in zwei Abschnitte unterteilt: die 'User Area' die für alle Benutzer sichtbar ist sowie 'Management' welcher nur für VO-Administratoren zu sehen ist. Der aktuelle Menüpunkt - in diesem Fall 'Overview' - wird jeweils farbig hervorgehoben.

Auf der Startseite werden zunächst vom Benutzer abgesendete Change Requests, welche noch nicht bearbeitet oder bestätigt wurden, mit ihrem Erstellungsdatum, Typ, zuständigen Administrator und dem Status angezeigt. Darunter folgen die Daten des Zertifikats welches für die Authentifizierung am VO-Layer verwendet wurde. Um die Übersichtlichkeit dieser Seite zu verbessern lassen sich bei Klick auf die jeweiligen Überschriften, beispielsweise 'Your Certificate', die darunterliegenden Informationen aus- bzw. einblenden. Im unteren Teil der Startseite werden dem Benutzer alle VOs aufgelistet in denen er Mitglied ist. Zu jeder einzelnen VO lassen sich desweiteren alle Ressourcen dieser VO anzeigen, Name und Email des Administrators sowie die FQANs die dieser Benutzer in der betroffenen VO besitzt. Außerdem wird dem Mitglied bei der entsprechenden VO ein Hinweis angezeigt, falls er sich in

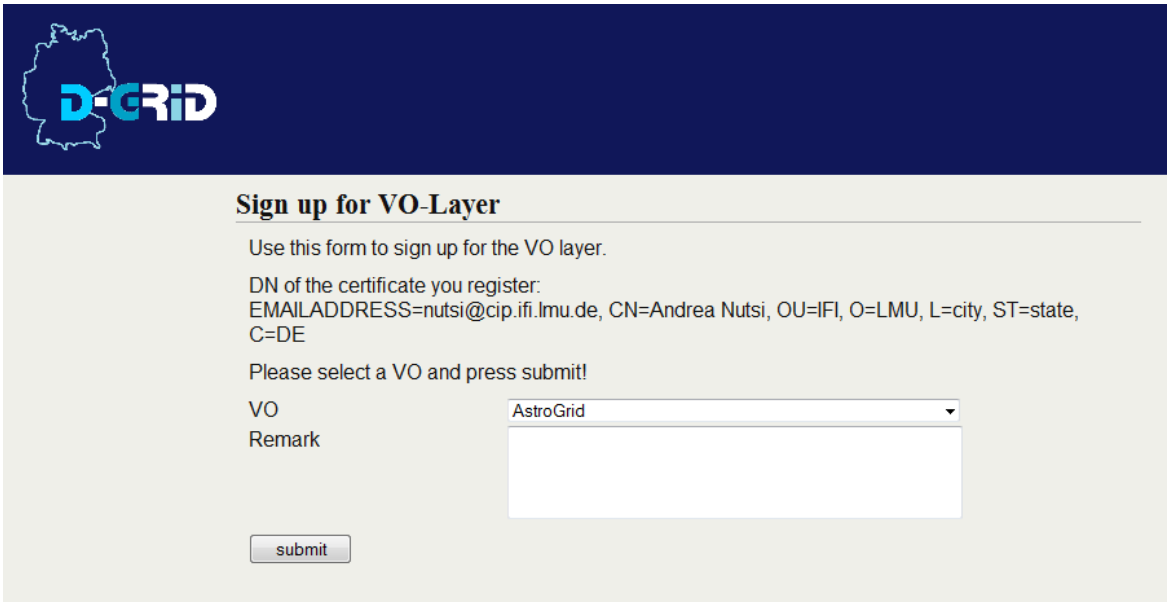

Abbildung 4.1: Screenshot der Registrierungsseite des VO-Layers

Abbildung 4.2: Screenshot der Bestätigung der Registrierung des VO-Layers

dieser auf der WatchList befindet.

Zur Demonstration des Einsatzes der JSTL sowie des jQuery Frameworks wird im Folgenden dargestellt, wie die Informationen der VOs auf der Startseite angezeigt werden. Dazu werden in der Methode `initEntity` der Klasse `ManageEntity` alle Informationen in das Request-Objekt gespeichert, die zur Anzeige benötigt werden. Ein Auszug der Methode wird in Listing 4.2 gezeigt. Dabei werden zwei HashMaps erstellt. Die erste mapped die VO-Id auf den Admin der VO, die zweite mapped die Id auf einen boolean, der angibt ob der User in der VO auf der WatchList steht. Da die jsp-Seite auch die Objekte in der Session zur Verfügung hat muss dieses nicht in das Request-Objekt gespeichert werden.

```
1  [...]
2  private String initEntity(String uri, HttpServletRequest request
       , HttpSession session){
3    [...]
4    if(uri.contains("/index.jsp") || uri.endsWith("/") ){
5
6      [...]
7      // Map fuer admin von vo
8      Map<Integer, User> admins = new HashMap<Integer, User>();
9      // Map fuer watchlist in vo
10     Map<Integer, Boolean> watchlist =
11       new HashMap<Integer, Boolean>();
12     for(VO vo : user.getVo_s()){
13       admins.put(vo.getId(), dataManager.loadAdminCR(vo));
14       watchlist.put(vo.getId(), dataManager.
15         load_watchListObject(userId, vo.getId())!= null);
16     }
17
18     request.setAttribute("admin", admins);
19     request.setAttribute("wl", watchlist);
20
21     [...]
22   }
23 }
24 [...]
```

Listing 4.2: Auszug aus ManageEntity.java für den Bereich index.jsp

Der Ausschnitt aus der `index.jsp` ist in Listing 4.3 dargestellt. Im Folgenden wird auf wichtige Eigenheiten der JSTL eingegangen. In Zeile 4 des Listings wird direkt auf das user-Objekt der Session zugegriffen. Durch die Angabe `user.vo_s` wird die Methode `user.getVo_s()` aufgerufen und das Ergebnis in der Variable vo dem Rumpf dieses foreach-Tags zur Verfügung gestellt. Der Schleifeninhalt wird für jede VO aus der Liste abgearbeitet. Das Tag `c:if` stellt eine einfache Abfrage dar, ob eine bestimmte Bedingung erfüllt ist. Eine Alternative ist dabei nicht vorgesehen und wird in diesem Fall auch nicht benötigt. Für solche Fälle wird das Tag `c:choose` herangezogen, welches auf dieser Seite aber nicht zum Einsatz kommt. Durch die Notation `${Variable.Eigenschaft}` kann der Wert direkt in der Seite ausgegeben werden.

Dies ermöglicht eine kompakte und besser wartbare Schreibweise. Da jeder User in seinen VOs mehrere FQANs besitzen kann, werden diese erneut durch eine forEach-Anweisung abgearbeitet.

Wie bereits weiter oben erwähnt, können zu jeder VO die FQANs und die Ressourcen angezeigt werden. Beim Laden der Seite sind diese ausgeblendet und können durch einen Klick auf den VO-Namen eingeblendet werden. Um dies zu ermöglichen, wird das jQuery Framework verwendet, welches JavaScript nutzt. Der Vorteil des Frameworks ist die Browserunabhängigkeit des Quellcodes sowie die kurze Schreibweise. Listing 4.4 zeigt einen Ausschnitt aus der JavaScript-Datei `script.js`, die in jede Seite eingebunden wird. In jQuery werden die zu bearbeitenden Elemente mit Hilfe von CSS Selektoren ausgewählt, so werden beispielsweise mit `$('.accordionHead')` alle Elemente des DOM-Baumes selektiert, welche die Klasse `accordionHead` besitzen.

```
1  [...]
2  <h1 class="accordionHead visible">Your VOs</h1>
3    <div>
4    <c:forEach items="${user.vo_s}" var="vo">
5      <div class="accordionHead">
6        <h3>${vo.name } (Admin: ${admin[vo.id].name} [...]
7            ${admin[vo.id].email} [...])</h3>
8        <c:if test="${wl[vo.id]}">
9          <p class="error">You are on the WatchList of ${vo.name }
               </p>
10       </c:if>
11     </div>
12     <div>
13       <strong>Your FQANs</strong>
14       <table>
15         <tr>
16           <th>Groups</th>
17           <th>Roles</th>
18           <th>Capabilities</th>
19         </tr>
20         <c:forEach items="${vo.fqan_s}" var="fqan">
21           <tr>
22             <td>${fqan.groupType.name}</td>
23             <td>${fqan.roleType.name}</td>
24             <td>${fqan.capabilityType.name}</td>
25           </tr>
26         </c:forEach>
27       </table>
28       <strong>Your Resources</strong>
29       <table>
30        [...]
31       </table>
32     </div>
```

```
33 </c:forEach>
34 [...]
```

Listing 4.3: Auszug aus index.jsp : Einsatz der JSTL bei der Anzeige der VO-Daten auf der Startseite

4.1.4 Change Requests

Eine wesentliche Änderung des bisherigen VO-Layers war die Einführung des Konzepts der Change Requests. Abbildung 4.4 zeigt die Sichtweise eines Benutzers bei der Erstellung eines neuen Change Requests. Dabei wird ein Großteil des Inhalts dynamisch nachgeladen. Zunächst muss der Benutzer die Art des Change Requests wählen, also 'Certificate', 'FQAN' oder 'VO'. Wurde ersteres ausgewählt so erscheint ein Formular mit der Möglichkeit eines Datei-Uploads sowie einem Feld für eventuelle Bemerkungen. Bei den beiden anderen Arten muss der Anwender sich zunächst zwischen 'Add' und 'Remove' entscheiden, bevor die Inhalte dynamisch geladen werden. Im Beispiel der Abbildung 4.4 wünscht der Benutzer einen weitereren FQAN für die VO AstroGrid. Nachdem der Benutzer die VO ausgewählt hat, wird der Administrator mit seiner Emailadresse dynamisch nachgeladen sowie alle verfügbaren FQANs dieser VO die der Benutzer noch nicht besitzt. Auch im Falle von FQAN und VO gibt es die Möglichkeit eine Bemerkung zu diesem Change Request zu speichern.

Nachdem der Benutzer den Change Request abgesendet hat, wird dieser dem betroffenen Administrator unter dem Menüpunkt 'Change Requests' im Abschnitt 'Management' angezeigt (siehe Abbildung 4.5). Auf dieser Seite wird das Absendedatum, Absender und Art sowie ein Link zur Detailansicht des Change Requests tabellarisch dargstellt. Wie die Detailansicht aussieht, ist in Abbildung 4.6 gezeigt. In dieser hat der Administrator die Möglichkeit den Antrag abzulehnen oder anzunehmen und dies mit einem Kommentar für den Antragsteller zu versehen. Die Entscheidung des Administrators wird dem Antragsteller in Folge auf der Startseite angezeigt, bis dieser die Kenntnisnahme bestätigt hat.

4.1.5 Entity Management

Eine weitere Anforderung an die GUI des VO-Layers ist es, das Management der Entitäten VO, User, Ressourcen, FQAN, GroupType, RoleType und CapabilityType zu ermöglichen. Am Beispiel User-Management ist in Abbildung 4.7 gezeigt wie dies umgesetzt wurde. Benutzer, die mindestens in einer VO des angemeldeten Administrators Mitglied sind, werden mit Namen, Beschreibung und Emailadresse aufgelistet. Der Administrator darf die Daten des Benutzers frei editieren. Außerdem können User deaktiviert werden, die Konsequenzen dieser Aktion wurden bereits im vorangegangen Kapitel besprochen.

Für jeden Anwender in der Tabelle lässt sich eine Detailansicht (Beispiel siehe Abbildung 4.8) aufrufen die auflistet, in welchen VOs der gewählte User Mitglied ist und die zugehörigen FQANs sowie den distinguished name anzeigt.

Ein weiteres Beispiel für das Entity-Management ist in Abbildung 4.9 dargestellt. FQANs werden nach VOs sortiert aufgeführt und können nicht editiert werden. FQANs welche die Rolle `VO_ADMIN` enthalten können nicht deaktiviert werden, die Konsequenzen einer Deaktivierung wurden bereits in Kapitel 3 besprochen. Nicht aktive FQANs werden am Ende der Tabelle aufgeführt und farblich gekennzeichnet.

```
1  [...]
2  $('.accordionHead').click(function(event) {
3    if($(event.target).is('a')){
4      return true;
5    }else{
6      $(this).next().toggle();
7    }
8    return false;
9  }).next().hide();
10
11 $('.accordionHead.visible').next().toggle();
12 [...]
```

Listing 4.4: Auszug aus script.js für Ein- und Ausblenden der Informationen

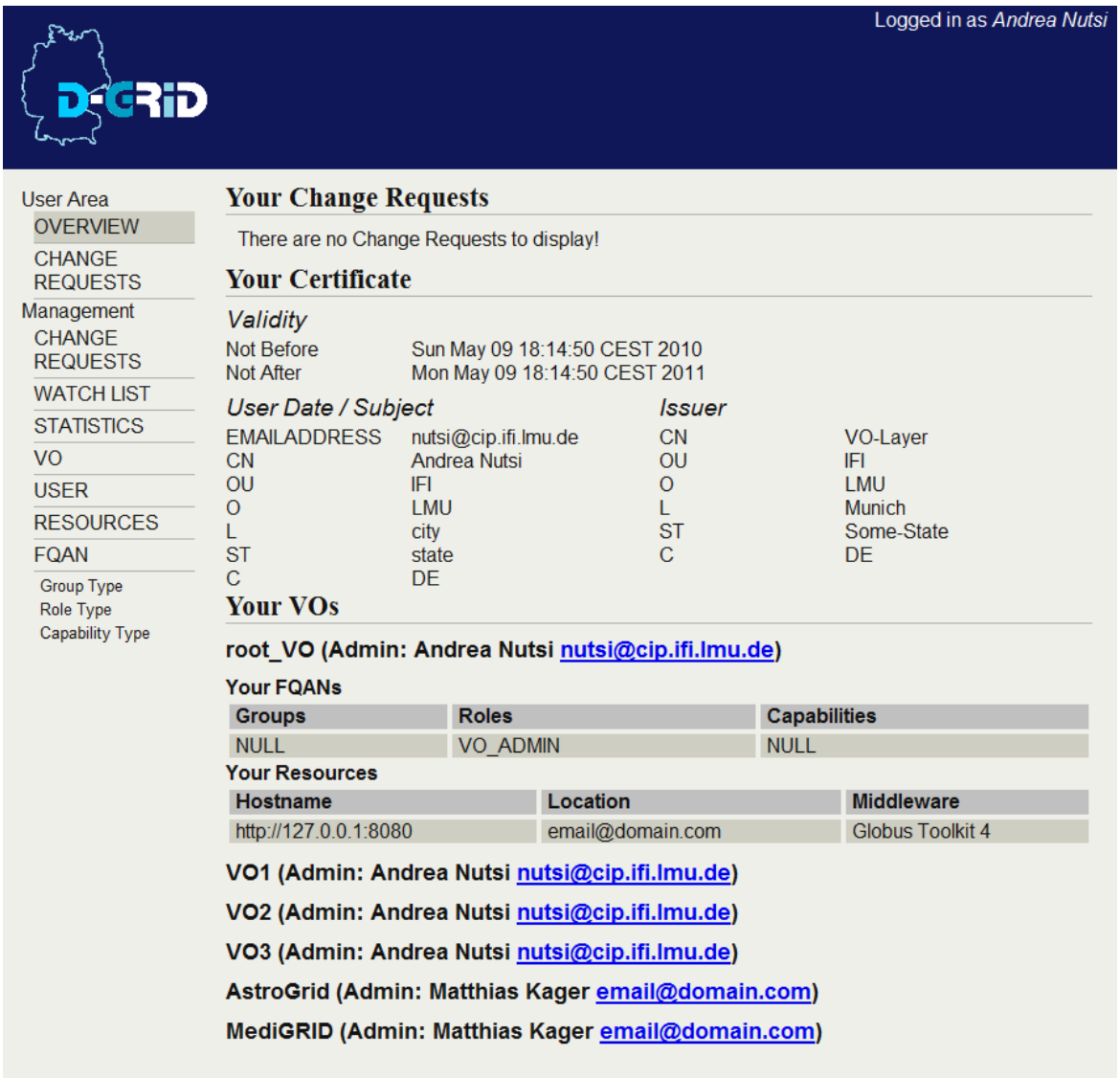

Abbildung 4.3: Screenshot des Startscreens des VO-Layers

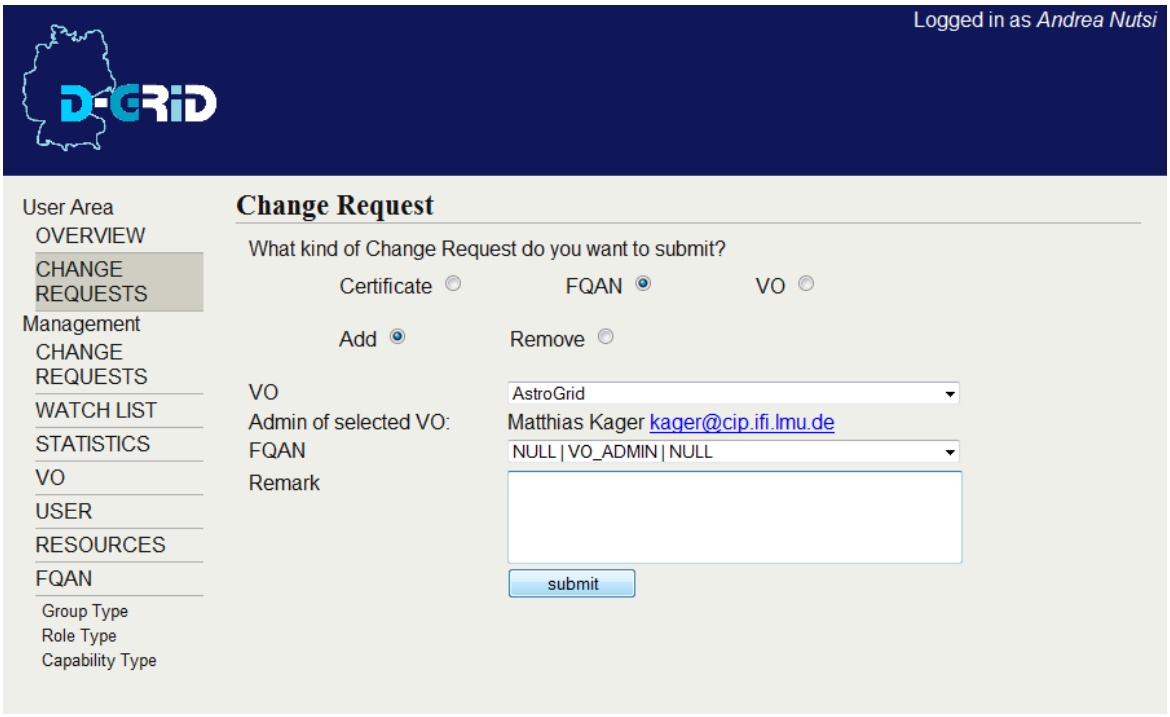

Abbildung 4.4: Screenshot der Erstellung eines Change Requests

Abbildung 4.5: Screenshot der Übersicht der Change Requests des Administrators

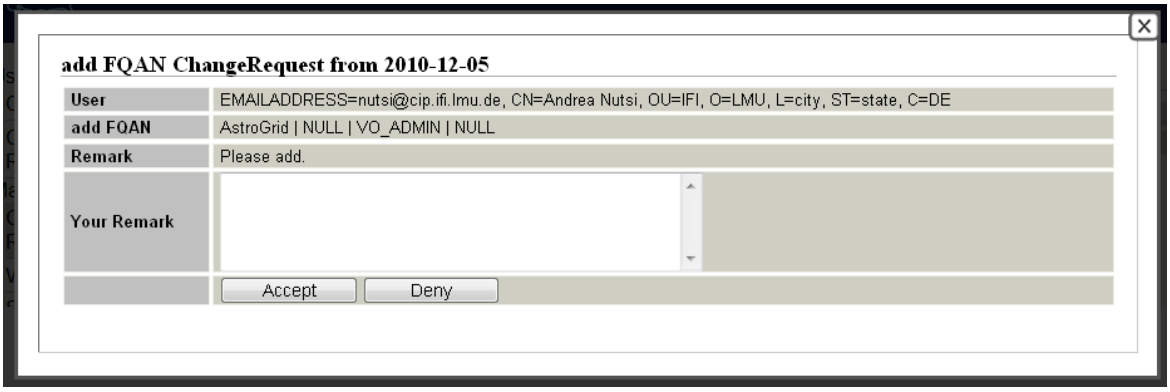

Abbildung 4.6: Screenshot der Detailansicht eines Change Requests

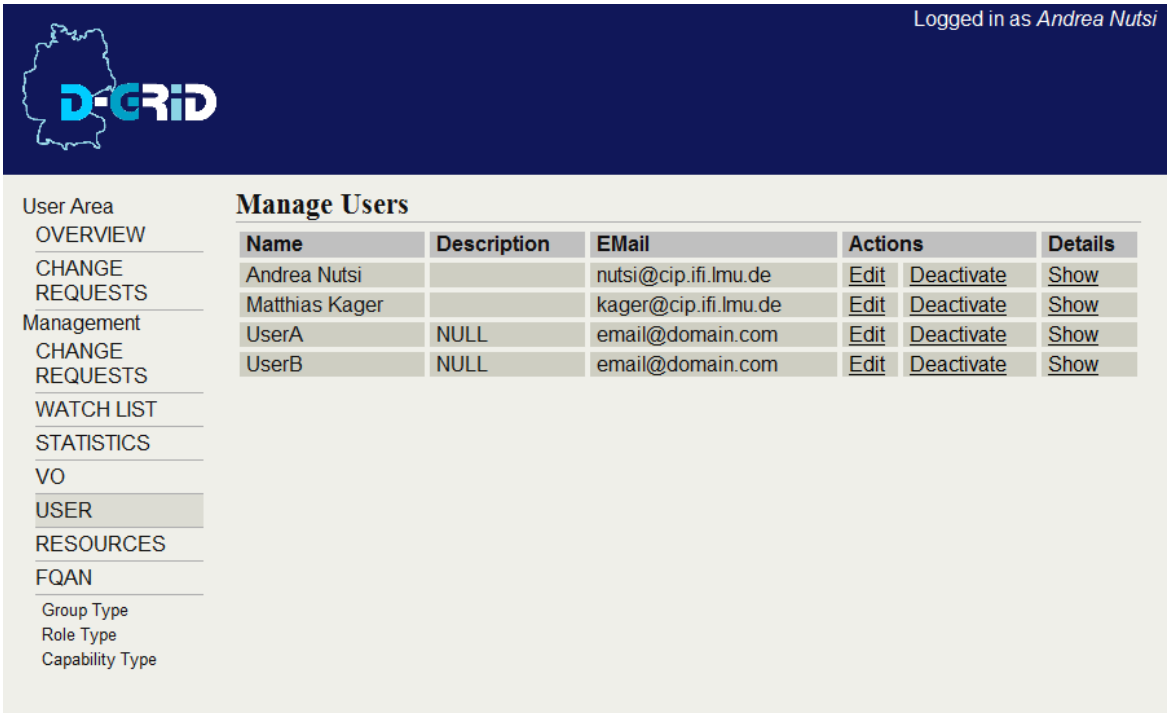

Abbildung 4.7: Screenshot des User-Management des VO-Layers

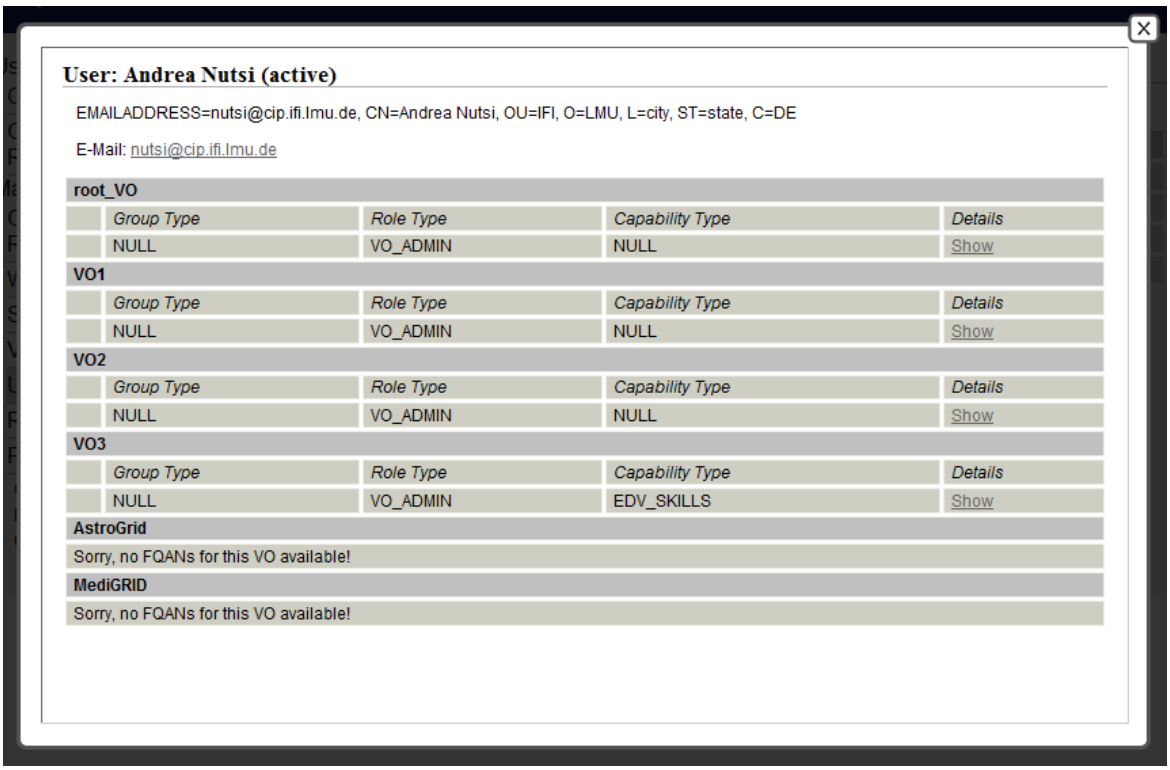

Abbildung 4.8: Screenshot der Detailansicht eines Users des VO-Layers

Alle hier nicht einzeln aufgeführten Entitäten werden in analoger Weise verwaltet. In allen Fällen wurde auf eine übersichtliche Darstellung geachtet und ein Verweis auf eine Detailansicht des Datensatzes angeboten.

4.1.6 WatchList

Neben den bereits besprochenen Change Requests war die Einführung der WatchList eine weitere wesentliche Änderung des bisherigen VO-Layers. Der Aufbau der Seite der WatchList lässt sich Abbildung 4.10 entnehmen. Zunächst befindet sich auf der Seite ein Button der zu einem Formular führt mit dem sich Benutzer der WatchList hinzufügen lassen. In diesem Formular kann ein Benutzer und eine VO ausgewählt werden sowie ein Kommentar angegeben werden. Das Eintrags-Datum wird dem Datensatz durch das System automatisch zugewiesen. In Abbildung 4.10 ist zu erkennnen, dass die Darstellung der WatchList-Einträge nach VOs einzeln aufgelistet ist. Für jeden Datensatz besteht die Möglichkeit den Kommentar zu editieren sowie den Eintrag zu löschen. Wurde ein Datensatz entfernt, so erscheint dieser auf der WatchList History welche im unteren Teil der Seite dargestellt ist. Um die Übersichtlichkeit zu verbessern lässt sich die Historie dynamisch ein- bzw. ausblenden. Beim Abruf der Übersicht werden diese Einträge noch nicht geladen, um die Antwortzeit niedrig zu halten.

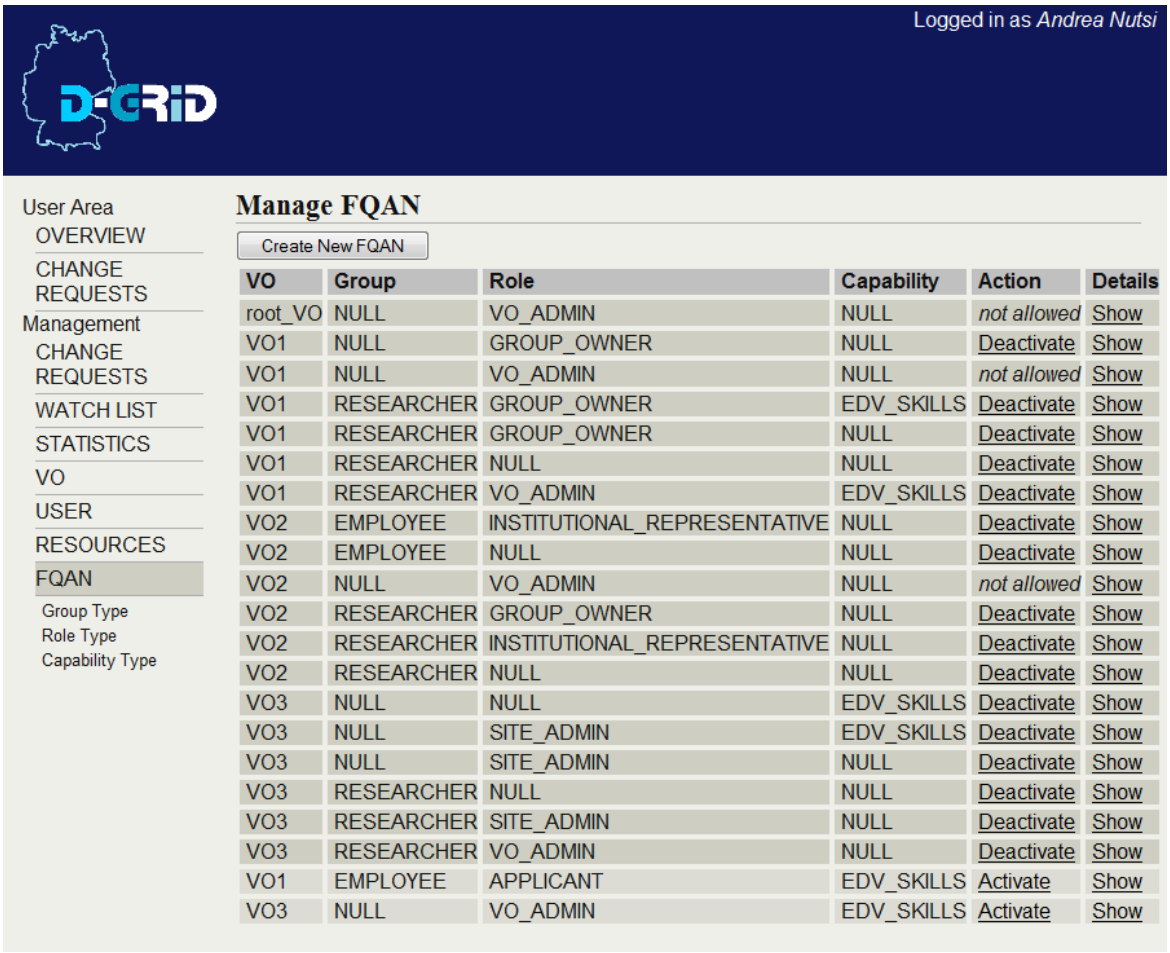

Abbildung 4.9: Screenshot des FQAN-Management des VO-Layers

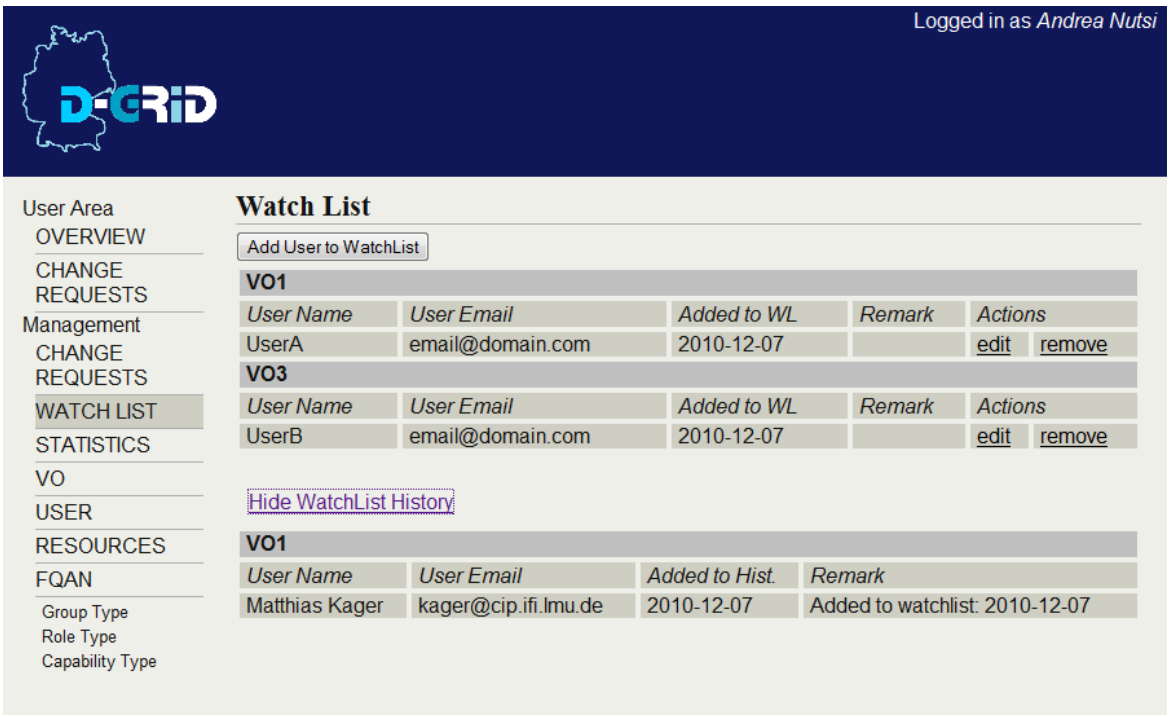

Abbildung 4.10: Screenshot des WatchList des VO-Layers

4.1.7 Statistik

In Abbildung 4.11 ist ein Ausschnitt der Statistik-Seite des VO-Layers zu sehen. Sehr wichtig ist hierbei, dass in der momentanen Implementierung alle Administratoren Einsicht in die komplette Statistik haben. An dieser Stelle werden also nicht - wie im restlichen Managementbereich - nur die Informationen der VO(s) angezeigt für welche der Anwender Administrationsrechte besitzt. Über einen Link rechts oben auf der Seite lassen sich die Daten der Statistik in eine csv-Datei exportieren. Auf der Seite selbst wird zunächst eine Übersichtstabelle für alle VOs angezeigt, die neben der Anzahl der Mitglieder die Beitritte und Austritte in den folgenden Zeiträumen enthält: vergangene Woche/ Monat/ 6 Monate/ Jahr. Anschließend werden diese Daten für jede VO einzeln aufgelistet. Nach den Tabellen für die VOs werden diese Daten für die einzelnen FQANs angezeigt. Für diese Tabellen ist noch anzumerken, dass VOs und FQANs erst auf dieser Seite erscheinen sobald mindestens ein initialer Beitritt registriert wurde.

4.1.8 Druckansicht

Eine weitere Anforderung an die GUI war es eine übersichtliche Druckansicht zu gestalten. Dies wurde mittels eines extra CSS-Stylesheets gelöst welches das Menü, den Banner und überflüssige Buttons ausblendet. Das Stylesheet wird im Header jeder jsp-Seite als Print-Stylesheet eingebunden, wie in Listing 4.5 gezeigt. Die Webbrowser selbst erstellen die Druckansicht ohne Hintergrundfarben und zeigen den Text in schwarz an. Ein Beispiel einer Druckansicht findet sich in Abbildung 4.12. Zum Vergleich kann Abbildung 4.10 herangezogen werden.

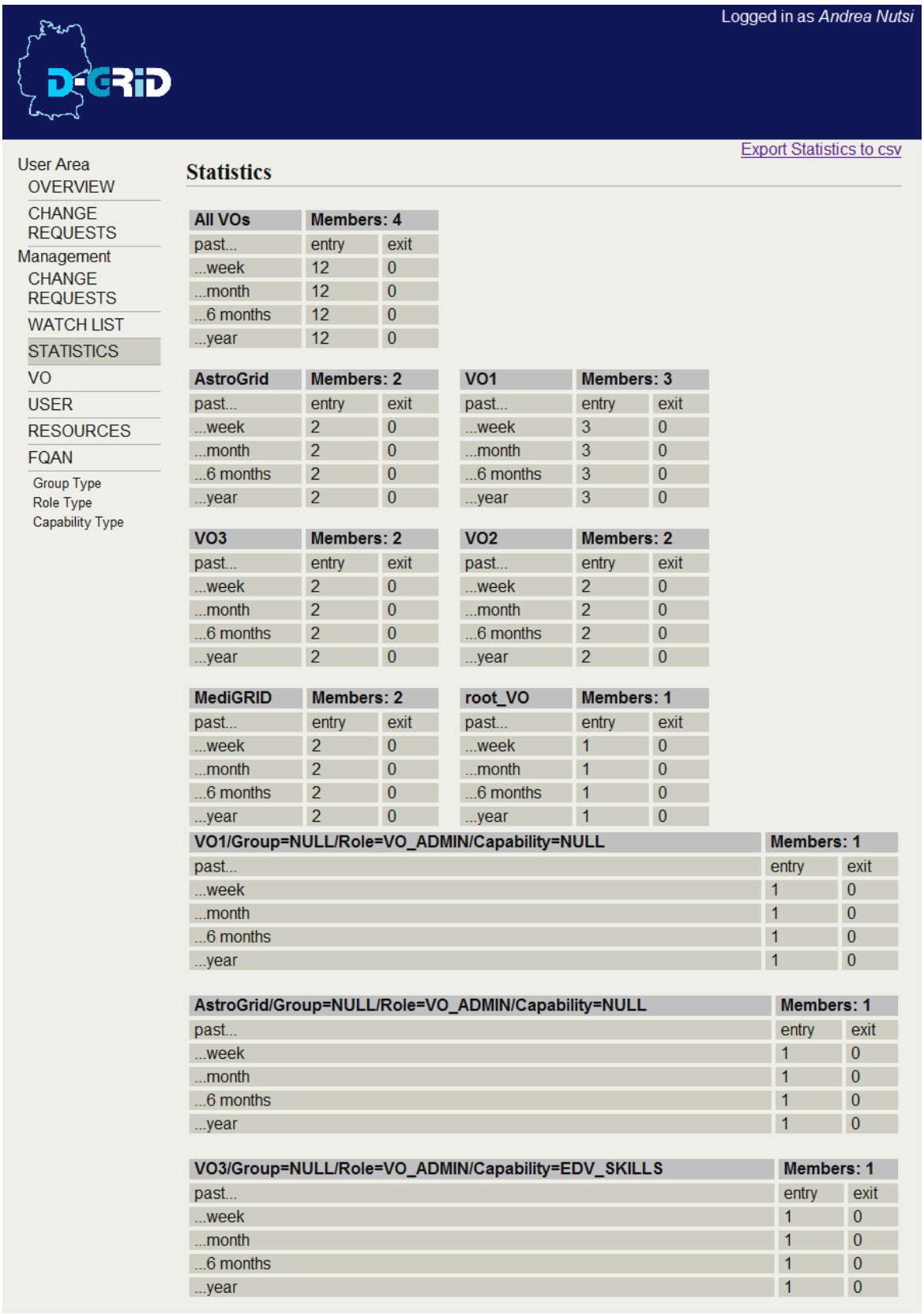

Abbildung 4.11: Screenshot des Statistik des VO-Layers

```
<html>
  <head>
    [...]
    <link rel="stylesheet" type="text/css"
      href="css/print.css" media="print" />
    [...]
  </head>
  [...]
</html>
```

Listing 4.5: Einbindung des print-Stylesheets in die jsp-Seiten

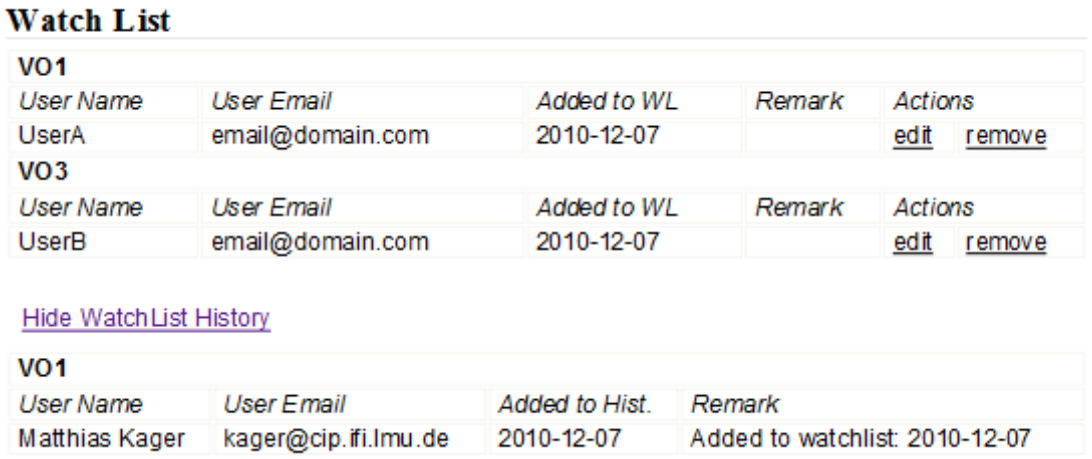

Abbildung 4.12: Screenshot der Druckansicht am Beispiel der WatchList

4.2 Systemumgebung

Im Folgenden wird auf die verwendete Software eingegangen und die Ordnerstruktur der Webanwendung erläutert.

4.2.1 Verwendete Software

Ursprünglich wurde der VO-Layer als Java-Anwendung implementiert. Im Rahmen dieser Arbeit wurde der Prototyp als Webanwendung realisiert, um den Benutzern eine einfachere Verwendung zu ermöglichen. Somit ist beispielsweise beim Client keine Installation notwendig, da ein Browser im Allgemeinen bereits vorhanden ist. Des weiteren können Updates der Software zentral und unabhängig von Benutzeraktionen erfolgen. Außerdem kann durch die Webanwendung für den Client eine bessere Plattformunabhängigkeit erreicht werden als mit dem ursprünglichen VO-Layer. Dieser ist aufgrund der Verwendung von Shell-Skripten, die nur unter Linux lauffähig sind, auf ein Linux Betriebssystem beschränkt.

Software	Version	Webadresse
Java SDK	1.6	http://www.java.com
MySql	5.1.41	http://www.mysql.com
MySql Connector	5.1.10	http://www.mysql.de/products/connector
Apache log4j	1.2.15	http://logging.apache.org/log4j
Apache Tomcat	6.0.24	http://tomcat.apache.org
Apache Commons FileUpload	1.2.1	http://commons.apache.org/fileupload
Apache Commons IO	1.4	http://commons.apache.org/io
JSTL core libary	1.1	http://java.sun.com/jsp/jstl/core
jQuery	1.4.2	http://jquery.com
CeeBox jQuery Plugin	2.1.4	http://github.com/catcubed/ceebox

Tabelle 4.1: Verwendete Software und Bibliotheken

In Tabelle 4.1 sind verwendete Software und Bibliotheken aufgeführt. Die GUI wird mittels Java Server Pages (JSP) realisiert. Hierfür wird der Apache Tomcat Server benötigt. Um die Übersichtlichkeit und Lesbarkeit des Codes zu verbessern wird die JSTL core libary verwendet. jQuery und das aufgeführte Plugin dienen der Darstellung der Detailansicht und der Ajax-Requests. Das jQuery Plugin CeeBox benötigt zusätzlich weitere Plugins (color, easing, metadata) von jQuery, welche aber von der CeeBox Webseite als Gesamtpaket geladen werden können.

Die Anwendung wurde im Firefox 3.6 sowie Internet Explorer 8 getestet. Um eine korrekte Funktionalität zu gewährleisten, muss JavaScript im Browser aktiviert sein.

4.2.2 Ordnerstruktur der Webanwendung

Die Ordnerstruktur der Webanwendung kann Abbildung 4.13 entnommen werden. Im Folgenden wird auf die einzelnen Verzeichnisse näher eingegangen.

Abbildung 4.13: Ordnerstruktur der Webanwendung

WebContent

Im Wurzelverzeichnis befinden sich alle direkt aufrufbaren jsp-Dateien sowie weitere Verzeichnisse. Aufgrund der überschaubaren Anzahl an Seiten wurde auf eine weitere Unterteilung in Unterverzeichnisse verzichtet. Dateien für den Adminbereich beginnen mit `admin_` und werden dadurch vom `VO-Tag` als Administrationsseite erkannt. Die Seiten, die von einem Ajax-Request aufgerufen werden, enden mit `_Ajax`.

WebContent/css

Die CSS Stylesheets werden im Ordner `css` abgelegt. Darin befinden sich die Styles für den VO-Layer (`style.css` und `print.css`) sowie die Stylesheets für das CeeBox jQuery Plugin (`ceebox.css` und `ceebox-min.css`).

WebContent/images

Die einzigen Grafiken die in der Webanwendung des VO-Layer vorkommen sind das D-Grid-Logo für den Banner der Webanwendung, eine Loading-Animation sowie die Bilder des CeeBox jQuery Plugins für die Detailansichten.

WebContent/js

Neben der eigenen `script.js` liegen in diesem Verzeichnis die Dateien von jQuery und den benötigten jQuery Plugins.

WebContent/WEB-INF

Dieses Verzeichnis und seine Unterverzeichnisse können vom Browser aus nicht aufgerufen werden. Die einzige Datei, die sich darin befindet ist die `web.xml`, in der die Webanwendung konfiguriert wird. Außerdem werden darin Parameter für die Applikation definiert. Die Beschreibung der Konfiguration erfolgt im Abschnitt 4.3.

WebContent/WEB-INF/includes

Wie bereits in einem vorangegangenen Abschnitt erwähnt befinden sich in diesem Verzeichnis die jsp-Dateien, welche in mehreren jsp-Seiten eingebunden werden. Dabei handelt es sich um die beiden Header und den Footer der Webanwendung.

WebContent/WEB-INF/lib

In diesem Verzeichnis werden die benötigten Bibliotheken für Java und JSP als jar-Archive abgelegt.

WebContent/WEB-INF/tld

Die Tag Library Definitions werden im `tld` Verzeichnis hinterlegt. Darin werden alle Definitionen der JSTL sowie die des `VOTags` abgelegt. Da in jeder JSP-Seite angegeben wird, welche Tags benötigt werden, ergibt sich daraus kein Performanzverlust. Auf die Definition des `VOTags` wird im Kapitel 4.3 eingegangen.

WebContent/WEB-INF/classes

Dieses versteckte Verzeichnis enthält den kompilierten Bytecode der definierten Java-Klassen. Aus Sicherheitsgründen sollte der Sourcecode nicht im Webverzeichnis liegen.

4.3 Deployment

In diesem Abschnitt werden die notwendigen Schritte zur Installation der Webanwendung auf einem Server erläutert und die notwendigen Konfigurationsdateien beschrieben.

Keystore erzeugen

Der Tomcat Server verwendet Zertifikate für die client- und serverseitige Authentifizierung. Für die serverseitige Authentifizierung wird ein Hostzertifikat benötigt, welches über das Programm `keytool` der Java Runtime erzeugt werden kann. Dabei wird das Zertifikat in einem Keystore gespeichert. Dieser wird im root-Verzeichnis des Servers abgelegt.

```
\Pfad\zu\jre6\bin\keytool -genkey -alias tomcat -keyalg RSA
  -keystore \Pfad\zu\apache-tomcat\truststore.jks -storetype JKS
```

Der Keystore kann weitere Zertifikate aufnehmen. Zur Realisierung der clientseitigen Authentifizierung wurde für das Testsystem eine eigene Simple CA erzeugt, die von der Webanwendung als ausstellende CA anerkannt wird. Dazu wird das erzeugte X509-Zertifikat dieser CA in den zuvor generierten Keystore importiert.

```
\Pfad\zu\jre6\bin\keytool -import -alias trustedCA1 -keystore
  \Pfad\zu\apache-tomcat\truststore.jks -trustcacerts -file
  \Pfad\zu\cert\myrootca.crt
```

Apache Tomcat konfigurieren

Dem Server muss nun der Ort des Keystore mitgeteilt werden. Dazu wird in der Datei `server.xml` im `conf` Ordner des Server-Verzeichnisses ein neuer Connector-Eintrag eingefügt, der Listing 4.6 entnommen werden kann. Da die Zertifikate des Hosts und der CA im selben Keystore liegen, werden die Attribute `keystoreFile` und `truststoreFile` mit dem selben Wert belegt. Die Angabe erfolgt in diesem Fall relativ zum Wurzelverzeichnis des Tomcat Servers.

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <Server port="8005" shutdown="SHUTDOWN">
3
4  [...]
5
6  <Connector SSLEnabled="true" clientAuth="true"
7      keystoreFile="truststore.jks"    keystorePass="hallo!?"
8      truststoreFile="truststore.jks" truststorePass="hallo!?"
9      maxThreads="150"   port="8443"
```

```
10        protocol="org.apache.coyote.http11.Http11NioProtocol"
11        scheme="https" secure="true" sslProtocol="TLS" />
12
13    [...]
14
15 </Server>
```

Listing 4.6: server.xml

Des weiteren muss für das Logging die Konfigurationsdatei `volayer_log4j.conf` in das `conf`-Verzeichnis kopiert werden. Dessen Inhalt kann Listing 4.7 entnommen werden. Dabei wird die Ausgabe in der Konsole ausgegeben sowie in die Datei `volayer.log` des Verzeichnisses `logs` im Wurzelverzeichnis des Apache Tomcat geschrieben.

```
1 # Set root logger level to DEBUG and its only appender to STDOUT
2 log4j.rootLogger=DEBUG, Console, File
3
4 # STDOUT is set to be a ConsoleAppender.
5 log4j.appender.Console=org.apache.log4j.ConsoleAppender
6 log4j.appender.Console.layout=org.apache.log4j.PatternLayout
7 log4j.appender.Console.layout.ConversionPattern=
8 %5p %c (%L) %m%n
9
10 # SYSTEM USES PatternLayout
11 log4j.appender.File=org.apache.log4j.RollingFileAppender
12 log4j.appender.File.file=logs/volayer.log
13 log4j.appender.File.layout=org.apache.log4j.PatternLayout
14 log4j.appender.File.layout.ConversionPattern=
15 %d %5p [%t] %c - %m%n
```

Listing 4.7: Konfiguration von log4j

Datenbank einrichten

Auf dem Testsystem wurde ein MySQL Server benutzt, auf dem die Datenbank `VOL_DB` angelegt werden muss. Um es dem Administrator zu ermöglichen, gleich Zugang zum VO-Layer zu erhalten, müssen die Daten seines DN im SQL-Dump eingetragen werden. Dabei handelt es sich um das Statement, das Daten in die Tabelle User einfügt. Anschließend kann der SQL-Dump in die Datenbank importiert werden.

Anwendung deployen

Bevor die Anwendung als Webarchiv mit der Endung .war gepackt wird, muss in der Konfigurationsdatei `web.xml` noch der entsprechende Port für den MySQL Server eingetragen werden. Die gesamte `web.xml` zeigt Listing 4.8. In der Konfigurationsdatei wird neben dem Kontext-Parameter mysql_port auch die Definition der verwendeten Tag-Bibliotheken eingebunden. Die Definition des Tags und die Zuordnung zum Taghandler kann Listing 4.9

entnommen werden. Des weiteren wird durch security constraints angegeben, dass die Anwendung nur über eine gesicherte Verbindung aufgerufen werden kann.

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <web-app id="WebApp_ID" version="2.4" xmlns="http://java.sun.com
       /xml/ns/j2ee" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
       instance" xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee
       http://java.sun.com/xml/ns/j2ee/web-app_2_4.xsd">
3   <display-name>VOLayer</display-name>
4   <welcome-file-list>
5    <welcome-file>index.html</welcome-file>
6    <welcome-file>index.htm</welcome-file>
7    <welcome-file>index.jsp</welcome-file>
8    <welcome-file>default.html</welcome-file>
9    <welcome-file>default.htm</welcome-file>
10   <welcome-file>default.jsp</welcome-file>
11  </welcome-file-list>
12
13  <jsp-config>
14   <taglib>
15       <taglib-uri>/volayer</taglib-uri>
16       <taglib-location>/WEB-INF/tld/volayer.tld</taglib-location>
17   </taglib>
18   <taglib>
19     <taglib-uri>http://jakarta.apache.org/taglibs/c</taglib-uri>
20     <taglib-location>/WEB-INF/tld/c.tld</taglib-location>
21   </taglib>
22  </jsp-config>
23
24  <security-constraint>
25   <web-resource-collection>
26     <web-resource-name>volayer</web-resource-name>
27     <url-pattern>/*</url-pattern>
28   </web-resource-collection>
29   <user-data-constraint>
30     <transport-guarantee>CONFIDENTIAL</transport-guarantee>
31   </user-data-constraint>
32  </security-constraint>
33
34  <context-param>
35   <param-name>mysql_port</param-name>
36   <param-value>3306</param-value>
37  </context-param>
38 </web-app>
```

Listing 4.8: web.xml

```
15
16  <!-- Namensraum der definierten Tags -->
17  <short-name>volayer</short-name>
18
19  <!-- Verknuepfung des Tags 'voLoggedIn' mit dem Tag-Handler -->
20  <tag>
21      <name>voLoggedIn</name>
22      <tag-class>vo_layer.gui.tags.VOTag</tag-class>
23  </tag>
```

Listing 4.9: volayer.tld: Definition des Tags volayer:voLoggedIn

Die Erzeugung des Webarchivs kann manuell durchgeführt werden, über ein Ant-Skript oder über das Entwicklungswerkzeug Eclipse[1]. An dieser Stelle wurde die einfachste Möglichkeit durch Eclipse verwendet. Der Sourcecode wird als Eclipse Projekt zur Verfügung gestellt, wodurch der Import in Eclipse und die weitere Bearbeitung vereinfacht werden. Wenn das Projekt ausgewählt wurde, kann über `File -> Export -> WAR File` das Webarchiv erstellt werden. Dabei sollte auf den Export der Source Files verzichtet werden, da es üblich ist, nur die kompilierten Dateien auf dem Server zu halten.

Das erhaltene Webarchiv kann nun auf den Server in das Verzeichnis webapps kopiert werden. Tomcat entpackt das Archiv automatisch und startet anschließend die Anwendung. Falls sich das Archiv ändert, wird die Anwendung gestoppt, das Archiv neu entpackt und neu gestartet.

Aufruf der Anwendung

Benutzer, die den VO-Layer verwenden wollen, müssen ein gültiges Zertifikat besitzen, welches von einer CA unterzeichnet wurde, deren Zertifikat in den Truststore importiert wurde. Der Anwender muss dieses als PKCS-12 in seinen Browser importieren und beim Aufruf der Anwendung auswählen.

[1] http://www.eclipse.org

5 Bewertung

In diesem Kapitel soll die neu entwickelte VO-Layer GUI abschließend evaluiert werden. Dazu wird noch einmal kurz aufgeführt was im Rahmen dieser Arbeit entwickelt wurde und welche Aspekte noch für einen Betrieb im D-Grid behandelt werden müssen. Außerdem wird diese GUI mit einer in [Lep10] entwickelten GUI verglichen und Möglichkeiten einer Integration aufgezeigt.

5.1 VO-Layer GUI

Diese Projektarbeit beschäftigte sich mit der Entwicklung einer GUI für den von Kirchler in [Kir08] entwickelten VO-Layer. Dabei beschränkte sich diese Arbeit auf den Management-Bereich und ermöglicht nun das einfache Erstellen, Modifizieren und Entfernen von Entities mittels einer grafischen Oberfläche. Zusätzlich wurde das Konzept des VO-Layers um Change Requests und eine WatchList erweitert. Es wurden vier Arten von Change Requests eingeführt: ein Register-, Certificate-, FQAN- und VO-Change Request. Dies geschah vor dem Hintergrund, das in dieser Arbeit zwischen normalen Benutzern und VO-Admins unterschieden wird, was im vom Kirchler entwickelten VO-Layer zwar vorgesehen aber noch nicht umgesetzt wurde. Daher können normale Benutzer keine Änderungen ihrer Rechte oder Mitgliedschaften direkt vornehmen, sondern müssen dies mittels Change Request bei dem jeweiligen VO-Admin beantragen. Eine zusätzliche Erweiterung ist das Konzept der WatchList welches es ermöglicht Benutzer temporär aus einer oder mehreren VOs auszuschließen, falls sie ein auffälliges Verhalten zeigen. Zusätzlich zu den Management-Funktionen der Entities, die einem VO-Admin zur Verfügung stehen, wurde noch eine rudimentäre Statistik umgesetzt, die Ein- und Austritte aus VOs sowie das Hinzufügen und Entfernen von FQANs aufführt.

Ein weiterer wichtiger Punkt ist die grundlegende Änderung des Konzepts des VO-Layers. Ursprünglich war von Kirchler angedacht den VO-Layer auf jeden Client separat zu installieren und nur die Datenbank zentral auf einem Server laufen zu lassen. Mit dieser erweiterten Version wurde eine Webanwendung geschaffen die zentral auf einem Tomcat-Server läuft und nicht mehr bei den einzelnen Clients. Die Gründe hierfür wurden bereits im Abschnitt 4.2.1 beschrieben.

5.2 Erweiterungen und Einschränkungen des Prototypes

Der von Kirchler entwickelte VO-Layer wurde in seiner Arbeit als Prototyp umgesetzt. Dieser wurde im Rahmen dieser Arbeit weiterentwickelt. Dabei ist ein recht umfangreicher und mit vielen Funktionen ausgestatteter Prototyp entstanden, welcher dennoch weiterentwickelt werden muss um im realen Betrieb einsatzfähig zu sein.

Um den Umfang dieser Arbeit in einem überschaubaren Rahmen zu halten, wurde die VO-Layer Datenbank als gegeben betrachtet. Die von Kirchler entwickelten Skripte zum

Datenaustausch mit den Middlewares wurden außer Acht gelassen, genauso wie das komplette Job-Management. An dieser Stelle besteht zukünftig Entwicklungsbedarf.

Nachdem beschlossen wurde, Entities nicht zu löschen sondern nur zu deaktivieren, verbleiben unter Umständen riesige Datenmengen in der Datenbank welche die Performanz erheblich beeinträchtigen können. Außerdem ist die Datenbank ein Single-Point of Failure und wie bereits in [Kir08] beschrieben, sollte für ein Einsatz im D-Grid ein kommerzielles Datenbankprodukt in Erwägung gezogen werden.

Ein Aspekt welcher in dieser Arbeit ebenfalls außer Acht gelassen wurde, allerdings für den Einsatz im D-Grid unerlässlich ist, ist das Ressourcen Host-Zertifikate benötigen. Um dies in einer zukünftigen Version des VO-Layers zu ermöglichen muss das Datenmodell erweitertet und die Managementfunktionen für Ressourcen angepasst werden. Um die Zuordnung von Ressourcen zu VOs zu erleichtern, sollte eine zusätzliche Administratorrolle für Resource-Provider eingeführt werden, die das Recht haben mit den Ressourcen zu arbeiten. Zurzeit können alle VO-Administratoren eine Ressource anlegen, diese aber nur den VOs hinzufügen, die sie administrieren dürfen. Die Rolle des Resource-Providers könnte also an Ressourcen gebunden werden. VO-Administratoren könnten anschließend mit Hilfe eines Change Requests an den Resource-Provider den Beitritt einer Ressource in seine VO beantragen. Ein weiteres Feature wäre die Möglichkeit, Change Requests an andere Administratoren weiterzuleiten. Dazu bedarf es nur weniger Anpassungen an das bisherige System. Zusätzlich wäre es für die Benutzerfreundlichkeit vom Vorteil, wenn der VO-Layer bei wichtigen Ereignissen, zum Beispiel die Abarbeitung der Registrierung, Emails an die Anwender verschicken würde.

Des Weiteren muss für den VO-Layer ein vollständiges Sicherheitskonzept entwickelt werden um den Einsatz in einer Grid-Umgebung zu ermöglichen. Ebenfalls zum Sicherheitskonzept gehört ein umfangreiches Logging. Dieses wird momentan überwiegend für das Debugging verwendet, da es sich noch um einen Prototyp handelt und für zukünftige Erweiterungen noch dazu benötigt werden wird. In einer finalen Version sollte aus den Logfiles konkret ersichtlich sein, welcher Admin wann welche Änderungen an den Datenbankinhalten vorgenommen hat. Diese Informationen könnten den entsprechenden Administratoren anschließend als Historie angezeigt werden.

Die Statistik wurde nur rudimentär umgesetzt. Sie muss zum einen noch mit einer Filterfunktion versehen werden. Außerdem muss ein Sichtbarkeitskonzept erarbeitet werden, da momentan alle VO-Admins Einsicht in alle Statistiktabellen haben.

Wie bereits besprochen wird zur Zeit als Server ein Apache Tomcat eingesetzt. Durch die im Verlauf dieser Arbeit gesammelten Erkenntnissen ist die Administration und Einrichtung sehr aufwändig und mäßig dokumentiert. Daher sollte überlegt werden, ob es eine bessere Lösung gibt.

Zukünftig sollte noch einmal darüber diskutiert werden, in wie weit der Einsatz des JavaScript-Frameworks jQuery sinnvoll ist. Neben den vielen bereits genannten Vorteilen von jQuery hat es den Nachteil das der Code bei komplexeren Vorgängen schnell unübersichtlich werden kann und somit die Wartbarkeit deutlich erschwert wird.

Ein grundlegendes Problem des VO-Layers ist die Beschränkung auf drei Middlewares in bestimmten Versionen. Soll zukünftig eine aktuellere Version einer Middleware verwendet werden, so muss der Code des VO-Layers angepasst werden. Des Weiteren wäre eine Anzeige der aktuell vom VO-Layer unterstützten Middlewares mit ihren Versionen wünschenswert. Diese Liste sollte in die VO-Layer GUI integriert werden.

5.3 Vergleich VO-Layer GUI mit VOMS-AJAX-GUI

Im Rahmen eines Fortgeschrittenenpraktikums an der LMU mit dem Titel "VO Membership Management mit VOMS und Ajax", zu finden in [Lep10], wurde ebenfalls eine GUI für ein VO-Management entworfen. Wie dem Titel der Arbeit bereits zu entnehmen ist, wurde die VOMS-Datenbank als Datenquelle herangezogen. Im Gegensatz dazu ist der VO-Layer unabhängig von VOMS entwickelt worden und stellt eher eine Alternative dazu dar. Die Vorteile des VO-Layers gegenüber VOMS können [Kir08] entnommen werden. Zum einen unterstützt der VO-Layer im Gegensatz zu VOMS das Ressourcen Bestandteile von VOs sein können. Zum anderen stellt der VO-Layer eine Abstraktionsschicht über den drei unterstützten Middlewares dar, während VOMS in jede Middleware einzeln eingebunden werden muss. Genau wie in der Arbeit von Lepiarczyk wurde auch in dieser Arbeit mit Ajax gearbeitet und auf eine übersichtliche Darstellung geachtet. Im Rahmen dieser Arbeit kam allerdings das JavaScript Framework jQuery zum Einsatz während Lepiarczyk das Framework Prototype verwendete. Da er dies bereits selbst als Kritikpunkt ansieht, sollte bei einer Integration seiner Arbeit in die hier entwickelte GUI, sein Code ebenfalls auf jQuery umgestellt werden. Die VO-Layer GUI nutzt Ajax nicht so intensiv wie die VOMS-AJAX-GUI, was etwas weniger Netzwerklast mit sich zieht. Beispielsweise werden bei der VOMS-AJAX-GUI alle Daten mittels AJAX-Requests abgerufen, während die JSP-Seite nur ein leeres HTML-Gerüst enthält. Bei der VO-Layer GUI enthält die erste Response bereits Daten und Ajax wird nur verwendet um weitere Daten nach Anforderung abzurufen. Die VOMS-AJAX-GUI bietet eine alternative Ansicht des Gruppen-, Rollen- und Attributmanagements und könnte auch als solche in die VO-Layer GUI integriert werden. So könnte ein Anwender zwischen zwei verschiedenen Ansichten wählen. Die VOMS-AJAX-GUI bietet den Vorteil einer integrierten Filterfunktion und eines InPlace-Editors, nachteilig ist allerdings das die Darstellung extrem breit und somit eventuell auch unübersichtlich werden kann. Dies ließe sich zukünftig, wie bereits in [Lep10] beschrieben, mit CSS3 lösen.

6 Zusammenfassung

Abschließend soll an dieser Stelle der Inhalt dieser Projektarbeit kurz zusammengefasst werden.

Im Rahmen dieser Arbeit wurde für den von Kirchler in [Kir08] entwickelten VO-Layer eine GUI zum Management von Entitäten entworfen und implementiert. Dabei wurde stets auf eine übersichtliche und intuitive Darstellung geachtet. Außerdem wurde das Konzept des VO-Layers erweitert. In dieser aktualisierten Version wird zwischen normalen Benutzern und Administratoren unterschieden, wobei die normalen Benutzern nur Einsicht in ihre Daten erhalten. Sie können mittels des neu geschaffenen Konzepts der Change Requests Änderungswünsche ihrer VO-Mitgliedschaft, FQANs oder ihres Zertifikats an den zuständigen Administratoren senden. VO-Administratoren können wiederum diese Change-Requests bearbeiten sowie auffällige Benutzer auf die WatchList setzen und diese somit aus einer oder mehreren VOs sperren. Außerdem haben Admins die Möglichkeit Entitäten zu erstellen, modifizieren und zu deaktivieren. Zusätzlich stehen ihm Statistiken zu VOs und FQANs zur Verfügung.

Der VO-Layer wurde nun als Webanwendung implementiert, der Zugriff darauf wird mittels eines Zertifikats ermöglicht.

Um den VO-Layer im D-Grid einzusetzen sind noch wichtige Erweiterungen notwendig, welche im Abschnitt 5.2 aufgeführt wurden.

Abbildungsverzeichnis

Listings

Literaturverzeichnis

[DGr09] *Betriebskonzept für die D-Grid Infrastruktur*, 2009. http://www.d-grid.de/fileadmin/user_upload/documents/Kern-D-Grid/Betriebskonzept/D-Grid-Betriebskonzept-V2.0.pdf.

[Kir08] Kirchler, Wolfgang: *Entwicklung einer einheitlichen Autorisierungs- und Authentifizierungsschnittstelle für heterogene Grids am Beispiel D-Grid*. Diplomarbeit Technische Universität München, September 2008. http://www.nm.ifi.lmu.de/pub/Diplomarbeiten/kirc08.

[Lep10] Lepiarczyk, Jakob: *VO Membership Management mit VOMS und AJAX*. Fortgeschrittenenpraktikum LMU München, August 2010. http://www.nm.ifi.lmu.de/pub/Fopras/lepi10/.