

**Dokumentation JProject**

*Entwurf und Implementierung einer Projektverwaltung mittels JEE 5 Webtechnologien*

**Team:**

*Andy Klay*

*Michael Koppen*

*Tino Reuschel*

Inhaltsverzeichnis

1. Einführung..............................................................................................................4

2. Portal-Funktionen...................................................................................................5

2.1 Funktionale Sicht/User-Sicht auf das Portal......................................................5

2.2 Funktionen Detailsicht.......................................................................................5

2.2.x Use-Case x.................................................................................................5

3. Architektur..............................................................................................................6

3.1 Schichten..........................................................................................................6

3.2 Komponenten....................................................................................................6

3.2.x Komponente <Name> (z.B. UserManager, ShoppingCard, etc) .................6

4. Systemvoraussetzungen........................................................................................7

4.1 Systemspezifische Komponenten.....................................................................7

4.2 Allgemeine Voraussetzungen............................................................................7

5. Anhang.......................................................................................................................8

A1 Execution Sicht/Deployment Sicht.....................................................................8

A2 Schnittsellen-Sprache (falls vorhanden z.B. DTD).............................................8

A3 Konfigurationseinstellungen...............................................................................8

A4 Testszenarien.....................................................................................................8

A5 Monitoring (welche Ports sind erreichbar, Prozesse, Directories, ...).................8

**1. Einführung**

JProjectist in erster Linie eine allgemeine Projektverwaltungsumgebung die von jedem Rechner, der Internet hat, erreicht werden kann.

JProject wird in 3 Teile unterteilt: JProject, Content und Wiki.

Mit JProject kann eine Benutzergruppe Teams zur Realisierung eines Projektes bilden und sich selbst organisieren. Die User können Projekte erstellen, Dokumente/Sourcecode hochladen und Aufgaben anlegen und zuordnen. Dazu steht ein flexibles Rechtesystem zur Seite.

Zum späteren Funktionsumfang soll eine Umfangreiche Datenbank mit Hilfestellungen, Anleitungen bzw. Tutorials zur Verfügung stehen. Dieser Bereich nennt sich Content.

Des Weiteren soll später ein Bereich namens Wiki eingerichtet werden, in dem die User allgemeine Informationen zum Ablauf und zur Organisation, in derenHerarchie das Projekt entworfen werden soll, finden.

***Funktionaler Umfang der Idee:***

- Rechtesystem für die Projekte

- Öffentlicher Bereich für Projekte

- Download/Upload des Contents (später mit Versionskontrolle)

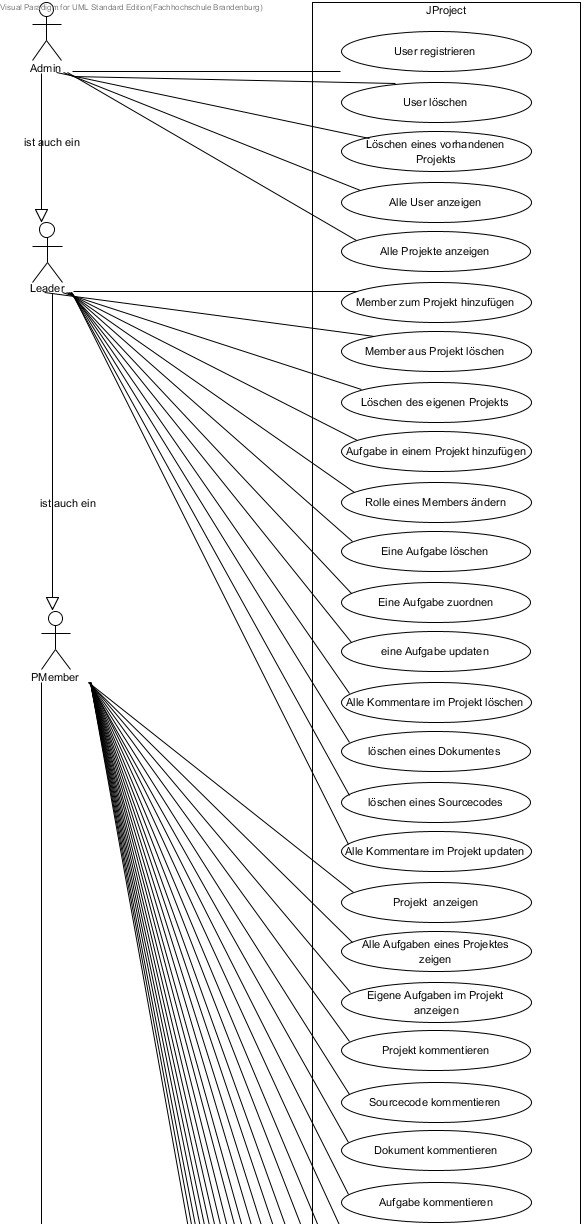
- Synchrones und Verlustfreies arbeiten am Projekt

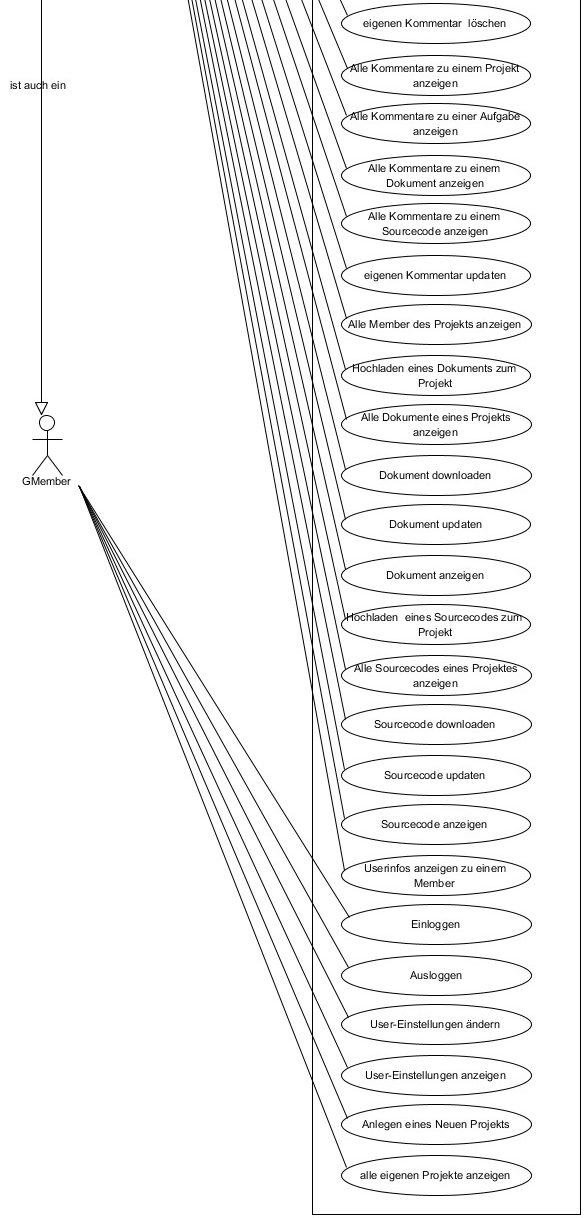
- Termin- und Aufgabenverwaltung innerhalb der Projekte



(Abb. 1 Ursprünglicher Ideen-Entwurf)

**2. Portal-Funktionen**

***2.1 Funktionale Sicht/User-Sicht auf das Portal***



In dem Usecasediagramm ist nur eine starre Konstellation der Rollen zu erkennen.

Implementiert wurde ein Rollensystem welches erlaubt dynamisch Rollen hinzu zufügen.

Es gibt ein globales Rollensystem (siehe auch GlobalRolesManager)in ihm wird unterschieden welche Aktionen der Benutzer im Allgemeinen tätigen darf. Beispielsweise ist hierim Usecasediagramm zu sehen der GMember (User ohne Projekt) und der Admin, dies sind zwei mögliche globale Rollen.

Als Weiteres gibt es aber noch ein Rollensystem nur für Projekte (siehe auch ProjectRolesManger) in ihm wird unterschieden welche Aktionen der Benutzer innerhalb eines Projektes tätigen darf. Als Beispiel dafür ist im Usecasediagramm der Leader und der PMember zu sehen.

Die globalen Rollen und Projektrollen sind für den Betrieb des Systems zwingend erforderlich d.h. es sind mindestens 2 Rollen Global (Admin und Member)und 2 Projektrollen (Leader und Member) erforderlich.

(Admin kann aber keine Tasks zuordnen!)

Der Admin hat durch seine Globale Rolle eine übergeordnete Funktion und darf grundsätzlich alles was andere Rollen auch dürfen, dann kommt der Projektleader, derinnerhalb eines Projektes wiederum mehr Rechte hat als ein Projektmember. User die in einem Projekt nicht involviert sind, also weder Member noch Leader (GMember) sind und haben daher nur grundlegende Aktionen wie z.B. Einloggen, Ausloggen, Projekt anlegen.

Die Rollen können derzeit von Ersteller des Systems umbenannt und auch deren Rechte dynamisch geändert oder auch komplett neue Rollen hinzugefügt werden.

***2.2 Funktionen Detailsicht***

**Admin**= d.h. also ein User mit entsprechenden Rechten, muss nicht Admin heißen.

**Projekt-Leader**=d.h. also ein User mit entsprechenden Rechten der in einem Projekt mehr Rechte hat als ein einfacher Teilnehmer(**Projekt-Member**).

Es wird hier im Folgenden nicht mehr immer explizit erwähnt, dass alle Aktionen die der Projekt-Leader machen kann, auch der Admin über die AdminConsole tätigen kann.

**2.2.1 User registrieren**

Über die AdminConsole kann ein Admin einen User im System erstellen, damit dieser das System nutzen kann.(Ein Externer kann sich nicht registrieren, da es sich in dieser Version um ein geschlossenes System handelt. Diese Option kann bei Bedarf hinzugefügt werden)

**2.2.2 User löschen**

Über die AdminConsole kann ein Admin einen User aus dem System löschen, damit dieser das System nicht mehr nutzen kann.

**2.2.3 Löschen eines vorhandenen Projekts**

Ein Projekt kann direkt gelöscht werden vom Admin über die AdminConsole.

**2.2.4 Alle User anzeigen**

Nur der Admin kann über die AdminConsole alle User anzeigen.

**2.2.5 Alle Projekte anzeigen**

Nur der Admin kann über die AdminConsole alle Projekte anzeigen.

**2.2.6 Member zum Projekt hinzufügen**

Der Projekt-Leader kann über das Projekt einen User als Projekt-Member hinzufügen.

**2.2.7 Member aus Projekt löschen**

Der Projekt-Leader kann über das Projekt einen Projekt-Member aus dem Projekt entfernen.

**2.2.8 Löschen des eigenen Projekts**

Ein Projekt kann gelöscht werden, indem der Projekt-Leader alle Member des Projektes löscht und zuletzt sich selbst.

**2.2.9 Aufgabe in einem Projekt hinzufügen**

Der Projekt-Leader kann über das Projekt und über den Reiter „Task“ eine Aufgabe zum Projekt hinzufügen.

**2.2.10 Rolle eines Members ändern**

Der Projekt-Leader kann über das Projekt und über den Reiter „Project“ im zweiten Abschnitt die Rolle der vorhandenen Member im Projekt ändern. Des Weiteren kann der Leader seine eigene Rolle im Projekt ändern, jedoch muss min. ein Projekt-Leader vorhanden sein.

**2.2.11 Eine Aufgabe löschen**

Der Projekt-Leader kann über das Projekt und über den Reiter „Task“ in der Leiste links eine Aufgabe des Projekts löschen durch klicken auf das Kreuz neben einer Aufgabe.

**2.2.12 Eine Aufgabe zuordnen**

Der Projekt-Leader kann über das Projekt und über den Reiter „Task“ eine Aufgabe des Projekts einem Member zuordnen. Durch die Auswahl der vorhandenen Aufgabe links und Auswahl des Members rechts und klicken auf Zuweisen kann dies bewerkstelligt werden.

2.2.13 eine Aufgabe updaten

2.2.14 Alle Kommentare im Projekt löschen

2.2.15 löschen eines Dokumentes

2.2.16 löschen eines Sourcecodes

2.2.17 Alle Kommentare im Projekt updaten

2.2.18 Projekt anzeigen

2.2.19 Alle Aufgaben eines Projektes zeigen

2.2.20 Eigene Aufgaben im Projekt anzeigen

2.2.21 Projekt kommentieren

2.2.22 Sourcecode kommentieren

2.2. 23 Dokument kommentieren

2.2. 24 Aufgabe kommentieren

2.2. 25 eigenen Kommentar löschen

2.2. 26 Alle Kommentare zu einem Projekt anzeigen

2.2. 27 Alle Kommentare zu einer Aufgabe anzeigen

2.2. 28 Alle Kommentare zu einem Dokument anzeigen

2.2. 29 Alle Kommentare zu einem Sourcecode anzeigen

2.2. 30 eigenen Kommentar updaten

2.2. 31 Alle Member des Projekts anzeigen

2.2. 32 Hochladen eines Dokuments zum Projekt

2.2. 33 Alle Dokumente eines Projekts anzeigen

2.2. 34 Dokument downloaden

2.2. 35 Dokument updaten

2.2. 36 Dokument anzeigen

2.2. 37 Hochladen eines Sourcecodes zum Projekt

2.2. 38 Alle Sourcecodes eines Projektes anzeigen

2.2. 39 Sourcecode downloaden

2.2. 40 Sourcecode updaten

2.2. 41 Sourcecode anzeigen

2.2.42 Userinfos anzeigen zu einem Member

2.2.43 Einloggen

2.2.44 Ausloggen

2.2.45 User-Einstellungen ändern

2.2.46 User-Einstellungen anzeigen

2.2.47 Anlegen eines Neuen Projekts

2.2.48 alle eigenen Projekte anzeigen

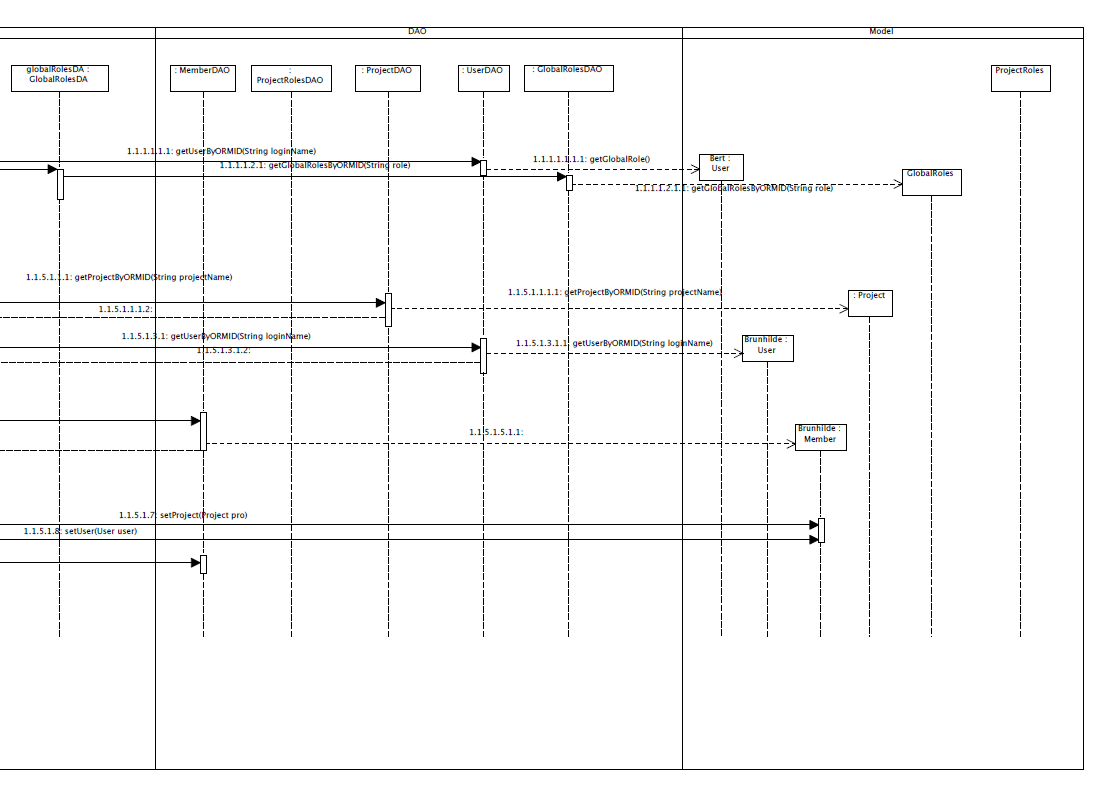
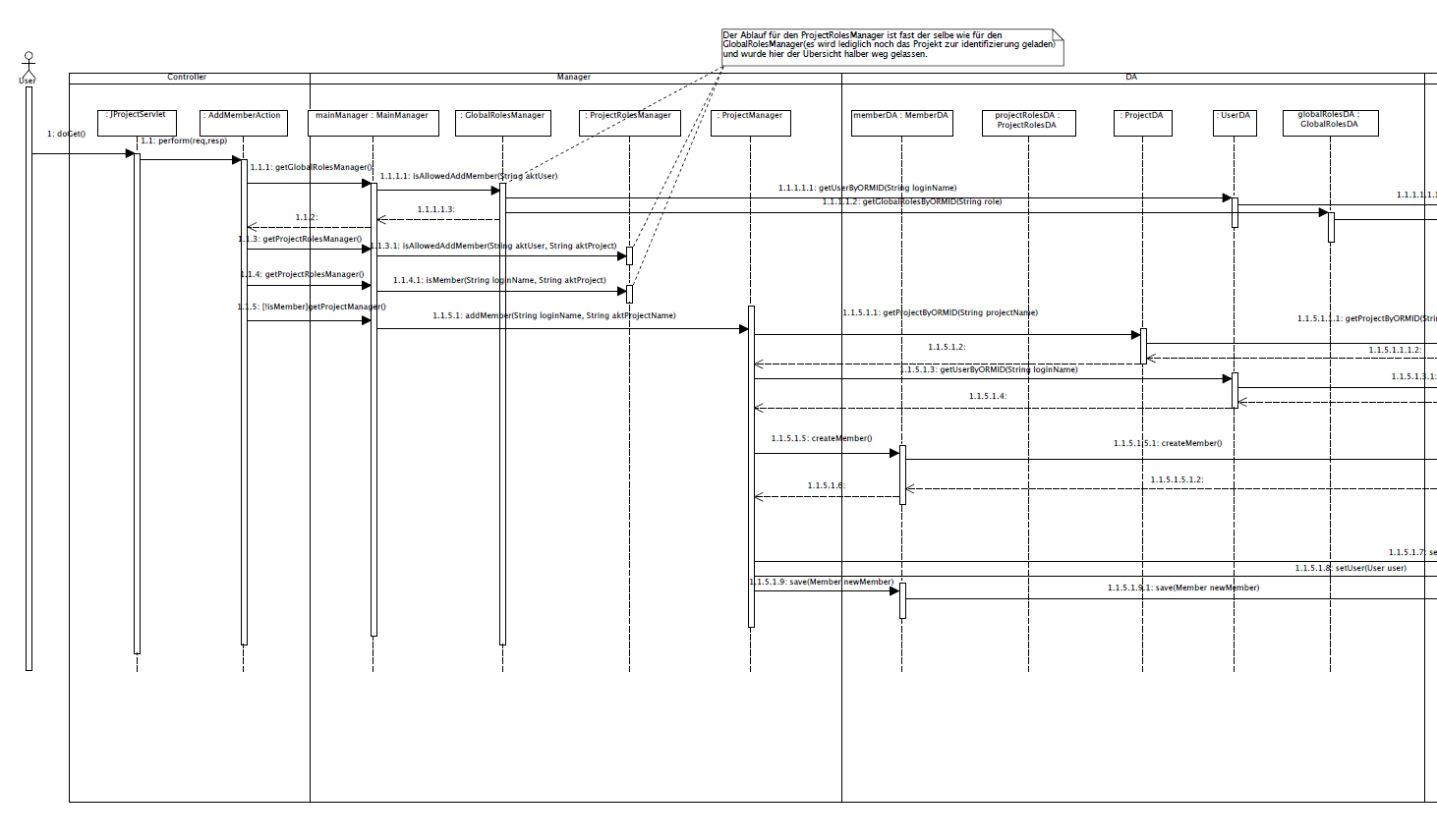
// wird spätestens Samstag 12 Uhr von Tino zugeschcikt

3. Architektur

*3.1 Schichten*



- Mindestens ein Sequenzdiagramm, das einen interessanten Ablauf über Schichten/Rechnergrenzen hinweg veranschaulicht, anhand dessen die Funktionalität des Systems erklärt wird (die genaue Sicht der einzelnen Kommunikationsszenarien würde man unter 3.2 angeben)



(Abb. 3 Sequenz-Diagramm)

|  |  |
| --- | --- |
| Name | AddMemberAction |
| Kurzbeschreibung | Dieses Sequenz-Diagramm(Abb. 2) stellt dar, wie der ein Projekt-Leader(Bert : User bzw. Bert : Member) einen Member(Brunhilde : User bzw. Brunhilde : Member) zu seinem Projekt hinzufügt. |
| Akteure | User |
| Auslöser | Eintragung des Usernamen und klick auf Button „Member hinzufügen“ |
| Vorbedingung | User eingeloggt, User hat Projekt indem er Leader ist, der hinzu zu fügende User ist nicht bereits im Projekt |
| Ergebnis | User wird dem Projekt zugeordnet(Member wurde erstellt) |
| Nachbedingung | keine |
| Essentieller Ablauf | Ablauf:   * ***1. doGet()*** * Der Request wird an das JProjectServlet gesendet. * Es überprüft ob der User einer Session zugeordnet ist. * Es ordnet dem Request die Aktion „AddMemberAction“ zu. * ***1.1 perform(req, resp)*** * Die Action zieht aus dem Request alle erforderlichen Parameter heraus. * Die Action AddMemberAction überprüft ob der User eingeloggt ist. * ***1.1.1 getGlobalRolesManager()*** und folgende - Die Action überprüft, ob der User Globale-Rechte hat um diese Action auszuführen. * ***1.1.3 getProjectRolesManager()*** und folgende - Die Action überprüft, ob der User Projekt-Rechte hat um die Action auszuführen, sofern er nicht schon Globale Rechte hatte. * ***1.1.4 getProjectRolesManager()*** und folgende - Die Action überprüft, ob der angegebene User eventuell schon Member in diesem Projekt ist. * ***1.1.5 getProjectManager()*** – Sofern der User nicht Member in diesem Projekt ist, wird der Member hinzugefügt (1.1.5.1) * ***1.1.5.1.1 getProjectByORMID(String projectName)*** und folgende - Das Projekt wird geladen. * 1***.1.5.1.3 getUserByORMID(String loginName)*** und folgende – Der hinzu zu fügende User wird geladen. * ***1.1.5.1.5 createMember()*** und folgende – Ein neuer Member wird erzeugt. * ***1.1.5.1.7 setProject(Project pro) –*** Das Attribut „Projekt“(PK) im neuen Member wird gesetzt. * ***1.1.5.1.8 setUser(User user) –*** Das Attribut „User“(PK) im neuen Member wird gesetzt. * ***1.1.5.1.9 save(Member newMember)*** und folgende – Der neue Member wird in die Datenbank gespeichert. |

Das Ganze nicht nur mit Diagrammen, sondern auch schriftlich erkären, was die Diagramme ausdrücken.

Dieses Unterkapitel soll zunächst einen schnellen Überblick über die Gesamtarchitektur vermitteln. In 3.2 wird dann näher auf die einzelnen Komponenten/Klassen eingegangen.

## 3.2 Komponenten

* siehe Grobansicht Paketdiagramm (Anhang A3)
* siehe Detailansicht Klassendiagramm(Anhang A???)

Eventuell hat Ihre Anwendung ausschließlich eine Komponente, da Se keine weiteren Schnitte zwischen einzelnen Anwendungsmodulen (z.B. UserManager/UserBO, SoppingCard, ...) vorgenommen haben. Dann würde man sicher hier von Klassen-Sicht und nicht von Komponenten sprechen und - wenn überhaupt - die Schichten als Unterkapitel aufteilen.

Es sollte hier auf jeden Fall herauskommen, wer welches Objekt benutzt, so dass man die Abhängigkeiten erkennen kann. Insbesondere ist das für die Value Objekte interessant, da diese die Datenstruktur für die Anwendung widerspiegeln. Da es sich hier um die technische Sicht geht, sollte man auch bereits mögliche Aggregationen und Kompositionen (siehe auch UML) berücksichtigen.

3.2.x Komponente<Name> (z.B. UserManager, ShoppingCard, etc)

-- Komponente und deren Hauptfunktionen sollten klar werden (Logische Sicht) -- Für die jeweiligen Komponenten ein Klassendiagramm oder grobes Klassendiagramm als Übersicht.

--Bitte bei verwendeten Mustern, diese beschreiben und deren Verwendung begründen, d.h. hier etwas mehr ins Detail gehen. li

3.2.1 GUI

3.2.2 Controller

Was wirdgemacht

3.2.2.1 Servlet

3.2.2.2 Actions

3.2.3. Manager

WIE wirdes gem8

Die Manager stellen die Verbindung zu DA-Schicht dar und greifen auf diese zu.

In den Managern werden Operationen und Zugriffe auf das Model über die DA-Sicht.

Weitere informationen siehe Javadoc

3.2.3.1 CommentManager

Der CommentManager halt Methodenbereitzu

3.2.3.2 DocumentManager

3.2.3.3 GlobalRolesManager

3.2.3.4 MainManager

3.2.3.5 ProjectManager

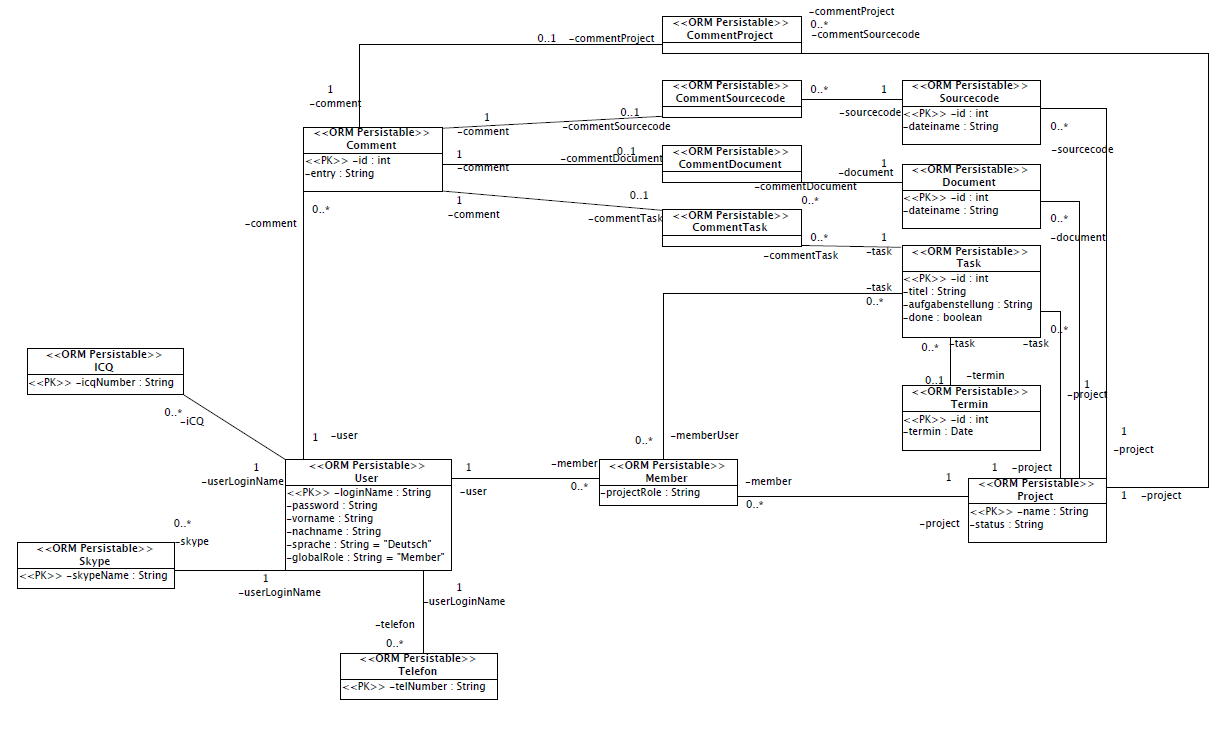
3.2.3.6 ProjectRolesManager

3.2.3.7 SourceManager

3.2.3.8 TaskManager

3.2.3.9 UserManager

3.2.4. Model





(3.2.4.1 Comment3.2.4.2 CommentDocument3.2.4.3 CommentProject3.2.4.4CommentSourcecode3.2.4.5 CommentTask3.2.4.6 Document3.2.4.7 GlobalRoles3.2.4.8 ICQ3.2.4.9 Member)

(3.2.4.10 Project3.2.4.11ProjectRoles3.2.4.12 Skype3.2.4.13Sourcecode3.2.4.14 Task3.2.4.15Telefon3.2.4.16Termin3.2.4.17 User)

3.2.5DA

3.2.6 DAO

**3.2.7 Datenbank**

In der Datenbankschicht wird MYSQL und Hibernate OR-Mapping genutzt.

Das Datenbankmodell auf das JProject aufgebaut ist im anhang A1 zu sehen.

4. Systemvoraussetzungen

*4.1 Systemspezifische Komponenten*

- welche Komponenten sind von der eigenen Software notwendig (die Komponenten liegen ür die Installation typischerweise als jar-/war-Dateien vor)

- externe Libraries, die von Ihrer Applikation benötigt werden

Verwendete Technologien:

Hibernate,

JavaScript mit Mootools,

JSTL,

HTML,

CSS,

(nochmal nachgucken und erweitern)

***4.2 Allgemeine Voraussetzungen***

- Browser:

GUI optimiert für Firefox 4 und 5.

- Ablaufumgebung (Java-Version)

1.5

- Frameworks

Hibernate

- Container

Glassfish/Tomcat

- DB

MySQL

- Betriebssystem

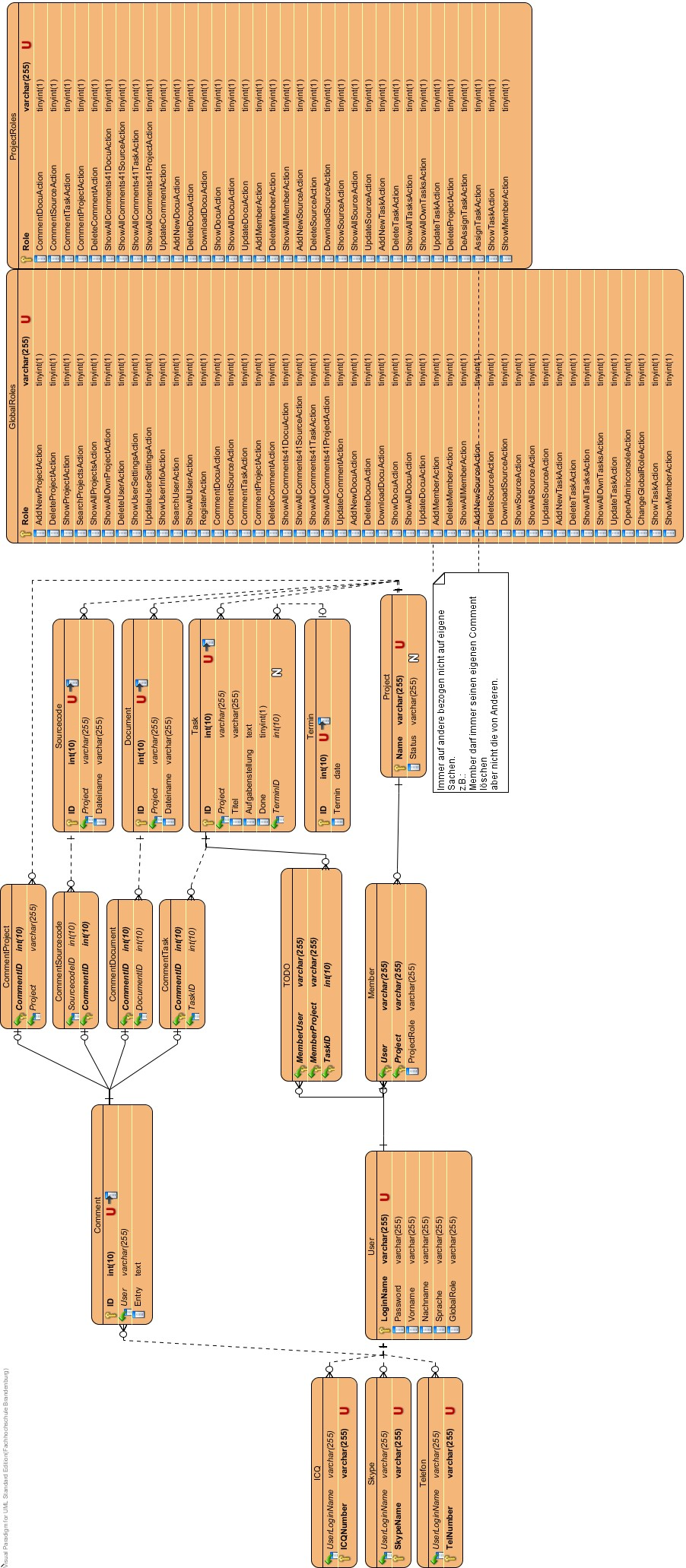
Plattformunabhängig

**5. Anhang**

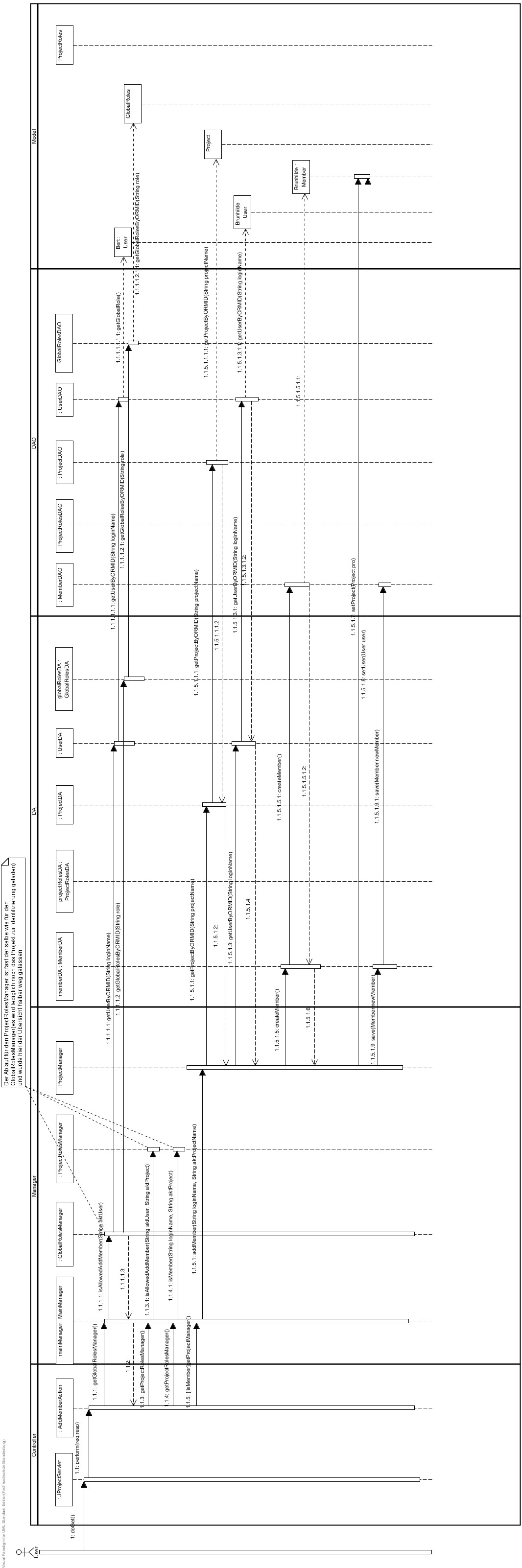
A1 Datenbank ERR-Diagramm

A2 Sequenz-Diagramm(AddMemberAction)

A3 Package-Diagramm

A1 Datenbank ERR-Diagramm

A2 Sequenz-Diagramm(AddMemberAction)



A3 Package-Diagramm

