Seminararbeit am

Institut für Wirtschaftsinformatik

Neue Medien

Elektronische Märkte von digitalen Gütern

Projekt JAPSTER

Studenten:	Bäck Christoph
	Gremmel Markus
	Moherndl Gernot
	Nentwich Thomas
	Seer Eugen
	Unger Robert
Betreuende Assistentin:	Dipl. Wirt.-Inform. Susanne Guth
Betreuender Professor:	Univ.-Prof. Dr. Gustaf Neumann
Semester:	WS 2000/01

INHALT

ABBILDUNGSVERZEICHNIS

1 Einleitung

Das Thema „Elektronische Märkte von digitalen Gütern“ wirft zunächst die Frage auf, wodurch sich digitale Güter von „normalen“ Gütern unterscheiden und wie diese über elektronische Märkte abgesetzt werden können.

Das Hauptmerkmal von digitalen Gütern ist, daß diese ausschließlich in digitaler Form genutzt und vertrieben werden können. Klassische Anwendungsbereiche sind hier beispielsweise Musik- und Filmdateien aber auch Zeitungsartikel in digitaler Form.

Nun wurden in der Vergangenheit diese digitalen Güter oft auf herkömmlichen „physikalischen“ Weg transportiert. Das beste Beispiel dafür sind CDs, auf denen die jeweiligen Dateien, Anwendungen, etc. zunächst gespeichert und danach vertrieben wurden.

Um diese teilweise unbefriedigende Situation (zum einem entstehen Vervielfältigungs- und Vertriebskosten, zum anderen Wartezeiten) zu entschärfen, werden immer öfter elektronische Märkte zum Vertrieb von digitalen Gütern eingesetzt. Dadurch werden die durch den physikalischen Vertrieb hervorgerufenen Wartezeiten beseitigt, die Vervielfältigungskosten gehen gegen Null, da kein physikalisches Medium wie CD oder Diskette dazu benötigt wird [Neum01].

Der elektronische Vertrieb von digitalen Gütern birgt jedoch auch eine Reihe von Problembereichen, die beim Vertrieb über das Internet zu beachten sind. Da diese Punkte jedoch nicht das Hauptthema dieser Arbeit sind, erfolgt nur eine stichworartige Aufzählung, die nicht den Anspruch auf Vollständigkeit erhebt:

Bezahlung:

Hier ist vor allem das Suchen und finden des geeigneten Zahlungsmittels gemeint. Nach wie vor bestehen große Ressentiments gegen die Bezahlung mit Kreditkarten, andere Zahlungsmittel haben sich bis dato ebenfalls noch nicht weltweit durchgesetzt.

Copyright:

Durch das zur Verfügung stellen von Files kann es bzw. kommt es sehr häufig zu Verletzungen des Urheberrechts. Selbst weniger interessierten Beobachtern ist wohl der „Fall Napster“ bekannt, wo es gerichtliche Klagen der Musikindustrie gegen Napster gab. Napster wurde beschuldigt, die Urheberrechte der Firmen durch kostenlose Weitergabe von Musikfiles im MP3-Format verletzt zu haben.

Abschluß von „digitalen Verträgen“

Das Problem besteht vor allem darin, welche Rechtsordnung im Streitfall angewendet wird, da es durch die weltweite Vernetzung sehr häufig zu Verträgen mit Auslandsberührung kommt.

Im nächsten Schritt dieser Arbeit wollen wir nun näher auf Distribution Channels oder Filesharing Applikationen näher eingehen, da dies die Kerntätigkeit der Seminargruppe bildete, die unter dem Titel „Distribution Channel Applications“ arbeitete.

1.1 Projekt Japster

Im Projekt Japster wird ein weitgehend plattformunabhängiger Distribution Channel entwickelt, der durch Metadatenunterstützung auch größtmöglicen Suchkomfort und –effizient bieten soll.

Die Verwendung von Java ermöglicht diese weitgehende Plattformunabhängigkeit, die jedoch Probleme in Hinblick auf den Zugriff von lokalen Ressourcen mit sich bringen wird.

Aus diesem Grund wird in einem kurzen Einschubkapitel die Funktionsweise von Java, vor allem in Hinblick auf die für Japster wichtigen Aspekte, erklärt.

1.2 Distribution Channels

Um im Internet digitale Güter transportieren zu können muss man sich sogenannter ‚Distribution Channels' bedienen. Eine Ausführung eines Distribution Channels ist File-Sharing. Beim File-Sharing tauschen eine große Anzahl an Usern ihre Daten Peer to Peer aus. Peer to Peer bedeutet, dass alle User gleichberechtigt nebeneinander das selbe File-Sharing-Tool verwenden. In den folgenden Kapiteln wird näher auf File-Sharing eingegangen.

1.2.1 Filesharing

Shawn Fanning, der Gründer von Napster, machte die Beobachtung, dass zum Download von Musikfiles (MP3-Format) eine Vielzahl von Homepages aufgerufen werden musste, von welchen Links auf die entsprechenden Files (Musikstücke) zeigten. Das heißt, diese Homepages mußten durch Suchmaschinen aufgefunden werden und erst durch den „Umweg" über die Homepages konnte das eigentliche Ziel, Download von Musikfiles, erreicht werden. Der Mehrwert von Napster besteht folglich darin, dass den Usern dieser Umweg erspart wird.

Durch Napster erfolgt die Suche, Download und das Abspielen durch eine integrierte, bedienerfreundliche Applikation. Potenzielle User von Napster sind sämtliche Personen, die über einen Computer mit Internetanbindung verfügen und regelmäßig Musikstücke anhören – also folglich, ein nicht unerheblicher Teil der gegenwärtig 407 Millionen Internetbenutzer [vgl. NUA01].

Der große Vorteil von Filesharing Applikationen besteht in der Verwendung des sogenannten Peer to Peer Modells. Peer to Peer ist ein Internet-Kommunikationsmodell, das einer Gruppe von gleichberechtigten Usern mit einer einheitlichen Netzwerk-Applikation die Möglichkeit bietet, direkt auf Festplatten anderer User zuzugreifen. Dadurch können Dateien untereinander ausgetauscht werden, ohne daß diese auf zentrale Server abgespeichert werden müssen.

1.2.2 Vergleich bestehender Distribution Channels

Aus der Vielzahl bestehender Filesharing Plattformen werden folgende drei Applikationen heruausgegriffen, genauer erklärt und die technischen Unterschiede herausgearbeitet:

- Napster
- Gnutella
- Freenet

1.2.2.1 Napster

Bei dieser Tauschbörse für Musikstücke im MP3-Format handelt es sich sicherlich um jene P2P-Plattformen, welche dieses Kommunikationsmodell einer breiten Öffentlichkeit erstmals ins Bewusstsein geführt hat. Napster wurde 1999 von Shawn Fanning begründet und hat inzwischen ca. 40 Millionen registrierte Benutzer [vgl. PTA01]. Die tauschbaren Inhalte beschränken sich auf Musikfiles im MP3-Format.

Napster beruht auf einer Server-Client Architektur. Der Server übernimmt dabei nur die Verwaltung der Benutzer und der dezentral vorhandenen Musikstücke. Das heißt dem Server sind die verfügbaren Musikfiles sämtlicher eingeloggter Benutzer „bekannt“ und diese werden auf Anfrage (Suchanfrage) anderen Benutzern mitgeteilt. Der Austausch der Files erfolgt dann direkt zwischen den Clients, was bedeutet, dass die Client-Software in diesem Fall auch als Server arbeitet (er bietet die Datei zum Download an). Abbildung 1 visualisiert dazu den Ablauf des Dateitransfers. Client A sendet bei seiner Anmeldung an das System den Usernamen und die IP-Adresse an den Server. Ebenfalls an den Server sendet Client B eine Suchabfrage einer gewünschten Datei. Als Antwort erhält er das Suchergebnis, versehen mit der IP-Adresse jenes Users, der die Datei auf seiner Festplatte abgespeichert hat. Der Dateitransfer erfogt direkt zwischen Client A und Client B.

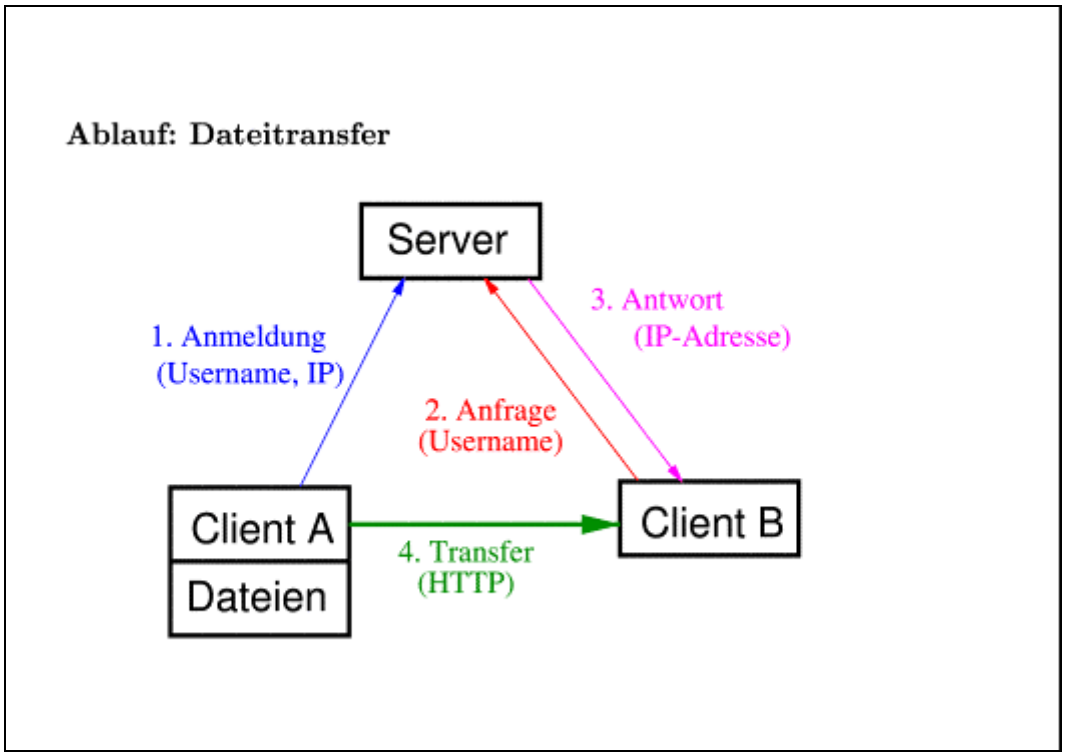


Abbildung 1: Software-Konzept von Napster [Schuh01]

Um das Tauschservice nutzen zu können, muss der potenzielle User die Applikation herunterladen und auf seinen Rechner installieren. Danach kann er/sie wählen, welche Ordner, Festplatten, etc. er mit anderen Benützern „teilen“ möchte. Neben der Funktionalität des Filesharing, bietet Napster noch eine Reihe sekundärer Services, z.B. Chat, MP3-Player.

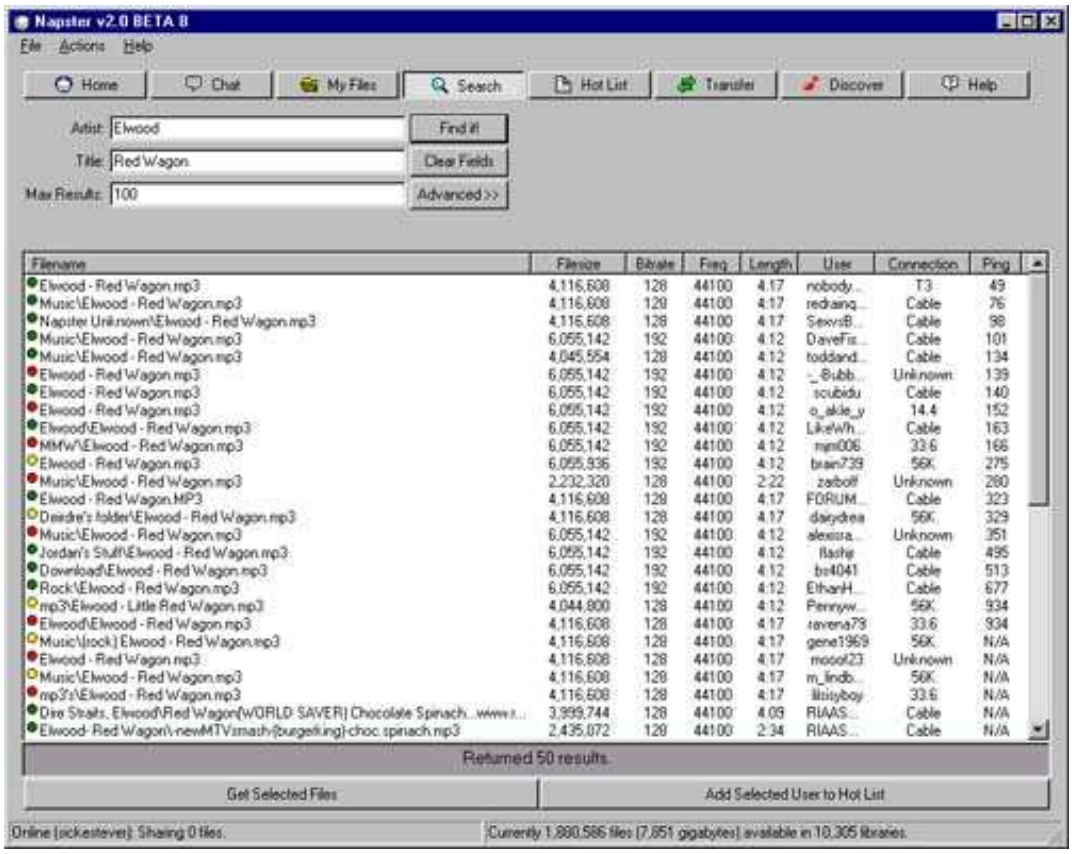


Abbildung 2: Screenshot von Napster

Zusammenfassend lässt sich Napster durch folgende Vor- und Nachteile gegenüber anderen Filesharing Applikationen charakterisieren:

Vorteile:

- Aufgrund der großen Anzahl an Usern bereits vielfältiges Anbot an Musiktiteln
- Benutzerfreundliche Oberfläche

Nachteile:

- Beschränkung auf Musikfiles im MP3-Format
- Um zu verhindern, dass der zentraler Server einen potenziellen Flaschenhals darstellt, sind bereits heute mehrere Server im Einsatz, die allerdings

wieder einer Synchronisation bedürfen, um den Usern alle Files zugänglich zu machen

- Download einer plattformabhängigen Client-Applikation erforderlich
- Verwendung eines proprietären Protokolls
- Sicherheit (Passwörter werden als Text übertragen)
- Da bei Napster lediglich Musikdateien im MP3-Format übertragen werden, ist eine Metadatenunterstützung nicht zwingend erforderlich. Dadurch wird jedoch eine mögliche Erweiterung auf andere Dateiformate erschwert, da hier der Dateiname allein nicht mehr als alleiniges Identifikationsmittel zur Verfügung steht.

1.2.2.2 Gnutella

Der Aufbau von Gnutella unterscheidet sich von Napster dahingehend, dass er auf einer vollkommen dezentralen Architektur beruht. Wie bei Napster muss der Client ein Applikation (plattformabhängig) herunterladen und auf seinem Rechner installieren. Im Gegensatz zu Napster, ist der Austausch nicht auf Musikfiles beschränkt, sondern jedes Dateiformat ist erlaubt.

Als weiteren Gegensatz zu Napster wird auch kein zentraler Server verwendet. Die Kommunikation zwischen den Computer erfolgt direkt, d.h. ein Computer baut zu einer Reihe „benachbarter“ Computer virtuelle Verbindungen auf, über die in Folge die Suchabfragen sowie die Dateitransfers laufen.

Schickt ein Benutzer eine Anfrage aus, so ergeht diese an die „verlinkten“ Benutzer. Diese sondieren die Files im eigenen Bestand auf Übereinstimmung und leiten die Anfrage wiederum an verlinkte Benutzer weiter – „Schneeballeffekt“. Wird das gesuchte File gefunden, läuft die Kommunikation in umgekehrter Reihenfolge zurück.

Abbildung 3 soll die Architektur darstellen, wobei die hier dargestellte Suchabfrage von Punkt C abgesetzt wird und auf Punkt G die gesuchte Datei gefunden und transferiert wird.

Anders als bei Napster, ist die Funktionsfähigkeit des Systems nicht von der Verfügbarkeit eines oder mehrerer zentralen Rechners (Server) abhängig, jedoch wird die Netzbelastung aufgrund des vielfachen Versendens und Weiterleiten von Suchabfragen und Antworten (so schickt z.B. der User C seine Suchabfrage gleich drei mal ab) um ein vielfaches im erhöht.

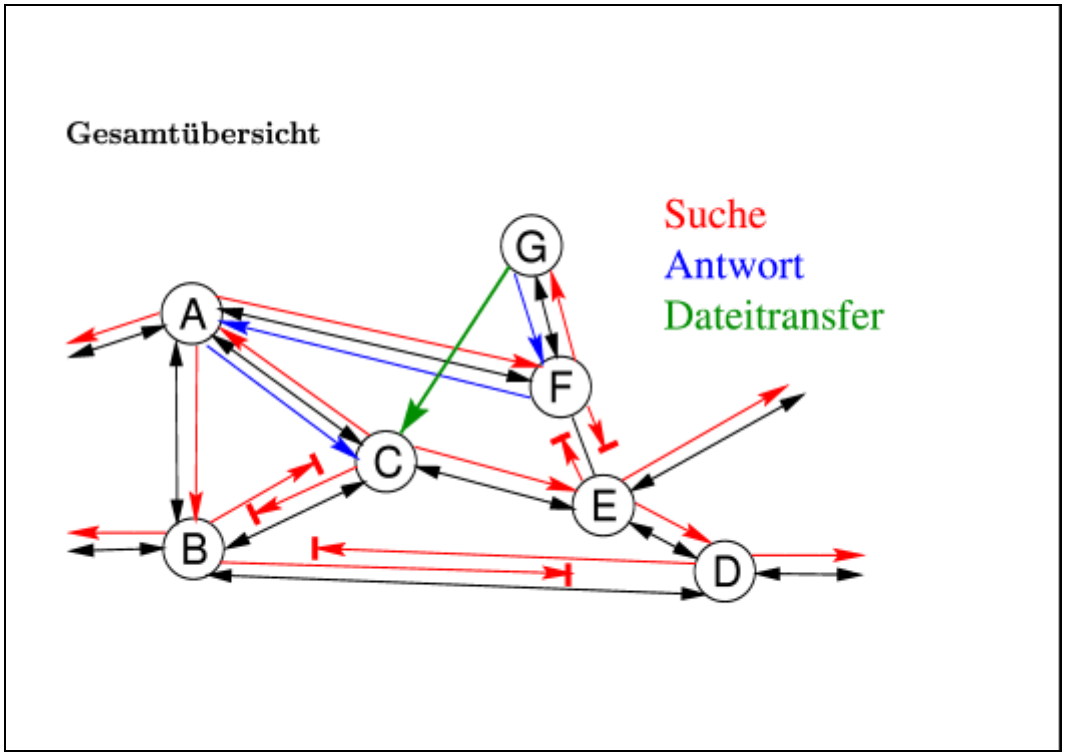


Abbildung 3: Software-Konzept von Gnutella [Schuh01]

Im Detail besitzt Gnutella folgende Vor- und Nachteile:

Vorteile:

- Vollständig dezentrale Struktur (Ausfallsicherheit)
- HTTP-basiertes Protokoll

- Keine Beschränkung auf Dateiformate

Nachteile:

- Erhöhte Netzbelastung aufgrund des vielfachen Aussendens und Weiterleiten der Abfragen und Antworten
- Suche immer im gleichen „Bereich“, da die Anzahl „Nachbarn“ sowie die Anzahl der Ebenen zum Weiterleiten der Abfrage beschränkt sind, um die bereits erhöhte Netzbelastung leicht einzuschränken.
- Keine Metadatenunterstützung, daher keine zusätzlichen Informationen über die Dateien erhältlich
- Download einer plattformabhängigen Client-Applikation erforderlich

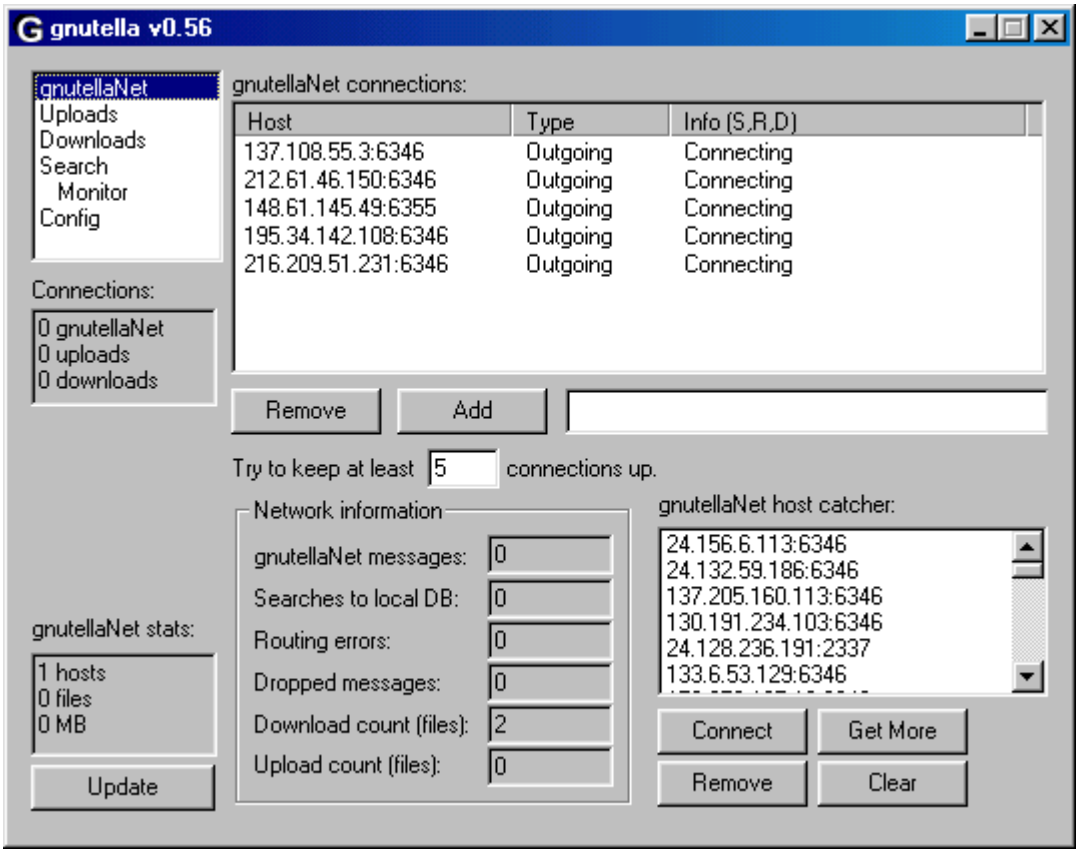


Abbildung 4: Screenshot von Gnutella

1.2.2.3 Freenet

Freenet folgt zwar dem Aufbau von Gnutella (dezentral), jedoch ist es gegenüber den anderen Beispielen ein Ausnahmefall, da die Zielsetzung dieses Systems eine andere ist. An erster Stelle steht in diesem System die Anonymität sämtlicher Teilnehmer, und die Unmöglichkeit von Zensierung von Inhalten. Stellt ein Teilnehmer Files zur Verfügung, so werden diese auf den Rechnern der User verteilt. Dadurch kann im Gegensatz zum WWW angebotener Inhalt nicht durch das „Ausschalten“ eines Servers vom Netz genommen werden.

Da es beinahe unmöglich ist, Inhalte zu lokalisieren und zu löschen, ist eine Zensurierung in diesem System unmöglich. Weiters garantiert Freenet auch die Anonymität sämtlicher Benutzer – Autor, Nachfrager, Anbieter. Das Vorgehen im Fall einer Anfrage ist ähnlich wie bei Gnutella, jedoch wird eine Anfrage nur an jeweils einen „verlinkten“ Benutzer weitergeleiten (zeitlich hintereinander), wodurch der bereits beschriebene Schneeballeffekt nicht eintritt. Freenet beruht derzeit ausschließlich auf Java und zum Betreiben des Clients ist ein Java Runtime Environment erforderlich.

Graphisch zusammengefasst lässt sich daher das Software-Konzept wie in Abbildung 5 darstellen, in der zu sehen ist, daß die Suchabfrage zunächst immer nur an einen benachbarten User abgesendet wird.

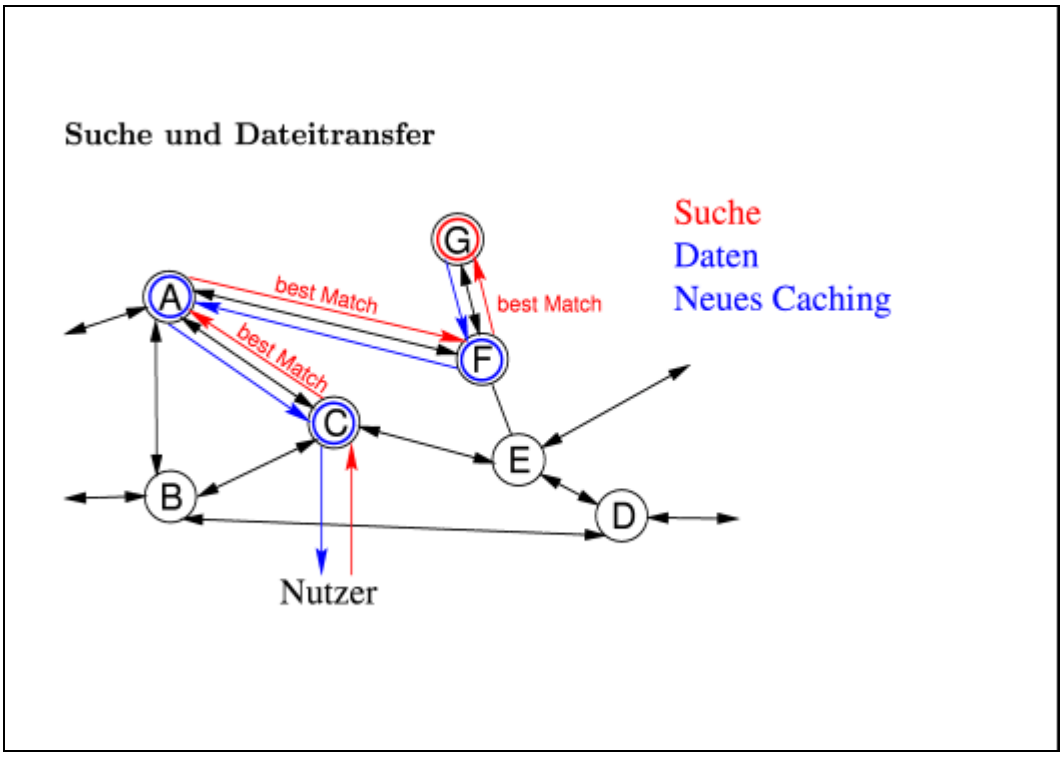


Abbildung 5: Software Konzept von Freenet [Schuh01]

Die Vor- und Nachteile von Freenet ähneln aufgrund der sehr ähnlichen Architektur natürlich Gnutella und lassen sich wie folgt zusammenfassen:

Vorteile:

- Vollständig dezentrale Struktur (Ausfallsicherheit)
- Keine Beschränkung auf Dateiformate
- Vollständige Anonymität der Teilnehmer

Nachteile:

- Suche immer im gleichen „Bereich“, da Anzahl der „Nachbarn“ und die der maximalen Weiterleitungen (Linktiefe) beschränkt.

- Keine Metadatenunterstützung, d.h. Dateinamen dient alleine zum Identifizieren der Dateien.
- Längere Suche als bei Gnutella, da sequentielle Anfrage bei anderen Teilnehmern

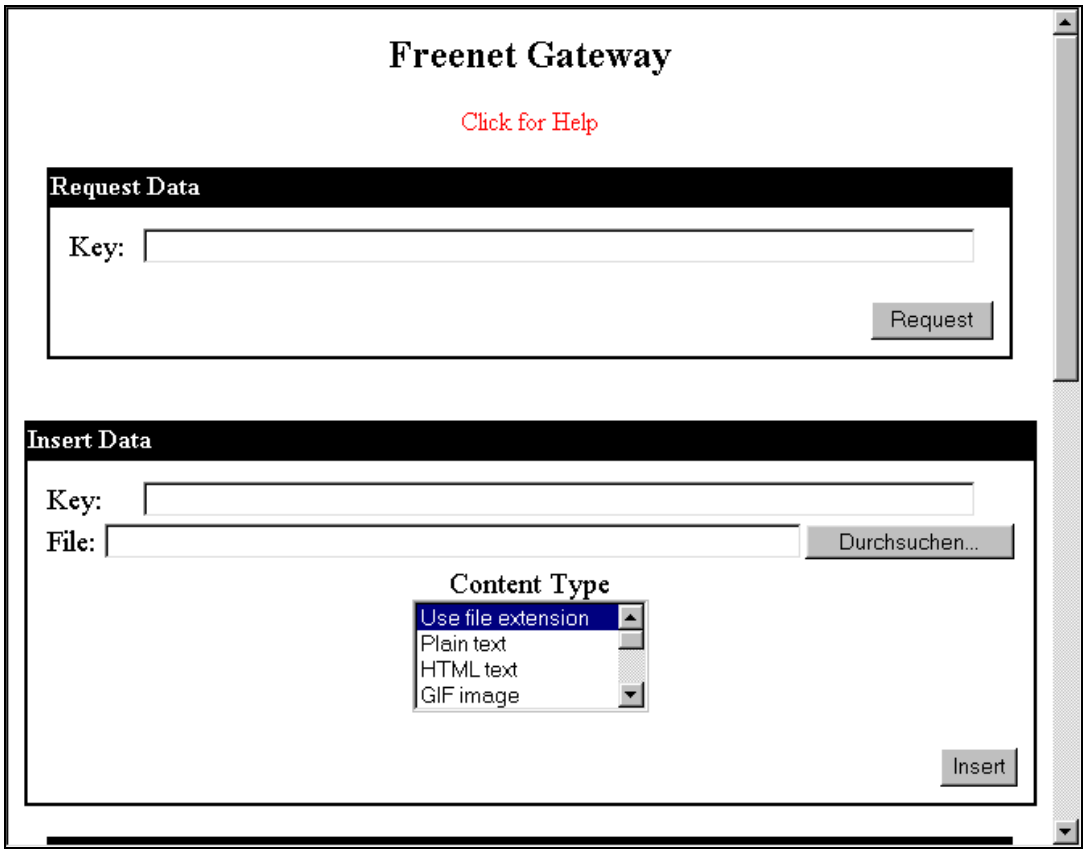


Abbildung 6: Screenshot von Freenet

1.3 Einschub: Kurzeinführung in Java

Da Japster mit Hilfe der Programmiersprache Java erstellt wird, soll an dieser Stelle das Grundkonzept von Java aber auch jene Details, die für Japster von Bedeutung sind näher erklärt werden.

1.3.1 Java Grundbegriffe

Java ist eine Universal-Programmiersprache, was bedeutet, daß man sie prinzipiell für jede Art von Anwendung einsetzen kann. Java wird jedoch nicht nur als Programmiersprache, sondern auch als Plattform bezeichnet [Hawl00].

Dem Programmierer werden eine Reihe von fertigen Klassenbibliotheken als API (Application Programmers Interface, engl. für Programmierschnittstelle) angeboten, die durch die sogenannte Laufzeitumgebung (die virtuelle Maschine) auf die reale Plattform (z.B. Betriebssystem Windows) umgesetzt.

Die Java Laufzeitumgebung besteht im wesentlichen aus der virtuellen Maschine, die der Zielplattform angepaßt ist. Durch diese virtuelle Maschine, die das Bindeglied zwischen dem Java-Code und dem Betriebssystem darstellt, wird die weitgehende Plattformunabhängigkeit erreicht. Der Java-Code bleibt unverändert, lediglich die virtuelle Maschine (Java VM) wird angepaßt. Der Java Quellcode wird vom Java Compiler in maschinenunabhängigen Bytecode für die Java VMumgewandelt. Das Laufzeitsystem (engl: Java Runtime Environment, kurz JRE) besteht aus dem Emulator und einer Reihe von Klassenbibliotheken und führt den Bytecode aus.

Java Applets sind spezielle Klassen, die in einem Java fähigen Webbrowser (z.B. Microsoft Internet Explorer oder Netscape Navigator) ablaufen und damit die Funktionalität des Browsers erhöhen können. Die Applets werden meist auf einem Web-Server zur Verfügung gestellt und durch einen spezielle HTML Befehl abgerufen. Sie unterliegen besonderen Sicherheitsrestriktionen, die verhindern sollen, daß der soeben geladene Code das eigene System ausspähen oder zerstören kann. Durch Java PlugIns besteht die Möglichkeit, eine akutelle Version der JRE auf seinem System zu installieren.

Java Servlets sind Klassen, die beispielsweise dynamisch Web-Seiten generieren. Der Unterschied zu den Applets ist, daß die Servlets am Server ausgeführt werden.

1.3.2 Sicherheitskonzept für Java Applets

Das sogenannte Sandbox-Modell der Java-Plattform garantiert für die risikolose Ausführung von heruntergeladenen bzw. nicht vertrauenswürdigen Programmen. Durch das Sandbox-Modell ist es Java-Programmen unmöglich direkt auf Computer-Hardware oder auf das Betriebssystem zuzugreifen [Guth00].

Erst ein Java Applet, daß digital signiert ist, wird einer lokalen Applikation gleichgesetzt und erhält damit uneingeschränkten Zugriff auf lokale Ressourcen. Nicht signierte Java Applets können weiterhin nur in der Sandbox laufen. Das digitale Signieren von Java Applets ist ab der Version 1.1 der Java Plattform möglich.

Durch digitales Signieren soll sichergestellt werden, daß der Java Code mit zusätzlichen Berechtigungen wirklich aus einer vertrauenswürdigen Quelle stammt. Zum Nachweis werden dazu alle notwendigen Klassen und Ressourcen in eine Java Archiv Datei (JAR) verpackt, die digital signiert wird. Um eine digitale Unterschrift leisten zu können, muß sich der Signierer bei einer autorisierten Zertifizierungsbehörde registrieren lassen, die seine Identität bestätigt. Das bedeutet, dem Signierer wird ein Zertifikat ausgestellt.

In der Folgeversion von Java (Version 1.2) durchläuft jede Java Anwendung die gleichen Sicherheitskontrollen, jedoch kann für lokale Applikationen auch eine Verfahrensanweisung (engl.: Policy) definiert werden. Dem Applet werden nur differenzierte Rechte (engl. Permissions) vergeben, die nur bestimmte Zugriffe erlauben. Dem Entwickler stehen vordefinierte Rechte zur Verfügung, die jedoch erweitert werden können.

1.4 Sicherheitsaspekte von Distribution Channels

Filesharing Applikationen arbeiten nach dem Prinzip, daß lokale Ressourcen (z.B. Musikfiles) den anderen Usern dieser Applikation zum Download zur Verfügung gestellt werden (vgl. Kapitel 1.2).

Ein Charakteristikum dieser Anwendungen ist dabei ein hoher Anonymitätsgrad der User, die sich sehr leicht hinter kryptischen User-Namen „verstecken“ können und sich lediglich durch Ihre IP-Adresse o.ä. identifizieren.

Dieser Punkt kann jedoch eine potentiell sehr hohe Gefahr bieten, da man allen Usern des Distribution Channels die Möglichkeit bietet, auf seine lokalen Ressourcen zuzugreifen. Eine latente Gefahr, die man als User von Filesharing Applikationen stets beachten sollte.

1.5 Begriffserklärungen

In diesem abschließenden Teil des Einführungskapitels sollen einige der in dieser Arbeit verwendeten Ausdrücke näher erklärt werden, um dem Leser den größtmöglichen Nutzen zu bieten.

Metadaten

Metadaten sind Daten, die andere Daten beschreiben und klassifizieren. Als Beispiele können unter anderem hier Titel, Autor, Beschreibung, Datum oder die Dateigröße angeführt werden.

Metadaten einer Seminararbeit sind zum Beispiel Titel, Thema, Autor, Semster, etc.

Profile

Die Auswahl zwischen mehreren Profiles (oder User-Profiles) gibt dem Japster Benutzer die Möglichkeit, Japster mit verschiedenen Grundeinstellungen zu nutzen. Die Grundeinstellungen betreffen v.a. den Anbindungstyp (Modem, Permanentverbindung, etc.) und das „shared dir“.

Parser

Ein Parser bietet die Möglichkeit, den Inhalt einer XML-Datei[1] auszulesen.

[1] XML steht Extented Markup Language

2 Projektziel

Die Zielsetzung dieses Seminars ist die Entwicklung einer Applikation als Vertriebskanal für digitale Güter.

Durch die Beschränkung auf digitale Güter kann der gesamte „Vertriebsprozess“ – Informations-, Verhandlungs– und Abwicklungsphase [Merz99, 26] – über Neue Medien erfolgen. Eine geeignete „Vertriebsapplikation“ für digitale Güter muß daher sämtliche Phasen des Vertriebsprozesses unterstützen, wodurch sich in den einzelnen Bereichen folgende Anforderungen an eine solche Applikation ergeben:

Informationsphase:

Es muss die Möglichkeit bestehen, Informationen über die erhältlichen Produkte/Dienstleistungen, deren Verfügbarkeit und über Geschäftspartner einholen zu können.

Verhandlungsphase:

Die Applikation ermöglicht den Usern durch „Verhandlung“ zu einer Einigung zu kommen. In der einfachsten Form bedeutet dies, dass Standardanbote zu einem vorgegebenen Preis durch einen Käufer angenommen werden können.

Abwicklungsphase:

Die Applikation muss in der Lage sein, die ausgetauschten Güter/Dienstleistungen an den Tauschpartner zu liefern. Auf digitale Güter bezogen bedeutet dies, dass die Daten/Informationen an den Tauschpartner übertragen werden können.[2]

Weiters sollte die Vertriebsapplikation auf einem Medium basieren, das eine möglichst große Anzahl von potenziellen Kunden anspricht. Damit diese auch bereit sind die Applikation anzunehmen, muss sich eine solche durch möglichst große Benutzerfreundlichkeit auszeichnen bzw. die Benutzung sollte einen möglichst geringen Lernaufwand erfordern. Damit ergibt sich die Forderung nach der Verwendung von Standardtechnologien, wodurch auch die Umstellungskosten bei Benutzern gering gehalten werden können und diese eher bereit sind, die Applikation zu verwenden.[3]

Aufgrund der aufgezählten Anforderungen an die Vertriebsapplikation haben wir das Internet als Vertriebskanal gewählt, da wir dadurch einen potenziellen Markt von 407 Millionen Usern ansprechen (vgl. NUA01). Um auch die Forderung nach Benutzerfreundlichkeit und geringen Umstellungskosten zu erfüllen, bauen wir auf (internetbasierten) Standardtechnologien auf, z.B. Java.

2.1 Technologische Zielsetzung

Wir wollen die bisher allgemein gehaltenen Forderungen an Filesharing Applikationen nun in konkrete Anforderungen an eine Distribution Channel Applicati-

[2] Es wird hier bewusst der Terminus Tauschpartner und nicht Käufer verwendet, da der letztere Entgeltlichkeit der Transaktion impliziert.

on, in Form eines Filesharing-Tools, überführen, und so die Anforderungen an Japster formulieren.

Keine Beschränkung der Dateiformate

Im Gegensatz zum First-Mover und Marktführer Napster wollen wir mit unserer Plattform des Austausch jeglicher Art von Files unterstützen (vgl. Kapitel 1.2.2.1).

Metadatenunterstützung

Die bestehenden Systeme bieten als Metadaten nur den Dateinamen, die Dateigröße, den Namen des Anbieters und das Dateiformat an (Napster). Hinsichtlich des Inhaltes und sonstiger Eigenschaften der Dateien erfolgt keinerlei nähere Spezifikation (z.B. Inhalt, Autor, Beschreibung, ...), wodurch eine effiziente Suche nach bestimmten Informationen nur beschränkt möglich ist. Wir wollen unseren Benutzern auf freiwilliger Basis ermöglichen, nähere Informationen zu den Files anzugeben, wodurch eine effiziente Suche gewährleistet wird.

[3] Geringe Umstellungskosten bringen zwar den Vorteil, dass User eher bereit sind, auf ein neues Produkt umzusteigen, jedoch weisen diese auch eine geringere Loyalität auf.

Plattformunabhängigkeit

Napster und Gnutella erfordern den Download einer plattformabhängigen Client-Applikation, wodurch verschiedene Versionen für sämtliche Betriebssysteme zur Verfügung stehen müssen. Dies erhöht die Komplexität und den Wartungsaufwand des Systems, weshalb wir auf weitgehend plattformunabhängige Technologien setzen werden.

Kein Installationsaufwand

Um potenzielle User nicht abzuschrecken, muss der erforderliche Aufwand eines Users bei der erstmaligen Benutzung unserer Plattform möglichst gering gehalten werden. Wir versuchen daher, durch den Einsatz von Java-basierten Technologien unseren Benutzern die Installation einer Client-Software zu ersparen. Dadurch wollen wir auch den Wartungsaufwand eines Users minimieren, da dieser nicht regelmäßig seine installierte Software updaten muss.

Abschließend möchten wir unser Projektziel in einem Missionstatement zusammenfassen:

Das auf Java basierende Japster ermöglicht weitgehend

plattform- und dateiformatunabhängiges Filesharing,

das durch Metadatenunterstützung

größte Sucheffizienz und -komfort bietet.

2.2 Services

Im Gegensatz zu anderen Systemen wollen wir unseren Usern erhöhten Suchkomfort durch die Metadatenunterstützung unseres Systems bieten, und es gibt keine Beschränkung hinsichtlich der tauschbaren Dateiformate.

Des weiteren wollen wir die Benutzer zur Registrierung bewegen. Dies ist insbesondere im Hinblick auf ein späteres Geschäftsmodell zur Erzielung von Einkünften erforderlich. Registrierte Benutzer haben die Möglichkeit Profile anzulegen. In einem solchen Profil werden die Daten eines Users hinterlegt, d.s. Profilname samt zugehörigem Arbeitsverzeichnis („shared dir“), Art der Internetverbindung, Anzahl der erfolgreichen Up- und Downloads. Dadurch reduziert sich der „Log-In“ Aufwand, da der User nicht bei jedem Log-In ein Arbeitsverzeichnis („shared dir“) freigegeben muss, sondern dieses im Profil hinterlegt ist. Weiters erfolgt die Bewertung der registrierten Benutzer aufgrund automatisch errechneter Kennzahlen. Beim Prototyp werden wir eine Kennzahl mit der Bezeichnung ASR angeben, welche Auskunft über die Verlässlichkeit eines Benutzers gibt. Diese errechnet sich als Verhältnis der abgebrochenen Uploads zu der Gesamtzahl der gestarteten Uploads (erfolgreich und abgebrochen).[4] Diese Kennzahlen öffnen im Rahmen eines späteren Geschäftsmodells die Möglichkeit eines Belohnungssystems für aktive User. Mögliche zusätzliche Services, die zu einem späteren Zeitpunkt implementiert werden können, wären ein Online-Chat, eine Abrechungsfunktion und Benutzergruppen.

[4] Upload bezeichnet einen Transfer, bei dem der betrachtete Client an einen fremden Client Daten überträgt.

Zusammenfassend möchten wir hier eine kurze Übersicht der Services von Japster darstellen, eine genauere Erklärung folgt in Kapitel xxx anschließend:

Basisdienste:

- Filesharing
- Benutzerbewertung
- Benutzerprofile für registrierte Benutzer

Zusatzdienste (zur späteren Implementierung):

- Online-Chat
- Benutzergruppen
- Abrechnungs- bzw. Belohnungssystem

2.3 Anwendungsfall

In den Schulungsräumen der Wirtschaftsuniversität Wien gibt es Hunderte Desktops, welche den Studenten zur freien Benutzung zur Verfügung stehen. In diesen Computerräumen gibt es keine spezifischen Benutzerprofile mit einem virtuellen Verzeichnis. Das heißt, ein Student loggt ein und erhält einen Standard-Desktop ohne spezifische Einstellungen und ohne ein eigenes, exklusives Verzeichnis für seine Daten. Da ein Student (beinahe) niemals auf dem gleichen Rechner arbeitet, wird für temporäre Benutzerfiles das Verzeichnis D:/Daten freigegeben, welches jeden Abend gelöscht wird. Aufgrund der Vielzahl der User sammelt sich folglich während des Tages eine Vielzahl von Files in diesen Ordnern an, welche auch für andere Benutzer von Interesse sind.

Durch die Verwendung unserer Plattformen in den Schulungsräumen könnten sämtliche User auf alle Verzeichnisse D:/Daten zugreifen, sofern die jeweiligen User eines Rechners an der Plattform angemeldet sind (Applet läuft) und das Verzeichnis freigegeben haben.

Weiters besteht auch die Möglichkeit, daß sich Studenten von ihren Heimarbeitsplätzen bei Japster anmelden und somit ebenfalls Zugriff auf sämtliche lokalen Ressourcen der WU-Rechner haben.

Abschließend möchten wir nochmals mit Nachdruck erwähnen, dass obiges Beispiel nur eine von jenen vielen Möglichkeiten darstellt, wo unsere Plattform Mehrwert schaffen kann. Da zur Benützung unserer Plattform nur der Aufruf der entsprechenden Homepage erforderlich ist, kann das Filesharing von jedem Computer mit Internetzugang und Webbrowser mit Java 2 Plug-In erfolgen. Im Bezug auf obiges Beispiel bedeutet dies, dass jeder eingeloggte Benutzer auf sämtliche freigegebenen Ressourcen zugreifen kann.

3 Software - Konzept

In diesem Kapitel wird auf die grundlegende Architektur bzw. Topologie von Japster eingegangen, d.h. man soll erkennen, wie die definierten Projektziele umgesetzt werden sollen.

3.1 Architektur

Bei der Auswahl des technologischen Design diente die Architektur von Napster als Vorbild, dessen Kernkonzept eine zentralisierte Serverstrukur mit lokal gespeicherten Meta - Daten darstellt.

Konzept von Japster ist es, vollkommen auf Java basiert zu sein, wodurch Client – seitig durch den Einsatz von Java Applets keine Installation von weiteren Softwarekomponenten notwendig ist, außer einem Browser mit Java2-Plug-In. Dies hat den Vorteil, daß User von Japster auf ein regelmäßiges Update der Client - Software verzichten können, da die aktuelle Version bei jedem Start von Japster vom Server bezogen wird.

Einen Überblick über die Architektur von Japster liefert Abbildung 7. Aus dieser Darstellung ist zu erkennen, daß jede Transaktion – bis auf den tatsächlichen Download – über den Server geführt werden muß.

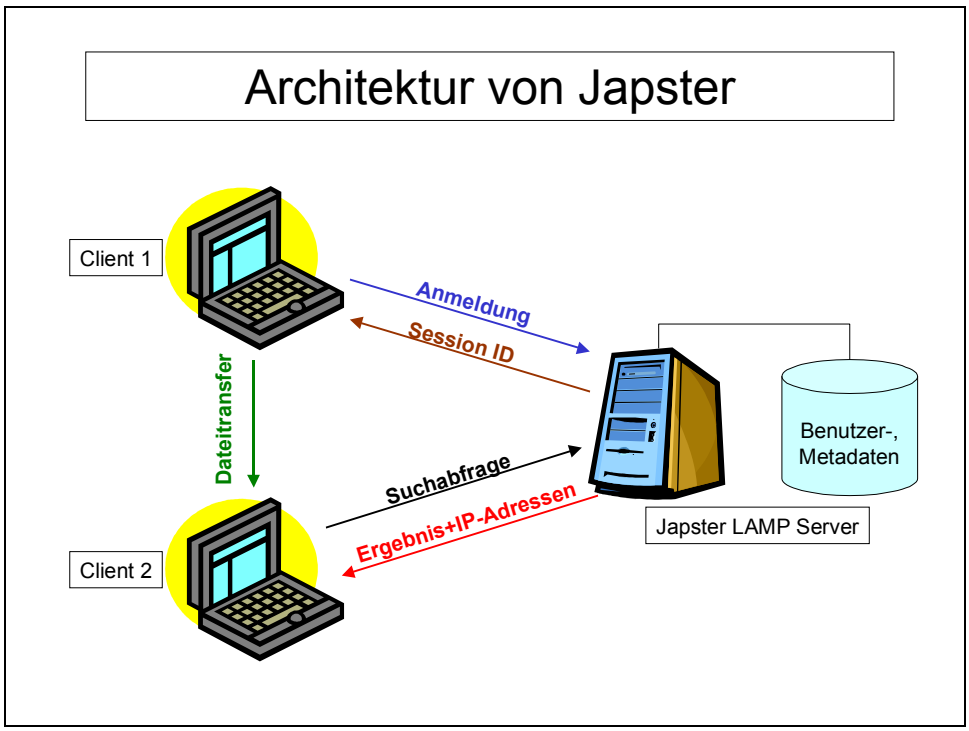


Abbildung 7: Architektur von Japster

An die technologische Zielsetzung (vgl. Kap. 2.1) anknüpfend, soll im folgenden versucht werden die Entscheidung für die gewählte Architektur zu begründen.

Metadatenunterstützung

Mit Japster sollte ein File-Sharing-Tool konstruiert werden, das die genauere Beschreibung der freigegebenen Dateien mittels Metadatenunterstützung ermöglicht. Es war notwendig eine Architektur zu wählen bei der sämtliche Metadaten aller freigegebenen Dateien effizient durchsuchbar gemachten werden. Als Lösung für diese Zielsetzung wurde entschieden einen zentralen Server mit einer MySQL – Datenbank zu verwenden. Startet ein User Japster, werden alle lokal vorhandenen Metadaten des freigegebenen Verzeichnis auf den Server übertragen und dort in die Datenbank abgelegt. Damit kann bei einer Suchabfrage eines Clients durch eine Datenbankabfrage relativ rasch das Suchergebnis geliefert werden.

Plattformunabhängigkeit

Um dem Ziel der Plattformunabhängigkeit möglichst nahe zu kommen wurde für die Ausführung des Clients ein Java – Applet gewählt. Um dieses auf dem Client lauffähig zu machen, reicht es aus, das dieser einen Browser mit einem Java2 – Plugin installiert hat und eine Reihe von Sicherheitseinstellungen vornimmt (vgl. Kap.4.1.3).

Kein bzw. geringer Installationsaufwand

Wurden einmal die erforderlichen Sicherheitseinstellungen getroffen, muß keine weitere Software installiert werden, da bei jedem Start von Japster das relativ kleine Applet vom Server geladen wird. Updates des Clients müssen nicht extra installiert werden, sondern werden automatisch beim nächsten Start von Japster wirksam. Würde man eine Architektur wählen bei der auf einen zentralen Server verzichtet wird, könnte es unter Umständen zu einem Versionskonflikt zwischen den einzelnen Clients kommen. Der User besitzt ja nur dann die neueste Version des Clients, wenn er diese installiert.

Ein weiterer Grund für die Wahl einer Architektur mit einem zentralem Server ist darin zu sehen, dass bei einer dezentralen Architektur sämtliche Informationsflüsse von einem Client zum nächsten weitergeleitet werden müssen und dadurch erstens diese länger dauern und zweitens eine höhere Netzbelastung zu erwarten ist. Bei Japster findet der Informationsfluß immer zwischen Client und Server statt. Nur der tatsächliche Download von Dateien erfolgt zwischen den einzelnen Clients.

Einhellige Meinung der Programmierer war es auch, dass es weniger kompliziert wäre eine System mit einem zentralen Server und lokalen Clients zu implementieren, als ein System ohne zentralem Server.

3.2 Metadaten

Wie schon aus dem Missionstatement des Projektes ersichtlich ist (vgl. Kap. 2.1), soll eine wesentliche Eigenschaft des Programmes, die Klassifizierbarkeit der Daten durch eine feste Anzahl von Metadaten – Tags, sogenannten „Qualifiern“, sein.

Dabei orientierten wir uns fast gänzlich an den Standard der „Dublin Core Metadata Initiative“ [Dublin]. Bereits 1995 wurden 13 formalbibliographische und inhaltliche Elemente definiert, die gezielt zur Beschreibung digitaler und digitalisierter Gegenstände angewandt werden können.

Zwischen Oktober und Anfang Dezember 1996 wurden 2 weitere Elemente hinzugefügt, womit nun diese 15 Qualifier den Kernsatz bilden (in Klammern stehen die englischen Begriffe der Elemente):

1. Titel (DC.Title)
2. Verfasser oder Urheber (DC.Creator)
3. Thema und Stichwörter (DC.Subject)
4. Inhaltliche Beschreibung (DC.Description)
5. Verleger bzw. Herausgeber (DC.Publisher)
6. Weitere beteiligte Personen und Körperschaften (DC.Contributors)
7. Datum (DC.Date)
8. Ressourcenart (DC.Type)
9. Format (DC.Format)
10. Ressourcen-Identifikation (DC.Identifier)
11. Quelle (DC.Source)
12. Sprache (DC.Language)
13. Beziehung zu anderen Ressourcen (DC.Relation)
14. Räumliche und zeitliche Maßangaben (DC.Coverage)
15. Rechtliche Bedingungen (DC.Rights)

Bei Japster werden die folgenden sechs Elemente aus dem Kernsatz der Dublin Core Metadata Initiative verwendet: *Title, Creator, Subject, Description, Format, Date*

Die Beschränkung ist damit zu begründen, dass bei Verwendung von allen Qualifiern der Programmieraufwand höher gewesen wäre und die Darstellung von 15 Qualifiern als zu unübersichtlich empfunden werden könnten. Mit der getroffenen Auswahl scheint eine weitreichende Genauigkeit der Beschreibung der Daten gegeben zu sein. Vor allem die Qualifier Format und Description erhöhen die Qualität von Suchergebnissen erheblich.

An dieser Stelle folgt nun eine genauere Beschreibung der 6 ausgewählten Qualifier nach Dublin Core [BIBDIENST]:

Titel (DC.TITLE)
Titel der Quelle; der vom Verfasser, Urheber oder Verleger vergebene Name der Ressource.

Verfasser oder Urheber (DC.CREATOR)
Die Person(en) oder Organisation(en), die den intellektuellen Inhalt verantworten. Z. B. Autoren bei Textdokumenten; Künstler, Photographen bzw. auch andere Bezeichnungen wie Komponist und Maler bei graphischen Dokumenten.

Thema und Stichwörter (DC.SUBJECT)
Thema, Schlagwort, Stichwort. Das Thema der Ressource bzw. Stichwörter oder Phrasen, die das Thema oder den Inhalt beschreiben. Die beabsichtigte Spezifizierung dieses Elements dient der Entwicklung eines kontrollierten Vokabulars. Das Element kann sowohl systematische Daten nach einer Klassifikation (scheme) wie Library of Congress Klassifikations-Nummer oder UDC-Nummer oder Begriffe aus anerkannten Thesauri (wie ***ME****dical* ***S****ubject* ***H****ea-*

dings (MESH) und ***A**rt and **A**rchitecture **T**hesaurus (AAT)* Deskriptoren) enthalten.

Inhaltliche Beschreibung (DC.DESCRIPTION)

Eine textliche Beschreibung des Ressourceninhalts inklusive eines Referats (Abstract) bei dokumentähnlichen Ressourcen oder Inhaltsbeschreibungen bei graphischen Ressourcen. Künftige Metadata-Sammlungen können auch numerische Inhaltsbeschreibungen (z. B. Spektralanalyse einer graphischen Ressource) enthalten, die eventuell noch nicht in bestehende Netzsysteme eingebettet werden können. In solchen Fällen kann dieses Feld einen Link zu einer solchen Beschreibung enthalten statt der Beschreibung selbst.

Datum (DC.DATE)

Das Datum, an dem die Ressource in der gegenwärtigen Form zugänglich gemacht wurde. Der empfohlene Eintrag des Datums wäre eine achtstellige Zahl JJJJMMTT, wie es in ANSI X3.30-1985 definiert ist, beispielsweise: 19961213 (=13. Dezember 1996). Andere Darstellungsschemata für das Datum sind erlaubt; werden sie angewandt, so sollten sie eindeutig identifiziert werden, damit keine Fehlinterpretationen auftreten.
N.B.: Beim 4. Dublin Core Workshop (DC4) wurde entschieden, dieses Element etwas offener, jedoch noch stark restriktiv zu definieren, und zwar in einer Unterteilung von zumindest DC.DATE.CREATION (Datum der Herstellung bzw. erstmaligen Veröffentlichung im Netz) und DC.DATE.LASTMODIFIED (Datum der letzten Änderung). Eine DC4-Arbeitsgruppe prüft z. Z., ob weitere Unterteilungen zu empfehlen sind.

Format (DC.FORMAT)

Hier wird das datentechnische Format der Ressource eingetragen, z. B. Text/HTML, ASCII, Postscript-Datei, ausführbare Anwendung, JPEG-Bilddatei etc. Die Angabe in diesem Feld gibt die erforderlichen Informationen, die Menschen oder Maschinen ermöglichen, über die Verarbeitungsmöglichkeiten der

kodierten Daten zu entscheiden (z. B. welche Hard- und Software benötigt werden, um diese Ressource anzuzeigen bzw. auszuführen). Ähnlich wie bei der Ressourcenart (TYPE) soll die Angabe in diesem Feld (FORMAT) aus festdefinierten Listen, wie die der angemeldeten Internet Media Types (MIME Types) entnommen werden. Grundsätzlich können Formate auch physische Medieneinheiten wie Bücher, Zeitschriften oder andere nicht elektronische Medien mit einschließen.

Im Rahmen der Überlegungen bei der Konzepterstellung wurde auch noch ein anderer Ansatz bzgl. der Metadaten diskutiert. Es wäre auch möglich die Qualifier frei definierbar (im Vergleich zu fix definierten Qualifiern wie bei Japster) zu gestalten. Dadurch wäre eine völlige Flexibilität bei der Beschreibung der Daten gewährleistet. Jedoch wurde beschlossen, im ersten Schritt zuerst eine funktionierende Basisversion von Japster zu schaffen und die frei definierbaren Qualifier erst in einem möglichen zweiten Schritt gemeinsam mit anderen Erweiterungen zu implementieren.

Eine weitere Begründung, warum eine fixe Anzahl von Qualifiern verwendet wird, liegt in der Verwendung einer starren relationalen Datenbank. Jeder der Qualifier ist ein Attribut innerhalb der Datenbank und als solcher nicht zu verändern. Um flexible Qualifier verwenden zu können, wäre es beispielsweise notwendig, die Metadaten in der Form von XML-Files am Japster Server abzulegen.

Da die Eingabe von Metadaten natürlich auf freiwilliger Basis geschieht, kann es vorkommen, dass nicht jeder sofort wenn er Dateien zur Verfügung gestellt, alle dazugehörigen Metadaten eingibt. In Hinblick auf die Vorteile, die die genaue Beschreibung von Dateien mit sich bringt, wäre es natürlich wünschenswert, wenn jeder User für seine Dateien auch die Metadaten dazu eingeben würde. Denn nur wenn alle Qualifier mit Informationen versehen sind, kann der volle Komfort beim Suchen erzielt werden. So ermöglichen die Qualifier bei-

spielsweise die Suche nach Dateien eines bestimmten Typs von einem bestimmten Urheber zu einem bestimmten Thema.

Um die Eingabe der Metadaten möglichst komfortabel zu gestalten, ermöglicht Japster das Ändern und Vervollständigen der Metadaten jederzeit, also auch dann wenn die Dateien schon freigegeben sind und die dazugehörigen Informationen bereits auf den Server übertragen wurden. Dies ergibt sich dadurch, dass bei jedem Login die gesamten Metadaten aus dem „Shared Dir“ gelesen und in die Datenbank auf dem Server übertragen werden. Zu jeder Datei zu der bereits Metadaten eingegeben wurden gibt es eine korrespondierende Metadatei, welche die Qualifier enthält. Das Format der Metadatei ist XML. Bei jedem Start von Japster wird das Verzeichnis gelesen und die bestehenden XML-Dateien an den Server gesendet. Dort werden die XML-Dateien geparst und die Inhalte der Qualifier in die Datenbank geschrieben.

3.3 Kommunikationsablauf

Um Japster zu starten ruft der User die URL des Applets auf (Abb. 8) und der Server überträgt daraufhin das Applet auf den Client (Abb. 9). Um das Applet starten zu können muß es die Rechte haben auf die lokale Festplatte zuzugreifen (vgl. Kap. 4.1.3). Nach dem Sicherheits-Check, in dem diese Rechte überprüft werden, startet das Applet mit dem Login – Screen (vgl. Kapitel 4.4.1, Abbildung 18).

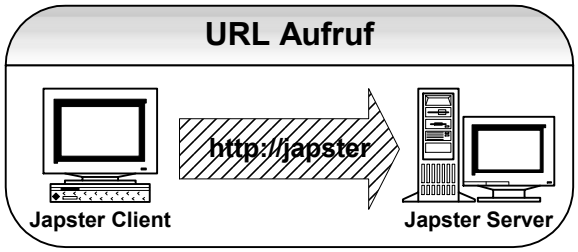


Abbildung 8:URL Aufruf von Japster

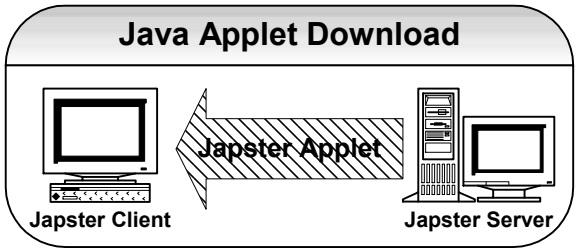


Abbildung 9: Java Applet Download

Ist der Benutzer bereits registriert, reicht es wenn er sich mittels Eingabe seines Usernamens und seines Passworts anmeldet (Abb.10). Nach der Eingabe sendet der Client den Usernamen und das Passwort an den Server (Abb. 11). Stimmt das Passwort mit dem in der Datenbank gespeicherten überein, liefert der Server die Session_ID zurück (Abb.12).

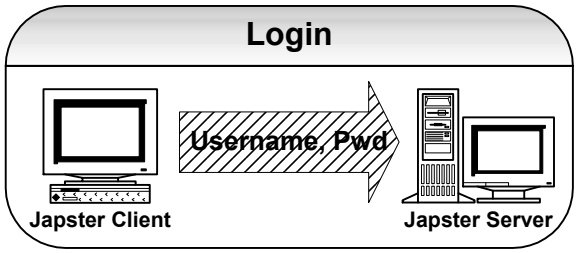


Abbildung 10: Login bei Japster

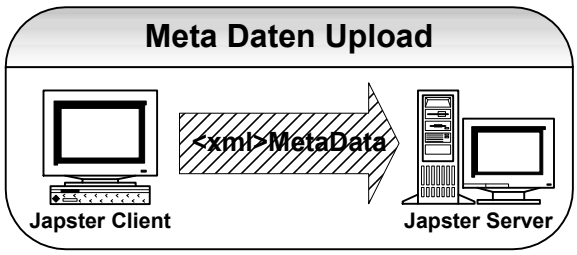


Abbildung 11: Upload der Meta Daten

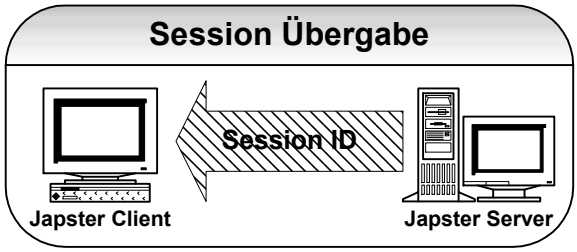


Abbildung 12: Übergabe der Session_ID

Nach dem erfolgten Login werden die Metadaten aus dem „shared dir“ gelesen und im XML-Format an den Server übertragen (Abb. 11). Dieser liest die Daten aus und schreibt sie in die Datenbank. Danach startet Japster.

Als nächstes kann der User die Datenbank nach Dateien durchsuchen (z.B. nach „Seminar.doc“), die jeder andere Japster – User anbietet.

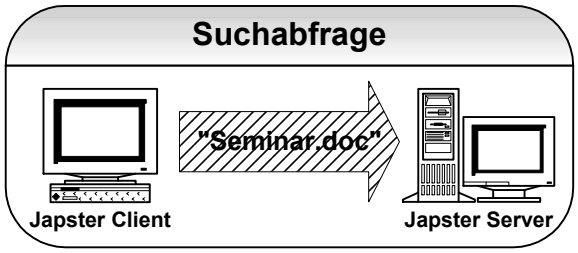


Abbildung 13: Suchabfrage an Japster Server

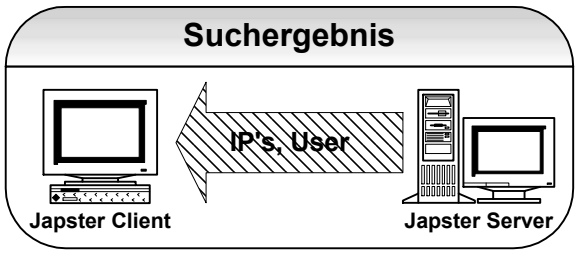


Abbildung 14: Japster Server liefert Suchergebnis

Wird der Button „Search“ gedrückt schickt das Applet die Suchabfrage in Form einer XML-Datei an den Server (Abb. 13). Der Server durchsucht die Datenbank nach übereinstimmenden Dateien und liefert das Suchergebnis ebenfalls in Form einer XML-Datei an den Client zurück (Abb. 14). Das Suchergebnis enthält die Dateinamen, die IP-Adressen der Clienten auf denen die Dateien liegen, deren Verbindungstyp und die Größe der Dateien.

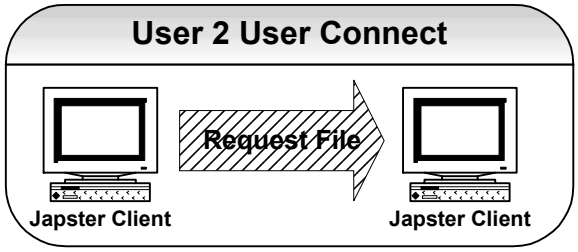


Abbildung 15: User to User Connect für Datei Download

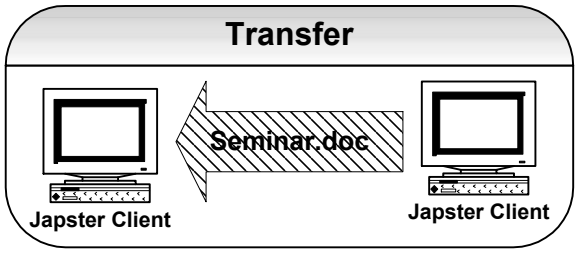


Abbildung 16: Dateitransfer

Ist die gewünschte Datei enthalten, kann der User diese direkt von dem anderen Japster Client anfordern (Peer to Peer, vgl. Kap. 1.2.1), auf dem die Datei gespeichert ist (Abb. 15).

Dieser überträgt dann die Datei und die dazugehörigen Metadaten auf den anfordernden Client (Abb. 16). Das Applet des anfordernden Clients speichert die übertragene Datei dann in das freigegebene Verzeichnis. Bei der nächsten Anmeldung an das System werden die Metadaten dieser Datei dann auch schon auf den Server gespielt und somit für andere User durchsuch- bzw. nutzbar gemacht.

Nach dem Download erfolgt dann das automatische Rating der User. Beim herunterladenden Client wird die Zahl Downloads und beim sendenden Client die Zahl Uploads um 1 erhöht.

4 Implementierung

Im folgenden Kapitel wollen wir den Vorgang der Implementierung etwas genauer beschreiben, jedoch sind in diesem Kapitel keine Source-Codes, Klassendiagramme o.ä. zu erwarten. Diese befinden sich partiell im Anhang.

4.1 Technologieentscheidung – Systemvoraussetzungen

Die grundlegende Entscheidung, Japster als Java Applet zu implementieren, beeinflußte die Entscheidung über die Technologieauswahl entscheidend. Im folgenden sind nun die Entscheidungen über die einzelnen Komponenten genauer erläutert.

4.1.1 Server

Der zentrale Server trägt – wie schon an anderer Stelle erwähnt – die Funktion der Verwaltung der eingeloggten User mit den Files, die auf den Clients abgelegt sind.

Um eine einheitliche Plattform zu schaffen, wurde entschieden, den Japster-Server als Java-Servlet zu betreiben. Um dieses Servlet zu betreiben, wurde folgende Software eingesetzt:

- Betriebssystem: Linux
- Servlet – Engine: Jrun 3.0
- XML-Parser: Java Extension JAXP
- SMTP Server zum automatischen Versenden von e-Mails. Als Inhalt der Mails wird bei Registrierung das Paßwort verschickt.

Die verwendeten Software-Pakete stellen allerdings keine unumstößlichen Normen dar, es können auch adäquate andere Produkte verwendet werden.

4.1.2 Datenbank

Hinsichtlich der Datenbank gibt es aus produkttechnischer Sicht nur die Einschränkungen, daß damit ein relationales Datenbanksystem mit Hilfe der Abfragesprache SQL realisiert werden kann.

Die Entscheidung fiel schließlich auf das Freeware Produkt MySQL, um die geringen Systemanforderungen an die Datenbank zu demonstrieren. Mit anderen Worten verzichteten wir auf den Einsatz eines High-End-Datenbankproduktes (wie beispielsweise Oracle), um bei potentiellen Interessenten keinen höheren Standard als notwendig zu implizieren.

Die Datenbank wurde auf oben beschriebenen Linux-Server eingerichtet.

4.1.3 Client

Der Client ist als Java Applet realisiert (siehe später) und wird auf einem Apache-Web-Server, der auf den bereits beschriebenen Linux-Rechner installiert ist, abgelegt.

Hinsichtlich der Clients gibt es lediglich jene Voraussetzung, daß nur Browser verwendet werden können, die bereits ein Java2-Plug-In installiert haben. Sollte das Plug-In noch nicht installiert sein, kann dies beim ersten Aufruf von Japster automatisch erfolgen.

Eine weitere Voraussetzung auf Seite der Clients besteht darin, daß die Permissions des Policy-Files (vgl. Kapitel 1.3, Sicherheitsaspekte) verändert werden müssen, um dem Java-Plug-In die Möglichkeit zu geben, auf lokale Ressourcen zuzugreifen. Der genaue Ablauf der Änderungen wird in einem eigenen Help-File dargestellt.

4.2 Serverseitige Implementierung

Die Package Struktur des Japster Servers unterscheidet (neben dem Basisnamensraum) zwischen drei Paketen – protocol, db und debug – deren Funktion anschließend kurz erklärt wird:

Protocol

Hier befinden sich alle Klassen, die das Protokoll von Japster kapseln (vgl. dazu die genaue Protokollspezifikation im Anhang). Das bedeutet, daß im Package protocol die Kommunikation zwischen Japster Server und Japster Client definiert wird.

Db

Das Package db beinhaltet alle Klassen und Methoden, die die Kommunikation zwischen zentralem Server und zentraler Datenbank regeln. Das bedeutet, daß jeder Request oder Update in der Datenbank in diesem Package implementiert.

Debug

In diesem Package befindet sich die Klasse debug, die die Suche nach Programmfehlern erleichtert.

4.3 Implementierung der Datenbank

Mit Vorhandensein von registrierten Usern, ist die Speicherung und Verwaltung der Stammdaten in einer Datenbank erforderlich. Nachfolgendes EER-Diagramm soll den Aufbau dieser Datenbank verdeutlichen:

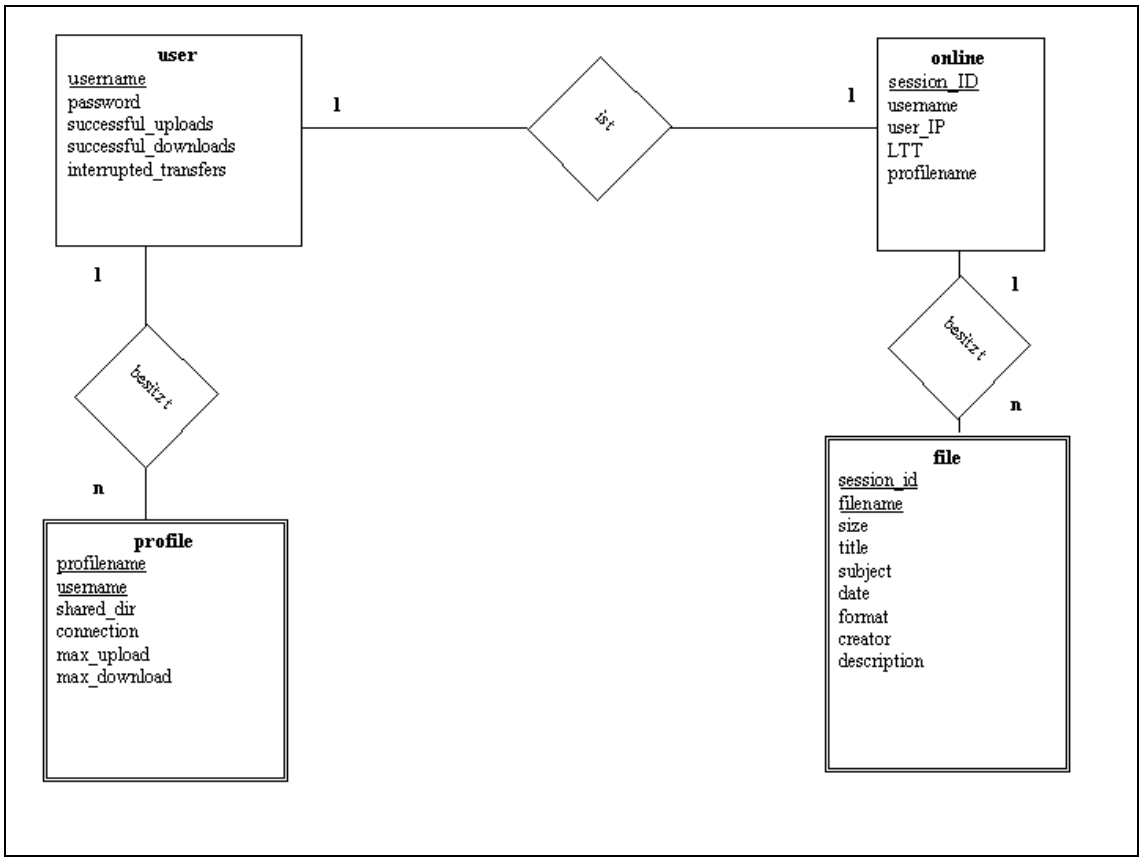


Abbildung 17: ER-Diagramm für Japster

In der Tabelle „User“ werden benutzerrelevante Daten gespeichert. Dies sind einerseits Username und Password, andererseits die Zahl der erfolgreichen Up- und Downloads sowie die Anzahl an unterbrochenen Versuchen, um die Kennzahl ASR oder ein weiterer Folge auch andere Kennzahlen zu berechnen.

In der Tabelle „profile“ werden jene Daten abgelegt, die für die Verwaltung der einzelnen Benutzerprofile (maximal drei pro User) notwendig sind.

Sowohl die Daten der Tabelle „user“ als auch jene der Tabelle „profile“ werden persistent abgespeichert und stehen daher auch dann zur Verfügung, wenn der jeweilige User nicht eingeloggt ist.

Die Daten der Tabellen „online“ und „file“ werden erst im Falle einer „Online-Session“ des Japster-Users ermittelt und nachher wieder verworfen. Das bedeutet, die Daten dieser beiden Tabellen sind nicht persistent. In der Tabelle „online“ wird für die Dauer in der der User online ist, eine eindeutige Session ID vergeben, um einen Dateinamen eindeutig einem User zuzuordnen (zusammengesetzter Schlüssel in der Tabelle „file“). Die Tabelle „file“ enthält schließlich die Meta-Daten der einzelnen Dateien im „shared dir“ des eingeloggten Users.

4.4 Clientseitige Implementierung

Der Client wurde gemäß anfänglicher Definition als Java Applet implementiert. Im Anhang sind dazu die Klassendiagramme einzusehen.

In diesem Kapitel möchten wir uns der graphischen Umsetzung von Japster widmen, um zu demonstrieren, wie die anfänglichen Forderungen, Zielsetzungen und Definitionen umgesetzt wurden.

Das Graphical User Interface (Abkürzung: GUI) wurde in Anlehnung an das Vorbild Napster in vier Einheiten geteilt:

- Das Search Panel
- Das Transfer Panel
- Das Share Panel
- Das Options Panel

4.4.1 GUI: Japster Login

Beim Japster Login hat der User zunächst die Möglichkeit zwischen dem anonymen oder dem registriertem Login. Dies ist sehr einfach durch unterschiedliche Buttons realisierbar (vgl. Abb. 17). Der dritte Button („Register“) dient für jene User, die bisher noch nicht bei Japster registriert waren, sich jedoch jetzt registrieren möchten.

Abbildung 18: Japster - Login

Nach Eingabe von Username und Paßwort bzw. nach dem anonymen Einstieg wird nochmals überprüft, ob Japster als Java Applet auf lokale Ressourcen des Rechners zugreifen darf. Ist diese Überprüfung positiv (d.h. das Zertifikat gültig, das keytool richtig definiert und alle Permissions gesetzt, vgl. Kapitel 1.3), startet Japster und man hat die Auswahl zwischen den vier Panels.

Abbildung 19: Japster - Security Check

4.4.2 GUI: Das Search-Panel

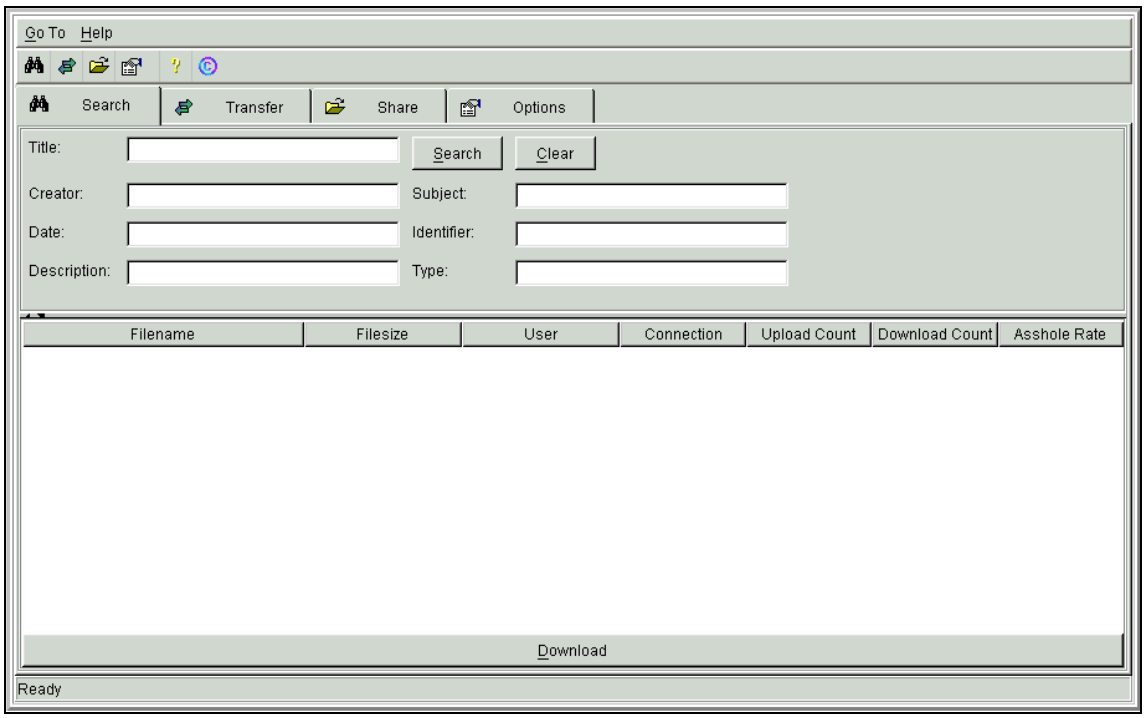


Abbildung 20: Japster - Search Panel

Das Search-Panel erscheint stadardmäßig beim Start von Japster. In den Feldern Title, Creator, Description, etc. können die Dateien nach definierten Meta-Daten gesucht werden.

Die Suchergebnisse (d.h. Übereinstimmungen) werden in der unteren Hälfte des Search-Panels mit weiteren Daten (Dateigröße, „Besitzer“ der Datei, Verbindungstyp, Anzahl Uploads, Anzahl Downloads sowie die Kennzahl ASR (vgl. Kaptitel 2.2) angezeigt.

Durch Doppelklick auf ein angezeigtes Suchergebnis wird automatisch der Download gestartet, der User gelangt automatisch in das Transfer-Panel.

4.4.3 GUI: Das Transfer-Panel

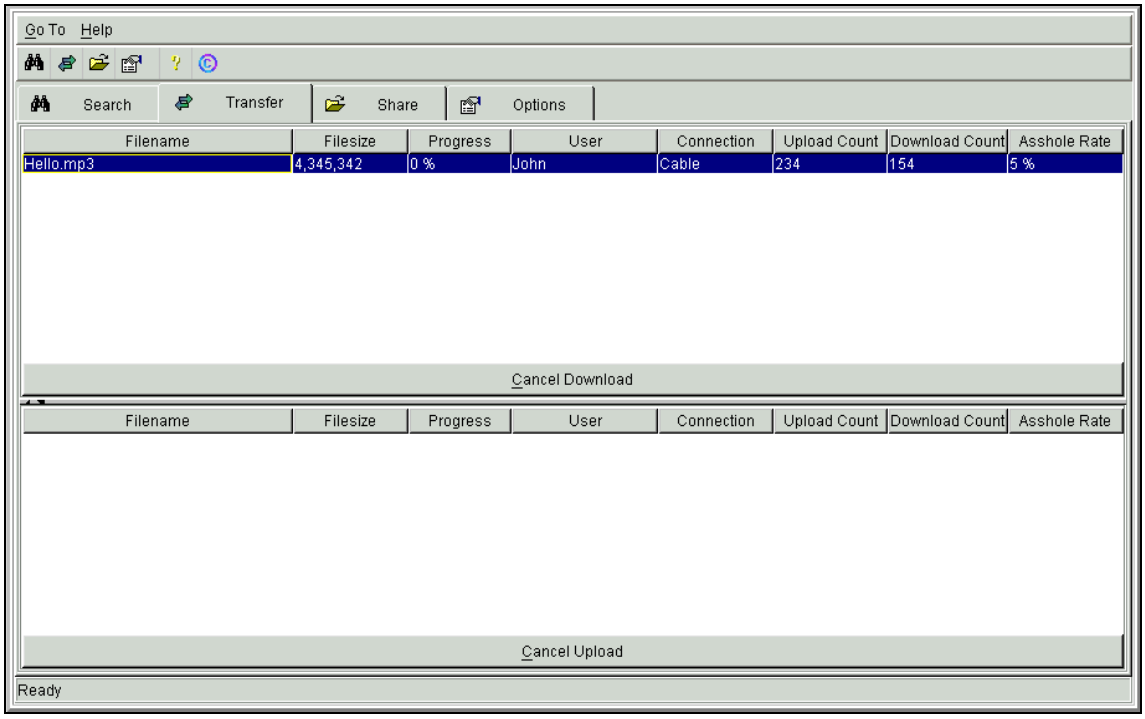


Abbildung 21: Japster - Das Transfer Panel

Im oberen Teilfenster des Transfer Panels sind alle Dateien angeführt, die der Japster User von anderen Usern auf seinen Rechner lädt. Durch das Feld

„Progress“ ist auch der Fortschritt des Ladevorganges zu ersehen. Durch Markieren eines Downloads und durch Drücken des Feldes „Cancel Download“ kann der Ladevorgang unterbrochen werden.

Im unteren Teilfenster sind jene Files angeführt, die andere Japster User von diesem Rechner „runterladen“. Durch Markieren eines Uploads und durch Drücken des Feldes „Cancel Upload“ kann der Ladevorgang unterbrochen werden. Dabei ist jedoch zu beachten, daß sich dies negativ auf die Kennzahl ASR auswirkt, da bei jedem begonnenen Upload die Anzahl der Uploads erhöht, jedoch nicht jene der „Successful Uploads“.

4.4.4 GUI: Das Share-Panel

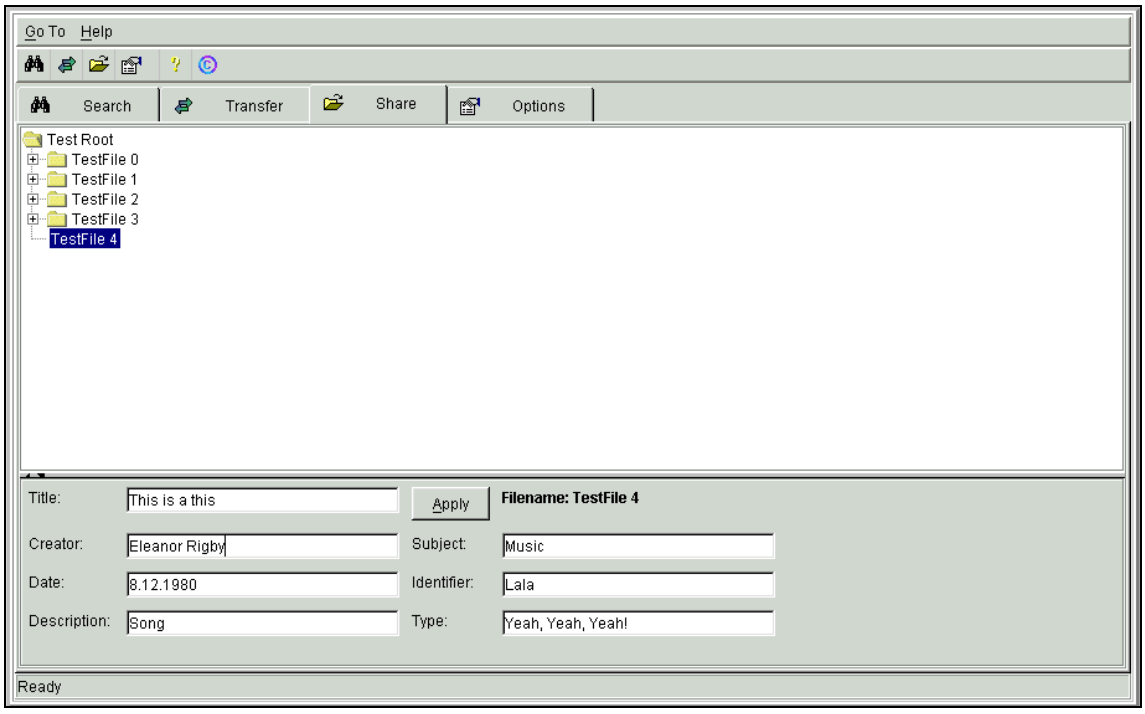


Abbildung 22: Japster - Das Share Panel

Das Share Panel zeigt all jene Dateien, die im sogenannten „shared dir“ abgelegt wurden. Im unteren Teil des Panels sind die Meta-Daten der einzelnen Files zu lesen. Will man für eine Datei Meta-Daten definieren bzw. ändern, füllt

man einfach die entsprechenden Felder aus und bestätigt durch den Button „Apply“.

Von all jenen Dateien, die von anderen Usern geladen werden, werden die Meta-Daten automatisch übernommen, d.h. wurden Meta-Daten für eine Datei einmal ausgefüllt, bleiben diese bei jedem Up- und Download erhalten.

4.4.5 GUI: Das Options-Panel

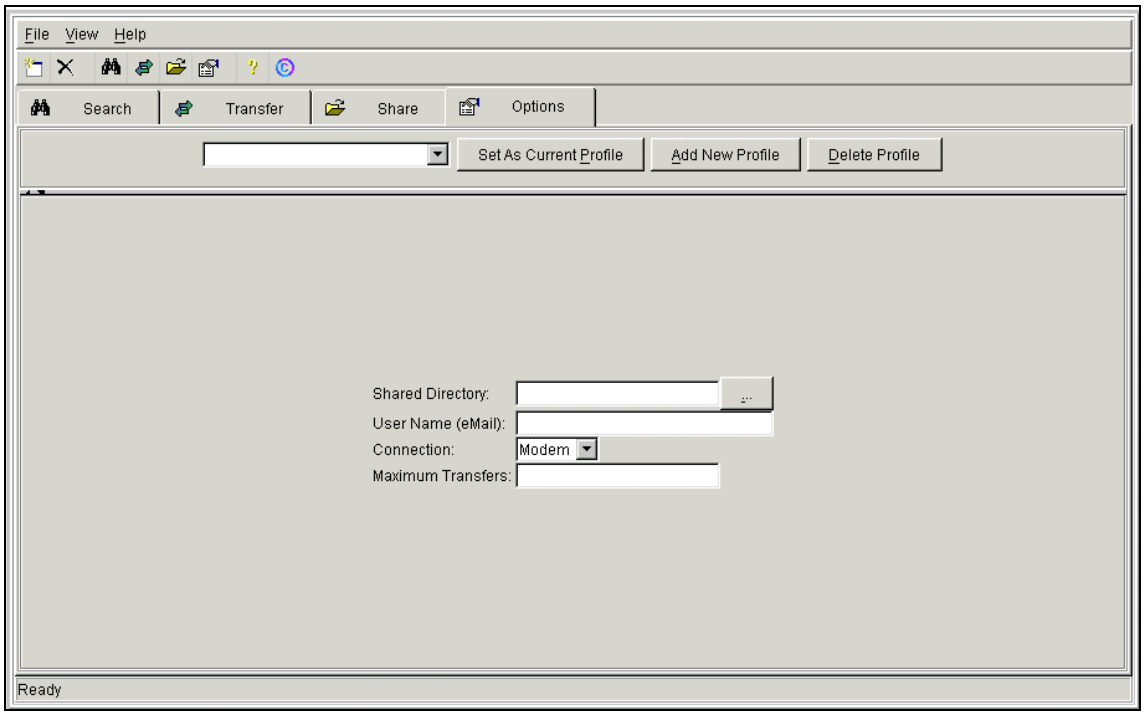


Abbildung 23: Japster - Das Options Panel

Im Options Panel kann der Japster User seine Profile editieren. Pro Profil (maximal sind drei Profile möglich) kann dabei das „shared directory“, der Username, der Connection-Type und die maximale Anzahl an Transfers (Uploads für andere User) geändert werden.

5 Zusammenfassung

Nach der Beschreibung der Technologie, der Auswahl des Konzeptes und der Implementation, sollen in diesem Kapitel vor allem die Probleme während der Implementation und die daraus resultierenden notwendigen Tätigkeiten näher beschrieben werden

Während der Implementationsphase mußte festgestellt werden, daß die teilweise hohen Zielsetzungen im Punkt „Geringer Installationsaufwand“ nicht erreicht werden.

Es stellte sich nach der Erstellung des ersten Prototyps heraus, daß die Installation eines Zertifikats inkl. der Konfiguration des Keytools und der Vergabe der Permissions für einen durchschnittlichen User nicht zumutbar ist. Es müßte in weiterer Folge eine Lösung gefunden werden, diesen Vorgang in weiteren Versionen von Japster zu automatisieren, wobei die Zielsetzung des möglichst geringen Installationsvorganges dennoch nicht eingehalten werden konnte.

Weiters stellte sich in diesem Zusammenhang heraus, daß sich der Vorgang des Zertifikatsimports bei den beiden Produkten „Microsoft Internet Explorer“ und „Netscape Navigator“ unterschiedlich abläuft bzw. nur Zertifikate unterschiedlichen Typs verwendet werden können. Aus diesem Grund ist es bisher nur möglich, Japster mit dem Internet Explorer von Microsoft zu verwenden. An einer Lösung für den Netscape Navigator wird nach wie vor gearbeitet. Diese Einschränkung ist auch als teilweise Verfehlung des Projektzieles Plattformunabhängigkeit zu sehen.

Die Grenzen von Japster sind vor allem beim Download gleichnamiger Dateien zu erreichen. Befindet sich beispielsweise im „shared dir“ eine Datei namens „Test.txt“ und wird vom Japster User eine gleichnamige Datei per Download geladen, wird die alte Datei durch die neue überschrieben. Das beutetet, daß die Dateien auf den lokalen Datenträgern nicht nach dem Inhalt bzw. nach den Meta-Daten durchsucht werden, sondern nach dem Dateinamen. Dies ist für den Japster-User so lange zu beachten, bis eine eventuelle zusätzliche Überprüfung der Meta-Daten erfolgen kann.

Zusammenfassend ist zu sagen, daß Japster aufgrund seines Konzeptes mit Java und der Verwendung von Meta-Daten, einen äußerst innovativer Ansatz innerhalb der Distribution Channels darstellt, jedoch noch ein wenig mehr Zeit bräuchte, um an eine „Serienproduktion“ bzw. kommerzielle Vermarktung zu denken, die jedoch einer getrennten Betrachtung bedürfe.

6 Bibliographie

[Hawl00] Hawlitzek F., Java 2, München 2000, Addison-Wesley

[Guth00] Guth S., Die Entwicklung der Sicherheitskonzepte für Java Applets, Universität Essen, Essen 2000

[StSc99] Strauß, R. E.; Schoder D.: Electronic Commerce – Herausforderung aus der Sicht der Unternehmen. In: Hermanns, A.; Sauter, M. (Hrsg.): Management Handbuch Electronic Commerce. München 1999: Vahlen.

[PTA01] Pressetext Austria: Napster und Gnutella verbreiten Kinderpornos. http://www.pressetext.com/open.php?pte=010109058. Abruf am 2000-01-10.

[NUA01] Nua Internet Surveys: How many online? http://www.nua.com/surveys/how_many_online/world.html. Abruf am 2000-01-23.

[EuKo97] Europäische Kommission: A European Initiative in Electronic Commerce. Http://www.ispo.cec.be/Ecommerce, 1997-04-14, Abruf am 2000-11-16.

Sonstige Hilfsmittel und Programmierhilfen

[Dublin] Dublin Core Metadaten Initiative: Dublin Core Metadaten Initiative. http://purl.org/dc/index.htm. Abruf am 2000-01-09.

[BIBDIENST] Rusch-Feja D., Dublin Core Metadata Initiative, Bibliotheksdienst Heft 4/97

[HeSC00] Hermes, A.; Schuhmacher, B.: Arbeitshefte Informatik. Java. Stuttgart 2000: Ernst Klett Verlag.

[Java] Java: Java Online Support : Online Support. http://java.sun.com. Abruf am 2000-01-09.

[Servlets] Online Tutorial: Servlets and JavaServer Pages (JSP) 1.0: A Tutorial. http://www.apl.jhu.edu/%7Ehall/java/Servlet-Tutorial. Abruf am 2000-01-09.

[Step00] Stepken, G.: MySQL Datenbankhandbuch. http://www2.little-idiot.de/mysql. Abruf am 2000-01-09.

[Neum01] Neumann G., Distribution Digitaler Güter – Foliensatz, Abteilung für Wirtschaftsinformatik, Wien 2001

7 Anhang

7.1 Protokollspezifikation

Dokumentenhistorie:

Version	Datum	Beschreibung	Autor
1.00	16.11.2000	Erstellung	Christoph Bäck
1.01	20.11.2000	Request DEL_PROFILE hinzugefügt	Christoph Bäck
1.02	21.11.2000	diverse Rechtschreibfehler ausgebessert Fußnoten für RFCs hinzugefügt Für SEARCH ein ACK NO_MATCH_FOUND hinzugefügt Fehler bei XML Daten spezifiziert DTDs hinzugefügt	Christoph Bäck
1.03	21.11.2000	Für LOGIN ein ACK: MAX_USER_SESSIONS und MAX_ANONYMOUS_SESSIONS hinzugefügt	Christoph Bäck
1.04	24.11.2000	Bei XML-Dokumenten von SEARCH wurde upload_speed hinzugefügt Bei profile.dtd (für GET_PROFILE und PUT_PROFILE) wurde die Entität dir zu shared_dir geändert	Christoph Bäck
1.05	24.11.2000	Bei LOGIN wurde Status-Code TOO_MANY_USERS entfernt	Christoph Bäck
1.06	26.11.2000	user_rating bei found_items hinzugefügt	Christoph Bäck

1.07	04.12.2000	Request CHANGE_PWD hinzugefügt Request REGISTER hinzugefügt Bei Request LOGIN existiert jetzt ein anonymer Benutzer der keinen Benutzernamen anzugeben hat Bei Request GET existiert jetzt eine ACK-Message TOO_MANY_USERS Entität date bei folgenden XML-Dokumenten hinzugefügt: share.dtd, search.dtd, found_items.dtd found_items.dtd umgestellt Verhalten von found_items.dtd bei anonymen User neu spezifiziert Request WHOIS hinzugefügt	Christoph Bäck
1.08	05.12.2000	REGISTER hat nun die Parameter PASSWD und USERNAME	Christoph Bäck
1.09	05.12.2000	REGISTER hat wieder die Parameter EMAIL_ADDRESS und USERNAME	Christoph Bäck

1.10	19.12.2000	REGISTER hat nur den Parameter USERNAME, wobei dieser als E-Mail Adresse des Benutzers definiert wird Entität <format> wurde in <type> geändert Bei SEARCH gibt es im XML-Dokument kein <upload_speed> mehr bzw. jetzt <connection>, da danach nicht gesucht wird <upload_speed> und <download_speed> zu <connection> vereinheitlicht	Christoph Bäck
1.11	22.12.2000	Status Code DOC_NOT_WELLFORMED ersatzlos gestrichen Anpassung bei den XML-Dokumenten bzgl. des DOCTYPEs und der DTD (DTD muss via HTTP erreichbar sein) Umlaut Unterstützung á la HTML in DTDs eingebaut (siehe neues Kapitel **Inhalt der XML-Dokumente**)	Christoph Bäck

1.12	28.12.2000	INVALID_PARAM genauer spezifiziert Bei LOGIN Parameter USER_IP in IP unbenannt Parameter PROFILE_NAME in PROFILENAME unbenannt Bei WHOIS Parameter USER in USERNAME unbenannt Bei RATE Parameter USER_TO_RATE in USERNAME unbenannt Bei GET Parameter USER in USERNAME unbenannt Bei SHARE und SEARCH date-Format in Form eines SQL-Dates XML Dokument und DTD von found_items, user_ip zu ip und user_rating zu rating unbenannt Bei XML Dokument von SEARCH wurde das Asterisk durch einen Leereintrag ersetzt XML Dokument und DTD von WHOIS (DTD heißt jetzt whois.dtd) whois_user zu whois und user_rating zu rating unbenannt	Christoph Bäck

1.13	29.12.2000	Anstatt von Leereinträgen werden Entitäten weggelassen, falls sie nicht beachtet werden sollen bzw. kein Eintrag existiert (SHARE (share.dtd), SEARCH (search.dtd, found_items.dtd)	Christoph Bäck
1.14	23.01.2001	Bei allen XML-Dokumenten wurde das Encoding (Zeichensatz) auf ISO-8859-1 (Latin-1) umgestellt	Christoph Bäck

7.1.1 Allgemeines zum Aufbau der Requests und der Kommunikation

Da Servlets zur Kommunikation verwendet werden, besitzt ein jeder Aufruf mindestens den Parameter `REQUEST`, dieser gibt den Typ der Request an, der Typ wird case-insensitive interpretiert, d.h. die Groß-/Kleinschreibung eines Requests wird ignoriert.

Anmerkung:

In diesem Text werden bestimmte Zeichen dazu verwendet um Optionalität bzw. mögliche Wiederholungen anzuzeigen, diese Zeichen sind an eine EBNF angelehnt.

Folgende Zeichen werden verwendet:

[optionaler Text**]** ... Zeigt Optionalität an

{optionale Wiederholung**}** ... Kann keinmal oder mehrmals Vorkommen

Die Beschreibung der einzelnen Requests an das Servlet haben immer folgende Form:

Format:

`REQUEST_TYPE` *`REQUEST_PARAMETER`* ***{****`REQUEST_PARAMETER`****}***

Parameter:

Parametername	**Beschreibung**
`REQUEST_PARAMETER`	Beschreibung des Parameters

Mögliche Status-Codes der folgenden ACK[5]-Message:

Status-Code	**Beschreibung**
STATUS_CODE	Ergebnis des Requests, bei Requests die Daten zurückliefern wird ein ACK nur im Fehlerfall geliefert, ansonsten existiert eine ACK OK Message.

Wie aus dem obigen Beispiel ersichtlich wird der `REQUEST_TYPE` immer in der Schriftart Courier dargestellt, die *`Parameter`* in kursiven Courier.

Nach der Kommunikationsrichtung werden die Requests in folgenden Kategorien aufgeteilt:

Kommunikation mit dem Server die vom Client initiiert wird

Kommunikation mit dem Client die vom Server initiiert wird

Kommunikation zwischen den Clients

Desweiteren wird zwischen Requests unterschieden die Daten zurückliefern oder keine zurückliefern.

Bei Requests die keine Daten (außer einer ACK-Message) zurückliefern, wird eine ACK-Message zurückgeliefert die angibt, ob die Operation erfolgreich oder erfolglos durchgeführt wurde.

Bei Requests die Daten zurückliefern, wird nur eine ACK-Message im Fehlerfall zurückgeliefert.

Es können folgende Daten durch Requests zurückgeliefert werden:

unstrukturierte Daten:
z.B. `session_id\n`

strukturierte Daten:
`<xml><entitaet>strukturierte Daten</entitaet></xml>`

Falls strukturierte Daten dem Servlet übergeben werden sollen, wird dies mittels des Parameter *`XML_DATA`* bewerkstelligt.

[5] ACK steht für Acknowledge, welches soviel wie mitteilen heißt

Parameterwerte sind **URL encoded[6]!**

7.1.2 Inhalt der XML-Dokumente

In den XML-Dokumente dürfen keine Sonderzeichen enthalten sein (analog zu HTML). Auf Grund dessen, sind in XML Dokumenten bestimmte Entitäten vordefiniert, diese sind in den Dokumenten auch so zu übertragen, da ansonsten Fehler beim Parsing auftreten können.

Sonderzeichen und deren Entsprechung:

Sonderzeichen	Entsprechung (analog zu HTML)
<	&lt;
>	&gt;
&	&amp;
”	&quot;
’	&apos;

[6] siehe **RFC1738**, abrufbar unter http://rfc.fh-koeln.de/rfc/html/rfc1738.html (Abrufdatum: 21.11.2000)

7.1.3 Die ACK-Message

Format:

`ACK` *`STATUS_CODE`*

Falls eine Request der keine Daten zurückliefert erfolgreich war, wird als Status-Code `OK` zurückgeliefert!

Allgemein gibt es global für alle Requests folgende Status-Codes die Fehler anzeigen:

Status-Code	Beschreibung
OK	Operation wurde erfolgreich durchgeführt, dieser Status-Code existiert nicht bei Requests die strukturierte oder unstrukturierte Daten zurückliefern
INVALID_PARAM	Es fehlt ein Parameter bzw. ein Parameterwert ist nicht innerhalb der erlaubten Wertemenge, er ist zu lang, besteht nur aus Whitespace oder ist leer usw.
UNKNOWN_REQUEST	Es existiert kein Request mit diesem Namen

Status-Codes die request-spezifische Fehler oder Zustände anzeigen, werden bei den einzelnen Requests gesondert behandelt!

Falls XML Dokumente zum Server übertragen werden (mittels des Parameters *XML_DATA*), ist immer folgender Status-Code definiert:

Status-Code	Beschreibung
INVALID_DOC	Es wurde ein ungültiges XML-Dokument empfangen, d.h. das XML Dokument war nicht valide, oder nicht Well-formed oder Werte die im Dokument definiert wurden (Attribute, Elemente) sind zu lang oder in einem ungültigen Format (nicht numerisch, nicht in einem erlaubten Bereich usw.)

Anmerkung:

Die Status-Codes sind Konstanten die beim Ausliefern zum Client durch die entsprechende IDs (=signed 32-Bit Integer, in Textform (nicht binär)) substituiert werden.

7.1.4 Kommunikation mit dem Server die vom Client initiiert wird

REGISTER

Format:

`REGISTER` *USERNAME*

Parameter:

Parametername	Beschreibung
USERNAME	Der gewünschte Benutzername, der zugleich auch die **E-Mail Adresse des Benutzers** sein muss, an die das automatisch generierte Paßwort gesandt wird

Mögliche Status-Codes der folgenden ACK-Message:

Status-Code	Beschreibung
MAX_USER_COUNT	Die Maximalanzahl an anlegbaren Benutzern wurde erreicht
NO_UNIQUE_NAME	Es wurde ein Username verwendet, der nicht eindeutig ist
SERVER_ERROR	Am Server ist ein Fehler aufgetreten (z.B.: DB konnte nicht erreicht werden, eine SQL-Exception oder sonstiges)

Retourniert:

Im Normalfall ‚`ACK OK\n`‘

Im Fehlerfall ‚`ACK STATUS-CODE\n`‘, mit dem Status-Code des Fehlers

7.1.5 LOGIN

Format:

`LOGIN` *IP* **[***USERNAME PASSWORD***]**

Parameter:

Parametername	Beschreibung
IP	IP Adresse des Benutzers
USERNAME^{optional}	Benutzername am System, falls kein PASSWORD und USERNAME angegeben wurde findet ein anonymes Login statt
PASSWORD^{optional}	Paßwort zum Benutzernamen, falls kein PASSWORD und USERNAME angegeben wurde findet ein anonymes Login statt

Mögliche Status-Codes der folgenden ACK-Message:

Status-Code	Beschreibung
ALREADY_LOGGED_IN	Es ist schon jemand unter dem angegebenen Account angemeldet
MAX_ANONYMOUS_SESSIONS	Die Maximalanzahl an anonymen Benutzern wurde erreicht, kann nur beim anonymen Login auftreten
MAX_USER_SESSIONS	Die Maximalanzahl an nicht-anonymen Benutzern wurde erreicht, kann nur beim nicht-anonymen Login auftreten
SERVER_ERROR	Am Server ist ein Fehler aufgetreten (z.B.: DB konnte nicht erreicht werden, eine SQL-Exception oder sonstiges)
USER_NOT_FOUND	Der angegebene Benutzername konnte in der Benutzer-DB nicht gefunden werden

WRONG_PASSWD	Das angegebene Paßwort stimmt nicht mit dem in der Benutzer-DB angegebenen überein

Retourniert:

Im Normalfall ‚`SESSION_ID\n`', `SESSION_ID` ist mit einer eindeutigen generierten ID zu ersetzen.

Im Fehlerfall ‚`ACK STATUS-CODE\n`', mit dem Status-Code des Fehlers

7.1.6 LOGOUT

Format:

`LOGOUT` *`SESSION_ID`*

Parameter:

Parametername	**Beschreibung**
`SESSION_ID`	Identifiziert einen eingeloggten Benutzer eindeutig

Mögliche Status-Codes der folgenden ACK-Message (außer OK):

Status-Code	**Beschreibung**
SERVER_ERROR	Am Server ist ein Fehler aufgetreten (z.B.: DB konnte nicht erreicht werden, eine SQL-Exception oder sonstiges)

SESSION_NOT_FOUND	Der durch die Session ID identifizierte Benutzer ist schon vom System abgemeldet oder er war nie angemeldet

Retourniert:

Im Normalfall ‚`ACK OK\n`'

Im Fehlerfall ‚`ACK STATUS-CODE\n`', mit dem Status-Code des Fehlers

7.1.7 CHANGE_PWD

Format:

`CHANGE_PWD` *`SESSION_ID NEW_PASSWD`*

Parameter (außer `SESSION_ID`):

Parametername	Beschreibung
`NEW_PASSWD`	Das neue Paßwort durch dass das Alte ersetzt werden soll

Mögliche Status-Codes der folgenden ACK-Message:

Status-Code	Beschreibung
PASSWD_NOT_ACCEPTED	Neues Paßwort wird vom System nicht akzeptiert (z.B. zu kurz)
SERVER_ERROR	Am Server ist ein Fehler aufgetreten (z.B.: DB

	konnte nicht erreicht werden, eine SQL-Exception oder sonstiges)
SESSION_NOT_FOUND	Der durch die Session ID identifizierte Benutzer ist schon vom System abgemeldet oder er war nie angemeldet
USER_IS_ANONYMOUS	Ein anonymer Benutzer versuchte ein Paßwort zu ändern, dies ist nicht erlaubt, da anonyme Benutzer keine Paßwörter besitzen

Retourniert:

Im Normalfall ‚`ACK OK\n`'

Im Fehlerfall ‚`ACK STATUS-CODE\n`', mit dem Status-Code des Fehlers

7.1.8 GET_PROFILES

Format:

`GET_PROFILES` *`SESSION_ID`*

Mögliche Status-Codes der folgenden ACK-Message:

Status-Code	Beschreibung
NO_PROFILES_FOUND	Für den durch die Session ID identifizierten Benutzer existieren keine Profile auf den Server
SERVER_ERROR	Am Server ist ein Fehler aufgetreten (z.B.: DB konnte nicht erreicht werden, eine SQL-Exception oder sonsti-

	ges)
SESSION_NOT_FOUND	Der durch die Session ID identifizierte Benutzer ist schon vom System abgemeldet oder er war nie angemeldet
USER_IS_ANONYMOUS	Ein anonymer Benutzer versuchte eine Profilliste zu bekommen. Dies ist nicht erlaubt, da anonyme Benutzer keine Profile verwenden dürfen

Retourniert:

Im Normalfall ‚`PROFILENAME\n{PROFILENAME\n}`‘

Falls keine Profile gefunden wurden, wird mit ‚`ACK NO_PROFILES_FOUND\n`‘ geantwortet

Im Fehlerfall ‚`ACK STATUS-CODE\n`‘, mit dem Status-Code des Fehlers (ohne NO_PROFILES_FOUND)

7.1.9 GET_PROFILE

Format:

`GET_PROFILE` *`SESSION_ID PROFILENAME`*

Parameter (außer `SESSION_ID`):

Parametername	Beschreibung
`PROFILENAME`	Name des zu holenden Profils

Mögliche Status-Codes der folgenden ACK-Message:

Status-Code	Beschreibung
NO_PROFILES_FOUND	Für den durch die Session ID identifizierten Benutzer existieren keine Profile auf den Server
PROFILE_NOT_FOUND	Es existieren Profile am Server, aber das Angegebene nicht
SERVER_ERROR	Am Server ist ein Fehler aufgetreten (z.B.: DB konnte nicht erreicht werden, eine SQL-Exception oder sonstiges)
SESSION_NOT_FOUND	Der durch die Session ID identifizierte Benutzer ist schon vom System abgemeldet oder er war nie angemeldet
USER_IS_ANONYMOUS	Ein anonymer Benutzer versuchte eine Profil zu bekommen. Dies ist nicht erlaubt, da anonyme Benutzer keine Profile verwenden dürfen

Retourniert:

Im Normalfall XML-Dokument für ein Profil, z.B.:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE profile SYSTEM „URL_DTDs[7]/profile.dtd">

<profile>
  <shared_dir>shared_dir</shared_dir>
  <connection>Cable</connection>
  <max_upload>50</max_upload>
  <max_download>50</max_download>
</profile>
```

[7] **URL_ DTDs** ist die URL zu den DTDs, die öffentlich zugänglich und zentral aufliegen

Im Fehlerfall ‚`ACK STATUS-CODE\n`', mit dem Status-Code des Fehlers

7.1.10 PUT_PROFILE

Format:

`PUT_PROFILE` *`SESSION_ID PROFILENAME XML_DATA`*

Parameter (außer `SESSION_ID`):

Parametername	Beschreibung
`PROFILENAME`	Name des zu setzenden Profils oder neu zu erzeugenden Profils
`XML_DATA`	XML-Dokument für ein Profil

Mögliche Status-Codes der folgenden ACK-Message (außer OK und INVALID_DOC):

Status-Code	Beschreibung
MAX_PROFILE_COUNT	Es wurde die Maximalanzahl an Profilen für diesen Benutzer erreicht
SERVER_ERROR	Am Server ist ein Fehler aufgetreten (z.B.: DB konnte nicht erreicht werden, eine SQL-Exception oder sonstiges)
SESSION_NOT_FOUND	Der durch die Session ID identifizierte Benutzer ist schon

	vom System abgemeldet oder er war nie angemeldet
USER_IS_ANONYMOUS	Ein anonymer Benutzer versuchte eine Profil zu speichern. Dies ist nicht erlaubt, da anonyme Benutzer keine Profile verwenden dürfen

XML_DATA Bsp.:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE profile SYSTEM „URL_DTDs/profile.dtd">

<profile>
  <shared_dir>shared_dir</shared_dir>
  <connection>Cable</connection>
  <max_upload>50</max_upload>
  <max_download>50</max_download>
</profile>
```

Retourniert:

Im Normalfall ‚`ACK OK\n`‘

Im Fehlerfall ‚`ACK STATUS-CODE\n`‘, mit dem Status-Code des Fehlers

7.1.11 DEL_PROFILE

Format:

`DEL_PROFILE` *`SESSION_ID PROFILENAME`*

Parameter (außer `SESSION_ID`):

Parametername	Beschreibung
PROFILENAME	Name des zu löschenden Profils

Mögliche Status-Codes der folgenden ACK-Message:

Status-Code	Beschreibung
NO_PROFILES_FOUND	Für den durch die Session ID identifizierten Benutzer existieren keine Profile auf den Server
PROFILE_NOT_FOUND	Es existieren Profile am Server, aber das Angegebene nicht
SERVER_ERROR	Am Server ist ein Fehler aufgetreten (z.B.: DB konnte nicht erreicht werden, eine SQL-Exception oder sonstiges)
SESSION_NOT_FOUND	Der durch die Session ID identifizierte Benutzer ist schon vom System abgemeldet oder er war nie angemeldet
USER_IS_ANONYMOUS	Ein anonymer Benutzer versuchte eine Profil zu löschen. Dies ist nicht erlaubt, da anonyme Benutzer keine Profile verwenden dürfen

Retourniert:

Im Normalfall ‚`ACK OK\n`'

Im Fehlerfall ‚`ACK STATUS-CODE\n`', mit dem Status-Code des Fehlers

7.1.12 SHARE

Format:

`SHARE` *`SESSION_ID XML_DATA`*

Parameter (außer `SESSION_ID`):

Parametername	Beschreibung
`XML_DATA`	XML-Dokument für die shared Items

Mögliche Status-Codes der folgenden ACK-Message (außer OK und INVALID_DOC):

Status-Code	Beschreibung
MAX_FILE_COUNT	Die Maximalanzahl an Dateien die vom System gespeichert werden wurde erreicht
SERVER_ERROR	Am Server ist ein Fehler aufgetreten (z.B.: DB konnte nicht erreicht werden, eine SQL-Exception oder sonstiges)
SESSION_NOT_FOUND	Der durch die Session ID identifizierte Benutzer ist schon vom System abgemeldet oder er war nie angemeldet

XML_DATA Bsp.:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE share SYSTEM „URL_DTDs/share.dtd">

<share>
  <item filename="share.pdf">
    <title>Share mechanisms in major OSs</title>
    <creator>John Doe</creator>
```

```
    <subject>CS          Computer          Science</subject>
    <description>Share  Locking  Files  OSs  OS</description>
    <type>application/pdf</type>[8]
    <date>2001-08-01</date>
    <size>1024</size>
  </item>
  {<item                                        filename="...">
    <title>...</title>
    <creator>...</creator>
    <subject>...</subject>
    <description>...</description>
    <type>...</type>

    <date>...</date>
    <size>...</size>
  </item>}
</share>
```

Retourniert:

Im Normalfall ‚`ACK OK\n`‘

Im Fehlerfall ‚`ACK STATUS-CODE\n`‘, mit dem Status-Code des Fehlers

7.1.13 SEARCH

Format:

[8] Type orientiert sich an die Internet Media Types abrufbar unter http://www.isi.edu/in-notes/iana/assignments/media-types/media-types (Abrufdatum: 21.11.2000)

`SEARCH` *`SESSION_ID XML_DATA`*

Parameter (außer `SESSION_ID`):

Parametername	Beschreibung
`XML_DATA`	XML-Dokument für die shared Items

Mögliche Status-Codes der folgenden ACK-Message (außer INVALID_DOC):

Status-Code	Beschreibung
NO_MATCH_FOUND	Es wurden keine zu dem Suchmuster passenden shared Items gefunden
SERVER_ERROR	Am Server ist ein Fehler aufgetreten (z.B.: DB konnte nicht erreicht werden, eine SQL-Exception oder sonstiges)
SESSION_NOT_FOUND	Der durch die Session ID identifizierte Benutzer ist schon vom System abgemeldet oder er war nie angemeldet

XML_DATA Bsp.:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE search SYSTEM „URL_DTDs/search.dtd">

<search>

   <title>Share mechanisms in major OSs</title>
   <creator>John Doe</creator>
   <subject>CS Computer Science</subject>
   <description>Share Locking Files OSs OS
```

```
                Network</description>
   <type>application/pdf</type>
   <date>2001-08-01</date>
</search>
```

Anmerkung:

In obigen Beispiel ist der filename für die Suche nicht relevant, da die Entität <filename> fehlt!

Retourniert:

Im Normalfall XML-Dokument (d.h. es wurden passende shared Items gefunden) für die gefundenen Dateien:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE found_items SYSTEM „URL_DTDs/found_items.dtd">

<found_items>
  <item filename="share.pdf" ip="111.111.111.111">
    [<user_desc>
      <username>Frank_Doolittle</username>
      <connection>Cable</connection>
      <download_count>20</download_count>
      <upload_count>100</upload_count>
      <rating>5</rating>
    </user_desc>]

    <title>Share mechanisms in major OSs</title>
    <creator>John Doe</creator>
    <subject>CS Computer Science</subject>
    <description>Share Locking Files OSs OS</description>
    <type>application/pdf</type>
    <date>2001-08-01</date>
    <size>1024</size>
```

```
  </item>

  {<item              filename="..."              ip="...">
   [<user_desc>
      <username>...</username>
      <connection>...</connection>
      <download_count>...</download_count>
      <upload_count>...</upload_count>
      <rating>...</rating>
    </user_desc>]

    <title>...</title>
    <creator>...</creator>
    <subject>...</subject>
    <description>...</description>
    <type>...</type>
    <date>...</date>
    <size>...</size>
  </item>}
</found_items>
```

Anmerkung:

Die Entität `user_desc` ist ein optionaler Bestandteil der XML-Daten, sie kommt nicht bei anonymen Benutzern vor, die die Items zur Verfügung gestellt haben!

Falls keine passenden shared Items gefunden wurden, wird mit ‚`ACK NO_MATCH_FOUND\n`' geantwortet.

Im Fehlerfall ‚`ACK STATUS-CODE\n`', mit dem Status-Code des Fehlers (ohne NO_MATCH_FOUND)

7.1.14 WHOIS

Format:

`WHOIS` *`SESSION_ID USERNAME`*

Parameter (außer `SESSION_ID`):

Parametername	Beschreibung
`USERNAME`	Benutzernamen zu dem Information angefordert wird

Mögliche Status-Codes der folgenden ACK-Message:

Status-Code	Beschreibung
SERVER_ERROR	Am Server ist ein Fehler aufgetreten (z.B.: DB konnte nicht erreicht werden, eine SQL-Exception oder sonstiges)
SESSION_NOT_FOUND	Der durch die Session ID identifizierte Benutzer ist schon vom System abgemeldet oder er war nie angemeldet, oder der mit USERNAME bezeichnete Benutzer ist nicht angemeldet!
USER_NOT_FOUND	Der Benutzer wurde in der DB nicht gefunden, oder ist nicht eingeloggt

Retourniert:

Im Normalfall XML-Dokument (d.h. es wurde ein passender Benutzer gefunden) für die Benutzerdaten (whois_user.dtd):

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE whois SYSTEM „URL_DTDs/whois.dtd">

<whois>
   <connection>Cable</connection>
   <upload_count>200</upload_count>
   <download_count>300</download_count>
   <rating>5</rating>
</whois>
```

Im Fehlerfall ‚ACK STATUS-CODE\n', mit dem Status-Code des Fehlers

7.1.15 RATE

Format:

RATE *SESSION_ID USERNAME DOWNLOAD_RESULT*

Parameter (außer SESSION_ID):

Parametername	Beschreibung
USERNAME	Loginname des zu bewertenden Benutzers
DOWNLOAD_RESULT	Eigentliche Bewertung, kann folgende Werte besitzen: SUCC ... Download erfolgreich FAIL ... Download nicht erfolgreich

Mögliche Status-Codes der folgenden ACK-Message (außer OK):

Status-Code	Beschreibung
SERVER_ERROR	Am Server ist ein Fehler aufgetreten (z.B.: DB konnte nicht erreicht werden, eine SQL-Exception oder sonstiges)
SESSION_NOT_FOUND	Der durch die Session ID identifizierte Benutzer ist schon vom System abgemeldet oder er war nie angemeldet
USER_IS_ANONYMOUS	Ein anonymer Benutzer versuchte eine Bewertung vorzunehmen oder es wurde versucht einen anonymen Benutzer zu bewerten. Dies ist nicht erlaubt, da anonyme Benutzer keine Bewertungen vornehmen dürfen und auch nicht bewertet werden dürfen
USER_NOT_FOUND	Der zu bewertende Benutzer wurde in der DB nicht gefunden

Retourniert:

Im Normalfall ‚`ACK OK\n`‘

Im Fehlerfall ‚`ACK STATUS-CODE\n`‘, mit dem Status-Code des Fehlers

7.1.16 Kommunikation mit dem Client die vom Server initiiert wird

PING

Der Server sendet zum Client ‚`PING\n`‘ und erhält als Antwort ‚`PONG\n`‘.

7.1.17 Kommunikation zwischen den Clients

GET

Format:

`GET` *`FILENAME`* **[***`USERNAME`***]**

Parameter:

Parametername	Beschreibung
`FILENAME`	Datei die downzuloaden ist
`USERNAME`^{optional}	Benutzername der Person die die Datei downloaden will, fällt weg falls der Benutzer anonym ist

Mögliche Status-Codes der folgenden ACK-Message:

Status-Code	Beschreibung
FILE_NOT_FOUND	Die Datei konnte nicht im shared Verzeichnis gefunden werden
TOO_MANY_USERS	Zu viele konkurrierende Benutzer beim Download

Retourniert:

Im Normalfall einen Header und dann die angeforderte Datei: `content-length: xxx bytes\n byte{byte}`

Im Fehlerfall ‚`ACK STATUS-CODE\n`', mit dem Status-Code des Fehlers

7.1.18 DTD[9] für profile

profile.dtd:

```
<!ELEMENT profile (shared_dir, connection, max_upload, max_download)>
<!ELEMENT shared_dir (#PCDATA)>
<!ELEMENT connection (#PCDATA)>
<!ELEMENT max_upload (#PCDATA)>
<!ELEMENT max_download (#PCDATA)>
```

7.1.19 DTD für share

share.dtd:

[9] DTD bedeutet Document Type Definition und ist eine Definition des Aufbaus eines XML Dokuments

```
<!ELEMENT share (item+)>

<!ELEMENT item (title?, creator?, subject?, description?,
type?, date, size)>

<!ATTLIST item filename CDATA #REQUIRED>

<!ELEMENT title (#PCDATA)>

<!ELEMENT creator (#PCDATA)>

<!ELEMENT subject (#PCDATA)>

<!ELEMENT description (#PCDATA)>

<!ELEMENT type (#PCDATA)>

<!ELEMENT date (#PCDATA)>

<!ELEMENT size (#PCDATA)>
```

7.1.20 DTD für search

search.dtd:

```
<!ELEMENT search (filename?, title?, creator?, subject?,
description?, type?, date?, size?)>

<!ELEMENT filename (#PCDATA)>

<!ELEMENT title (#PCDATA)>

<!ELEMENT creator (#PCDATA)>

<!ELEMENT subject (#PCDATA)>
```

```
<!ELEMENT description (#PCDATA)>

<!ELEMENT type (#PCDATA)>

<!ELEMENT date (#PCDATA)>

<!ELEMENT size (#PCDATA)>
```

7.1.21 DTD für found_items

found_items.dtd:

```
<!ELEMENT found_items (item+)>

<!ELEMENT item (user_desc?, title?, creator?, subject?, de-
scription?, type?, date, size)>


<!ATTLIST item filename CDATA #REQUIRED

 ip                          CDATA             #REQUIRED>

<!ELEMENT user_desc (username, connection, download_count,
upload_count, rating)>


<!ELEMENT username (#PCDATA)>

<!ELEMENT connection (#PCDATA)>

<!ELEMENT download_count (#PCDATA)>

<!ELEMENT upload_count (#PCDATA)>

<!ELEMENT rating (#PCDATA)>
```

```
<!ELEMENT title (#PCDATA)>

<!ELEMENT creator (#PCDATA)>

<!ELEMENT subject (#PCDATA)>

<!ELEMENT description (#PCDATA)>

<!ELEMENT type (#PCDATA)>

<!ELEMENT date (#PCDATA)>

<!ELEMENT size (#PCDATA)>
```

7.1.22 DTD für whois

whois.dtd:

```
<!ELEMENT whois (connection, upload_count, download_count,
rating)>

<!ELEMENT connection (#PCDATA)>

<!ELEMENT upload_count (#PCDATA)>

<!ELEMENT download_count (#PCDATA)>

<!ELEMENT rating (#PCDATA)>
```

7.1.23 Übersicht der Requests die anonymen Benutzern nicht zur Verfügung stehen

CHANGE_PWD

GET_PROFILES

GET_PROFILE

PUT_PROFILE

DEL_PROFILE

RATE

Übersicht der Status-Codes und deren Verursacher

Status-Code	Beschreibung	Requests
ALREADY_LOGGED_IN	Es ist schon jemand unter dem angegebenen Account angemeldet	LOGIN
FILE_NOT_FOUND	Die Datei konnte nicht im shared Verzeichnis gefunden werden.	GET
INVALID_DOC	Es wurde ein ungültiges XML-Dokument empfangen, d.h. das XML Dokument war nicht valide, oder Werte die im Dokument definiert wurden (Attribute, Elemente) sind zu lang oder in einem ungültigen Format	PUT_PROFILE SHARE SEARCH

	(nicht numerisch, nicht in einem erlaubten Bereich usw.)	
INVALID_PARAM	Es fehlt ein Parameter bzw. ein Parameterwert ist nicht innerhalb der erlaubten Wertemenge	alle (außer PING)
MAX_ANONYMOUS_SESSIONS	Die Maximalanzahl an anonymen Benutzern wurde erreicht, kann nur beim anonymen Login auftreten	LOGIN (**nur beim anonymen Login**)
MAX_FILE_COUNT	Die Maximalanzahl an Dateien die vom System gespeichert werden wurde erreicht	SHARE
MAX_PROFILE_COUNT	Es wurde die Maximalanzahl an Profilen für den angegebenen Benutzer erreicht	PUT_PROFILE
MAX_USER_COUNT	Die Maximalanzahl an registrierten Benutzern wurde erreicht	REGISTER
MAX_USER_SESSIONS	Die Maximalanzahl an nicht-anonymen Benutzern wurde erreicht, kann nur beim nicht-anonymen Login auftreten	LOGIN (**nur beim nicht-anonymen Login**)
NO_MATCH_FOUND	Es wurden keine zu dem Suchmuster passenden shared Items gefunden	SEARCH
NO_PROFILES_FOUND	Für den durch die Session ID identifizierten Benutzer existieren keine Profile auf den Server	GET_PROFILES GET_PROFILE DEL_PROFILE
NO_UNIQUE_NAME	Es wurde bei der Registrierung	REGISTER

	ein Benutzername beantragt, der nicht eindeutig ist	
OK	Operation erfolgreich durchgeführt, kann nur bei Requests auftreten die keine Daten zurückliefern	REGISTER LOGOUT PUT_PROFILE DEL_PROFILE SHARE RATE
PASSWD_NOT_ACCEPTED	Neues Paßwort wird vom System nicht akzeptiert (z.B. zu kurz)	CHANGE_PWD
PROFILE_NOT_FOUND	Es existieren Profile am Server, aber das Angegebene nicht	GET_PROFILE DEL_PROFILE
SERVER_ERROR	Am Server ist ein Fehler aufgetreten (z.B.: DB konnte nicht erreicht werden, eine SQL-Exception oder sonstiges)	REGISTER LOGIN LOGOUT GET_PROFILES GET_PROFILE PUT_PROFILE DEL_PROFILE SHARE SEARCH WHOIS RATE

SESSION_NOT_FOUND	Der durch die Session ID identifizierte Benutzer ist schon vom System abgemeldet oder er war nie angemeldet	LOGOUT GET_PROFILES GET_PROFILE PUT_PROFILE DEL_PROFILE SHARE SEARCH WHOIS RATE
TOO_MANY_USERS	Zuviele konkurrierende Benutzer beim Download	GET
UNKNOWN_REQUEST	Es existiert kein Request mit diesem Namen	alle (außer PING)
USER_IS_ANONYMOUS	Ein anonymer Benutzer versuchte Operation aufzurufen die ihm nicht zur Verfügung steht	CHANGE_PWD GET_PROFILES GET_PROFILE PUT_PROFILE DEL_PROFILE RATE
USER_NOT_FOUND	Der angegebene Benutzername konnte in der Benutzer-DB nicht gefunden werden	LOGIN WHOIS RATE
WRONG_PASSWD	Das angegebene Paßwort	LOGIN

	stimmt nicht mit dem in der Benutzer-DB angegebenen überein	

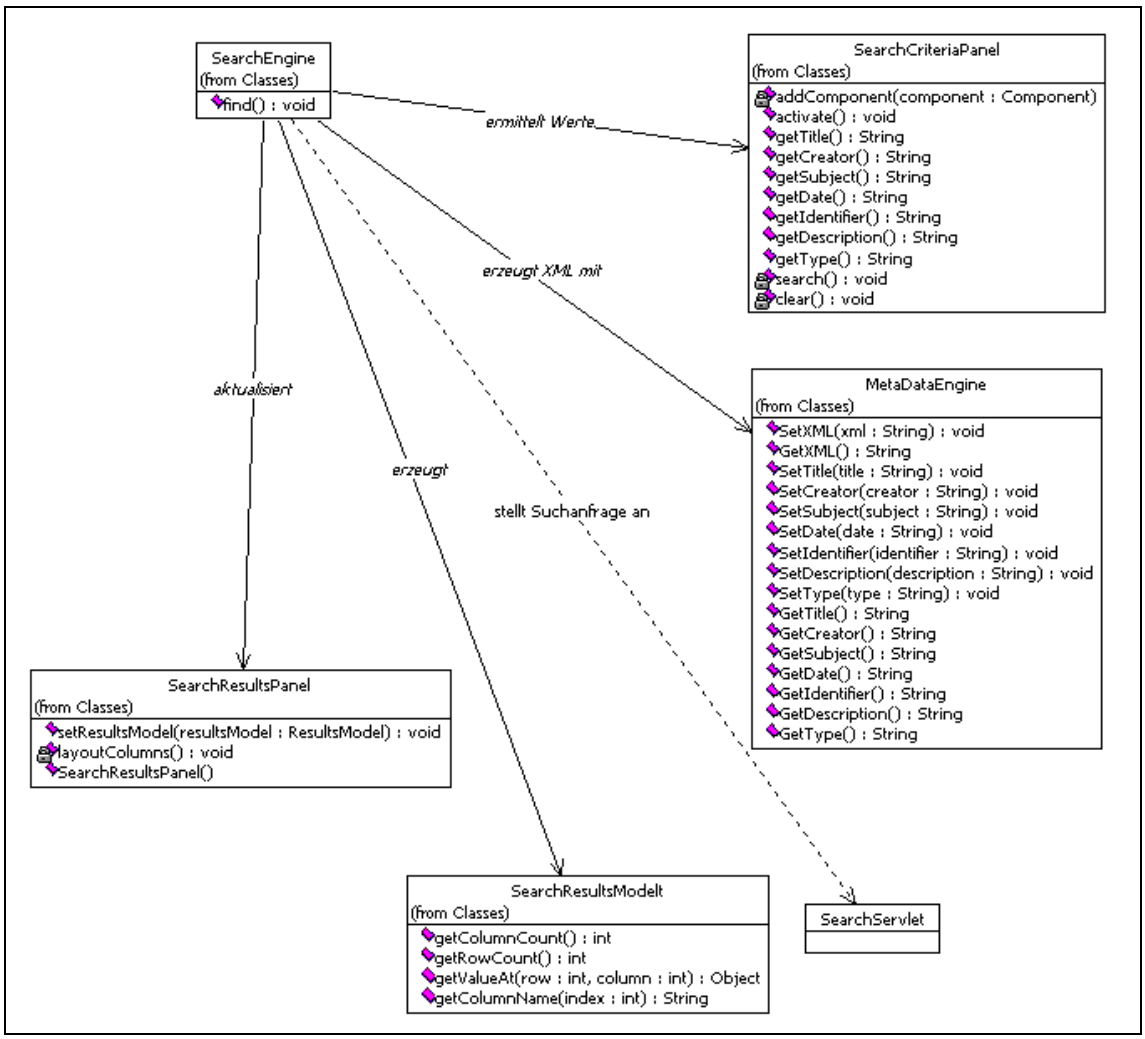
7.1.24 IDs zu den Status-Codes

Status-Code	**ID**
ALREADY_LOGGED_IN	1
FILE_NOT_FOUND	2
INVALID_DOC	3
INVALID_PARAM	4
MAX_ANONYMOUS_SESSIONS	5
MAX_FILE_COUNT	6
MAX_PROFILE_COUNT	7
MAX_USER_COUNT	8
MAX_USER_SESSIONS	9
NO_MATCH_FOUND	10
NO_PROFILES_FOUND	11
NO_UNIQUE_NAME	12
OK	**0**
PASSWD_NOT_ACCEPTED	13

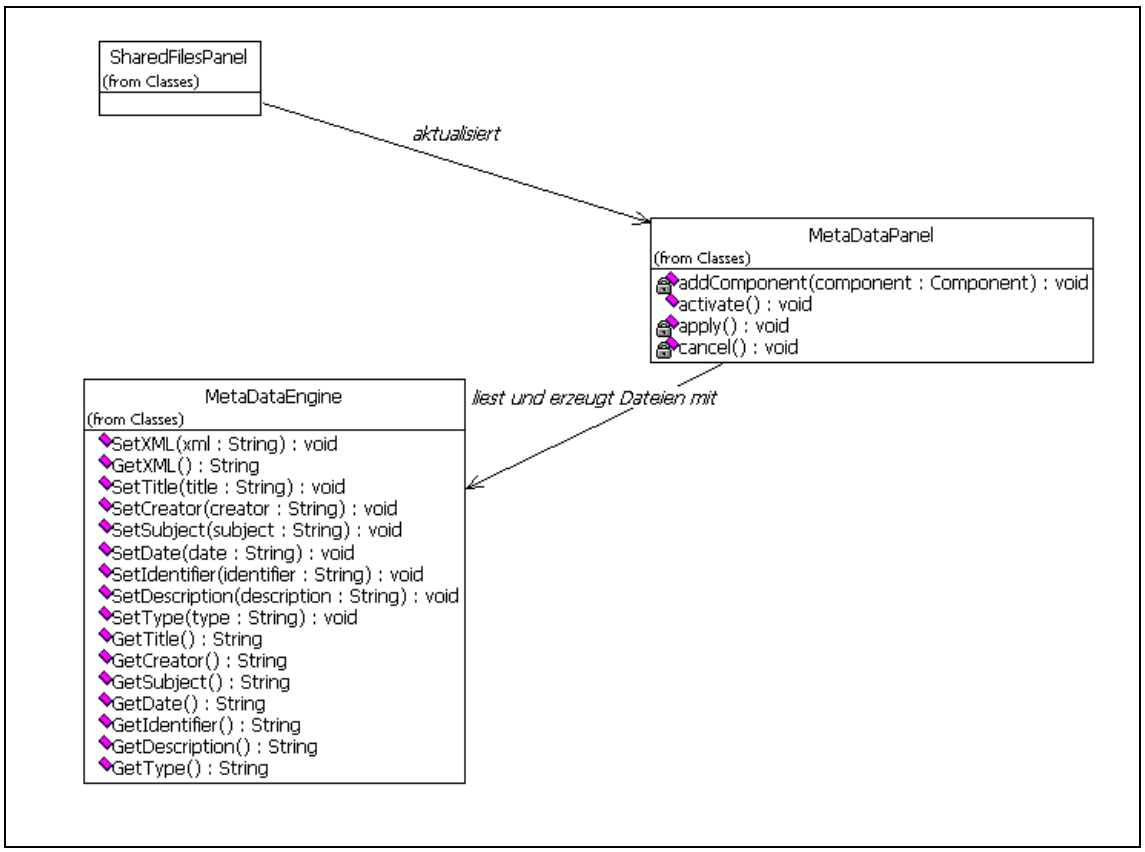
PROFILE_NOT_FOUND	14
SERVER_ERROR	15
SESSION_NOT_FOUND	16
TOO_MANY_USERS	17
UNKNOWN_REQUEST	18
USER_IS_ANONYMOUS	19
USER_NOT_FOUND	20
WRONG_PASSWD	21

7.2 Klassendiagramme – Client

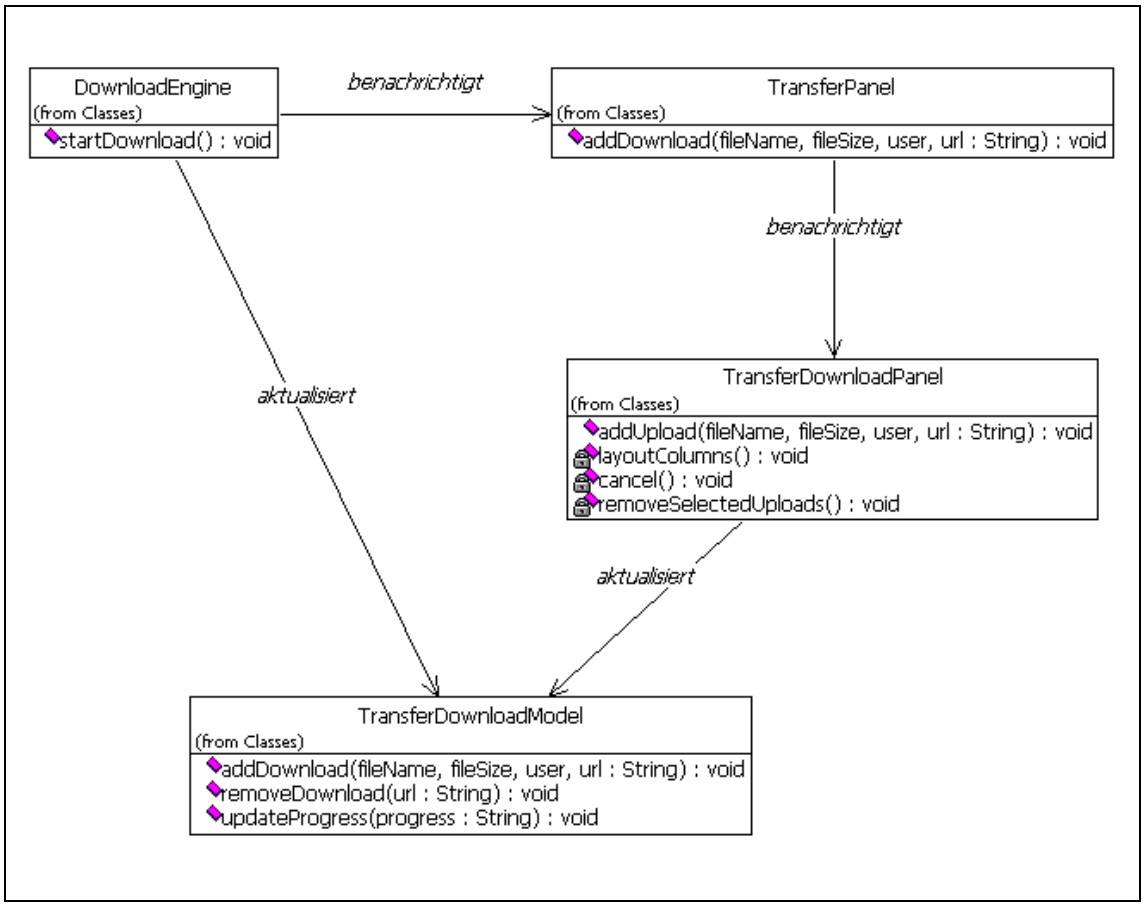
7.2.1 Funktionalität Search



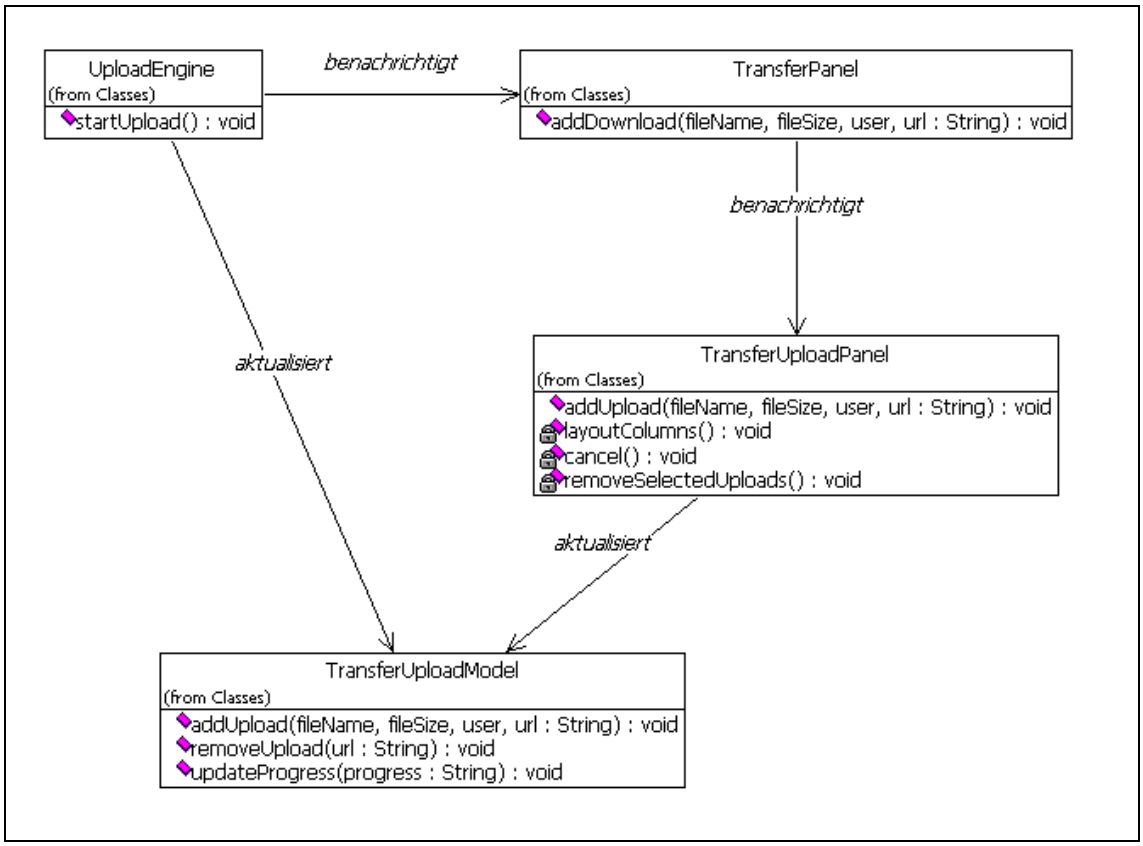
7.2.2 Funktionalität Share



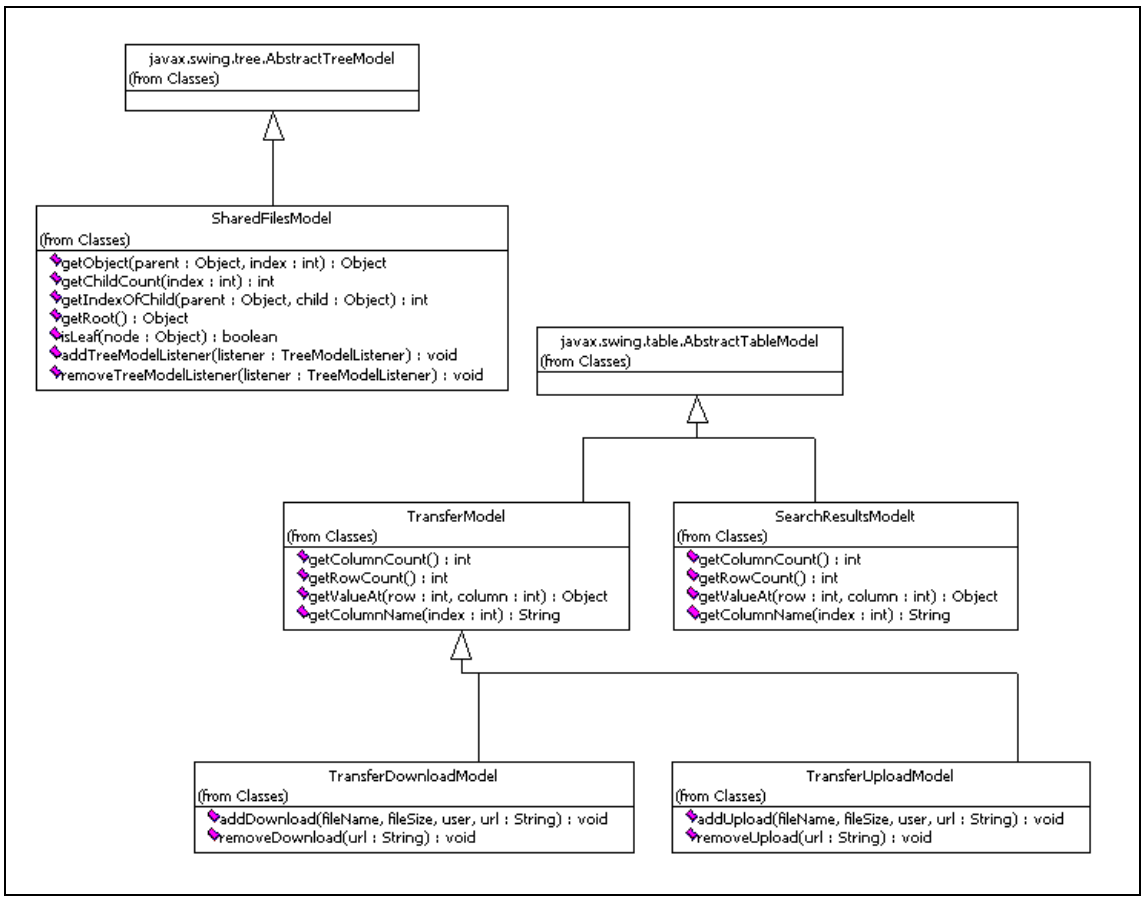
7.2.3 Funktionalität Download



7.2.4 Funktionalität Upload



7.2.5 Funktionalität Vererbung



7.3 Japster Server – Grob Entwurf V 1.02

7.3.1 Dokumentenhistorie:

Version	Datum	Beschreibung	Autor
1.00	06.12.2000	Erstellung	Christoph Bäck
1.01	19.12.2000	Diverse Änderungen (Mutex für DB Zugriff, Schnittstellen, Parametrisierung des Systems usw.)	Christoph Bäck
1.02	27.12.2000	Parameter hinzugefügt und korrigiert	Christoph Bäck

7.3.2 Zusammenhang mit anderen Dokumenten

Dieses Dokument basiert auf der **Japster Protokoll Spezifikation V 1.13**, falls sich die zugrundeliegende Spezifikation ändert, ist auch dieses Dokument zu prüfen und nötigenfalls zu ändern.

7.3.3 Projektumfeld

Es wurde entschieden den Server als Java Servlet zu betreiben, die Zielmaschine ist ein Linux Rechner mit der Servlet Engine JRun 3.0 von Allaire, als Datenhaltungssystem wird eine MySQL Datenbank eingesetzt. Das Servlet verwendet als XML Parser die Java Extension JAXP (Version 1.1 Early Access) von Sun, des weiteren wird ein SMTP Server zum Versenden der E-Mails mit dem automatisch generierten Paßwort bei der Registrierung verwendet.

7.3.4 Package Struktur des Japster Server

Es wird zwischen 3 Packeten (außer dem Basisnamensraum) unterschieden:

protocol

db

debug

Das Package **protocol** beinhaltet alle Klassen die zur Abwicklung des Protokolls benötigt werden, das Package **db** stellt den Persistenzmechanismus der Applikation dar und schließlich sind Debughilfen im Package **debug** vorhanden.

7.3.4.1 Package protocol

In diesem Package befinden sich Klassen die das Protokoll kapseln, ein jeder zu bearbeitender Request muss vom Interface *Request* abgeleitet worden sein. In diesem Interface ist eine abstrakte Methode *doRequest(...)* definiert, die dann in der Applikation angesprungen wird, und für den Ablauf der Requests zuständig ist. Um die Persistenzmechanismen zu nutzen, erhält diese Methode als Parameter einen *SessionManager*, *UserManager* und einen *SharedItem-Manager*.

Um das PING durchzuführen, existiert eine Klasse *PingThread* die nebenläufig zur Applikation alle überfälligen Clients anpingt, und falls keine Antwort erfolgt die zur Session gehörenden shared Items löscht, und danach auch die entsprechende Session. Da I/O in Java eine blockierende Operation ist, findet die Kommunikation mittels eines eigenen Threads namens *PingIOThread* statt.

7.3.4.2 Package db

Beinhaltet Klassen die die Persistenz der Objekte *User*, *Session*, *Profile* und *SharedItem* managen.

Es existiert ein Interface *SharedItemManager*, *UserManager* und die abstrakte Klasse *SessionManager;* per Konvention tragen die konkreten Klassen, die von

diesen Managern abgeleitet wurden, den Namen **dbPräfix...Manager** d.h. dass bezogen auf die Zielmaschine alle Manager Klassen das dbPräfix MySQL haben. Mittels des dbPräfix soll das dynamische wählen der Ziel-Datenbank ohne Neukompilation ermöglicht werden.

Da das Instantiieren von Datenbank-Connection in Java ein Performance Hindernis darstellen kann, werden die Connections von einem sogenannten Connection Pool gemanagt, in diesem Pool werden bei Initiierung der Applikation eine voreingestellt Anzahl an Connections generiert, die dann von anderen Objekten genutzt werden können, desweiteren können noch zusätzlich Connections im Connection Pool bis zu eine Maximalanzahl angelegt und genutzt werden.

Um zu garantieren, dass die einzelnen Methoden als eine Transaktion ungestört von anderen Methoden einer Manager Klasse ablaufen, ist ein Mutex in Form einer Klasse namens *DBMutex* eingeführt worden.

7.3.4.3 Package debug

Dieses Package enthält eine Klasse *Debug* die Funktionalität wie assert(..) und trace(..) zur Verfügung stellt.

7.3.4.4 Die Klassen im Basisnamensraum

Das Servlet *Japster* implementiert einen *RequestDispatcher* und initialisiert, dann das System mit *GlobalConfig*, d.h. es werden alle Parameter eingelesen,

überprüft und dann in der System Konfiguration gesetzt, danach werden an Hand der Parameter folgende Objekte bzw. davon abgeleitete Konkrete instantiiert:

UserManager -> siehe Parameter **db_prefix**

SessionManager -> siehe Parameter **db_prefix**

SharedItemManager -> siehe Parameter **db_prefix**

Anschließend werden die zu verarbeitenden Requests extrahiert, und die dazugehörigen Klassen instantiiert, und schließlich mittels *addRequest(..)* registriert.

Danach wird *DBConnectionPool* instantiiert, abschließend wird dann ein *PingThread* gestartet der das Überprüfen von Timeouts bei Benutzern übernimmt.

Wenn das Servlet nun einen Request ausführen soll, so ruft es die Methode *dispatchRequest(..)* mit dem Request Type auf, um die entsprechende Request Implementierung anzuspringen, existiert der Request nicht wird ein ‚`ACK UNKNOWN_REQUEST`‘ zurückgesendet.

Das System wird mittels *initArgs* der Servlet Engine parametrisiert.

7.3.5 Parametrisierung des Systems

Name	Typ	Beschreibung
db_name	Zeichenkette	DB-Name (muss vollständig sein, d.h. z.B. jdbc:mysql://host/db)
db_passwd	Zeichenkette	Paßwort für die DB **(db_user und db_passwd sind zusammen optional)**
db_prefix	Zeichenkette	Präfix für die Manager Klassen die zu instantiieren sind, d.h. wenn MySQL angegeben ist, wird versucht u.a. *MySQLUserManager* im Package **db** zu instantiieren. **Klassen für die Manager werden immer Package db gesucht!**
db_user	Zeichenkette	Username für die DB **(db_user und db_passwd sind zusammen optional)**
debug	Zeichenkette	Debugmodus aktiviert oder nicht, kann den Wert „true“ oder „false“ haben (ist nicht case-sensitive)
init_connections	Numerisch (signed 32-Bit Integer)	Initiale Anzahl an Connections die der Connection Pool beinhalten soll (muss größer 0 sein)
japster_email	Zeichenkette	E-Mail Adresse, die als Absender der bei der Registrierung generierten Mail mit dem Paßwort für den User aufscheint
japster_port	Numerisch (unsigned 16-Bit Integer)	Auf welchen Port der Japster Client angepingt werden soll (muss größer 0 sein, und ein gültiger Port d.h. im Bereich eines unsigned 16-Bit Shorts)
jdbc_driver	Zeichenkette	Name des zu verwendenden JDBC-Treiberes

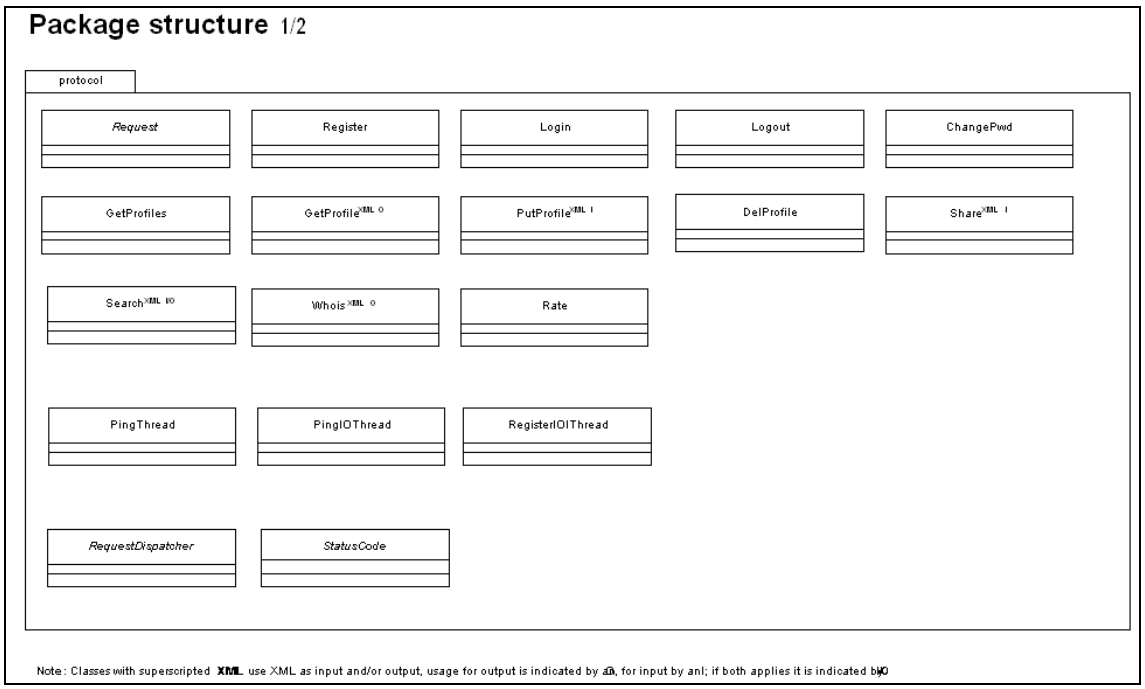
max_anonymous_sessions	Numerisch (signed 32-Bit Integer)	Maximalanzahl an Sessions von anonymen Benutzern (muss größer 0 sein)
max_connections	Numerisch (signed 32-Bit Integer)	Maximale Anzahl an Connections die der Connection Pool beinhalten und managen soll (muss größer 0 und größer gleich init_connections sein)
max_file_count	Numerisch (signed 32-Bit Integer)	Maximalanzahl an Dateien die vom System gehandhabt werden (global) (muss größer 0 sein)
max_length_connection	Numerisch (signed 32-Bit Integer)	Maximallänge der Zeichenkette die den Type der Verbindung angibt (muss größer 0 sein)
max_length_creator	Numerisch (signed 32-Bit Integer)	Maximallänge des Creators eines shared Items (muss größer 0 sein)
max_length_description	Numerisch (signed 32-Bit Integer)	Maximallänge des Beschreibung (Whitespace-separierte Keywords) eines shared Items (muss größer 0 sein)
max_length_filename	Numerisch (signed 32-Bit Integer)	Maximallänge des Dateinamens eines shared Items (muss größer 0 sein)
max_length_profilename	Numerisch (signed 32-Bit Integer)	Maximallänge einer Profilbezeichnung (muss größer 0 sein)
max_length_shared_dir	Numerisch (signed 32-Bit Integer)	Maximallänge des Verzeichnis für die shared Items (muss größer 0 sein)
max_length_subject	Numerisch (signed 32-Bit Integer)	Maximallänge des Subject eines shared Items (muss größer 0 sein)
max_length_title	Numerisch (signed 32-Bit Integer)	Maximallänge des Titels eines shared Items (muss größer 0 sein)

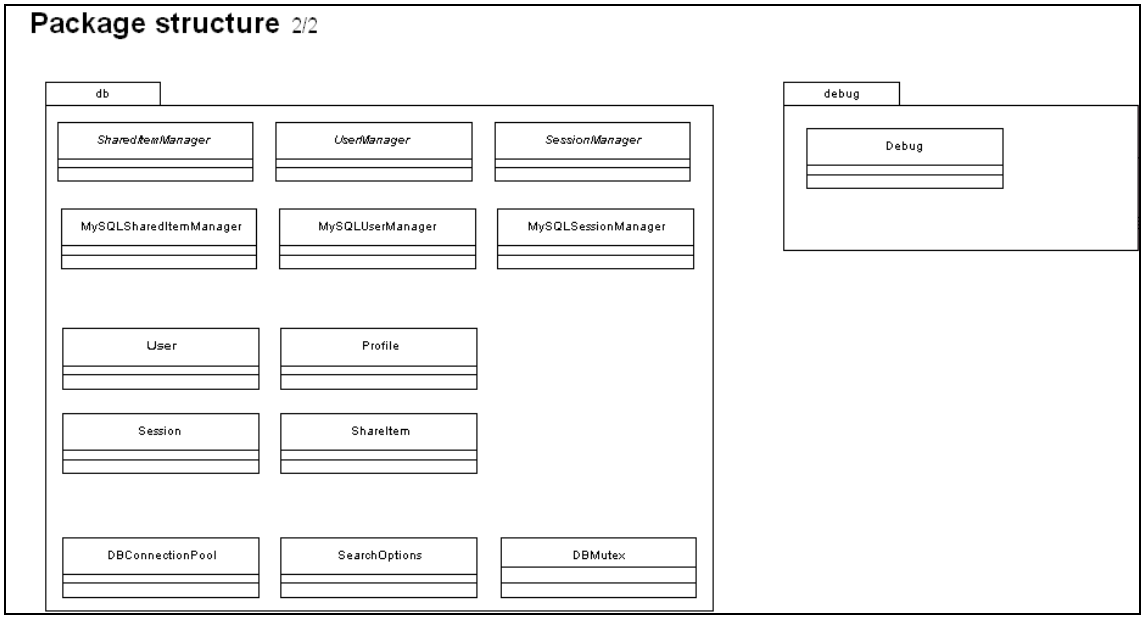
max_length_type	Numerisch (signed 32-Bit Integer)	Maximallänge des Types (MIME-Type) eines shared Items (muss größer 0 sein)
max_length_username	Numerisch (signed 32-Bit Integer)	Maximallänge des Benutzernamens (ist seine E-Mail Adresse) (muss größer 0 sein)
max_passwd_length	Numerisch (signed 32-Bit Integer)	Maximale Länge eines Paßwortes (muss größer 0 und größer gleich min_passwd_length sein)
max_profile_count	Numerisch (signed 32-Bit Integer)	Maximalanzahl an Profilen für einen User (muss größer 0 sein)
max_user_count	Numerisch (signed 32-Bit Integer)	Maximalanzahl an registrierten Usern für die DB (muss größer 0 sein)
max_user_sessions	Numerisch (signed 32-Bit Integer)	Maximalanzahl an Sessions von registrierten Benutzern (muss größer 0 sein)
min_passwd_length	Numerisch (signed 32-Bit Integer)	Minimale Länge eines Paßwortes (muss größer 0 sein)
ping_session_timeout	Numerisch (signed 32-Bit Integer)	Ab welchen Zeitraum der Überfälligkeit in Sekunden ein PING durchgeführt werden soll (muss größer 0 sein)
ping_sleep	Numerisch (signed 32-Bit Integer)	Wieviele Millisekunden der PING-Thread schlafen soll, bevor er wieder nach überfälligen Sessions sucht und diese anpingt (muss größer 0 sein)
request_**...**_class	Zeichenkette	Für eine jede Request die in request_types definiert wurde existiert so ein Parameter der die zu registrierende Klasse zu dem Request angibt. **(Die Requests-Types werden immer als Kleingeschrieben gehandhabt, auch wenn sie in Großbuchstaben angegeben wurden)**

		Z.B.: request_login_class=Login request_logout_class=Logout **Klassen für die Request Types werden immer im Package protocol gesucht!**
request_types	Zeichenkette	Beschreibt die Requests die die Applikation verarbeiten kann, die einzelnen Requests sind durch Whitespace getrennt. **(Die Requests werden immer als Kleingeschrieben gehandhabt, auch wenn sie in Großbuchstaben angegeben wurden)** Z.B.: login logout share
session_id_length	Numerisch (signed 32-Bit Integer)	Länge der zu generierenden Session ID (muss größer 0 sein)
smtp_port	Numerisch (unsigned 16-Bit Integer)	Port auf den der SMTP-Server Verbindungen entgegen nimmt (muss größer 0 sein, und ein gültiger Port d.h. im Bereich eines unsigned 16-Bit Short)
smtp_server	Zeichenkette	Adresse des SMTP-Servers zum Versenden des Paßworts nach REGISTER
trace_file	Zeichenkette	Trace Datei, wird nur im Debug Modus benötigt
url_dtd	Zeichenkette	URL der DTDs für die XML-Dokumente, um diese validieren zu können. **Die URL muss immer in der Form: http://host/ sein!**

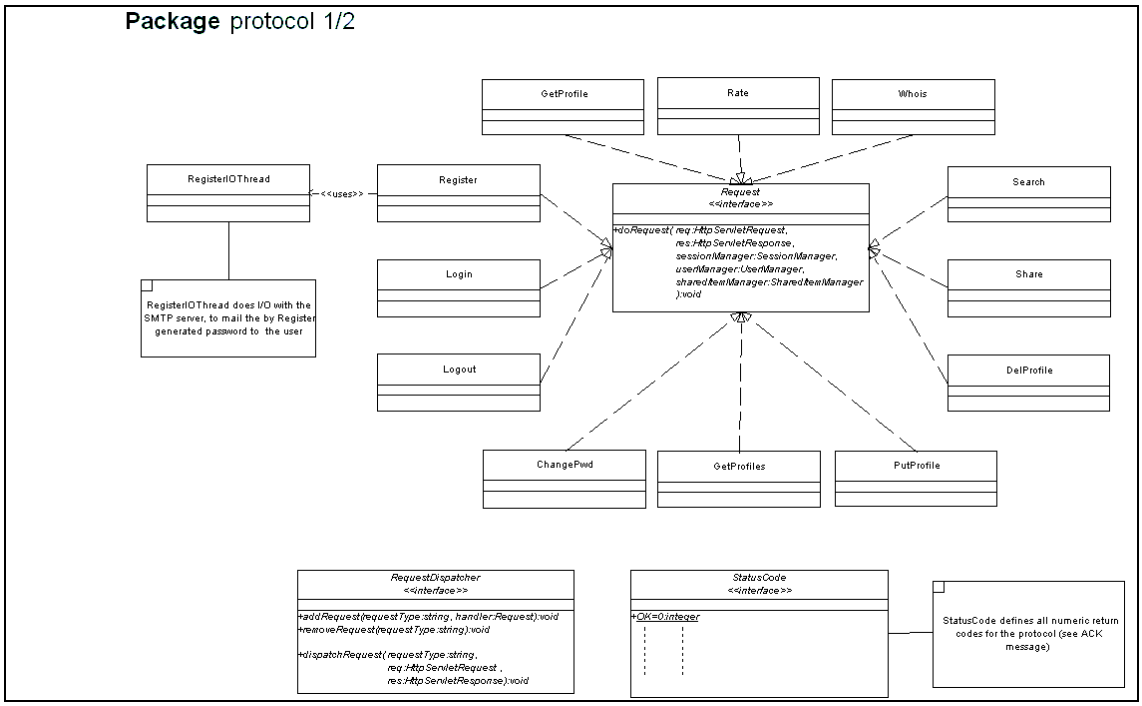
7.4 Klassendiagramme Server

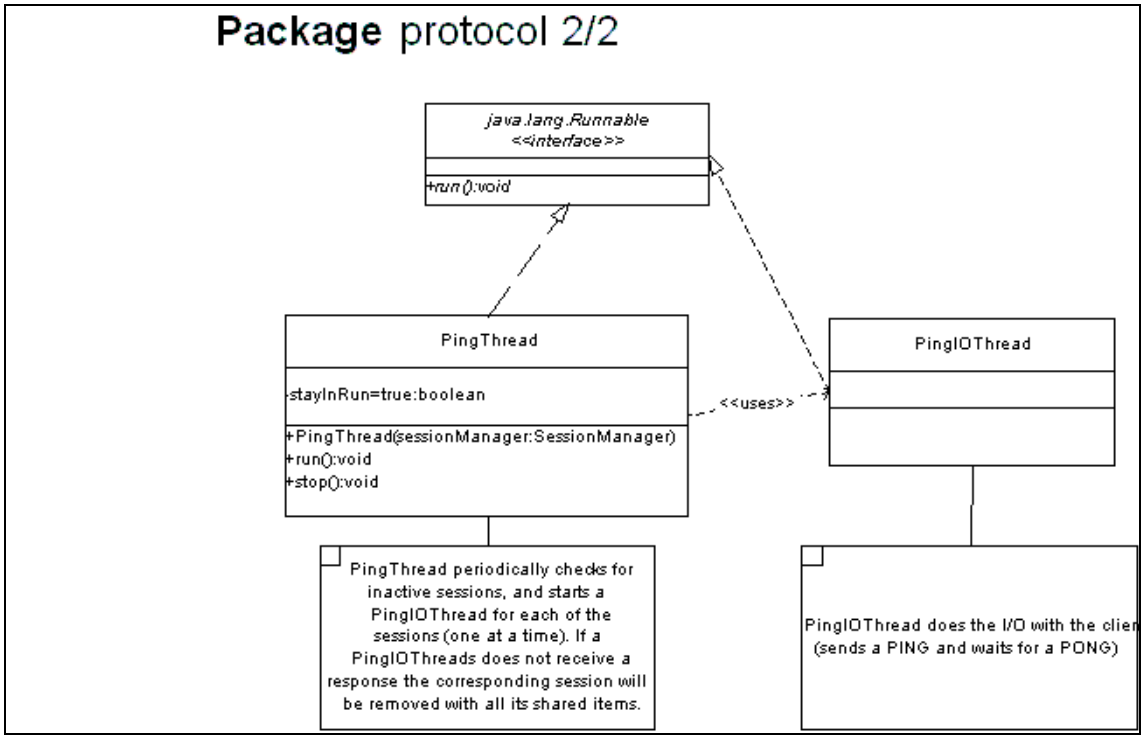
7.4.1 Package Structure



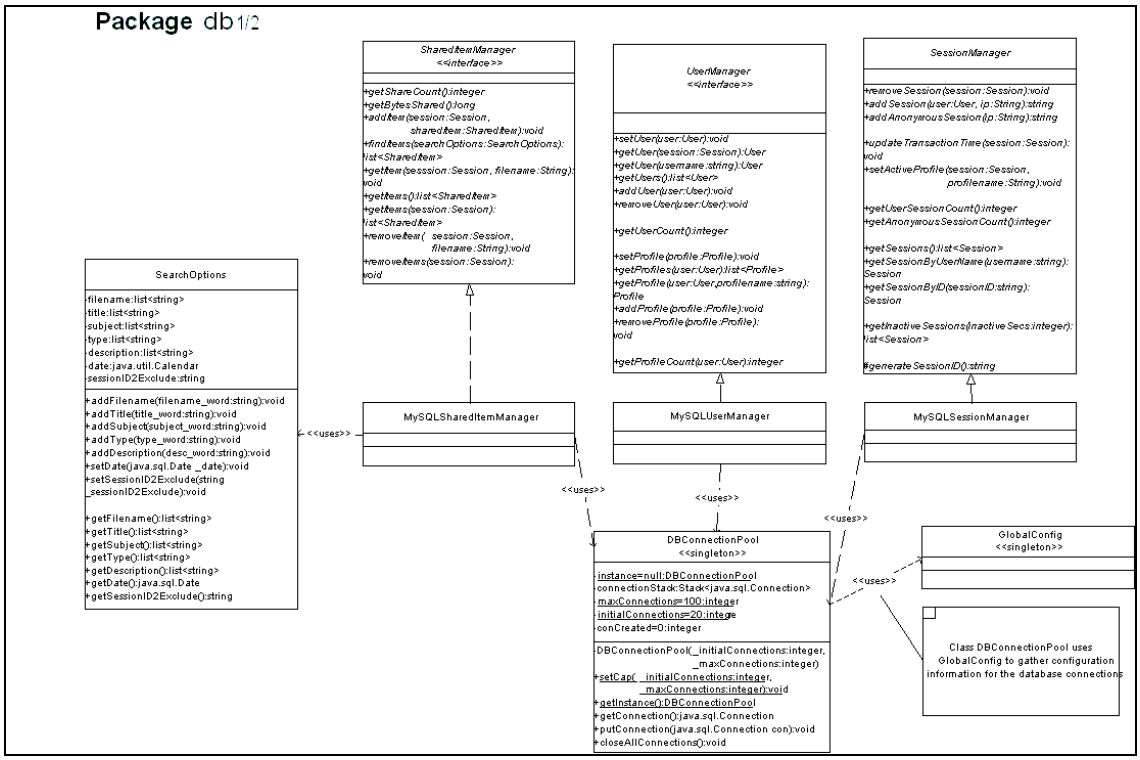


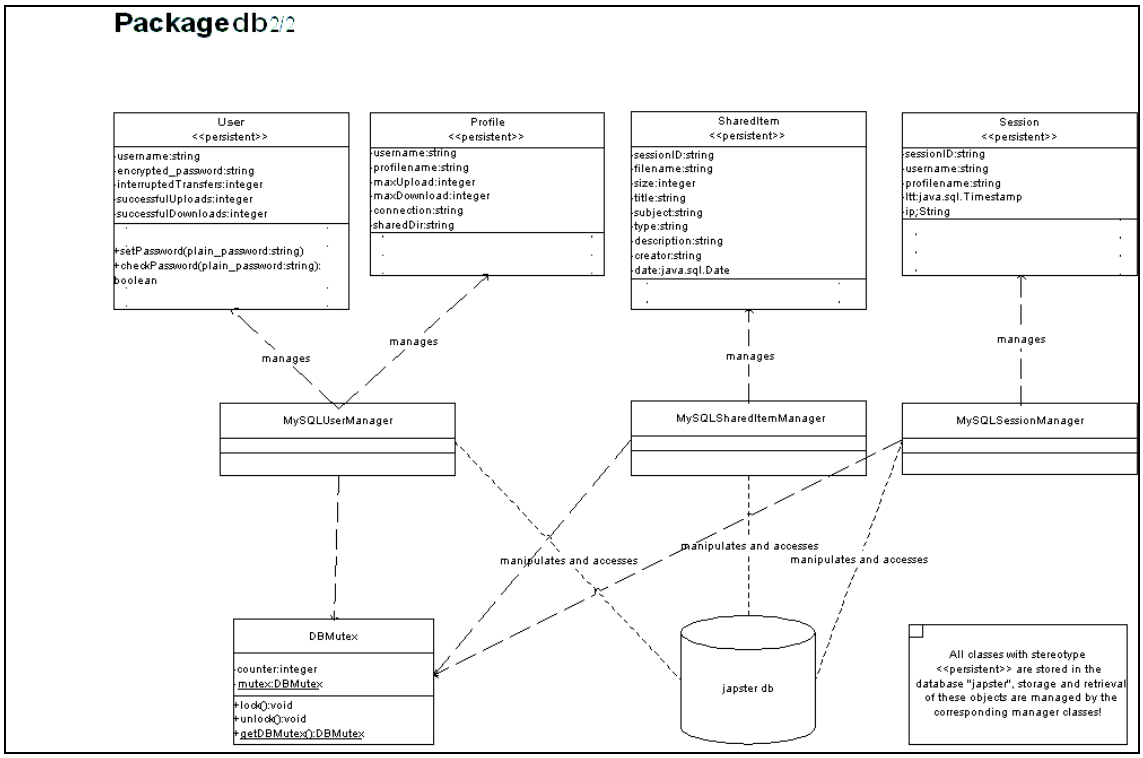
7.4.2 Package protocol



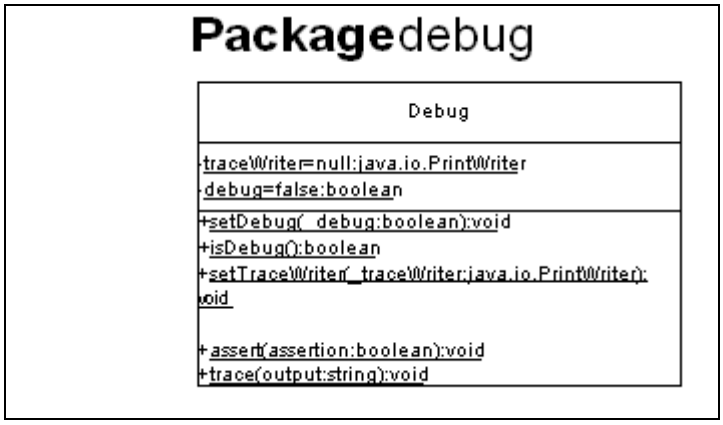


7.4.3 Package db





7.4.4 Package debug



7.4.5 Basisnamensraum