DIPLOMARBEIT

Datenbankbasiertes Informationssystem für ein Universitätsinstitut auf der Basis von WWW unter Verwendung von MySQL und PHP4

ausgeführt zum Zwecke der Erlangung des akademischen Grades eines Diplom-Ingenieurs unter der Leitung von

Ao.Univ.Prof. Dipl.-Ing. Dr.techn. Karl Riedling

E 366

Institut für Industrielle Elektronik und Materialwissenschaften

eingereicht an der Technischen Universität Wien
Fakultät für Elektrotechnik

Von

Vladimir Nisevic
Matrikelnummer 9127493
Jurekgasse 4/16, A-1150 Wien

Wien, im Juni 2001

Alles sollte so einfach wie
möglich gemacht werden.
Aber nicht einfacher.

Albert Einstein

Kurzfassung

IDM (steht für „Internet-based Document Management“) ist eine Web-orientierte Dokumentenmanagement-Applikation, entwickelt an der TU Wien, am Institut für Industrielle Elektronik und Materialwissenschaften.
Die Applikation basiert auf dem Internet als Kommunikationsmedium und nutzt daher die gängigen Standardbrowser als Client. Dadurch ist eine clientseitige Plattform-Unabhängigkeit gegeben.
Auf der Serverseite befinden sich eine relationale Datenbank (MySql), ein Webserver (Apache), und ein PHP Interpreter. Als Betriebssystem wurde die bekannteste Open-Source Software, Linux, eingesetzt.
Die gesamte Implementierung basiert daher auf Open-Source Software.
Im Rahmen der Applikationsentwicklung ist ein neues Framework in PHP entstanden, dessen Hauptmerkmal ist, daß es vollständig objektorientiert ist.
Das Framework ermöglicht eine Trennung der Aufgaben zwischen den Applikations-Entwicklern und Web-Designern, sowie ein modulares Design von Seiten. Weitere Merkmale des Frameworks sind sehr einfache Wartbarkeit und Erweiterbarkeit.
Die Tabellen der relationalen Datenbank sind ebenfalls durch entsprechende Klassen in PHP abstrahiert.

Summary

Today, the World Wide Web is doubtlessly the standard for electronically providing of easily accessible information to a general public.
Much more efficient than the standard way to publish information, by many HTML pages, is a server-side application with an relational database as a backend.
This allows a centralized data storing, ease of use and maintenance, and many other benefits.

IDM is a web-based document management application developed at the Institute of Industrial Electronics and Material Science at the Vienna University of Technology.
The application has been developed in PHP, and is based on an object oriented framework which has been developed in the same diploma thesis.
The main features of this framework are ease of use, high level of modularity, and distinction between PHP code and HTML pages. This permits application developers and web page designers to work independently on their part of code.
The application is using Internet as a communication medium, and a standard browser as a client.
This allows to use it on almost any computer and operating system respectively.

Inhaltsverzeichnis

1 Einleitung

In der heutigen Zeit ist das World Wide Web zweifelsohne der Standard für die Bereitstellung der Informationen für ein möglichst breites Publikum. Um den Benutzern eine möglichst komfortable Benutzerschnittstelle zur Verfügung zu stellen, muß man sich einige zusätzliche Fragen stellen, wie zum Beispiel: Für wen ist eine bestimmte Information relevant? Wie hoch ist die Relevanz? Und wann hört sie auf, relevant zu sein?

Hier ist der Einsatz von Informationsmanagementsystemen angesagt.

Was ist ein Informationssystem?
Ein Informationssystem ist ein Anwendungsprogramm, das die Bereitstellung von Informationen aus Dateien und die Ablage von Informationen in Form von Dateien ermöglicht. Ein Informationssystem unterstützt also die Vorgänge der Interpretation von Dateien und der Repräsentation von Informationen.

Übersicht der einzelnen Kapitel
Im nächsten Kapitel „Motivation und Zielsetzung“ werden die grundlegende Anregungen dargestellt, wie: Warum man überhaupt ein Informationssystem braucht, und welche Funktionen es erfüllen soll.
Im Kapitel „Begriffe“ werden die eingesetzte Software und ihre Entstehung vorgestellt.
Im Kapitel „Aspekte beim Entwurf einer Web-Anwendung“ sind die spezifische Methoden und ihre Anwendung im Hinblick auf den Entwicklungsprozeß einer Web-Applikation dargelegt.
Im Kapitel „Entwicklungsprozeß“ ist der vorliegenden Arbeit beschrittene Weg der Entwicklung eines objektorientierten Frameworks und einer darauf basierenden Beispielanwendung, eines Web-basierten Dokumentenmanagementsystems, dargestellt.
Im „Schlußbemerkungen und Ausblick“ sind weitere Möglichkeiten aus diesem Gebiet dargestellt worden.
Anschließend ist im Kapitel „Benutzerhandbuch“ auch ein Handbuch zu der entwickelten Applikation beigelegt.

2 Motivation und Zielsetzung

Ein Dokumentenverwaltungssystem ermöglicht einer beliebigen Anzahl von Clients, Dokumente jeglicher Art zu verwalten und anderen Benutzern zur Verfügung zu stellen. Auf Grund der rasanten Entwicklung des weltübergreifenden Netzwerks Internet, versucht man immer mehr Applikationen internetfähig zu machen.
Die Idee ist, über das standardisierte Browser-Interface eine Art Dokumentenverwaltung an den Benutzer zu bringen. Heutige netzwerkfähige Dateiverwaltungssysteme, wie Novell Netware oder Microsoft Netzwerk, haben einen entscheidenden Nachteil – eine unvollständige Plattform-Unabhängigkeit. Außerdem diese Systeme sind ja letztendlich nur netzwerkfähige Dateisysteme und besitzen daher keine Informationsmanagement-Aspekte. Weitere Nachteile eines Dateisystems sind eine feste Zuordnung der Dateinamen mit der Beschreibung der Datei, bzw. eine beschränkte Möglichkeiten bei der Benennung der Dateien.
Bei einer wachsenden Anzahl der Dateien in einem Ordner verliert man schnell die Übersicht, und das Auffinden einer bestimmten Datei wird immer schwieriger.

Ein Web-basiertes Informationssystem versucht diese Probleme durch den Einsatz von Techniken, wie zum Beispiel für jeden Benutzer angepaßte Datensicht und ständige Verfolgung seiner Schritte innerhalb des Informationssystems, zu lösen. Außerdem sollen die Benutzer durch eine aktive E-Mail-Benachrichtigung immer die neuesten Informationen erhalten, ohne sich an Informationssystem anmelden zu müssen.

Die Architektur eines solchen Systems besteht aus folgenden Komponenten: einem Web-Server, einer relationalen Datenbank, und einer serverseitigen Applikation, die den Clients eine Web-basierte Arbeitsumgebung zur Verfügung stellt.

Das Informationssystem sollte folgende Aspekte berücksichtigen:

2.1 Funktionalität

Gewünscht ist es, unterschiedlichste Dokumente, wie zum Beispiel Institutsprotokolle, Formulare oder Publikationen, bestimmten Benutzergruppen zur Verfügung zu stellen.

Am Informationsserver kann es Informationen in zwei Grundformen geben: entweder als eine Datei, die intern am Server gespeichert wird, oder als ein Hyperlink, das auf einen Server im Internet zeigt.

Diese Informationen werden in Ordnern, ähnlich wie bei einem Dateisystem, gehalten. Die Ordner können wiederum untergeordnete Ordner beinhalten. Hier besteht die Gefahr, durch die Komplexität die Benutzerfreundlichkeit zu verlieren. Im internen Berechtigungssystem werden die Benutzerrechte und sonstige Eigenschaften der neuangelegten Ordner standardmäßig von den übergeordneten Ordnern geerbt. Man soll jedoch jederzeit eine andere Einstellung vornehmen können.

Wenn eine neue Information auf den Server gestellt wurde, soll es die Möglichkeit geben, die berechtigten Benutzer per E-Mail darüber zu informieren. Diese Funktion soll periodisch von selbst ausgeführt werden.

Wenn sich der Benutzer entscheidet, eine Datei vom Server zu holen, wird ein neues Fenster geöffnet. Dadurch ist der Benutzer weiterhin im Informationssystem angemeldet, kann aber den angebotenen Informationseintrag separat betrachten. Nachdem ein Benutzer eine bestimmte Datei einmal vom Server geholt hat, merkt sich das das System, und ordnet diesem Informationseintrag und genau für diesen Benutzer eine niedrigere Priorität zu.

Der Benutzer soll nach Stichworten suchen können. Im Suchkriterium können sowohl die Namen von Ordnern als auch die Informationseinträge enthalten sein.

Ein Informationseintrag kann ein Ablaufdatum haben. Nach diesem Datum ist die Information nur bedingt sichtbar. Die Ordner können ebenfalls ein Standardablaufdatum haben, dieses ist aber nicht mit dem Ablaufdatum der Information gleichzusetzen. Vielmehr handelt es sich hier um ein Zeitintervall.
Wenn das Standardablaufdatum eines Ordners gegeben ist, wird das Ablaufdatum eines neuen Informationseintrags in diesem Ordner auf das Anlagedatum erhöht um dieses Zeitintervall gesetzt. Dieses geschieht nur, falls man beim Anlegen eines Informationseintrags kein Ablaufdatum angegeben hat.

Auf der Serverseite befindet sich ein Linux-Rechner mit einem Web Server (Apache) und einer relationalen SQL-Datenbank. Über Browser haben die Benutzer die Möglichkeit, verschiedene Dateien herunterzuladen oder auf den Server aufzuspielen.

Die Benutzer können einer oder mehreren Gruppen angehören. Die Ordner unterliegen einem bestimmten Zugriffsmechanismus, und zwar kann ein Ordner einer oder mehreren Gruppen zugeordnet werden. Dadurch sind die Rechte, zwecks einfacheren Wartung zusammengefaßt.
Man unterscheidet zwischen keinem Zugriff, einem Lesezugriff und einem Lese-/Schreibzugriff.
Jeder Benutzer wird durch seinen Benutzernamen und Passwort identifiziert. Außerdem kann man noch eine Email-Adresse des Benutzers angeben. Diese Email-Adresse wird bei einem Mail-Versand verwendet. Der Email-Versand soll ebenfalls auf der Benutzerebene einstellbar sein.

Die Applikation soll eine Authentifizierung unterstützen. Um dem Benutzer die ständige Anmeldung zu ersparen, soll es auch eine Cookie-Unterstützung geben. Nachdem aber der Benutzer die Cookie-Unterstützung ausschalten kann, soll die Applikation auch ohne Cookies problemlos funktionieren.

2.2 Benutzerfreundlichkeit / Ergonomie

Die Informationen auf dem Server müssen einfach, mit einem möglichst geringen Aufwand für den Benutzer und ohne spezielles Fachwissen, auffindbar und zugänglich sein.
Die Organisation der Informationen auf dem Server muss logisch und intuitiv erfassbar sein; das heißt, die angewandten Kriterien müssen (zumindest innerhalb der Benutzergruppe) als plausibel erachtet werden. Die Kriterien, die von einer speziellen Gruppe der Anwender - Administratoren - festgelegt werden, müssen flexibel definierbar sein.

2.3 Sicherheit

Der Zugriff auf das interne Informationssystem erfolgt grundsätzlich authentisiert, wobei einer Authentisierung über Benutzername und Passwort der Vorzug gegenüber Cookies zu geben ist, weil sie zuverlässiger und personen- statt maschinenbezogen funktioniert. Für jeden

Benutzer wird von der Backend-Datenbank protokolliert, auf welche Einträge bereits zugegriffen wurde; damit können neue oder seit dem letzten Zugriff geänderte Beiträge benutzerspezifisch als solche markiert werden.
Das Informationssystem soll ein Berechtigungssystem haben. Bestimmten Benutzergruppen sollen individuell eingeschränkte Informationen zugänglich sein.
Die Zugangsbeschränkung wird durch die Zugriffsrechte auf bestimmter Ordner, ähnlich wie bei den heutigen Dateisystemen, gelöst.

2.4 Plattform-Unabhängigkeit, Wartung

Der Zugriff auf die Daten und ihre Darstellung soll weitestgehend Plattform-unabhängig, also mit beliebiger Hardware und unter beliebiger Betriebssystems-Software, möglich sein. Die Installation zusätzlicher Softwarekomponenten auf den Client-Rechnern sollte unbedingt vermieden werden.
Die Administration des Informations-Servers, also die Erstellung und Änderung von Organisationsstrukturen oder das Einbringen neuer Informationen, soll einfach, mit einem Minimum an EDV-spezifischen Fachkenntnissen, und jedenfalls ohne Programmierkenntnisse möglich sein. Auch die Administration des Servers muß Plattform-unabhängig, via Web-Interface, möglich sein.
Anzustreben ist eine verteilte Administration der Informationen auf dem Server. Neue Informationen sollten von allen dazu berechtigten Benutzern auf den Server gestellt werden können; die Aufgaben eines zentralen Server-Administrators sollten sich im Wesentlichen auf die Wartung der Strukturen beschränken.

2.5 Technik

Eine flexible Organisation der Informationen auf dem Server und die einfache Wartung dieser Informationen setzt ein Datenbankkonzept für die Realisierung des Servers voraus.
Ein Datenbank-basierter Webserver bietet zudem zahlreiche Vorteile:

- Die Information ist zentral gespeichert
- Zugriffe auf Informationen können benutzerspezifisch registriert werden. Ein Benutzer kann daher gelesene und ungelesene Informationen in unterschiedlicher Form (z.B. mit unterschiedlichen optischen Markierungen) präsentiert bekommen
- Aktuelle Informationen können ohne zusätzlichen Wartungsaufwand genau für die Dauer ihrer Relevanz ausgegeben werden
- Informationen können personalisiert werden; jeder Benützer des Servers kann also vorzugsweise oder ausschließlich bestimmte, für ihn relevante, Informationen erhalten

Jede Information auf dem Server kann einem „Besitzer" zugeordnet werden, der allein berechtigt ist, sie in irgendeiner Form zu modifizieren.
Zusätzliche Dienste sind möglich, beispielsweise eine E-Mail-Benachrichtigung jener Benutzer, für die eine bestimmte Information gedacht ist, wenn diese Information neu erstellt oder geändert wurde.

2.6 Aussehen

Der Zugriff auf die Informationen auf dem Server ist über eine in mehreren Ebenen organisierte Menüstruktur möglich. Aus Übersichtlichkeitsgründen sollte sich die Anzahl der Menü-Ebenen möglichst in Grenzen halten; zwei Ebenen sollten das Optimum sowohl an Menü-Kapazität als auch an Übersichtlichkeit darstellen. Jeder Eintrag in einem Menü einer

Ebene führt entweder auf ein Menü der darunter liegenden Ebene oder auf eine Liste von Einträgen. Je nach Kontext können Einträge chronologisch, alphabetisch oder in einer frei definierten Anordnung sortiert angeboten werden. Bei Gruppen von Einträgen, die in größerer Zahl chronologisch angelegt werden (z.B. Sitzungsprotokolle), können ältere Einträge automatisch in einem „Archiv" zusammengefasst werden, so dass im Auswahlmenü nur die jüngsten (und daher wahrscheinlich relevantesten) Einträge sowie ein Archiv-Eintrag aufscheinen. Einträge können auf Dateien verweisen, die (in einem gängigen Format, z.B. HTML, PDF, Microsoft® Word (DOC), JPEG, GIF, usw.) mittels eines Administrations-Tools auf den Server geladen wurden, oder aus reinem Text bestehen, der unter Verwendung von vordefinierten „Schablonen" Web-gerecht formatiert ausgegeben wird.

3 Begriffe

3.1 Linux

Linux ist ein frei verfügbares Unix-Betriebssystem, das ursprünglich von dem finnischen Studenten Linus B. Torvalds entworfen wurde. Seine Arbeiten begannen 1991, im März 1994 konnte er die Kernelversion 1.0 vorstellen, 2.0 folgte im Juni 1996. Derzeit liegen die Versionsnummern jenseits der 2.4.0. Linux unterliegt der ‚GNU General Public License', das heißt, der Quellcode ist für jedermann zugänglich. Die freie Verfügbarkeit hat dazu geführt, daß immer mehr Programmierer die Entwicklung zu einer leistungsfähigen und stabilen Plattform vorangetrieben haben. Hierzu leistete das Internet einen entscheidenden Beitrag, da die auf der ganzen Welt verteilten Entwickler ihre Kommunikation über dieses Medium abwickeln.
Mit der zunehmenden Verbreitung nahm auch die Anzahl der für Linux verfügbaren Anwendungen zu. So steht Linux immer öfter auf den Listen der unterstützten Betriebssysteme auch bei renommierten Herstellern.

Die Kernel-Entwicklung findet normalerweise zweigleisig statt. Varianten mit gerader zweiter Release-Nummer (z. B. 2.2.x) sind die Anwender-Kernel, bei denen der Schwerpunkt auf Stabilität liegt. Die zweite Version (z. B. 2.3.x) sind die Entwickler-Kernel, bei denen die Aktualität der unterstützten Hardware Vorrang hat, beziehungsweise neue Konzepte integriert werden.

Einen Hinweis auf die derzeit aktuellen Kernel-Versionen erhält man - direkte Netzanbindung vorausgesetzt – mit dem Kommando : finger @ftp.kernel.org

3.2 Free Software / Open Source Software

Es ist wichtig zu wissen, daß GNU-Linux nur die Spitze des Eisbergs ist, den man "Free Software" (http://www.fsf.org) oder ‚Open Source' (http://www.opensource.org/osd.html) nennt.
Die bestimmenden Eigenschaften der Open Source Software sind: sie darf nach freiem Willen und kostenlos weitergegeben werden, ihr Quell-Code ist verfügbar und darf von jedem genügend erfahrenen Programmierer verändert werden.
Dieses Konzept ist hauptsächlich in GPL-Lizenz-Systemen (General Public License) enthalten, obwohl diese nicht die Einzigen sind. Ein Juwel der Open Source Software neben GNU-Linux, ist "Apache" (http://www.apache.org), ein Web Server Programm für das Internet, das einen Marktanteil von 50% in seinem Bereich hält.
Die Mehrzahl aller Software, die das Internet in allen Bereichen am Leben erhält, ist Open Source Software.

3.3 Apache

Der Apache HTTP Server (kurz Apache) ist eines der Aushängeschilder der Open-Source Community. Sein Marktanteil von bald 60 Prozent (siehe Abbildung) macht ihn zum unangefochtenen Leader unter den Web-Servern.
Das Apache Server Project, das sich um die Entwicklung des Apache HTTP Servers kümmert, ist eines von mehreren Projekten der „Apache Software Foundation", kurz ASF.
Die erst im Juni 1999 in Delaware (USA) gegründete Institution, die sich selbst als „not-for-profit corporation" bezeichnet, ist aus der im Jahr 1995 entstandenen und weitaus bekannteren

Apache Group hervorgegangen. Alle Projekte des ASF, zur Zeit acht an der Zahl, gründen auf den Prinzipien der freien, kollaborativen Software-Entwicklung, verwenden das Internet zur Kommunikation und leben vom Geist des „open-source“ Paradigmas. Das bekannteste mit dem „Apache Brand“ versehene Produkt ist der Apache HTTP Server, der üblicherweise vereinfachend nur „Apache“ oder „Apache Server“ genannt wird.

3.3.1 Ziele des Apache Server Project

Das Ziel des Projekts ist die Realisierung eines sicheren, effizienten und erweiterbaren HTTP Servers, der die aktuellen und gültigen HTTP (derzeit HTTP/1.1) Standards implementiert.

3.3.2 Entstehung

Der im Jahre 1995 meistgenutzte Web-Server im Internet war der durch Rob McCool entwickelte NCSA Server. Nachdem McCool bereits Mitte 1994 NCSA verlassen hatte, kam die weitere Entwicklung zum Erliegen und viele Webmaster, die bis dato die Version 1.3 des NCSA Servers benutzten, begannen den Server in eigener Regie zu „patchen“.
Zur Kommunikation unter diesen Webmastern wurde von Brian Behlendorf und Cliff Skolnick eine Mailing Liste und ein Rechner zur gemeinsamen Entwicklung eingerichtet.
Die Apache Group mit acht Core Contributors war geboren. Aus dem „gepatchten“ Server (im Englischen „a patchy server“) entstand auch die Bezeichnung Apache.
Nun erfolgte die Entwicklung Schlag auf Schlag. Im April 1995 erfolgte die erste öffentliche Release (Version 0.62) und bereits im Dezember 1995 war Apache 1.0 mit neuem Modul-Konzept, API, verbesserter Dokumentation und weiteren Erneuerungen verfügbar. Im selben Zeitraum löste Apache den NCSA Web-Server von der Spitze der benutzten Server Software ab. Apache 2.0 Beta ist die derzeit letzte Version des Servers.

Einsatz von Apache Web-server
Quelle : http://www.netcraft.co.uk/survey/ [31. März 2001]

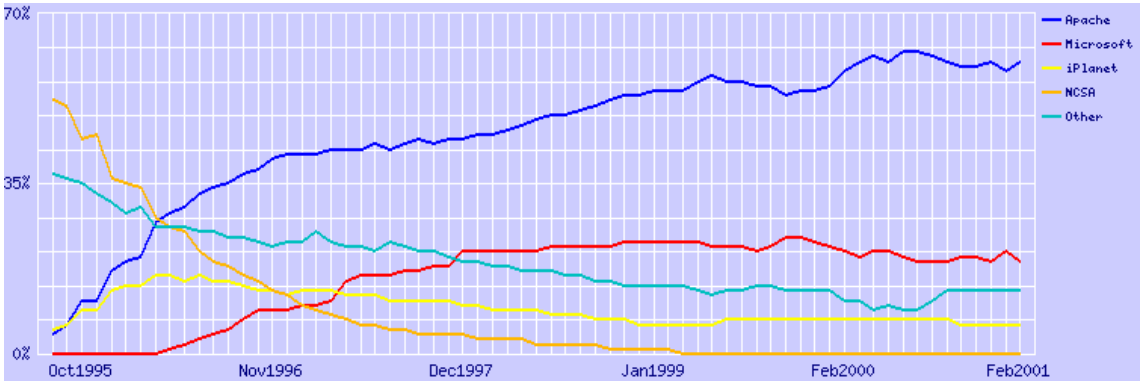


Abbildung 1 Market Share for Top Servers Across All Domains August 1995 - February 2001

3.3.3 Top Servers

Server	January 2001	Percent	February 2001	Percent	Change
Apache	16207982	58.75	16871744	59.99	1.24

Microsoft-IIS	5901507	21.39	5522069	19.63	-1.76
Netscape-Enterprise	1733097	6.28	1751123	6.23	-0.05
WebLogic	1004571	3.64	1039605	3.70	0.06
Zeus	693684	2.51	801215	2.85	0.34
Rapidsite	371441	1.35	380217	1.35	0.00
Thttpd	343172	1.24	367724	1.31	0.07
Tigershark	150937	0.55	166465	0.59	0.04
AOLserver	127980	0.46	153296	0.55	0.09
WebSitePro	113480	0.41	114655	0.41	0.00

Tabelle 1 Active Sites

Developer	January 2001	Percent	February 2001	Percent	Change
Apache	6380004	58.49	6716312	58.82	0.33
Microsoft	2682869	24.59	2776159	24.31	-0.28
Iplanet	263588	2.42	278738	2.44	0.02

Tabelle 2 Totals for Active Servers Across All Domains January 2001 - February 2001

3.4 MySQL

MySQL ist eine echte Multi-User, Multi-Threaded SQL Datenbank und wird von vielen großen Providern oder auch Suchmaschinenbetreibern eingesetzt. MySQL ist eine Client/Server Implentierung, die aus einem Server-Dämon mysqld und vielen Client Programmen, sowie Bibliotheken für PERL, PHP/3, PHP/4 sowie ASP besteht.

Die wichtigsten Eigenschaften von MySQL sind Geschwindigkeit, Stabilität und einfache Bedienbarkeit. MySQL wurde ursprünglich entwickelt, weil auf TCX (dem Server der Entwickler) ein SQL Server benötigt wurde, der sehr große Datenmengen handeln konnte, und zwar um eine Größenordnung schneller, als die Datenbankhersteller damals liefern konnten.
MySQL ist nun nun seit 1996 auf vielen Tausend Sites im Internet und Intranet im Einsatz und erfreut sich hier wachsender Beliebtheit. MySQL ist bei vielen tausend Unternehmen im täglichen Einsatz, von denen über 500 mehr als 7 Millionen Einträge bzw. mehr als 100 Gigabyte an Daten managen.
MySQL ist geradezu prädestiniert zur Beschleunigung bestehender MS-Access Datenbanken. Diese Kombination ermöglicht es, mit nur kleinen Änderungen in MS-Access die Datenbankabfragen um einen Faktor 3-100 zu beschleunigen.
Der Grundstock, um den MySQL herum gebaut worden ist, ist eine Liste von Routinen, die sich im täglichen Einsatz seit Jahren bewährt haben. Obwohl MySQL dauernd weiter entwickelt wird, hat es zu jedem Zeitpunkt der Entwicklung stets zuverlässig und stabil gearbeitet.

SQL ist eine standardisierte Datenbanksprache, die das Speichern, Updaten und den Zugriff auf Informationen erleichtert. Beispielsweise kann man Produktinformationen eines Kunden auf einem WWW-Server speichern und abrufen. MySQL ist äußerst schnell und flexibel genug, um sogar Bilder und Log-Dateien darin abzulegen. In der Praxis ist MySQL sehr viel schneller als z.B. ORACLE oder INFORMIX.

SQL steht für Structured Query Language und wurde Ende der 70´er Jahre bei IBM in San Jose, Kalifornien als Abfragesprache für die relationale Datenbank DB2 entworfen. Es

handelte sich hier ursprünglich um eine nichtprozedurale Sprache, die also keine Schleifen, Unterprogramme, Funktionen und Funktionsübergabeparameter u.s.w. enthält.

SQL ist also in dem Sinne keine Programmiersprache und dementsprechend einfach zu erlernen. Die SQL Befehle setzen sich aus zwei Teilen zusammen, der Data Definition Language (DDL) und der Data Manipulation Laguage (DML). Die DDL dient dem Aufsetzen der Datenbankstruktur, die DML dient der Manipulation der darin enthaltenen Daten.

3.5 PHP

PHP ("Personal Home Page"; "PHP: Hypertext Preprocessor") ist eine serverseitig interpretierte, in HTML eingebettete Skriptsprache. Die Syntax ist ähnlich zu C, Java und Perl, erweitert durch PHP-eigene Features wie z.B. Kommandos zur Integration von Datenbanken. PHP gibt dem WWW-Anwendungs-Entwickler einfach erlernbare und gleichzeitig mächtige Werkzeuge zur Erstellung von Web-Seiten dynamischen Inhalts an die Hand.

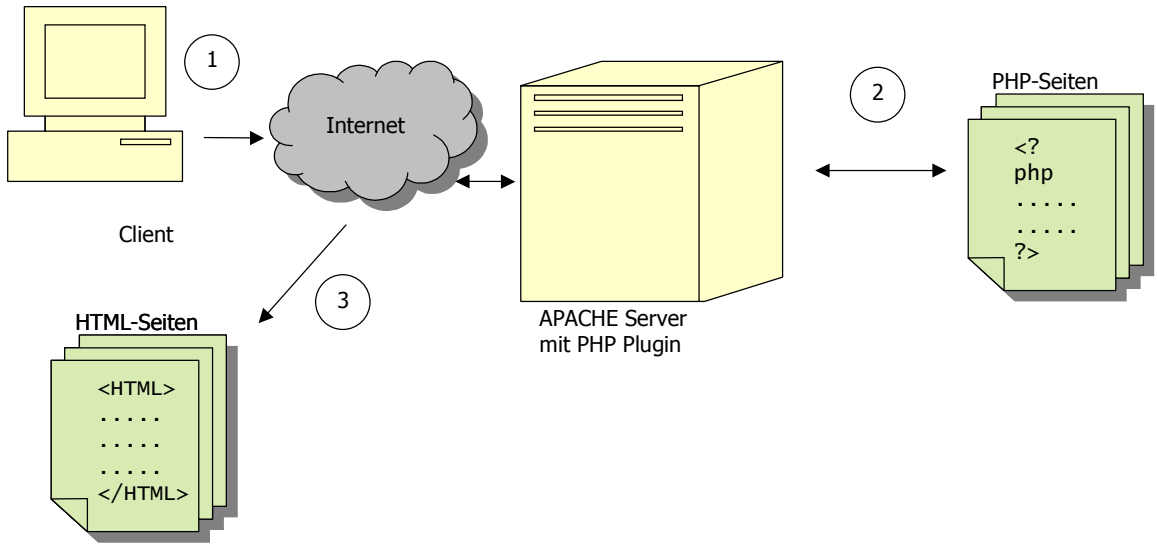


Abbildung 2 Wie entsteht aus einer PHP- eine HTML-Seite

PHP existiert sowohl für Unix als auch für die Windows-Plattform, sehr beliebt ist die Integration von PHP als Modul in den Apache-Webserver (http://www.apache.org), des weiteren ist die Ausführung via CGI möglich.
PHP wurde irgendwann im Herbst des Jahres 1994 von Rasmus Lerdorf konzipiert. Frühe, nicht veröffentlichte Versionen wurden auf seiner Homepage genutzt, um die Leser seiner Online-Bewerbung festzustellen. Die erste Version, die von anderen genutzt wurde, war Anfang 1995 verfügbar und wurde unter dem Namen "Personal Home Page Tools" bekannt. Sie bestand aus einem extrem simplifizierten Parser, der ausschließlich einige spezielle Macros verstand, und einigen Werkzeugen, die damals häufig auf Homepages genutzt wurden: ein Gästebuch, ein Counter und einige andere. Der Parser wurde Mitte 1995 neu programmiert und in PHP/FI umgenannt. Das FI kam von einem anderen Paket, das Rasmus geschrieben hatte und HTML-Formulardaten interpretierte. Er kombinierte die "Personal Home Page Tools"-Skripts mit dem Formular Interpreter und fügte noch mSQL-

Unterstützung hinzu und PHP/FI war geboren. PHP/FI wuchs in mit unglaublicher Geschwindigkeit, und immer mehr Leute begannen Code beizusteuern.
Es ist schwierig, harte Fakten zu liefern, aber es wird angenommen, dass PHP/FI Ende 1996 für mindestens 15.000 Websites auf ganzen Welt genutzt wurde. Mitte 1997 ist diese Zahl auf über 50.000 angewachsen. Um die gleiche Zeit gab es auch Veränderungen in der Weiterentwicklung von PHP. Es wandelte sich von Rasmus kleinem Privatprojekt, zu dem eine Handvoll Programmierer beigetragen hatte, zu einer wesentlich besser organisierten Anstrengung eines Teams.

Von Zeev Suraski und Andi Gutmans wurde der Parser von Grund auf neu geschrieben und bildete die Basis für PHP Version 3. Ein Teil des Codes der PHP/FI-Werkzeuge wurde auf PHP3 portiert und ein Teil wurde komplett neu erstellt.
Heute wird PHP/FI oder PHP3 mit einigen kommerziellen Produkten wie z.B. C2´s StrongHold Webserver und RedHat Linux eingesetzt. Eine vorsichtige Schätzung, basierend auf einer Hochrechnung der Zahlen, die von NetCraft veröffentlicht wurden, sagt aus, dass PHP für 150.000 Websites auf der ganzen Welt genutzt wird. Um das in Relation zu setzen: das sind mehr Sites, als alle die, die auf Netscapes Flagschiff, dem "Enterprise Server", laufen.
Während das hier geschrieben wird, ist PHP auf dem Weg zur nächsten Generation, welche die mächtige “Zend“ scripting engine nutzen wird, um höhere Geschwindigkeit zu erreichen, und auch mit anderen Servern als dem z.Zt. unterstützten Apache Server, als natives Modul laufen wird.

Mit der am 22. Mai 2000 freigegebenen Version 4.0 wandelt sich PHP von der Skriptsprache für kleine und mittlere Webanwendungen zur umfangreichen Middleware-Technik.
Version 4.0 enthält einen neuen Parser, eine Schnittstelle zu Webservern sowie eine Fülle an neuen Funktionen.
Zu den erweiterten Bereichen gehört neben dem Session-Management die Objektorientierung, darunter die Unterstützung von COM.
Mit PHP 4.0 können Webentwickler auch komplexe Anwendungen ohne die bisherigen Leistungseinbußen erzeugen.

Ein Beispiel für PHP Syntax:

```
<html>
   <head>
      <title>Beispiel</title>
   </head>
   <body>
      <?php echo "Hallo, das ist ein PHP-Skript!"; ?>
   </body>
</html>
```

Dieser Skript unterscheidet sich von einem CGI-Skript, der in einer Sprache wie Perl oder C geschrieben wurde -- anstatt ein Programm mit vielen Anweisungen zur Ausgabe von HTML zu schreiben, schreibt man einen HTML Code mit einigen, eingebetteten Anweisungen, um etwas zu auszuführen (z.B. um -wie oben- Text auszugeben). Der PHP Code steht zwischen speziellen Anfangs- und Schlusstags, mit denen man in den PHP-Modus und zurück wechseln kann.
Was PHP von einer client-seitigen Sprache wie Javaskript unterscheidet ist, daß der Code vom Server ausgeführt wird. Wird ein Skript wie den obige auf einem Server ausführt,

empfängt der Besucher nur das Ergebnis des PHP-Codes, ohne die Möglichkeit zu haben, herauszufinden, wie der zugrundeliegende Code aussieht.

4 Aspekte beim Entwurf einer Web-Anwendung

4.1 Einführung

Web-Anwendungen können sehr unterschiedliche Ausprägungen haben. Die Bandbreite reicht von statischen, dokumentenorientierten Hypertextsystemen bis hin zu interaktiven, verteilten Anwendungen (z.B. mit Hilfe von Formularen und CGI-Skripten realisiert). Bezüglich der unterschiedlichen Ausprägungen existieren verschiedene Werkzeuge, die bei der Erstellung von statischem Hypertext, bei der Integration von Datenbanken als Informationssysteme oder bei der Entwicklung von hochdynamischen Anwendungen (z.B. sessionsbasierte Anwendungen) unterstützen. Statischer Hypertext ist dadurch charakterisiert, daß nur statische Links und Seiten angewendet werden. Im Vergleich hierzu sind bei einem Web-Informationssystem die Seiten dynamisch, lediglich die Linkstruktur bleibt statisch. Dynamische Web-Anwendungen sind dadurch gekennzeichnet, daß sowohl die Seiten als auch die Linkstruktur dynamisch erzeugt werden.

Im Rahmen dieser Diplomarbeit wird eine dynamische Web-Anwendung entwickelt.

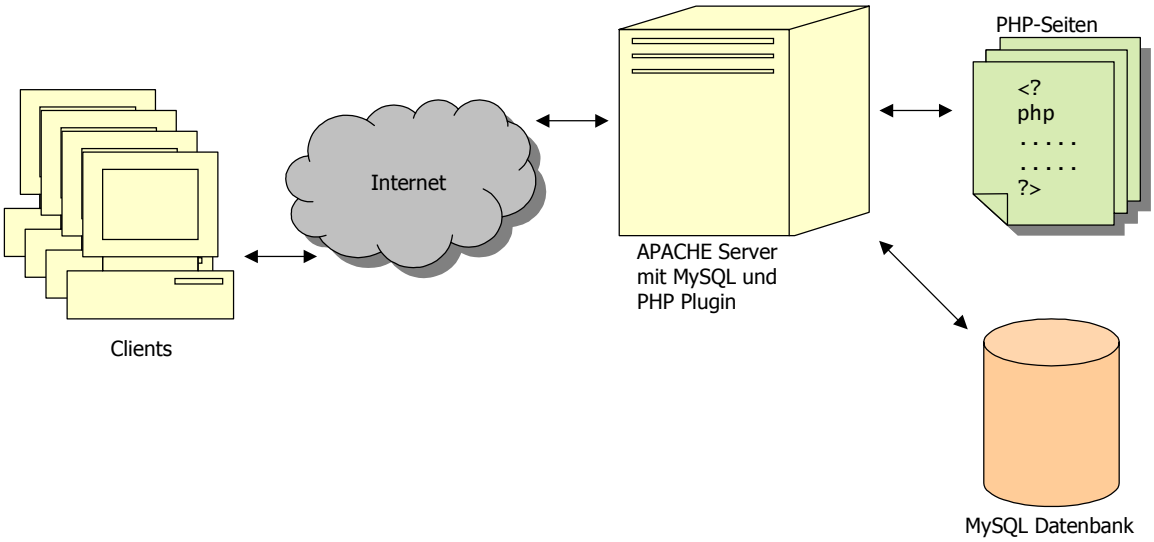


Abbildung 3 Die Architektur in der ein Web-Server, eine relationale Datenbank und eine Ansammlung von serverseitigen Befehlen zur Ein- und Ausgabe von Daten vorkommen

Beim Entwurf einer Web-basierten Applikation gibt es einige grundlegende Unterschiede zu einer “konventionellen“ Client-Server Applikation. Diese Unterschiede können in zwei große Gruppen unterteilt werden:

- Unterschiede in der Mensch-Maschine-Kommunikation
- Unterschiede in Client-Server Aufbau und Persistenz der Applikation

4.2 Mensch-Maschine-Kommunikation

Durch das Benutzerinterface eines Browsers kann man die Funktionalität einer klassischen Applikation nicht nachbilden. Man kann nicht mehrere gleichzeitig geöffnete Fenster haben, wie z.B. ein zweites Dialogfenster, das den Benutzer zwingt, genau in diesem eine Aktion zu setzen, oder ein Fehlermeldedialog, der den Benutzer z.B. über eine unvollständige Eingabe benachrichtigt, und diesen wieder zwingt im ursprünglichen Fenster die Aktion zu wiederholen.
Vielmehr muß man versuchen, die Benutzeraktionen so zu definieren, daß sie sich über klar definierte und abgegrenzte Schritte abspielen. Die einzige Interaktionsmöglichkeit des Benutzers mit dem System (außer, man benutzt die Java-Script Erweiterungen oder Java-Applets) sind die Links, die der Benutzer 'anklicken' kann, bzw. die Formulare mit den entsprechenden Buttons, was eigentlich den Links entspricht. In den Links können beliebig viele Parameter enthalten sein, und durch diese Parameter gibt der Client dem Server die eigentliche Information weiter. Diese Parameter sind dann in den 'PUT' oder 'GET' Variablen am Server wieder enthalten.

4.3 Persistenz der Applikation

Bei einer Web-basierten Applikation existieren keine Clients und Server wie bei den klassischen Applikationen. Was es gibt, ist eine Ansammlung von Skripts am Server, die der Benutzer, vor einem Browser sitzend ausführen kann. Die Skripts können verschiedenste Anweisungen beinhalten, wie zum Beispiel das Öffnen einer Datenbankverbindung, Auslesen der Daten, Schreiben der Daten in die Datenbank, Erstellen oder Löschen einer Datei am Server. Das Problem einer fehlenden Persistenz im Client kann man mit Hilfe einiger Techniken wie 'Sessions', 'Cookies' , oder eine Serialisierung der Objekte und ihre temporäre Abspeicherung am Server, in den Griff bekommen. Diese Begriffe werden weiter unten näher erläutert.

4.4 Das HTTP-Protokoll

Das Hypertext Transfer Protocol wurde 1990 mit folgender Zielsetzung bzw. Defnition entworfen: " The Hypertext Transfer Protocol (HTTP) is an application-level protocol for distributed, collaborative, hypermedia information systems". HTTP zählt zur Gruppe der request/response/-Protokolle.
Die aktuelle Version des HTTP-Protokoll ist 1.1. Hier sind im Vergleich zu 1.0 insbesondere Verbesserungen in Bezug auf Proxying, Caching, persistente Verbindungen und virtuelle Hosts implementiert worden.
Eine Besonderheit des HTTP-Protokolls ist die Zustandslosigkeit ('statelessness'), d.h. nachdem ein Client eine Anfrage an einen Server geschickt hat, sendet der Server die Antwort und beendet die Verbindung. Eine weitere Anfrage desselben Clients sieht somit aus Sicht des Servers wie eine vollkommen neue Anfrage eines neuen Clients aus. Dies ist insbesondere bei der Entwicklung von Web-Applikationen zu berücksichtigen.

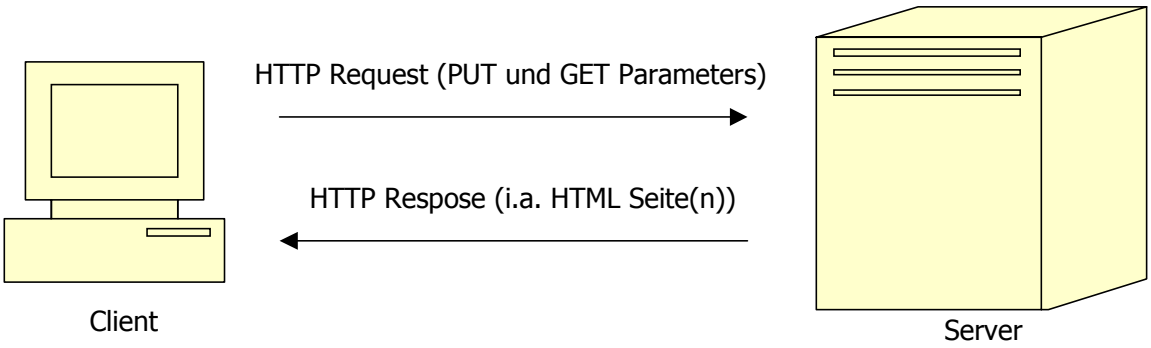


Abbildung 4 Kommunikation zwischen Client und Server über HTTP Protokoll

4.5 Session

Eine Session ist ein zeitlich begrenzter, autorisierter Zugang zu einer Internet-Applikation.
Technisch werden Sessions gerne folgendermaßen realisiert:
Beim ersten Aufruf einer Seite, die die PHP-Funktion session_start() beinhaltet, ist der Session-Schlüssel im Request beigelegt und wenn nicht, generiert der Server einen neuen Schlüssel (‚Session-Key') und legt ihn in Form einer Variable diesem Request bei. Diese Variable wird dann entweder im Cookie (weiter unten wird erläutert, was ein Cookie ist) am Client gespeichert, oder bei jedem Client-Request an den Server weitergegeben.
Somit weiß der Server, daß es sich immer wieder um einen und desselben Client handelt, obwohl es zwischen zwei Requests beliebig (eigentlich einstellbar) lang dauern kann. Zusätzlich zu dem Session-Schlüssel kann der Server beliebig viele Session-Variablen oder sogar ganze Objekte in einer lokalen Datei speichern, und somit eine Art Client-Persistenz aufbauen. PHP 4 bietet Möglichkeit die Objekte in eine Datei temporär zu speichern (‚serialisieren').

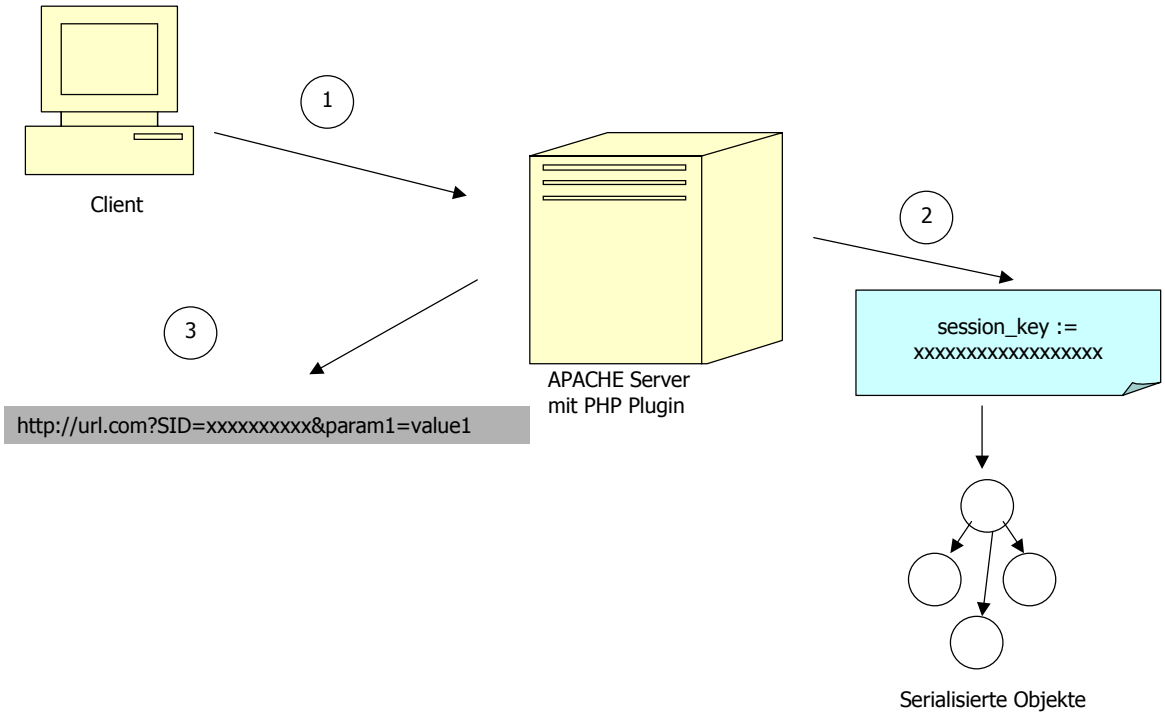


Abbildung 5 Ablauf einer Session

In der vorliegenden Applikation wird diese Möglichkeit folgendermaßen ausgenutzt:
Jedem Client, das sich erfolgreich angemeldet hat, wird ein ‚Session-Key' zugeordnet. Dieser Schlüssel wird in Form einer Datei am Server gespeichert. Außerdem wird ein neues Objekt der Klasse *‚IdmApplication'* erstellt, und darin die Informationen über den Benutzer, seine Rechte, seine aktuellen Einstellungen der Arbeitsumgebung, und sonstige applikationsübergreifende Informationen gespeichert. Diese Objekte werden dann ebenfalls in eine temporäre Datei am Server abgespeichert, und werden bei jedem neuen Request neu aufgebaut. Somit ist es nicht notwendig, immer wiederkehrende Informationen aus der Datenbank zu holen. Durch den objektorientierten Ansatz ist es sehr einfach möglich, weitere Objekte an das ‚IdmApplikation'-Objekt anzuhängen und diese ebenfalls automatisch abspeichern lassen.
Wenn sich der Benutzer explizit abmeldet, wird die Datei am Server, in der Session-Schlüssel und die serialisierten Objekte gespeichert waren, gelöscht. Wenn er aber nur seinen Browser schließt, befinden sich Dateien noch eine Zeit am Server, und werden dann anschließend von einem ‚cron'-job gelöscht .

4.6 Cookies

Cookies stellen Variable dar, die vom Webserver initiiert und wieder auslesbar sind.
Sie werden vom Browser lokal in einer Datei abgespeichert und können als eine Art Gedächtnisstütze für die Webapplikationen verwendet werden.
Es lassen sich allerdings nur begrenzte Informationen in Cookies speichern und wieder auslesen.
Durch den Begriff ‚gläserner Surfer' sind Cookies in Verruf gekommen, denn durch die Auswertung von Cookies lässt sich das Surfverhalten von Benutzern im Internet analysieren.
Für den Programmierer stellt dieser Mechanismus allerdings eine praktische und einfache Variante zur Speicherung von Informationen beim Client dar.
Als alleinige Zugangskontrolle eignen sich Cookies nicht, in Verbindung mit anderen Mechanismen können sie aber einen erleichterten Zugang (z.B. durch Vorbelegung des Loginfeldes) erzeugen.
In IDM kommen die Cookies zum Einsatz, und zwar wird in einem Cookie der zuletzt verwendete ‚Session-Key' abgespeichert.
Dem Benutzer, der sich bereits einmal erfolgreich angemeldet hat, ist es möglich, das nächste Mal ohne Anmeldung ins IDM einzusteigen.
Hat sich der Benutzer abgemeldet, oder hat er die ‚Cookie-Unterstützung' in seinem Browser ausgeschaltet, muß er sich nach jedem Neustart (von Rechner oder Browser) wieder anmelden.
Die Cookie–Unterstützung ist im IDM nicht unbedingt notwendig, der in den Cookies gespeicherte Session-Key wird, falls notwendig, in HTTP-Request Variablen propagiert.

5 Entwicklungsprozeß

Um komplexe Aufgaben wie die Entwicklung einer Web-Applikation besser in den Griff zu bekommen, geht man am besten systematisch vor. Man unterteilt die Entwicklung in verschiedene Phasen, um eine schrittweise Durchführung zu ermöglichen. Die Phasen lassen sich folgendermaßen einteilen:

- Problemanalyse und Planung
- Systemspezifikation
- System- und Komponentenentwurf
- Implementierung
- Systemtest
- Betrieb und Wartung

5.1 Problemanalyse

Die vorliegende Applikation soll eine Client – Server Applikation werden. Eine der grundlegenden Eigenschaften ist durch das vorliegende HTTP-Protokoll bestimmt, der Wegfall von Persistenz sowohl am Client als auch am Server.

Weitere Probleme, die auftreten können, sind schlechte Trennbarkeit zwischen Funktionalitätscode, also jenen Funktionen die das Verhalten des Systems definieren, und GUI-Code, also in vorliegendem Fall HTML-Seiten oder –Fragmenten, die das Aussehen der Applikation definieren. Hier muß man ein System entwickeln, das eine Trennung zwischen diesen unterschiedlichen Aspekten einer Applikation ermöglicht.

Die Systemfunktionen, die man für die Applikation braucht, wie z.B. das Kopieren einer Datei, Starten einer Session, die Anbindung an die relationale Datenbank, sind weitgehend durch die mitgelieferten PHP-Funktionen abgedeckt.

Durch die bereits erwähnte ‚Request/Response' Eigenschaft des HTTP-Protokolls gilt es, neue Wege bei der Entwicklung der Benutzerschnittstelle sowie darunterliegender Applikationsschichten zu suchen.

Die Zustandslosigkeit, die bei den Web-basierten Applikationen vorkommt, bedeutet, daß jedes Mal, wenn ein Benutzer ein Link aufruft oder einen Button drückt, eine Verbindung zum Web-Server aufgebaut wird, Daten dort entsprechend ausgewertet werden und ein Ergebnis zurückgeschickt wird.

Was kann der Client dem Server übermitteln ?

Die Möglichkeiten, vom Client aus dem Server etwas bekannt zu geben, sind ebenfalls durch den HTTP-Protokoll relativ beschränkt.
Der Client kann eine spezifische Seite am Server aufrufen und auf diese Art und Weise unterschiedliche Aktionen bekannt geben. Unterschiedliche Seiten können eben unterschiedliche Code beinhalten und dadurch ist eine Verhaltenstrennung gegeben.

Zum Beispiel: eine HTML-Seite beinhaltet zwei Links:

```
<HTML Seite>
http://server/link1.php
http://server/link2.php
</HTML Seite>
```

Wenn ein Benutzer den ersten Link aufruft, setzt er am Server Aktion 1 (die im Code link1.php beschrieben ist), respektive Aktion 2.
Der Nachteil einer solchen Programmierung ist, daß man sich relativ schnell zwischen vielen Codeseiten befinden kann. Ein mögliche Vorteil ist eine Art Modularität, die aber mit einem großen Programmierungsaufwand verbunden ist. Dieser Aufwand würde mit der wachsendem Komplexität wahrscheinlich überproportional wachsen.
Ein zweiter möglicher Vorteil ist, dass der PHP-Parser jene Seiten, die er bei diesem Aufruf nicht braucht, auch nicht durchlaufen muss.

Eine weitere Möglichkeit ist, daß man auf der betreffenden Seite auf ein und denselben Link verweist, jedoch mit unterschiedlichen ‚GET'-Parametern. Durch die verschiedene Parameter gibt man implizit dem Server bekannt, welche Interaktion man wählt.

```
<HTML Seite>
http://server/link.php?parameterName=parameterWert1
http://server/link.php?parameterName=parameterWert2
</HTML Seite>
```

Der Vorteil einer solchen Programmierung ist, dass man auf eine und dieselbe Datei bei verschiedenen Interaktionen verweist, und auf diese Art eine bedeutend bessere Übersichtlichkeit bei der Applikationsentwicklung erreichen kann.
Diese Art hat auch ihre Nachteile. Eine ist, daß der PHP-Parser über den gesamten PHP-Code laufen muß, auch über den Teil, der für die nicht-gewünschten Interaktionen da ist.

Eine dritte Möglichkeit wäre, beide oben angeführten Arten zu mischen, d.h. viele Dateien (aber funktional unterschiedlich), die aber entsprechende Parameter verstehen können.
Neben klaren Vorteilen, ist die schwierige Trennung ein Nachteil, d.h. welche ‚Use-Cases' in welcher Datei werden beschrieben, und wo zieht man da die Grenze durch? Insbesondere, was passiert, wenn man im Laufe der Entwicklung zu den Grenzfällen kommt, die mehrere Funktionen, zerstreut über mehrere Dateien einsetzen?

Bei der Entwicklung der vorliegenden Applikation ist die Entscheidung für die Version 2 gefallen, jedoch mit einigen Anpassungen, die weiter unten erläutert werden.

Die zweite große Frage lautet, wie man eine große Web-Applikation macht, die auch wartbar und erweiterbar ist.

Bei der Entwicklung einer Web-basierten Applikationen spielen nicht nur die Programmierer eine Rolle. Vielmehr ist es ein Zusammenspiel zwischen vielen Disziplinen:

- Funktionalität, die durch die ‚klassische Programmierer' implementiert wird
- Design der Benutzerschnittstelle, bei dem viele andere Disziplinen vorkommen (u.a. Psychologen, GUI-Designer, Graphiker ...)
- Implementierung der Benutzerschnittstelle, wo die Web-Designer zu Zug kommen

Wie man sehen kann, ist die Entwicklung einer Web-basierten Applikation mittlerweile ein interdisziplinäres Projekt, und man kann allen diesen Leuten das Leben sehr schwer machen, wenn man sie alle an einer und denselben Datei arbeiten läßt.
Deshalb ist es notwendig, die Applikation so zu entwerfen, daß die Programmierer nur mit ihrem Code arbeiten, in vorliegendem Fall mit den PHP-Seiten, und die Web-Designer mit HTML-Seiten. Die unvermeidbare Mischung soll nur über klar definierten Schnittstellen passieren. Die Realisierung einer solchen Applikation führt unvermeidlich zu einer Lösung mit sehr vielen Dateien.

Deshalb stellt sich die Frage, wie man die Applikation entwerfen kann, daß man einerseits einen zentralen Einstiegspunkt hat, die aber trotzdem modular aufgebaut ist, und so die Entwicklung für alle Beteiligten mit möglichst wenig Aufwand verbunden ist?

Einen möglichen Ausweg kann man mit Abhilfe eines objektorientierten Ansatzes finden.

5.2 Objektorientierter Ansatz

Je mehr Projekte man mit PHP umsetzt, um so mehr merkt man, daß der gleiche Code immer und immer wieder vorkommt. Da fragt man sich, ob es nicht möglich ist, den Code, den man in verschiedenen Projekten wiederverwendet, irgendwie zu kapseln, um ihn dann auslagern zu können.

Die erste Möglichkeit hierfür wären einfache Funktionen, die man in eine externe Datei einbindet und diese dann in jede PHP-Datei, in der man den Code braucht, einbindet. Die andere Möglichkeit ist, die Funktionen zu kapseln und sie in ein Klasse zu packen. Das führt dazu, daß man quasi einen Topf hat, in dem Daten und Funktionen stecken, auf die man ganz einfach zurückgreifen kann.

Die Definition einer Klasse erfolgt in PHP mit dem Schlüsselwort class, gefolgt vom Namen der Klasse. Dazu ein Beispiel-Code:

```
class IdmChangeDesktopActivity extends IdmSilentActivity {
        var classVar1;
        function initialize (){
                global $idmApp;
                if ($this->application->currentUser->lastdesktop == 'public') {
                        if ($this->application->currentUser->userIsAdmin()) {
                                $this->application->currentUser->lastdesktop = 'admin';}
                }
                else {$this->application->currentUser->lastdesktop = 'public';}
                $idmApp->currentUser->lastdesktop = $this->application->currentUser->lastdesktop;
                $this->application->currentUser->sqlUpdateLastdesktop();
        }
        function processData (){
                $anActivity = new IdmPrintDesktopActivity;
                $anActivity->startActivityForApplication($this->application);
}
```

Das Schlüsselwort class wird hier gefolgt vom Namen der Klasse und von den üblichen geschweiften Klammern, die auch bei Klassen von elementarer Bedeutung sind. Danach erfolgt die Definition der Variablen, die zur Klasse gehören sollen. Hierbei muss vor den

Namen der Variable noch das Schlüsselwort var gesetzt werden. Danach folgen Deklarationen von Funktionen, die für das Verhalten stehen. Die Funktionensdeklarationen entsprechen ansonsten voll und ganz denen, die man von der normalen, prozeduralen Programmierung mit PHP gewohnt ist.

Grundsätzlich gesehen ist eine Klasse eine Art Bauanleitung für ein Objekt. Deswegen kann man mit einer Klasse an sich noch nichts anfangen. Eine Klasse beschreibt Eigenschaften und Funktionen eines Objektes. Aus einer Klasse lassen sich beliebig viele Objekte bauen.

Das Schlüsselwort *,$this'*

Hierbei handelt es sich um einen Zeiger, der immer auf das aktuelle Objekt zeigt. Wenn man innerhalb einer Klasse auf deren Variablen zugreifen möchte, so kann dies nicht einfach mit dem Benutzen des Variablennamen geschehen, sondern hierfür muss das Schlüsselwort $this verwendet werden.

5.3 Model - View - Controller

Die Model-View-Controller-Architektur (MVC-Architektur) ist ein eigentlich schon ziemlich altes objektorientiertes Programmierkonzept, das schon 1981 mit dem Smalltalk-System der Welt vorgestellt wurde. Dabei handelt es sich um eine konsequente Trennung der Daten von der Oberfläche, die diese Daten darstellt.

Prinzip

In der MVC-Architektur besteht jedes Programm aus mindestens drei Objekten:

- **Model** (Modell, das die Daten enthält)
- **View** (Anzeige der Daten)
- **Controller** (Steuerung)

Im Model werden die Daten gespeichert und intern verarbeitet. Dabei kann sich das Model auch noch anderer Objekte bedienen. Das View-Objekt sorgt für die Darstellung der Daten auf dem Bildschirm. Der Controller nimmt die Benutzereingaben entgegen und setzt sie in die entsprechenden Aktionen um. View und Controller sind während der Laufzeit des Programms ein fest verdrahtetes Paar. Ihnen ist genau ein Model zugeordnet. Einem Model wiederum können mehrere View/Controller-Paare zugeordnet sein.

Das bedeutet, daß ein- und dieselben Daten gleichzeitig auf verschiedene Weise angezeigt und von den Controllern bearbeitet werden können. Zum Beispiel in einem Fenster als Listbox und in einem anderen Fenster als Icons. Wird nun irgend etwas in der Listbox geändert, so wird diese Änderung sofort im Icon-Fenster sichtbar.

Für den Programmierer ist dieses Konzept von großem Vorteil, da sich die zentrale Logik des Programms in der Model-Ebene abspielt und von der verwendeten Oberfläche vollständig und der Hardware-Plattform weitgehend unabhängig ist. Die Aufteilung führt automatisch zu sauberen Schnittstellen und zu leichter wartbarem Programmcode.

View und Controller müssen voneinander wissen und verweisen deshalb mit Hilfe von Zeigern direkt aufeinander. Bei der Verbindung zwischen Model und View/Controller-Paar

ist das ganz anders: Erstens dürfen für ein Model beliebig viele solcher Paare existieren, zweitens soll das Model ja unabhängig von der Oberfläche programmiert werden können. Deshalb darf es von diesen Paaren nichts wissen.

Hier kommt der ‚Dependenz-Mechanismus' als eine Art Oberaufseher ins Spiel. Die Beziehung ‚Dependenz-Mechanismus' kommt aus dem Englischen (dependency = Abhängigkeit). Es wird eine Dependenzliste erzeugt, die alle Fäden in der Hand hält. Dort sind sämtliche View/Controller-Paare für alle Models in einer Liste eingetragen. Das Model braucht dann nur die Dependenzliste zu unterrichten, daß seine Daten verändert wurden. Das geschieht einzig über die Methode changed(). Die Methode changed() des Models ruft die Methode update() der Dependenzliste auf, die ihrerseits die Methode update() aller dem Model zugehörigen Views aufruft. Damit sind alle Anzeigen der Daten des Models wieder auf dem neuesten Stand. Schematisch sieht das folgendermaßen aus:

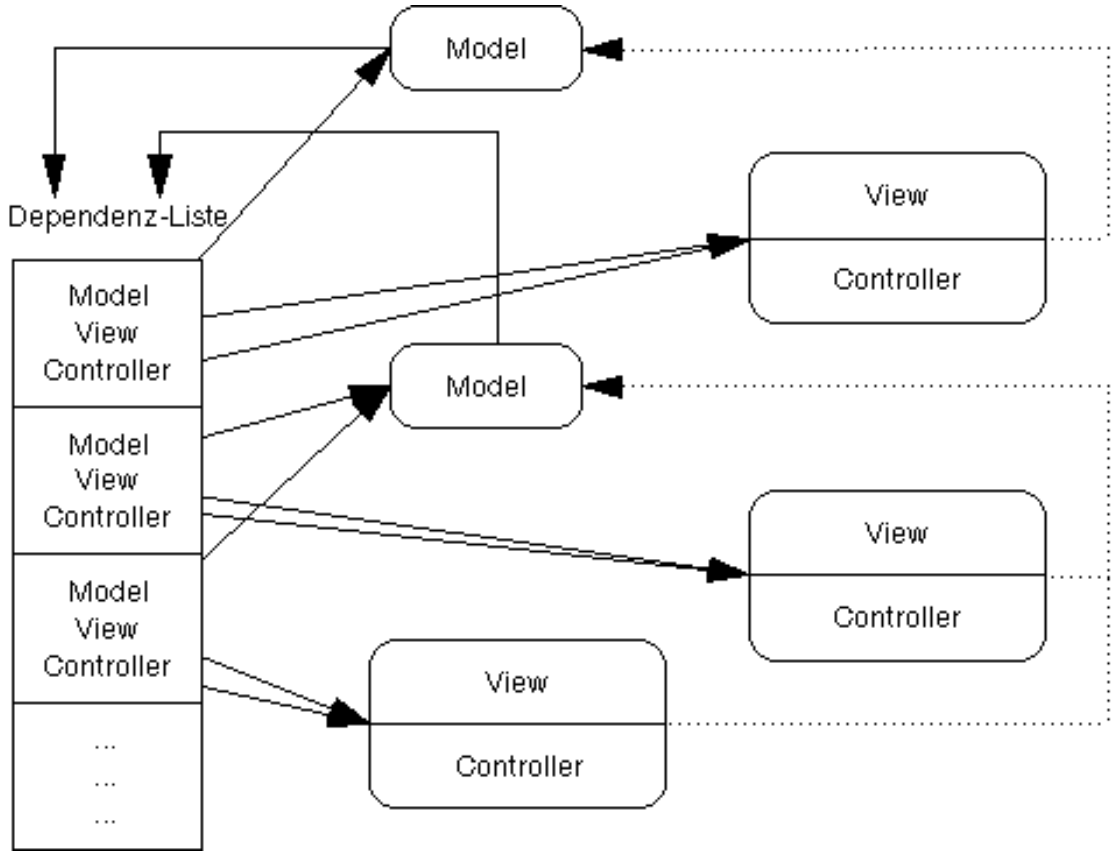


Abbildung 6 MVC Architektur

Ein Model kann mehrere View/Controller-Paare bedienen, Jedem View/Controller-Paar ist aber genau ein Model zugeordnet. Die Models wissen nur von der Dependenzliste, während die View/Controller-Paare ihr Model "kennen".

5.4 Systemspezifikation

Hier sind einige Anwendungsfälle (engl. ‚Use-Case'), die bei der Applikation vorkommen, aufgeführt. Prinzipiell kann man zwischen zwei Arbeitsumgebungen (‚Desktops') unterscheiden: Administrationsumgebung und Standardumgebung.

Die Administrationsumgebung ist nur den Benutzern der Gruppe ‚Administratoren' vorbehalten, während in der Standardumgebung alle Benutzer arbeiten können.

Administrationsumgebung

- Benutzer anlegen/ändern/löschen
- Benutzergruppe anlegen/ändern/löschen
- Benutzer einer Gruppe zuordnen/löschen
- Gruppenrechte auf Ordnern anlegen/ändern/löschen
- Dateiverwaltung: Datei in einen Ordner anlegen/ändern/löschen
- Applikationsübergreifende Einstellungen vornehmen
- Mailing an die Benutzer
- Zugriff-Logging

Standardumgebung:

- Ordner anlegen
- Datei anlegen
- Datei öffnen
- Link anlegen
- Link öffnen

Neben den oben angeführten gibt es noch eine Reihe von kleineren ‚Use-Cases', die durch eine kurze Interaktion des Benutzers mit dem System darstellbar sind, wie z.B., ‚Fragen an Benutzer stellen', oder ‚Antwort auswerten und entsprechend reagieren'.

5.5 System- und Komponentenentwurf

5.5.1 Activities

Wie bereits besprochen, sind eines der grundlegenden Probleme in der Entwicklung einer webbasierten Applikation die verteilten Daten und eine Mischung zwischen Code und Oberflächendesign (HTML-Seiten). Ein weiteres Problem sind die vielen Einstiegspunkte in die Applikation, die durch viele Dateien scheinbar unvermeidlich sind.
Die Lösung dieser und einiger anderen Probleme ist in der Abstrahierung dieser Interaktionen und Festlegen eines passenden objektorientierten Modells.

Die Activity ist die Abstrahierung jeder Interaktion zwischen Client und Server, sowie zwischen dem Server und der Datenbank. Die Activity repräsentiert den Controller-Teil eines MVC-Frameworks. Sie ist einerseits für das Holen der Daten aus der Datenbank zuständig. Andererseits weiß sie wer für die Datenrepräsentation zuständig ist. Sie weiß, aus welchen HTML-Dokumententemplates eine Seite gebaut wird, gleichzeitig stellt sie in diesen Templates die Daten zur Verfügung. Deshalb wird auf alle Daten in den Templates über die ‚$this'–Anweisung (dasselbe wie ‚self' in Java) zugegriffen. Das bedeutet, daß der Ort, wo man die Daten holt, Objekte dieser Klasse sind, gleichzeitig aber sind sie der Ort, in dem man die Daten repräsentiert.

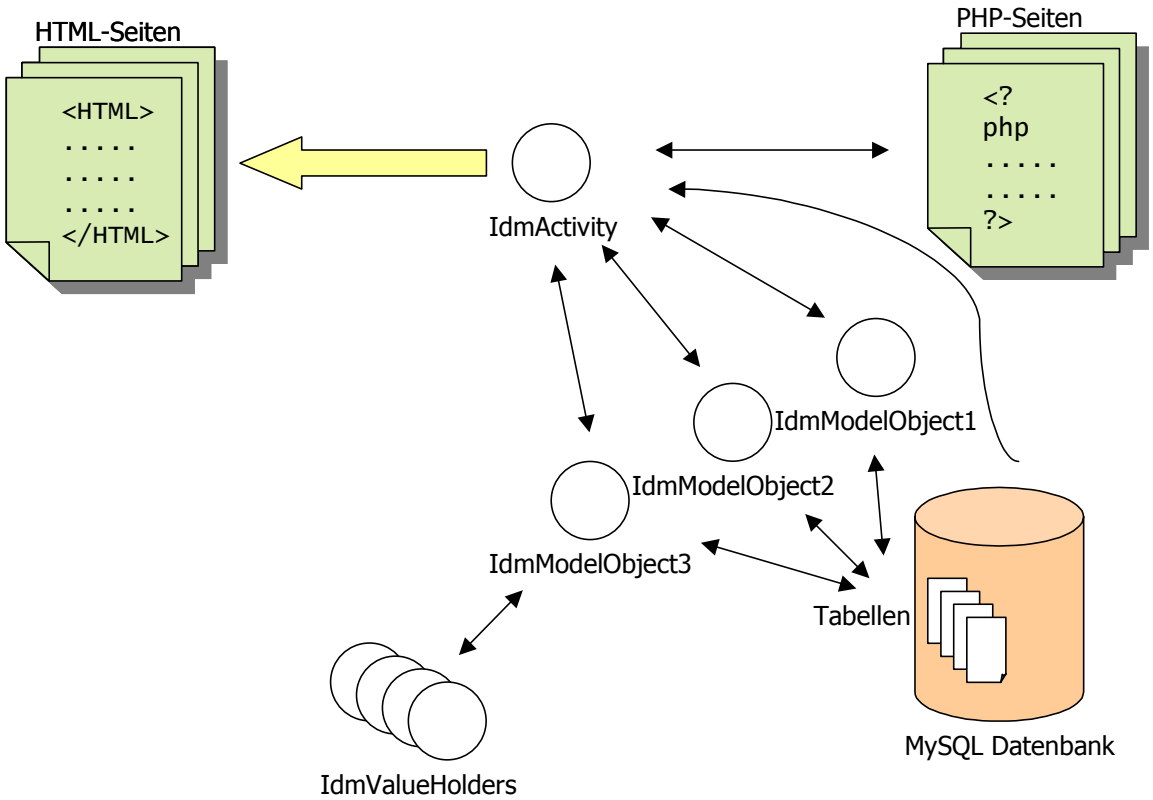
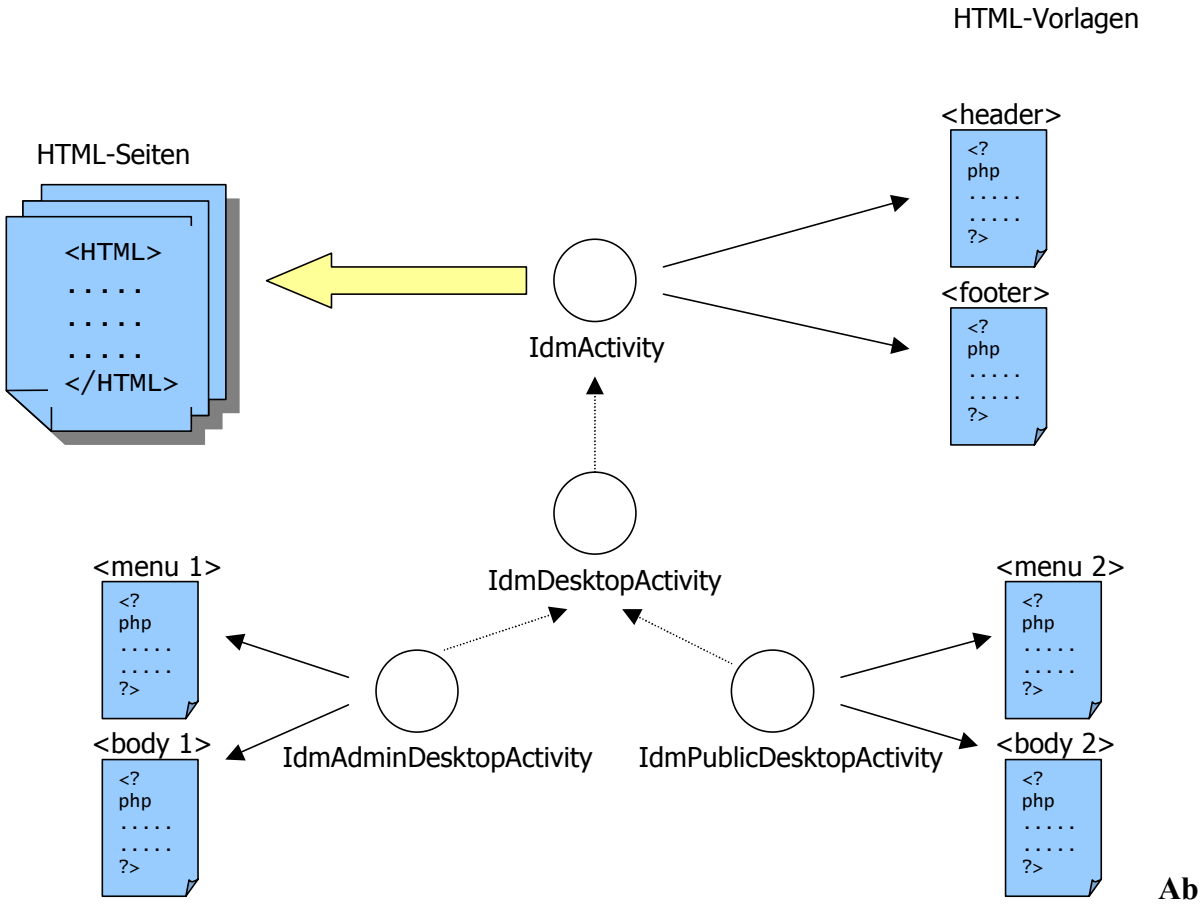


Abbildung 7 Funktion der Activity

Nachdem für die Repräsentation der Daten in einem MVC-Framework Viewer-Klassen zuständig wären, simuliert man solche durch die HTML-Templates, die wiederum für die HTML-Designer relevant sind und somit keine größeren Programmierkenntnisse voraussetzen dürfen. Das bedeutet, daß die gesamten Algorithmen, Holen der Daten, Speichern der Daten,

Abbildung verschiedenster Geschäftsprozessen in den Methoden der Activity-Klasse implementiert sein sollen, und nicht in die HTML-Templates ausgelagert werden dürfen.



Ab

Abbildung 8 Funktionsweise der HTML Vorlagen

Diese Klasse ist auch für die Verarbeitung der Clientantwort zuständig. Sie besitzt wichtige API-Methoden, die GET- und POST-Variablen aus dem Client-Request auslesen. Diese Variablen sind die einzige Möglichkeit für den Client, dem Server etwas zu übermitteln.

Man unterscheidet zwischen folgenden Grundformen (Subklassen) der Activity:

SilentActivity
Die ‚SilentActivity' bildet jene Interaktionen ab, die keine HTML-Seiten als Ergebnis liefern, sondern nur am Server (in aller Ruhe – ‚silent') abgearbeitet werden. Beispiel einer solchen Activity wären interne Batch-Jobs, die regelmäßig und ohne Client-Eingriff ausgeführt werden. Ein zweites Beispiel sind die Interaktionen, die man separat kapseln und dann in unterschiedlichen Kontexts ausführen will.

PrintActivity
Die Klassen der PrintActivity sind, wie der Name sagt, für die standardisierte Datenausgabe (printing) zuständig. Eine solche Activity holt bestimmte Daten aus der Datenbank, und stellt sie den HTML-Templates zur Verfügung. Welche HTML-Templates zum Einsatz kommen, entscheidet ebenfalls die Activity. Außerdem kann sie, wie jede andere Activity, eventuelle

Meldungen vom Client in Form von PUT und GET Variablen in lokale Variablen holen, und diese entsprechend weiterverarbeiten.
Diese Klassen wurden in der vorliegenden Applikation am meisten verwendet. Welche HTML-Telmplates zum Einsatz kommen, wird in den folgenden Methoden beschrieben:

- headerFileName()
- bodyFileName()
- footerFileName()

PromptActivity
Die Subklassen der PromptActivity werden bei der Interaktion mit dem Benutzer eingesetzt. Zum Beispiel, wenn der Benutzer ein Formular ausfüllt und diese Daten verschicken will, startet er eine PromptActivity. Nachdem die Daten, die der Benutzer geschickt hat, nicht unbedingt gültig sein müssen, muß die Activity entsprechend reagieren, und dem Benutzer das halbausgefüllte Formular wieder zur Verfügung stellen, und ihn auffordern fehlende oder falsche Daten zu korrigieren. Der Benutzer kann sich auch entscheiden, eine bestimmte Interaktion abzubrechen. Dann muß die Activity dem Benutzer die ursprüngliche Umgebung zur Verfügung stellen.
Alle diese Aufgaben übernimmt die PromptActivity.

5.5.2 Starten der Applikation

Starten und jeder weiterer Aufruf der Applikation geschieht immer über die *index.php* – Datei. Diese Datei ist der einzige Einstiegspunkt in die Applikation. Aus diesem Grund wird sie hier näher erklärt.

Die globale Variable *$idmApp* enthält das Objekt IdmApplikation, das eingentlich die Applikation selbst darstellt. *Mit session_start()* wird eine neue Session gestartet oder - wenn ein gültiger Session-Schlüssel im HTTP-Request vom Client geschickt wurde - eine bestehende Session wiederhergestellt Dabei werden ihre Variablen aus der temporären Datei initialisiert.

Die einzige Variable, die im IDM temporär gespeichert wird, ist das Objekt *$idmApp*, das alle weiteren Objekte enthält.

Das nächste Schlüsselwort ist die Variable *$activityName*. Diese wird, wie der Session-Schlüssel, als ein PUT- oder GET-Parameter im HTTP-Request vom Client gesendet. Sie kann unterschiedliche Funktionen ausführen:

- wenn ihr Wert ‚login' oder ‚login2' ist, dann versucht System einen neuen Benutzer anzumelden. Bei ‚login' werden Benutzername und Paßwort über zwei proprietäre Variablen (*txtUsername* und *txtPassword*) dem System übergeben. Auf diese Weise ist eine ‚Schnellanmeldung', wo Benutzername und Paßwort bereits im Hyperlink enthalten sind, möglich. Bei ‚login2' wird eine standardisierte Anmeldeprozedur gestartet, wie sie in vielen PHP-Handbüchern beschreiben ist.

- bei allen anderen Werten der Variable *$activityName* bedeutet dies das Starten einer neuen Activity mit dem Klassennamen, der in der Variable *$activityName* steht.

```
<?
include("./application.php");
global $idmApp;
session_start();
```

```
$ME = $SCRIPT_NAME:
if (!isset($idmApp)) {
        session_register("idmApp");
        $idmApp = new IdmApplication;
        $idmApp->isLogged = false;
}
if (!isset($activityName)) $activityName = '';
switch ($activityName) {
    case 'login':
        if (isset($txtUsername) &  isset($txtPassword)) {
                $loginSuccess = $idmApp->startWithLogin($txtUsername,$txtPassword);
                $currentUser  = $idmApp->getCurrentUser();
            }
            if ($idmApp->isLogged) {
                $activity = new IdmPrintDesktopActivity;
                $activity->startActivityForApplication($idmApp);
            } else {
                if (isset($currentUser)) unset($currentUser);
                include("templates/header.php");
                include("templates/loginForm.php");
                echo "<h1 ALIGN=center> Login failed ! Check your Username/Password </h1>";
                include("templates/footer.php");
            }
    break;
            case 'login2':
            if (isset($PHP_AUTH_USER) and isset($PHP_AUTH_PW)) {
                    $loginSuccess = $idmApp-
>startWithLogin($PHP_AUTH_USER,$PHP_AUTH_PW);
                    $currentUser  = $idmApp->getCurrentUser();
                }
                if ($idmApp->isLogged) {
                    $activity = new IdmPrintDesktopActivity;
                    $activity->startActivityForApplication($idmApp);
                } else {
                    unset($PHP_AUTH_USER);
                    unset($PHP_AUTH_PW);
                    header( 'WWW-Authenticate: Basic realm="Private"' );
                header( 'HTTP/1.0 401 Unauthorized' );
                    include("templates/header.php");
                    include("templates/loginForm.php");
                    echo "<h1 ALIGN=center> Login failed ! Check your Username/Password
</h1>";
                    include("templates/footer.php");
                    exit;
                }
        break;
    default :
        if (!$idmApp->isLogged) { //Falls nicht eingeloggt versuche erstens sich anzumelden !
                                    if (isset($PHP_AUTH_USER) and isset($PHP_AUTH_PW)) {
                                        $loginSuccess = $idmApp-
>startWithLogin($PHP_AUTH_USER,$PHP_AUTH_PW);
                                        $currentUser  = $idmApp->getCurrentUser();}
                                    else {
                                        unset($PHP_AUTH_USER);
                                        unset($PHP_AUTH_PW);
                                        header( 'WWW-Authenticate: Basic realm="Private"' );
                                    header( 'HTTP/1.0 401 Unauthorized' );
                                        include("templates/header.php");
                                        include("templates/loginForm.php");
                                        echo "<h1 ALIGN=center> Login failed ! Check your
Username/Password </h1>";
                                        include("templates/footer.php");
```

```
                                                exit:}}
        if ($idmApp->isLogged) {
                        if (class_exists($activityName)) {$activity = new $activityName;}
                else {$activity = new IdmPrintDesktopActivity;}
                                                if (!is_subclass_of ($activity,'IdmActivity')) $activity = new
IdmPrintDesktopActivity;

                                                if (isset($HTTP_POST_VARS)) {$activity->postVars =&
$HTTP_POST_VARS;}
                                                if (isset($HTTP_GET_VARS)) {$activity->getVars =&
$HTTP_GET_VARS;}
                                                $activity->startActivityForApplication($idmApp);}
                                        else {
                                                if (isset($currentUser)) unset($currentUser);
                                                include("templates/header.php");
                                                include("templates/loginForm.php");
                                                include("templates/footer.php");}
}
?>
```

5.5.3 Starten einer bestimmten Acitivity

Eine Activity ist die einzige Möglichkeit, irgend etwas im IDM auszuführen. Sie startet über die standardisierte Methode *startForApplication($anApplication)*. Der neuen Activity wird gleich beim Start die aktuelle Applikation übergegeben. In diesem Objekt sind die fast immer gebrauchten Objekte enthalten, wie z.B. der aktuelle Benutzer und einige seiner Eigenschaften. Auf diese Weise spart man sich einen Zugriff auf die Datenbank wegen jener Daten, die immer wieder gebraucht werden.

Wenn man eine neue Funktionalität in IDM beschreiben möchte, muß man eine neue Subklasse der Klasse Activity anlegen. Je nachdem, welches Verhalten man implementieren will, ist die Klasse eine Subklasse der IdmPrintActivity, der IdmPromptActivty, oder einer anderen.

5.5.4 Abstraktion der relationalen Datenbank

Um eine saubere Schnittstelle zwischen der relationalen Datenbank und den Activities zu ermöglichen, wurden die Zwischenobjekte der Klasse IdmModelObject definiert. Diese stellen einerseits die Daten den Activities zur Verfügung, und anderseits greifen sie auf die relationale Datenbank zu.
Die IdmModelObject-Objekte, die auf die Datenbank zugreifen können, tun dies über die Methoden sqlSelect(), sqlInsert(), sqlUpdate() und sqlDelete().

Einige Objekte, die die oben angeführte Funktionalität erfüllen, sind:

- IdmUser – ein Benutzer in der Applikation
- IdmUserGroup – die Benutzergruppe
- IdmFolder – Ordner
- IdmFile - Datei
- IdmHyperlink - Hyperlink
- IdmViewSetting - benutzerspezifische Ansicht
- IdmFolderGroup – Beziehung zwischen Benutzergruppe und Ordner (Berechtigung)

Neben den Datenbankzugriffsfunktionen verstehen diese Objekte zusätzliche Methoden, wie zum Beispiel *IdmFile* (Datei) kann durch die Methode *deleteFromFilesystem()* von dem Server gelöscht werden. Von dieser Methode macht man Gebrauch in der entsprechenden Activity, die für das Löschen einer Datei zuständig ist - *IdmDeleteFileActivity*.

Neben den Objekten, die für den Datenbankzugriff dienen, existieren allgemeine Objekte, wie zum Beispiel *IdmApplication* – Abstraktion der aktuellen Applikation.

5.5.5 ValueHolders

Die Objekte der Klasse IdmValueHolder sind ein Versuch, die gängigen Attribute objektorientiert zu beschreiben. Zum Beispiel wird durch die Klasse *IdmBoolean* versucht, eine Boolean-Variable und ihre möglichen Ausprägungen im HTML zusammenzufassen.

Ein zweites Beispiel ist die *IdmEnumerate*-Klasse. Durch diese Klasse werden in der IDM vorkommenden Aufzählungen beschrieben.

Die *ValueHolders* sind nur ein kleiner Schritt in eine mögliche Richtung, wie man die HTML-Darstellung von *IdmModelObject*-Objekten implementieren könnte.

5.5.6 Dateienstruktur

In diesem Abschnitt wird die entstandene Dateienstruktur und die darin liegende Dateien und ihre Bedeutung erklärt.

/idm
... ist der Hauptordner der gesamten Applikation. Dieser Ordner sollte im Webserver als Einstiegsordner definiert werden. Alle anderen Ordner, die PHP-Seiten beinhalten, sind in diesem Ordner, bzw. in den untergeordneten Ordnern enthalten. Das ist jedoch keine Regel, sondern nur eine Konvention, die zur Übersichtlichkeit beitragen soll.

./idm/index.php ist der zentrale Einstiegspunkt der gesamten Applikation. Jede Seite wird eigentlich durch diese Seite aufgerufen, bzw. aufgebaut. Das Prinzip, wie die Seiten aufgebaut werden, wird im Abschnitt Entwurf erläutert.

```
<?
error_reporting(15);

class Object {
}

$CFG = new Object;

$CFG->dbhost = "linux2";
$CFG->dbname = "idm";
$CFG->dbuser = "mysql";
$CFG->dbpass = "";

$CFG->wwwhost = "linux2";
$CFG->wwwroot = "/www/idm";
$CFG->wwwdownloaddir = "/tmp";
$CFG->dirroot = "/home/www/idm";
$CFG->filedir = "/home/www/idm/document";
$CFG->downloaddir   = "/tmp/idm";

$DB_DEBUG = true;
```

```
$DB_DIE_ON_FAIL = true:

$ME = $SCRIPT_NAME;

require("./lib/modelObject.php");
require("./lib/valueHolders.php");
require("./lib/activities.php");
require("./lib/classes.php");
require("./lib/stdlib.php");
require("./lib/dblib.php");

/* connect to the database */
db_connect($CFG->dbhost, $CFG->dbname, $CFG->dbuser, $CFG->dbpass);

?>
```

./idm/application.php ist die zentrale Konfigurationsdatei. In der werden alle relevanten systemspezifische Einstellungen vorgenommen.

./idm/lib

In diesem Ordner befindet sich der gesamte PHP-Code. Dateien in diesem Ordner betreffen im allgemeinen den Applikationsentwickler und den Frameworkdesigner. Hier werden die Activities, Datenbankzugriffe und spezielle Strukturen als Subklassen von ValueHolders implementiert.

./idm/lib/dblib.php

In dieser Datei sind alle Datenbankfunktionen nochmals abstrahiert. Jede Datenbankzugriffsfunktion, die in der restlichen Applikation verwendet wird, muß hier implementiert werden. Darin enthaltene Funktionen greifen wiederum auf die datenbankspezifischen PHP-Funktionen zu - in diesem Fall auf die MySQL-Funktionen. Durch den Austausch dieser Datei ist es möglich, eine andere relationale Datenbank einzusetzen.

./idm/lib/activities.php

In der activities.php-Datei sind alle Activities (was eine Activity ist, wird im Abschnitt ‚Entwurf' erklärt) implementiert. Das ist die Schlüsseldatei für den Applikationsentwickler. Wenn man eine neue Funktionalität (z.B. Auslesen der Daten aus der Datenbank und ihre Darstellung in einer Seite) hinzufügen will, macht man es in dieser Datei.

./idm/lib/modelObject.php

Diese Datei beinhaltet das objektorientierte Modell, dem eine relationale Datenbank zugrunde liegt. Die Abstrahierung der Tabellen in die Klassen ist nicht starr, d.h. es muß nicht für jede Tabelle eine äquivalente Klasse existieren.

./idm/lib/valueHolders.php

ValueHolder-Subklassen stellen spezielle Strukturen dar, die man entweder in der Datenbank abspeichert oder die man in der Applikation braucht. Alle Subklassen werden in dieser Datei zentral gespeichert.

./idm/templates

In diesem Ordner befinden sich die HTML-Templates in Form von PHP-Dateien. Im Gegensatz zu dem lib-Ordner werden diese Dateien i.a. von Webdesignern angelegt, geändert und gewartet.

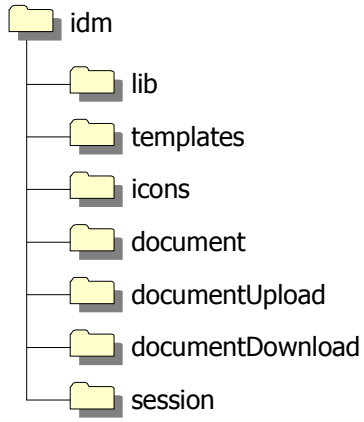


Abbildung 9 IDM - Odnerstruktur

./idm/icons
Hier sind, an einer zentralen Stelle, alle Icons und sonstigen Bilder, die in der Benutzeroberfläche eingesetzt werden, gespeichert.

./idm/documentDownload
Dieses Verzeichnis dient zur temporären Speicherung von Dateien, die der Benutzer von dem Informationssystem herunterladen will. Es ist nur ein Beispiel und muß sich nicht unbedingt an dieser Stelle befinden, es muß aber auf jeden Fall für den Webserver freigegeben werden. In der Konfigurationsdatei (application.idm) kann man einen alternativen Pfad angeben. Dieser ist dann mit der Systemeinstellung in der Datei php.ini gleichzusetzen. Es ist zu beachten, daß die Daten in diesem Verzeichnis mit der Zeit anwachsen können. Daher müssen Inhalte aus diesem Verzeichnis regelmäßig gelöscht werden.

./idm/documentUpload
Dieses Verzeichnis ist ebenfalls nur ein Beispiel. Es beinhaltet die temporär gespeicherten Dateien während eines Upload der Daten auf den Server. Die Einstellung, wo sich dieses Verzeichnis befinden soll, steht in der PHP-Konfigurationsdatei - *php.ini*. Das Verzeichnis sollte nicht durch den Webserver freigegeben werden.

./idm/session
In diesem Verzeichnis sind temporäre Session-Daten abgespeichert. Jedes Mal, wenn sich ein Benutzer erfolgreich anmeldet, wird eine Datei mit dem Session-Schlüssel als Namen erzeugt. In dieser Datei werden dann die Session-spezifischen Daten temporär abgespeichert. Dieses Verzeichnis sollte, wie das Verzeichnis ./idm/documentDownload, regelmäßig ausgeleert werden. Die Information darüber, wo der Pfad zur Speicherung von Session-Daten ist, befindet sich ebenfalls in der PHP-Konfigurationsdatei php.ini.

./idm/document
In diesem Verzeichnis befinden sich alle Dateien, die über den IDM-Server abgespeichert wurden. Dieses Verzeichnis sollte nicht für den Webserver freigegeben sein.

5.6 Datenbankdesign

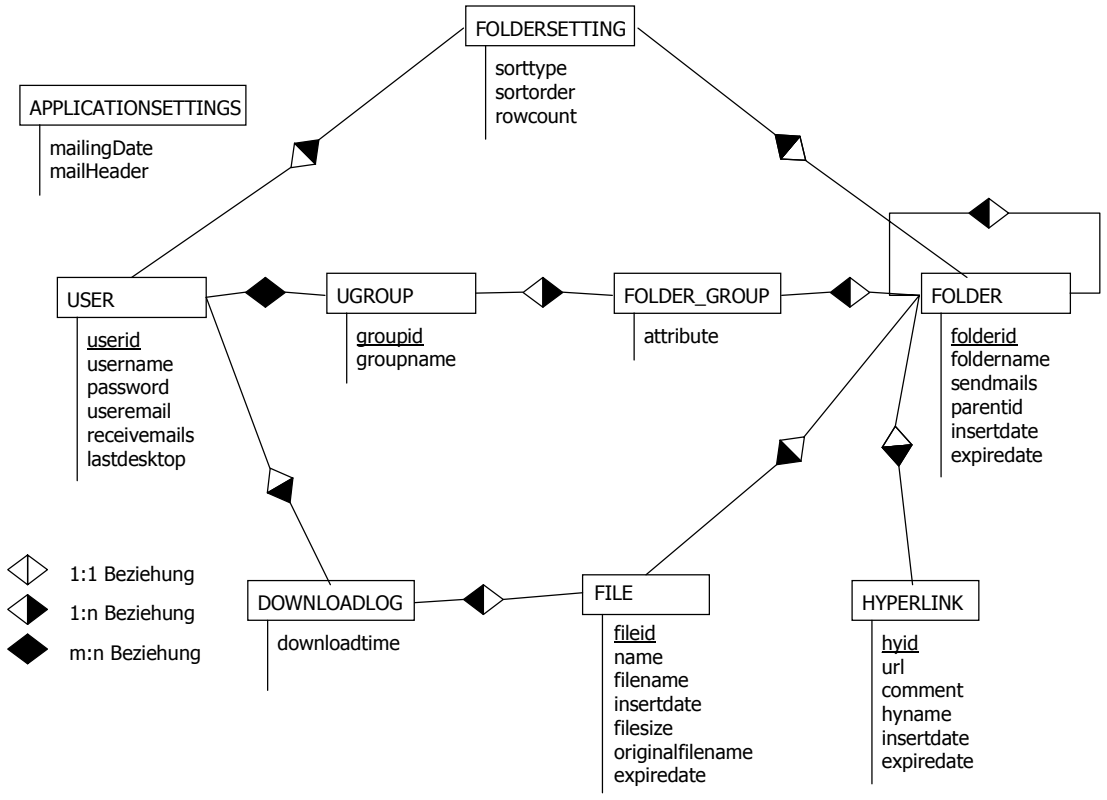


Abbildung 10 EER Diagramm des Datenbankmodells

5.6.1 Bedeutung einzelnen Tabellen

APPLICATIONSETTINGS
In dieser Tabelle werden applikationsübergreifende Parameter gehalten. Man hätte statt dieser Tabelle alle Einträge in der Form von globalen PHP-Variablen im PHP-Code speichern können, hier sind sie aber übersichtlich alle an einer zentralen Stelle abgespeichert, und man greift nur auf jene zu, für die es notwendig ist.

DOWNLOADLOG
In dieser Tabelle werden alle Downloads von dem IDM protokolliert. Dadurch ist es möglich, eine detaillierte Liste von allen Benutzern, die eine bestimmte Datei heruntergeladen hatten, zu erstellen.

FILE
In der FILE-Tabelle sind Verweise auf alle Dokumente, die sich im Informationssystem befinden, abgespeichert. Wichtig: Die Dateien selbst befinden sich im Verzeichnis, das in der Konfigurationsdatei angegeben ist *(./idm/document)*.

HYPERLINK
Der andere Informationstyp, den das Informationssystem behandeln kann, sind die Hyperlinks. Sie werden in dieser Tabelle gespeichert. Die Tabelle ist sehr ähnlich der FOLDER-Tabelle.

FOLDER
Diese Tabelle speichert die Ordner des Informationssystems ab. Ein Ordner kann weitere Ordner beinhalten. Diese Beziehung ist durch den Eintrag parentid abgebildet. Nachdem der Ordner immer den Zeiger auf den ‚oberen' Ordner haben muß, bedeutet das es mindestens einen Eintrag in dieser Tabelle geben muß – den, der auf den Wurzelordner der gesamten Struktur zeigt. Das ist der Eintrag mit der folderid = 1. Dieser Eintrag darf nicht gelöscht werden, in der Applikation geht man davon aus, daß es diesen Eintrag geben muß.

USER
Das Informationssystem ist eine mehrbenutzerfähige Applikation. Alle vorkommenden Benutzer sind in dieser Tabelle abgespeichert.

UGROUP
Die Benutzergruppen werden in dieser Tabelle gespeichert.

USER_GROUP
Eine Benutzergruppe kann beliebig viele Benutzer beinhalten, bzw. ein Benutzer kann beliebig vielen Benutzergruppen zugeordnet sein. Die Beziehung ist in dieser Tabelle abgeleitet.

FOLDER_GROUP
Die zweite Beziehung, die die Berechtigungen abbildet, ist die Beziehung zwischen den Benutzergruppen und den einzelnen Ordnern. Eine Benutzergruppe kann eine bestimmte Berechtigung für einen Ordner haben, andersrum kann ein Ordner vielen Benutzergruppen zugänglich sein. Die Beziehung wurde in dieser Tabelle aufgelöst.

Das entsprechende SQL-Code zur Datenbankerstellung sieht folgendermaßen aus:

```
CREATE TABLE applicationSettings (
  mailingDate varchar(20) NOT NULL,
  mailHeader text NOT NULL
);

CREATE TABLE downloadlog (
  userid int(11) NOT NULL,
  fileid int(11) NOT NULL,
  downloadtime timestamp(14),
  KEY userid (userid),
  KEY fileid (fileid),
  KEY downloadtime (downloadtime)
);

CREATE TABLE file (
  fileid int(11) NOT NULL auto_increment,
  name varchar(40) NOT NULL,
  filename varchar(40) NOT NULL,
  folderid int(11) NOT NULL,
  insertdate date,
  filesize int(11),
  originalfilename varchar(40) NOT NULL,
  userid int(11) DEFAULT '1' NOT NULL,
  expiredate date,
  PRIMARY KEY (fileid),
  KEY folderid (folderid),
```

```
  KEY insertdate (insertdate)
);

CREATE TABLE folder (
  folderid int(11) NOT NULL auto_increment,
  foldername varchar(40) NOT NULL,
  sendmails int(1) DEFAULT '1' NOT NULL,
  parentid int(11) NOT NULL,
  insertdate date,
  expiredate varchar(20),
  PRIMARY KEY (folderid),
  KEY parentid (parentid)
);

CREATE TABLE folder_group (
  folderid int(11) NOT NULL,
  groupid int(11) NOT NULL,
  attribute int(2) NOT NULL,
  KEY groupid (folderid),
  KEY userid (groupid)
);

CREATE TABLE foldersetting (
  folderid int(11) NOT NULL,
  userid int(11) NOT NULL,
  sorttype tinyint(4) NOT NULL,
  sortorder tinyint(4) NOT NULL,
  rowcount smallint(6) NOT NULL,
  KEY folderid (folderid),
  KEY userid (userid)
);

CREATE TABLE hyperlink (
  hyid int(11) NOT NULL auto_increment,
  url mediumtext NOT NULL,
  comment mediumtext,
  folderid int(11) NOT NULL,
  hyname varchar(40) NOT NULL,
  insertdate date,
  expiredate date,
  PRIMARY KEY (hyid),
  KEY folderid (folderid)
);

CREATE TABLE ugroup (
  groupid int(11) NOT NULL auto_increment,
  groupname varchar(40) NOT NULL,
  PRIMARY KEY (groupid)
);

CREATE TABLE user (
  userid int(11) NOT NULL auto_increment,
  username varchar(40) NOT NULL,
  password varchar(10) NOT NULL,
  useremail varchar(40) NOT NULL,
  receivemails int(1) DEFAULT '1' NOT NULL,
  lastdesktop varchar(8) NOT NULL,
  PRIMARY KEY (userid)
);

CREATE TABLE user_group (
  groupid int(11) NOT NULL,
```

```
  userid int(11) NOT NULL.
  KEY groupid (groupid),
  KEY userid (userid)
);
```

5.7 Betrieb und Wartung

5.7.1 Voraussetzungen

Der Einsatz der IDM-Applikation ist vorerst am Institut für Industrielle Elektronik und Materialwissenschaften an der Technischen Universität Wien vorgesehen. Die gesamte Applikation ist letztendlich eine Ansammlung von Dateien, die sich an den entsprechenden Stellen am Server befinden sollten. Außerdem gibt es noch einige Voraussetzungen für die Inbetriebnahme der Applikation:

Server

- Ein netzwerkfähiges Betriebssystem mit TCP/IP Unterstützung , am Institut ist das ein Linux Rechner, Distribution Red Hat 6.1
- Apache Web-Server, ab Version 1.13
- Relationale Datenbank MySQL, ab Version 3.23.x
- Installierter PHP Interpreter und seine Anbindung an dem Apache Web Server (etwa durch ein Modul), ab Version 4.04

Client
Clientseitig reicht ein beliebiger, zeitgemäßer Rechner mit einem Browser. Empfohlen sind ein Netscape ab Version 4 oder Internet Explorer, ebenfalls ab Version 4.

5.7.2 Installation

Zu beachten bei der Installation ist, daß die angelegten Ordner entsprechenden Benutzerrechte besitzen müssen. Dem ./idm Ordner müssen die Leserechte für den Benutzer, unter dem der Webserverprozeß läuft (üblicherweise: *http* oder *httpd*), zugeordnet sein. Das Hauptverzeichnis, in dem alle Dokumente gespeichert werden, muß für diesen Benutzer Schreib- und Leserechte freigegeben haben. Weiters müssen die Verzeichnisse zur temporären Abspeicherung der Sessiondaten und Dateiuploads auch für diesen Benutzer mit Schreb- und Leserechten markiert sein.

Bei der MySQL Datenbank muß man beachten, daß die Rechte für die Anbindung an die Datenbank vorhanden sind. Die Verbindungsparameter sind einerseits in der Konfigurationsdatei *application.php* anzugeben, anderseits muß man in den Systemtabellen der MySQL Datenbank die entsprechende Verbindungsparameter setzen.

5.7.3 Temporäre Dateien am Server

Für die Benutzer-Authentifizierung werden die Session-Schlüssel eingesetzt. Diese Schlüssel werden mit jeder erfolgreichen Anmeldung lokal am Server als Dateien abgespeichert. In diesen Dateien werden noch die benutzerspezifische Einstellungen in Form von serialisierten Objekten abgelegt. Wenn der Benutzer an seinem Client die Cookie-Unterstützung eingeschaltet hat, werden immer wieder dieselben Session-Schlüssel, bzw. Dateien verwendet. Anderseits, wenn der Benutzer die Cookies ausgeschaltet hat, wird bei jeder

neuen Anmeldung eine neue Datei am Server erstellt. Neben diesen gibt es noch eine zweite Art der temporären Dateien, und zwar die Dateien, die dem Benutzer zur Verfügung gestellt werden, wenn er sich zum Herunterladen einer Datei entscheidet. Die Benutzer haben keinen Zugriff auf das Verzeichnis, wo die gesamten Dateien zentral abgespeichert sind, sondern, wenn er eine bestimmte Datei herunterladen will, wird diese Datei aus diesem Verzeichnis in ein temporäres kopiert. Das Zielverzeichnis muß von dem Webserver freigegeben werden, weil dieser Vorgang ebenfalls auf dem HTTP-Protokoll basiert. Nach dem erfolgreichen Download kann diese temporäre Datei ebenfalls gelöscht werden.

Die Dateien, die nicht mehr verwendet werden, sollten daher regelmäßig vom Server gelöscht werden. Für diesen Vorgang gibt ein Löschskript - *cleanup.sh*, das sich im *./idm/config/* Verzeichnis befindet. Diese Datei sollte angepaßt, und dann regelmäßig ausgeführt werden. Die beste Lösung ist es, das Starten dieser Datei dem *cron*-Daemon zu überlassen.

Folgende Pfade sollten in der Datei angepaßt werden:

- Der Pfad zur Speicherung der Session-Schlüssel. Diesen Pfad gibt man in der allgemeinen *php.ini* Datei an, und von dort kann man die Einstellung entnehmen, bzw. vornehmen.
- Der Downloadpfad, der in der Konfigurationsdatei *application.php* unter dem Eintrag: *$CFG->downloaddir* abgespeichert ist.

```
sessionKeyPath="/tmp/idm/session"
downloadDirPath="/tmp/idm"

# Lösche alle Session-Schlüssel, die in den letzten 15 Tagen nicht zugegriffen wurden
find $sessionKeyPath -type f -name 'sess*' -daystart -not  -mtime -15 -print0 | xargs --null --no-run-if-empty rm -fr

# Lösche alle temporäre Download-Dateien älter als 2 Tage
find $downloadDirPath -type d -name '*.dir' -daystart -not -mtime -2 -print0 | xargs --null --no-run-if-empty rm -fr
```

Die Datei *cleanup.sh* sieht folgendermaßen aus:

5.7.4 Webserver

Der Webserver ist das Interface zum Internet. Daher können eventuelle Mißbräuche oder Angriffe auf den Rechner sich vor allem auf den Webserver beziehen.
Es gilt daher, den Webserver immer auf dem neuesten Stand zu halten, mögliche Sicherheitslücken zu suchen und alle freigegebenen Patches einzuspielen.
Die Auflistung der Verzeichnisse sollte ausgeschaltet sein, insbesondere des temporären Verzeichnisses, das zum Download der Dateien eingerichtet ist.

IDM kann auch über den verschlüsselten HTTPS-Protokoll betrieben werden. Dazu muß man die SSL-Unterstützung für den Apache-Server installieren und den Rechner entsprechend konfigurieren.

Genauso wie für den Webserver, sollten auch für den gesamten Server die Sicherheitsaspekte stets vor den Augen gehalten werden.

5.8 Eingesetzte Entwicklungstools

5.8.1 CVS

CVS ist ein Versionsverwaltungssystem.
Es lassen sich damit Projekt jeglicher Art und Größe, nicht nur Softwareprojekte, verwalten und kontrollieren.
Jeder Schritt des Projektes, bzw. jede Projektversion kann wiederhergestellt werden, wenn sie im CVS gespeichert wurde.
Außerdem ist es theoretisch möglich, mit einer fast unbegrenzten Anzahl von Entwicklern zu arbeiten, die über den gesamten Globus verteilt sind.

WinCVS ist ein GUI-orientierter CVS Client.

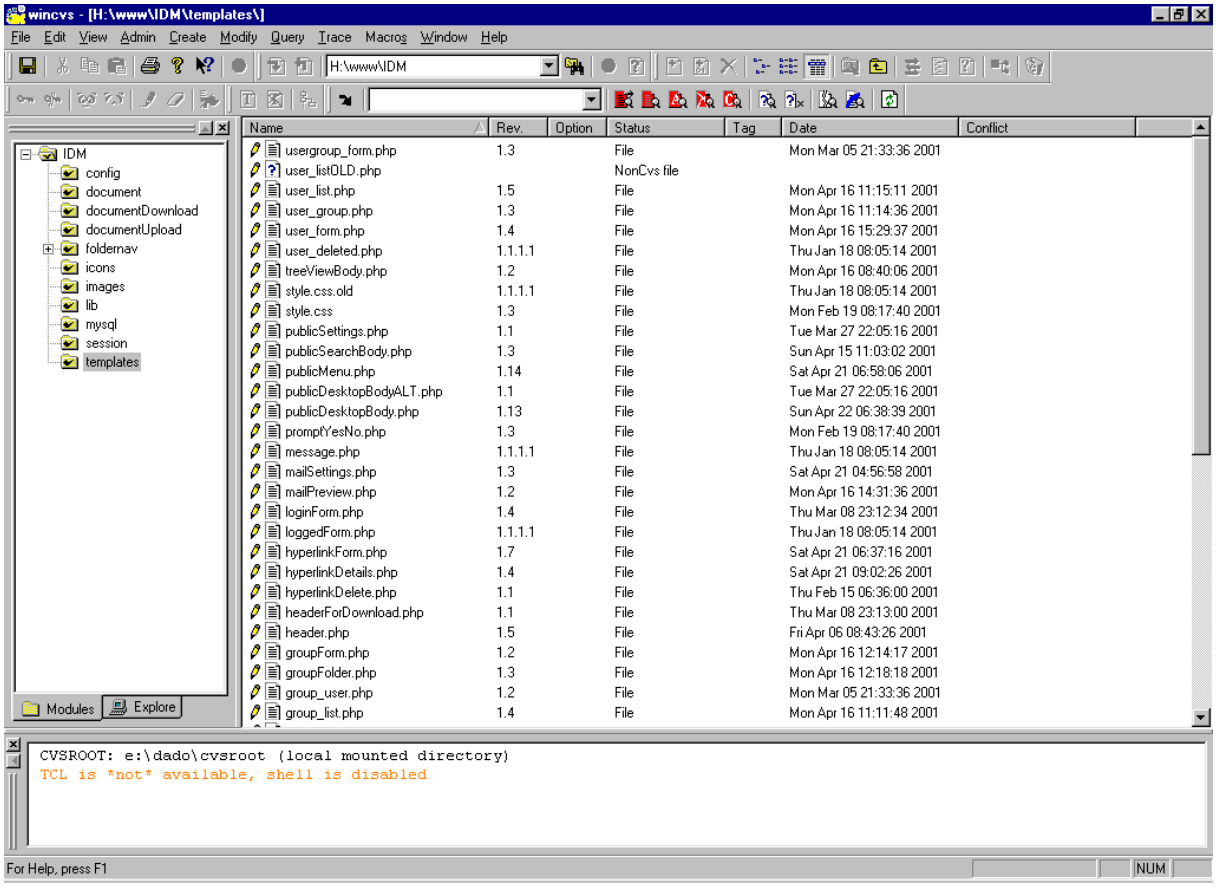


Abbildung 11 CVS Umgebung (Windows 98)

5.8.2 PhpMyAdmin

phpMyAdmin ist eine in PHP geschriebene Verwaltungsoberfläche für die relationale Datenbank MySQL. phpMyAdmin ist ein webbasierter Client.

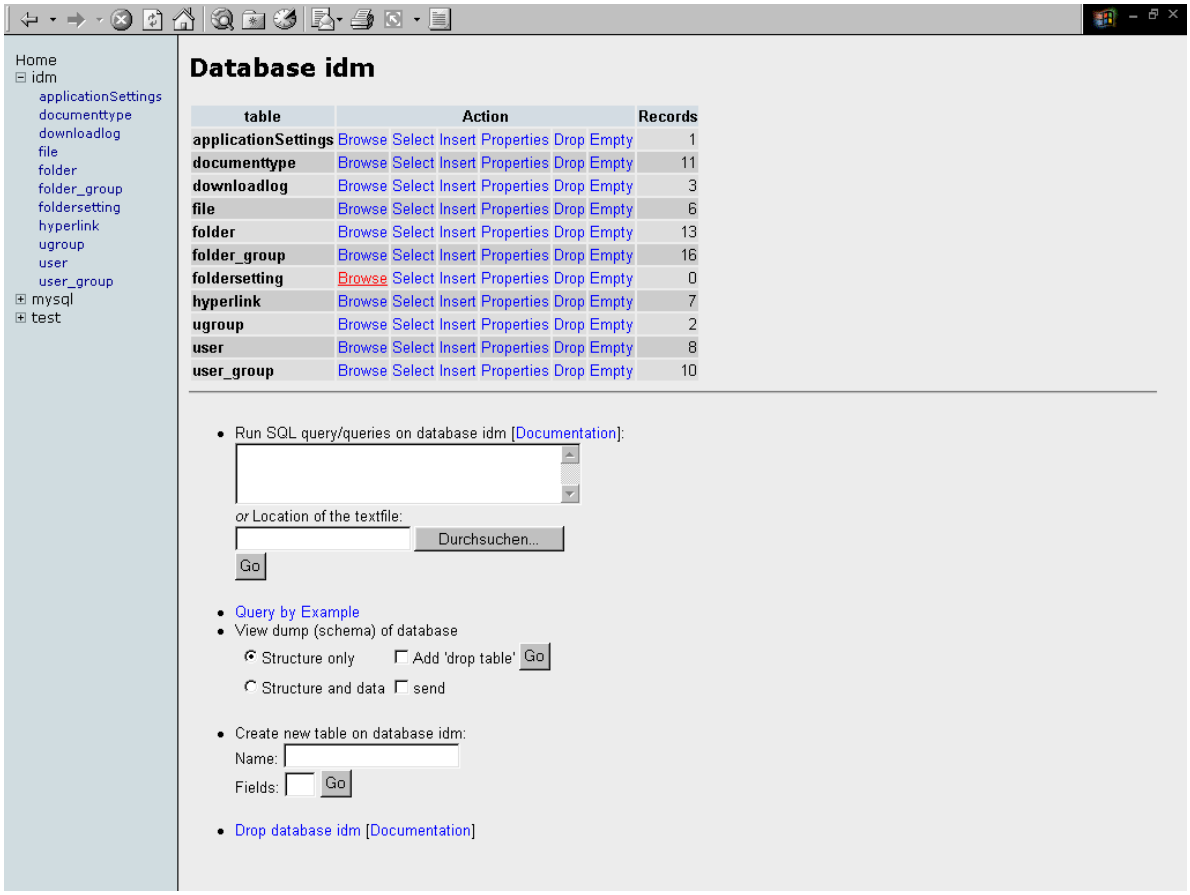


Abbildung 12 phpMyAdmin - Benutzeroberfläche

5.8.3 Allaire Homesite

Das Produkt Homesite der Firma Allaire ist ein Werkzeug zur Erstellung von Webseiten. Es sind HTML-Kenntnisse nicht nur nötig, sondern Voraussetzung. Anfänger, wie auch den Profi unterstützt Homesite mit guter Benutzerführung, einfachen Hilfsmitteln und produktivitätssteigernden Features.
Homesite ist ein sehr verbreitetes HTML-Entwicklungswerkzeug.

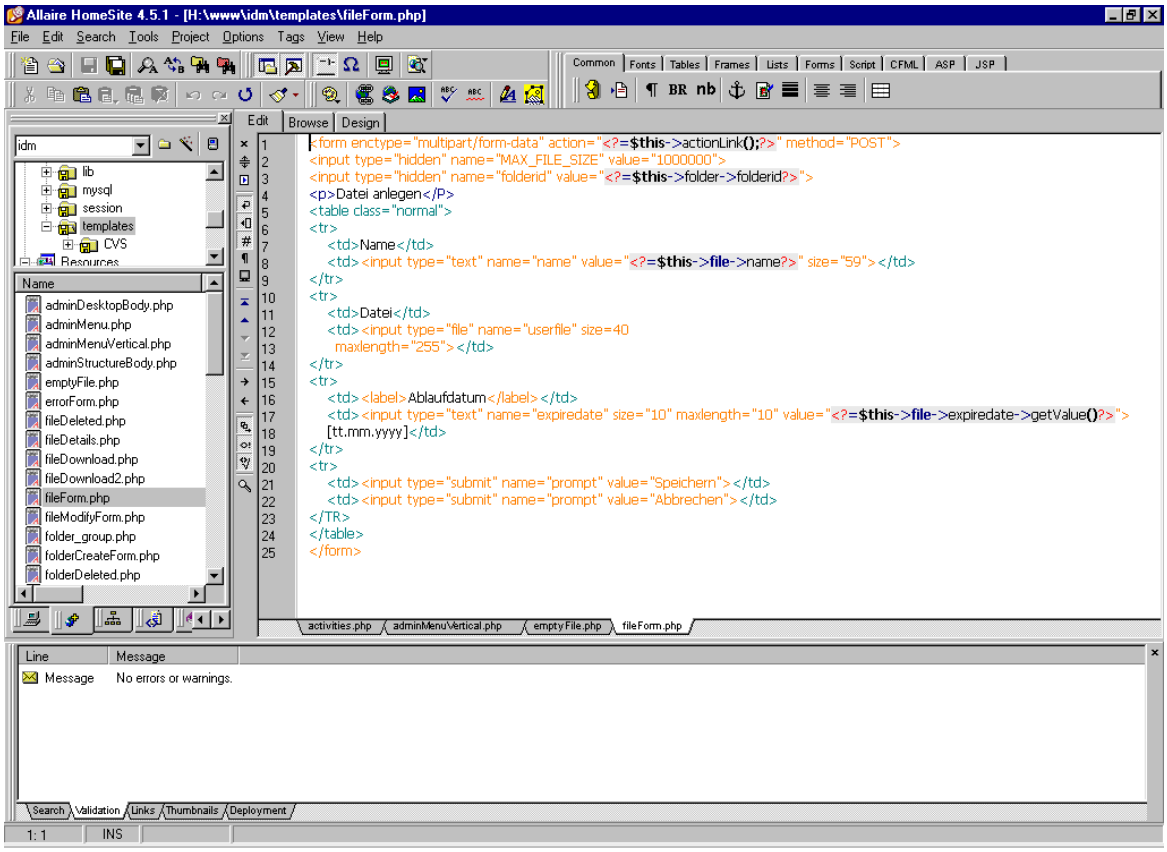


Abbildung 13 Allaire Homesite Entwicklungsumgebung

5.8.4 PHP Coder

PHP Coder eines der seltenen PHP Entwicklungswerkzeuge. Es ist noch in der Entwicklungsphase, läuft aber bereits sehr stabil. Sein Hauptmerkmal ist der Klassenbrowser. Nachdem die Entwicklung der vorliegenden Applikation vollständig objektorientiert ist, ist der Klassenbrowser ein sehr nützliches Werkzeug.

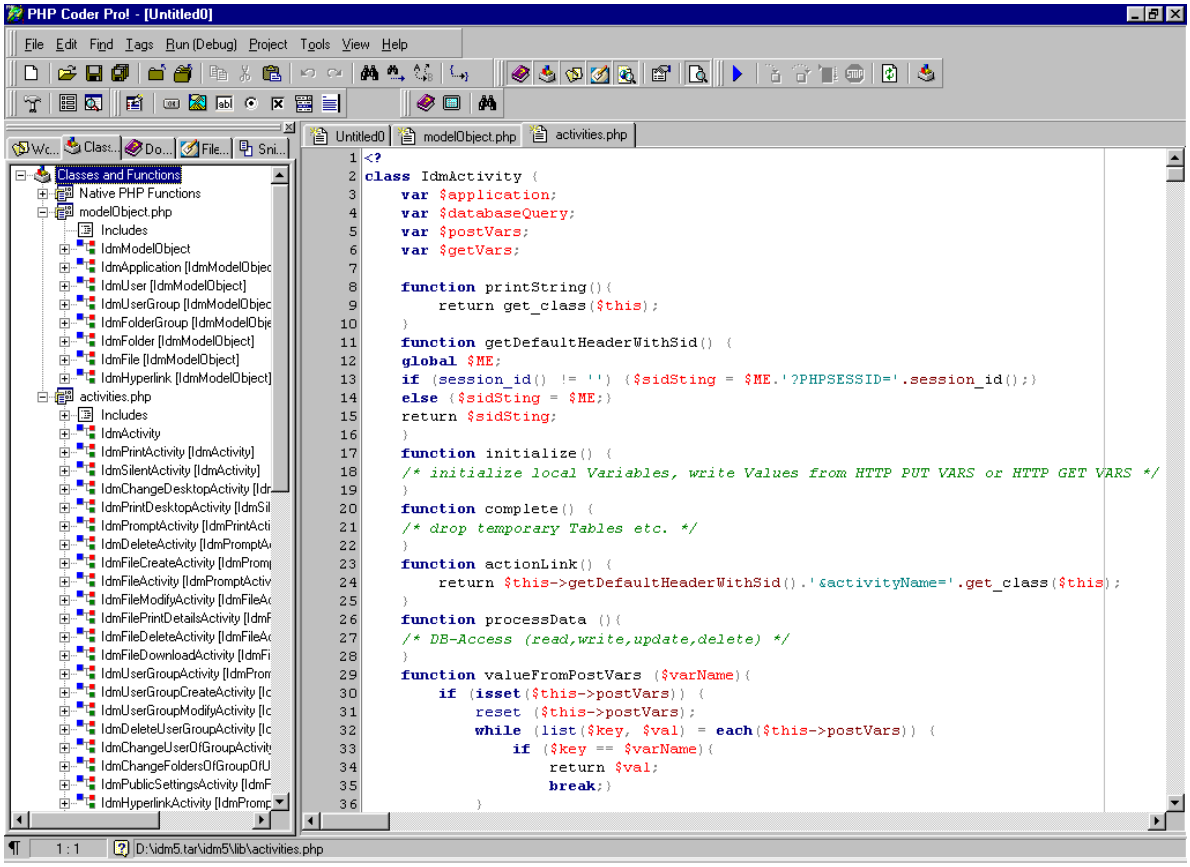


Abbildung 14 PHP Coder Entwicklungsumgebung

5.8.5 IDM – Internet-based document Management

IDM ist die aktuelle Applikation. Weitere Details und Features stehen im Abschnitt ‚Benutzerhandbuch'.

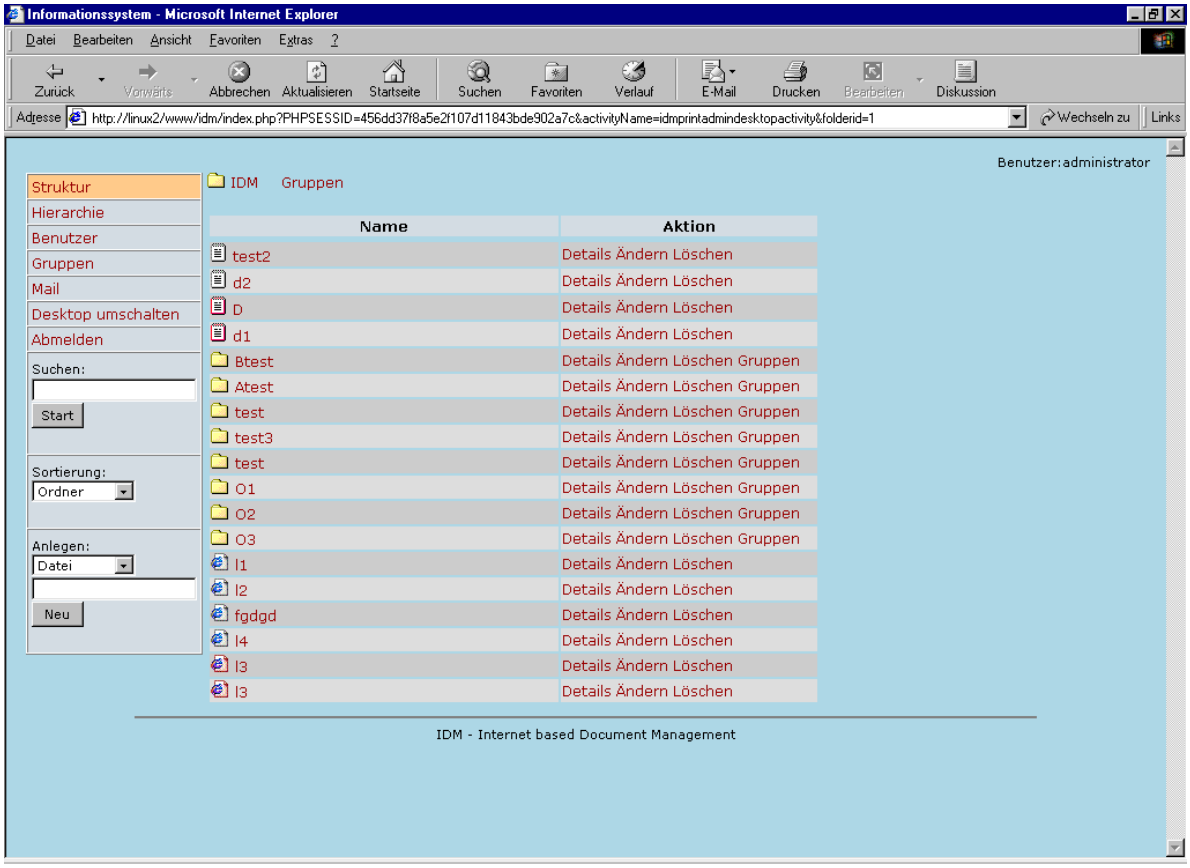


Abbildung 15 IDM - Applikation

6 Benutzerhandbuch

Dieses Handbuch ist eine kurze Einführung in das IDM (Internet-based Document Management System), die im Rahmen dieser Diplomarbeit entwickelte Dokumentenmanagement - Applikation.

Im IDM unterscheidet man prinzipiell zwischen zwei Benutzergruppen:

- Administratoren
- normale Benutzer.

Jeder Benutzer, der der Gruppe ‚Administrator' zugeordnet ist, wird zu einem Administrator.

Administratoren können, neben Aufgaben, die den Standard-Benutzer zur Verfügung stehen, einige zusätzliche Tasks vornehmen.

Durch die vorliegende Benutzeraufteilung ergibt sich auch eine Aufteilung der Arbeitsumgebung. Im IDM findet man daher zwei unterschiedliche Arbeitsumgebungen:

- Administrationsumgebung
- Standardumgebung

Im vorliegendem Text werden die Einträge im IDM – Ordner, Dateien und Hyperlinks – gemeinsam als *Informationseinträge* bezeichnet.

6.1 Anmeldung

IDM ist eine mehrbenutzerfähige Applikation. Das Starten der Applikation folgt erst nach einer erfolgreichen Anmeldung.
Wenn der Benutzer ein Administrator ist, hat er die Möglichkeit, zwischen zwei Arbeitsumgebungen zu wechseln. IDM führt den Benutzer zu dem zuletzt verwendeten Arbeitsumgebung. Wenn der Benutzer die Cookie-Unterstützung in seinem Browser eingeschaltet und sich nicht nach der letzten Session abgemeldet hat, bleibt ihm die neuerliche Anmeldung erspart, und das System öffnet automatisch seine Arbeitsumgebung, sobald er die entsprechende HTTP-Adresse in seinem Browser eingegeben hat.

Abbildung 16 Anmeldedialog

6.2 Administrationsumgebung

Nur die Benutzer, die der Gruppe ‚Administratoren' zugeordnet sind, haben einen Zugang zu der Administrationsumgebung.

IDM bietet ein detailliertes Schreib- und Leseberechtigungssystem. Das bedeutet, daß ein bestimmter Benutzer nur bestimmte Ordner lesen oder ändern darf. Dieses Berechtigungssystem gilt jedoch nur in der Standardumgebung. In der Administrationsumgebung gelten keine Rechtseinschränkungen!

Aus dieser Umgebung kann man folgende Aufgaben erledigen:

- Benutzer anlegen / ändern / löschen
- Benutzergruppe anlegen / ändern / löschen
- Benutzer einer Gruppe zuordnen / aus einer Gruppe löschen
- Versand von E-Mails
- Ordner anlegen / ändern / löschen
- Dateien anlegen / ändern / löschen
- Hyperlinks anlegen / ändern / löschen

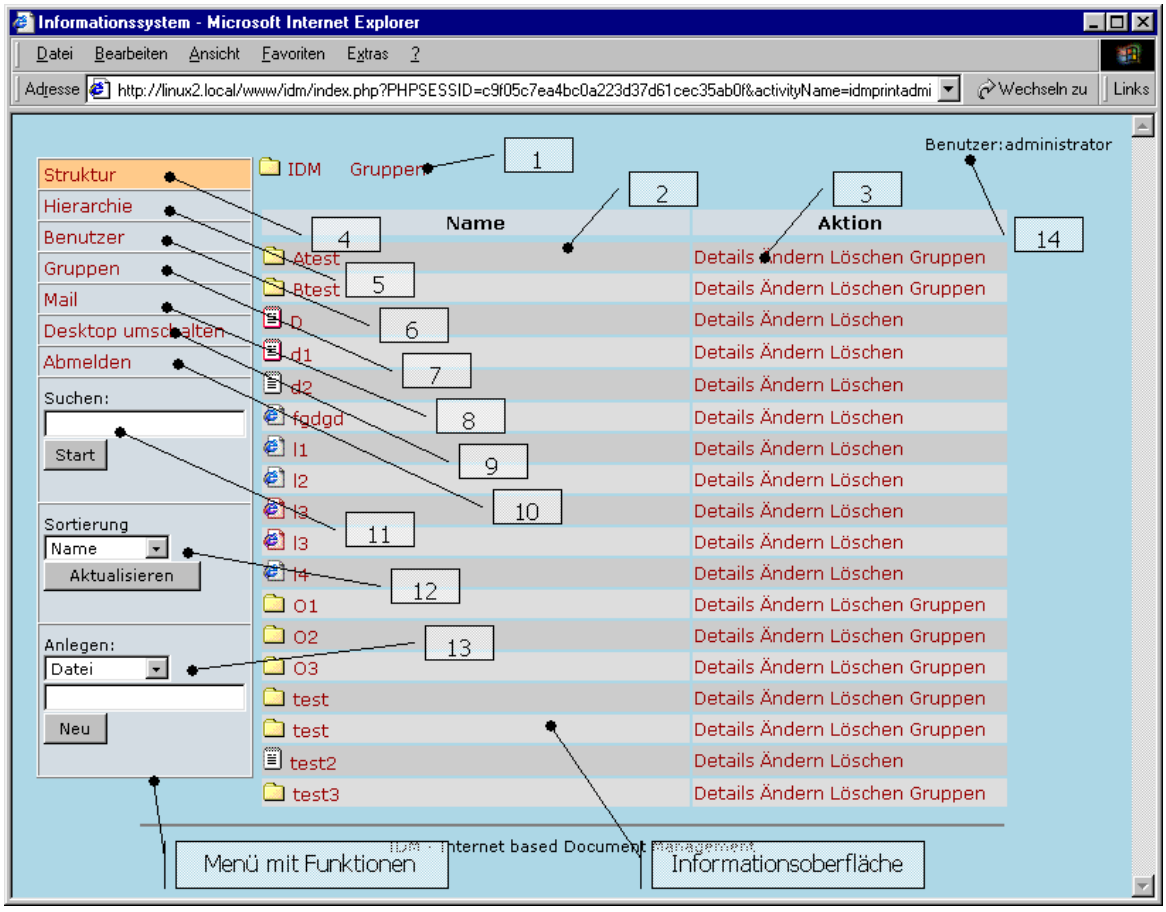


Abbildung 17 Administratordesktop

Die Administrationsumgebung kann man in zwei große Einheiten aufteilen: das Menü und die Informationsoberfläche. Die Informationsoberfläche ist Kontext-abhängig, d.h. je nachdem,

welches Menü ausgewählt wurde, erscheinen hier unterschiedliche Informationen, wie zum Beispiel die Ordnerliste, die Benutzerliste oder die Liste aller Benutzergruppen.

In der Abbildung 17 vorkommende Links und ihre Bedeutung werden hier kurz genannt:

1-3 Informationsoberfläche, wenn man den Menüpunkt ‚Struktur' anklickt

1. Der aktuelle Ordner, bzw. der Pfad zu diesem Ordner
2. Liste einzelner Informationseinträge
3. Aktionsleiste, jede von dieser einzelnen Aktionen bezieht sich auf den links stehenden Informationseintag

4 – 13 Menü mit Funktionen

4. Auflistung der aktuellen Informationseinträge
5. Einblendung der hierarchischen Ansicht der gesamten Datenstruktur, und gibt dem Benutzer die Möglichkeit, schnell einen bestimmten Ordner auszuwählen.
6. Benutzerverwaltung – Auflistung aller Benutzer
7. Gruppenverwaltung – Auflistung aller Benutzergruppen
8. Versand von E-Mails
9. Umstieg auf die Standardumgebung
10. Abmeldung beim IDM
11. Suche von Informationseinträgen nach einem Begriff
12. Ansichtseinstellungen
13. Allgemeines Anlage – Formular, dient zur schnellen Anlage eines Benutzers, einer Gruppe, oder eines Informationseintrags
14. Angemeldeter Benutzer

6.3 Standardumgebung

Die Standardumgebung ist die allgemeine Benutzerumgebung, zu der alle Benutzer einen Zugang haben. Der Unterschied zwischen diesen zwei Umgebungen liegt vor allem in den Menüs und darin angebotenen Funktionen.

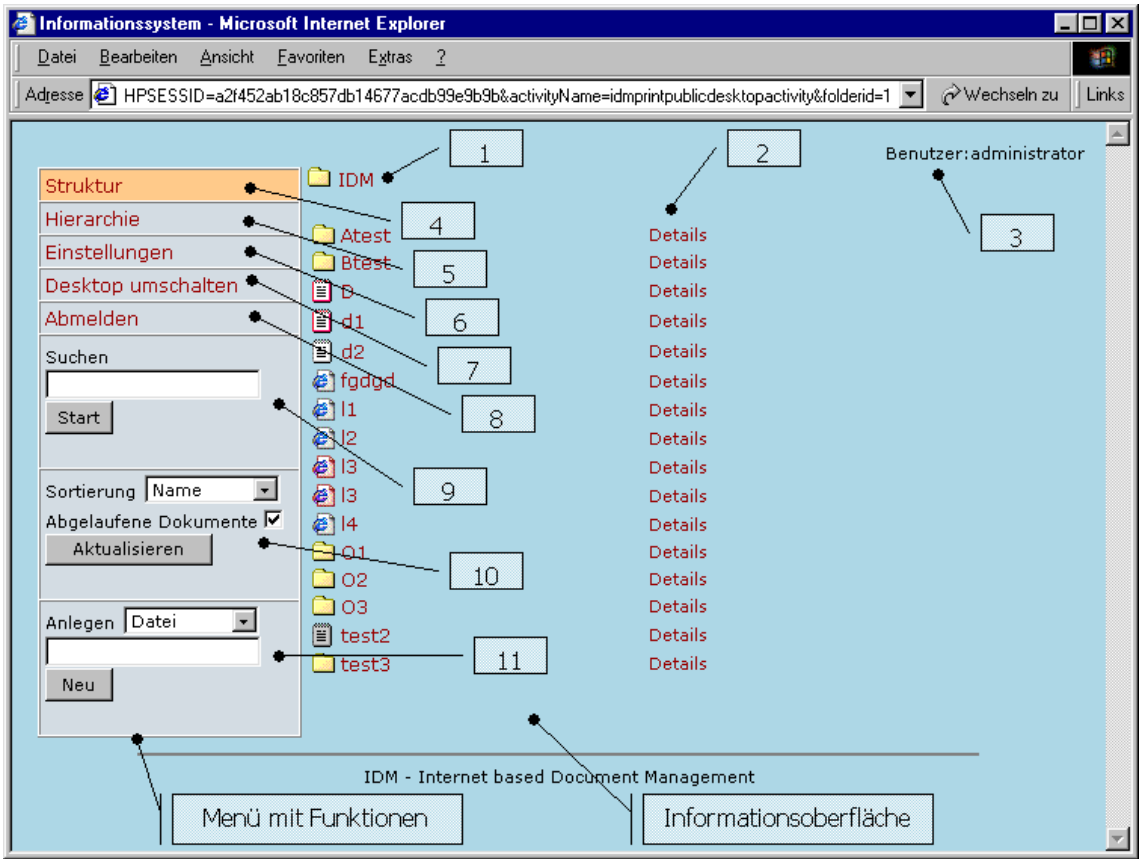


Abbildung 18 Standarddesktop

In dem Standarddesktop (Abbildung 18) vorkommende Links und ihre Bedeutung werden hier kurz genannt:

1-2 *Informationsoberfläche*

1. Der aktuelle Ordner, bzw. der Pfad zu diesem Ordner
2. Liste einzelner Informationseinträge mit dem Link auf Details
3. Angemeldeter Benutzer

4 – 11 *Menü mit Funktionen*

4. Auflistung aktueller Informationseinträge
5. Einblendung der hierarchischen Ansicht der gesamten Datenstruktur; gibt dem Benutzer die Möglichkeit, schnell einen bestimmten Ordner auszuwählen
6. Benutzerspezifische Einstellungen
7. Umstieg auf die Administrationsumgebung

8. Abmeldung beim IDM
9. Suche von Informationseinträgen nach einem Begriff
10. Ansichtseinstellungen
11. Allgemeines Anlage – Formular, dient zur schnellen Anlage von Ordnern, Dateien oder Hyperlinks

6.4 Struktur *(Administratordesktop [4]; Standarddesktop [4])*

Durch diese Funktion bekommt man eine Liste der Einträge des Wurzelordners. Beim Neustart der Applikation wird ebenfalls der Inhalt dieses Ordners aufgelistet.

Der Wurzelordner der gesamten Struktur heißt IDM und kann nicht gelöscht oder geändert werden
In der Administrationsumgebung wird zusätzlich die Möglichkeit für die Änderung der Berechtigungen des Wurzelordners (IDM) angeboten.

6.5 Informationsoberfläche

Die Informationsoberfläche liefert eine Liste der aktuellen Informationseinträge und die Links für die erlaubten Funktionen, die man an diesen Einträgen ausführen kann.

Aus der Standardumgebung kann folgende Funktionen ausführen:

- Dateien holen
- Hyperlinks öffnen
- untergeordnete Ordner öffnen
- Details einzelner Einträge ansehen

Aus der Administrationsumgebung heraus kann man zusätzlich zu diesen weitere Funktionen ausführen:

- Informationseinträge ändern und löschen
- Benutzerberechtigungen einzelner Ordner ändern
- Benutzer auflisten und Funktionen starten
- Benutzergruppen auflisten und Funktionen starten.

6.5.1 Ordnerpfad *(Administratordesktop [1]; Standarddesktop [1])*

Die Informationseinträge sind in einer beliebig tiefen hierarchischen Liste abgespeichert. Um die Übersichtlichkeit zu gewähren, zeigt der Ordnerpfad den gesamten Pfad von dem Wurzel- bis zu dem aktuellen Ordner.

6.5.2 Liste aktueller Einträge *(Administratordesktop [2]; Standarddesktop [2])*

Die Liste der Einträge zeigt alle Informationseinträge des aktuellen Ordners. In der Administrationsumgebung sind alle Einträge dargestellt, während in der Standardumgebung nur jene aufgelistet sind, für welche der aktuelle Benutzer mindestens ein Leserecht hat.

Außerdem sind die Einträge mit Icons versehen, die eine zusätzliche farbliche Codierung besitzen:

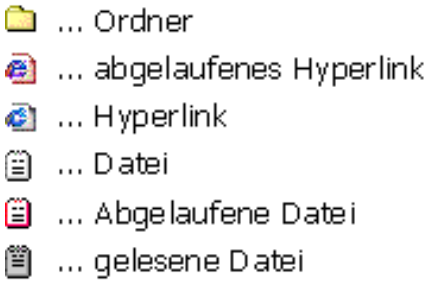


Abbildung 18 Bedeutung der einzelnen Icons

Hyperlinks und Dateien können ein Ablaufdatum besitzen. Wenn dieses Datum abgelaufen ist, werden sie nur bedingt dargestellt (siehe Ansichtseinstellungen), und werden zusätzlich mit einem anderen Icon markiert.
In der Administrationsumgebung werden stets ALLE Dateien bzw. Hyperlinks angezeigt, aber noch immer entsprechend dargestellt.
In der Standardumgebung unterscheidet man zusätzlich zwischen Dateien, die der angemeldete Benutzer bereits heruntergeladen hat, und jenen die er noch nicht gesehen hat. Bei diesen beiden Arten darf das Ablaufdatum nicht abgelaufen sein, ansonsten wird die Datei als abgelaufen markiert.

Wenn man die einzelnen Einträge ‚anklickt' startet man, je nach dem von welchem Typ der Eintrag ist, folgende Funktionen:

- wenn der Eintrag ein Ordner ist, 'öffnet' man diesen Ordner und die Liste zeigt nun seinen Inhalt
- wenn der Eintrag eine Datei ist, wird das Download dieser Datei in einem neuen Browserfenster gestartet
- wenn der Eintrag ein Hyperlink ist, wird ein neues Browserfenster mit dem URL von dem ausgewählten Hyperlink geöffnet

6.6 Benutzerspezifische Einstellungen *(Standarddesktop [6])*

Jeder Benutzer hat die Möglichkeit, sich jederzeit aus der Mailingliste auszuschließen, bzw. sich wieder in die Mailingliste einzuschließen. Außerdem kann er seine E-Mail Adresse ändern.

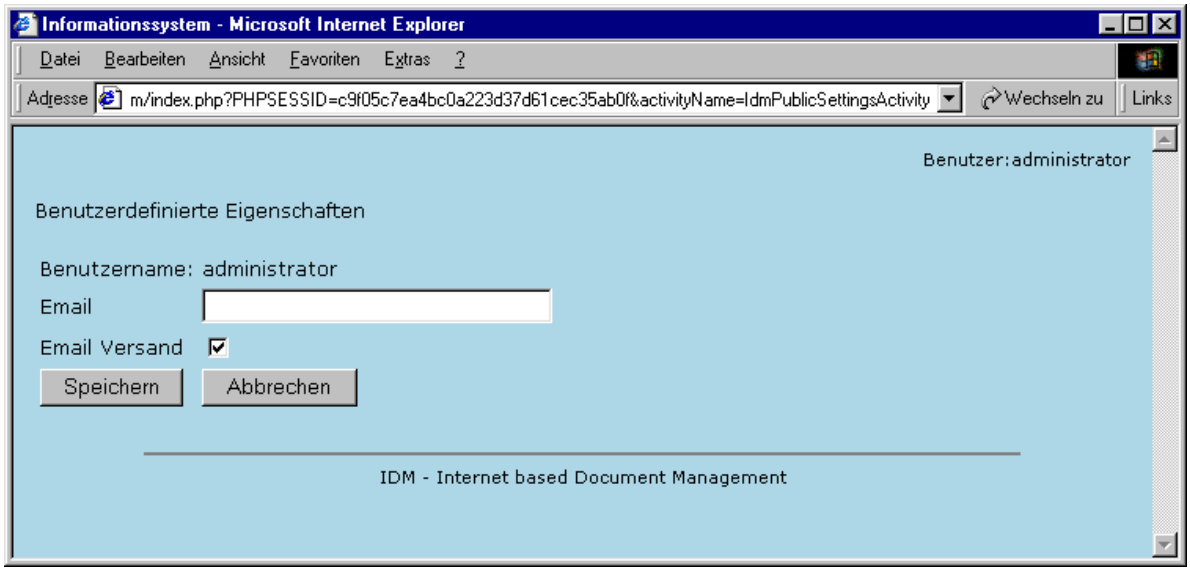


Abbildung 19 Benutzerspezifische Einstellungen

6.7 Ansichtseinstellungen *(Administratordesktop [12]; Standarddesktop [10])*

Jeder Benutzer hat die Möglichkeit, die Sortierung der aktuellen Eintragsliste zu ändern. Derzeit sind folgende Sortierungen möglich: nach Typ der Information, nach dem Anlagedatum und nach der Bezeichnung der Information.
In der Standardumgebung kann man noch die Einblendung der abgelaufenen Einträgen ein- bzw. ausschalten.
Die Ansichtseinstellung ist benutzerabhängig. Sie wird vom System abgespeichert, und bei der nächster Anmeldung entsprechend gesetzt.

6.8 Hierarchie *(Administratordesktop [5]; Standarddesktop [5])*

Der Nachteil einer Standard-Ansicht ist, daß sie nur eine (aktuelle) Ebene anzeigen kann. Bei einer tief verschachtelten Struktur kann man schnell die Übersicht verlieren. Die Hierarchie stellt die gesamte Ordnerstruktur in einer hierarchischen Liste dar, und hilft auf diese Art dem Benutzer, schnell zu einem bestimmten und i.a. schwer auffindbarem Ordner zu kommen.

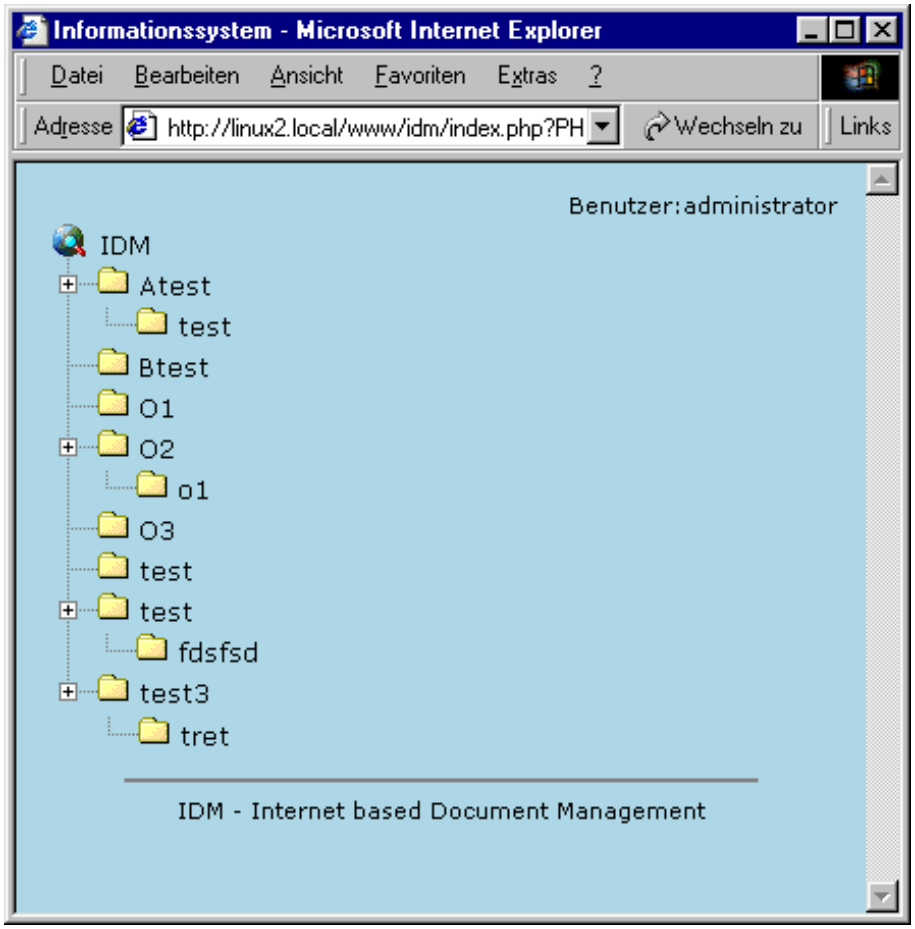


Abbildung 20 Hierarchieansicht

6.9 Anlage-Formular *(Administratordesktop [11]; Standarddesktop [13])*

Das Anlage-Formular ist das universelle Werkzeug, um ein beliebiges Objekt im IDM zu erstellen. Über dieses Formular können folgende Objekte angelegt werden:

- Benutzer (nur Administrationsumgebung)
- Benutzergruppe (nur Administrationsumgebung)
- Ordner
- Datei
- Hyperlink

Aus der Administrationsumgebung kann man jederzeit ein beliebiges Objekt anlegen.
In der Standardumgebung kann man einen neuen Informationseintrag nur dann anlegen, wenn der Benutzer ein Schreibrecht für den aktuellen Ordner besitzt. Wenn er nur ein Leserecht hat, wird das Anlage-Formular aus dem Menü ausgeblendet.

6.10 Verwaltung der einzelnen Informationseinträge

Die Ordner, Dateien und Hyperlinks können sowohl aus der Administrations- als auch aus der Standardumgebung angelegt werden. Dies geschieht über das oben beschriebene Anlage-Formular.

Die Änderungen bzw. Löschung einzelner Einträge ist nur aus der Administrationsumgebung möglich.

6.10.1 Ordner

Neben seinem Namen hat der Ordner zwei weitere Parameter, die jederzeit änderbar sind:

- Email-Versand – mit diesem Parameter entscheidet man, ob die darunterliegenden Einträge in den Email-Versand-Funktion einbezogen wurden oder nicht.

- Standardablaufdatum – alle Einträge, die ohne ein bestimmtes Ablaufdatum in diesem Ordner abgelegt werden, bekommen ein Ablaufdatum, das aus dem Anlagedatum erhöht um das Zeitintervall, das hier angegeben wurde, berechnet wird.

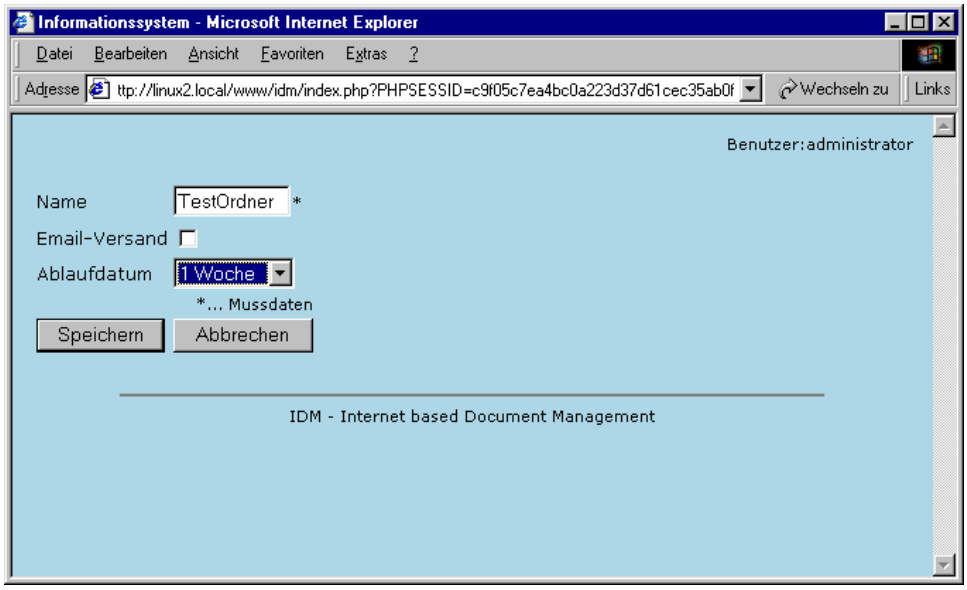


Abbildung 21 Ordneranlage, -änderung

6.10.2 Datei

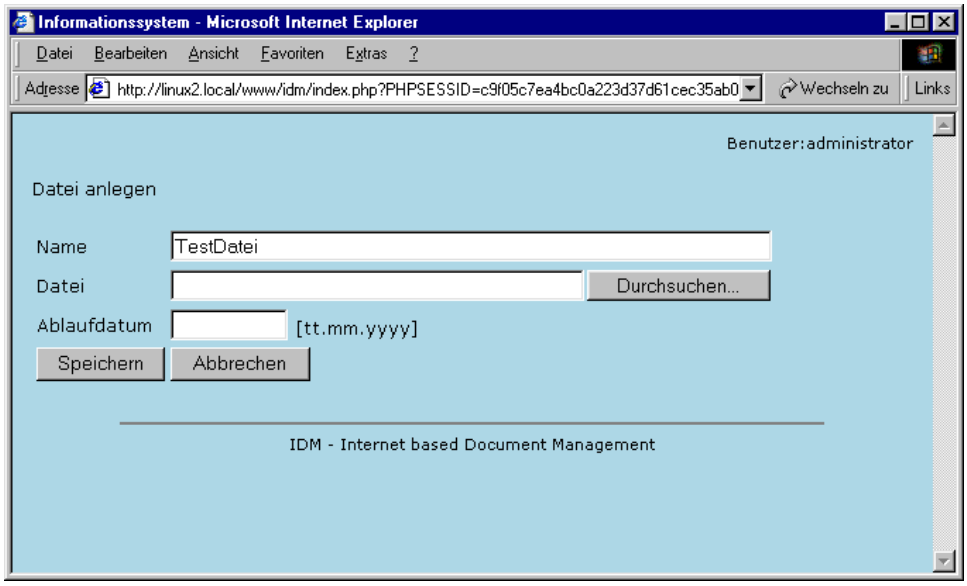


Abbildung 22 Dateianlage, -änderung

6.10.3 Hyperlink

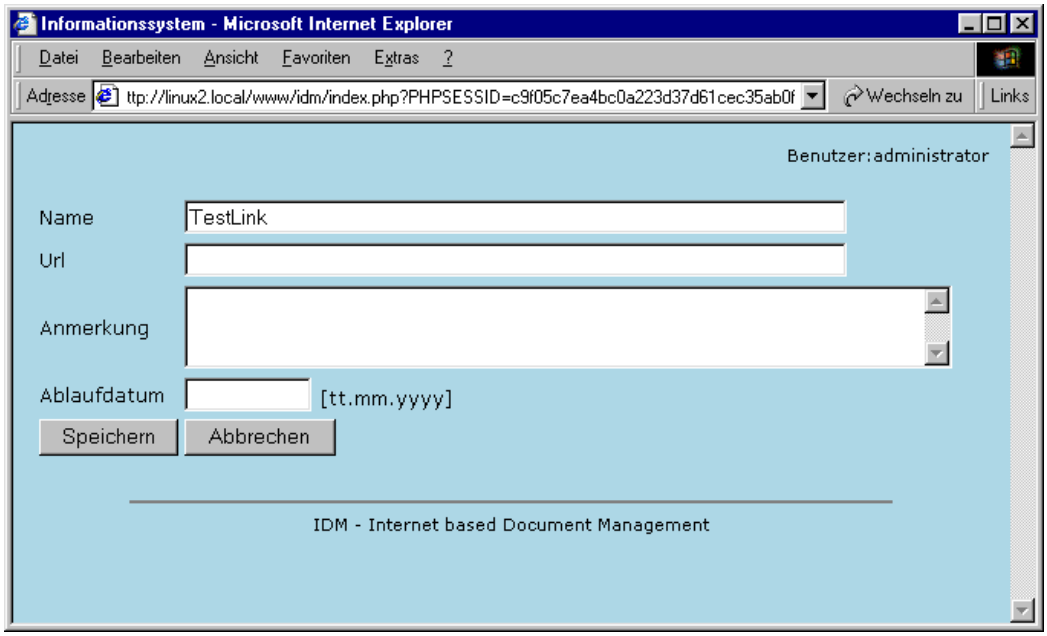


Abbildung 23 Hyperlinkanlage, -änderung

6.11 Benutzerverwaltung *(Administratordesktop [11])*

Die Benutzerverwaltung ist eine Aufgabe, die nur den Administratoren vorbehalten ist. Die Anlage eines neuen Benutzers startet man aus dem allgemeinen Anlage-Formular (Abbildung 17 [13]).

Im IDM befinden sich im Initialzustand zwei Benutzer: *administrator* und *guest*.

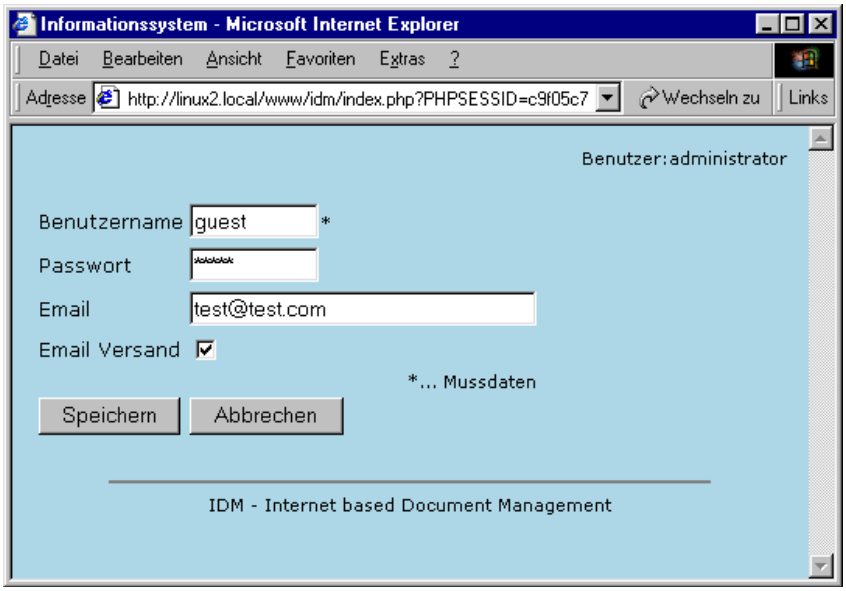


Abbildung 24 Anlage eines neuen Benutzers

Die Änderung oder das Löschen eines Benutzers führt man durch die Funktionen *‚Ändern'*, bzw. *‚Löschen'*, aus der aktuellen Benutzerliste (Abbildung 25 [1],[2]) durch.

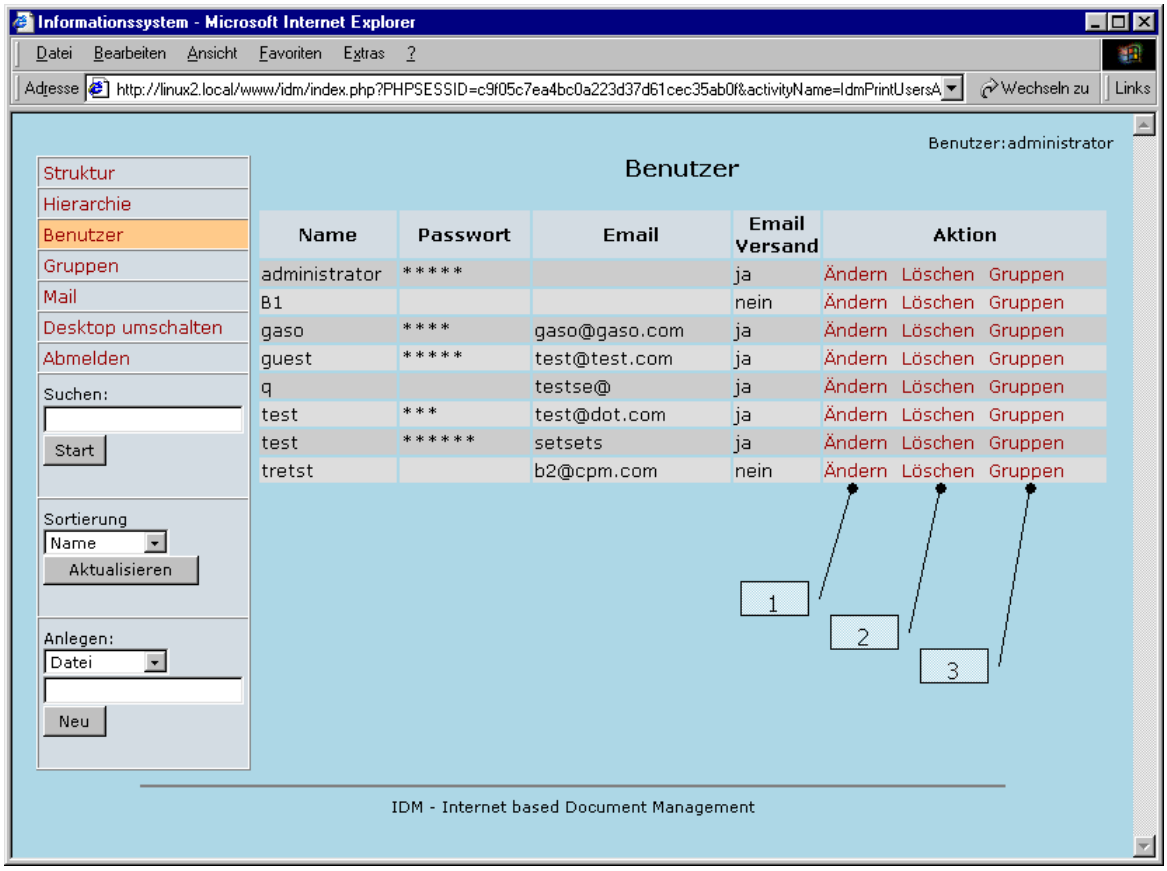


Abbildung 25 Benutzerverwaltung

6.12 Benutzergruppenverwaltung *(Administratordesktop)*

Wie bei der Benutzerverwaltung, ist auch die Gruppenverwaltung auch eine Aufgabe, die nur die Administratoren ausführen dürfen.

Die Anlage einer Gruppe findet über das Anlage-Formular (Abbildung 17 [13]) statt, das Ändern und Löschen startet man über die entsprechenden Funktionen aus der Gruppenliste (siehe Abbildung 26 [1],[2]).

Bei der Gruppenverwaltung muß man vorsichtig sein, weil eine Gruppe die zentrale Rolle im Berechtigungssystem spielt, und jede Änderung sich an vielen anderen Stellen auswirken kann.

Im IDM befinden sich im Initialzustand zwei Benutzergruppen: *admin* und *public*.
Zur Gruppe *admin* gehört der Standardbenutzer *administrator*, und zur Gruppe *public* der Benutzer *guest*.

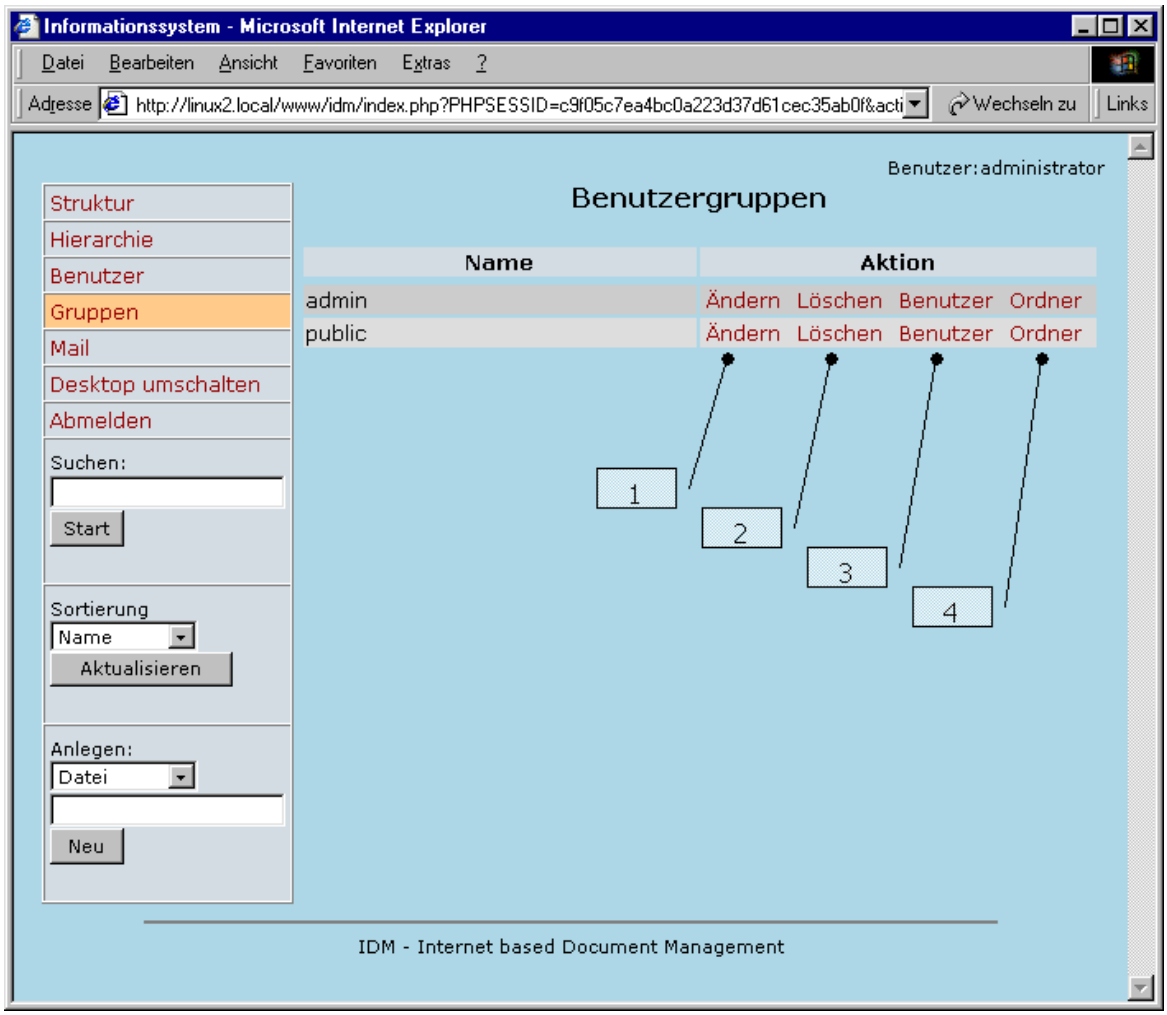


Abbildung 26 Benutzergruppenverwaltung

6.13 Berechtigungssystem *(Administratordesktop)*

Das Berechtigungssystem im IDM ist folgendermaßen aufgebaut: Ein Benutzer kann einer oder mehreren Benutzergruppen gehören. Eine Benutzergruppe kann wiederum ein bestimmtes Zugriffsrecht an einem oder mehreren Ordnern haben. Bei Zugriffsrechten unterscheidet man zwischen:

- kein Zugriff
- Lesezugriff
- Schreib- und Lesezugriff

Berechtigungen kann man nur aus der Administrationsumgebung ändern.

Ein Benutzer kann einer Benutzergruppe über zwei gleichartige Wege zugeordnet werden:

a) Benutzer auflisten, Funktion ‚Gruppen' auswählen (Abbildung 25 [3])
b) Benutzergruppen auflisten, Funktion ‚Benutzer' auswählen (Abbildung 26 [4])

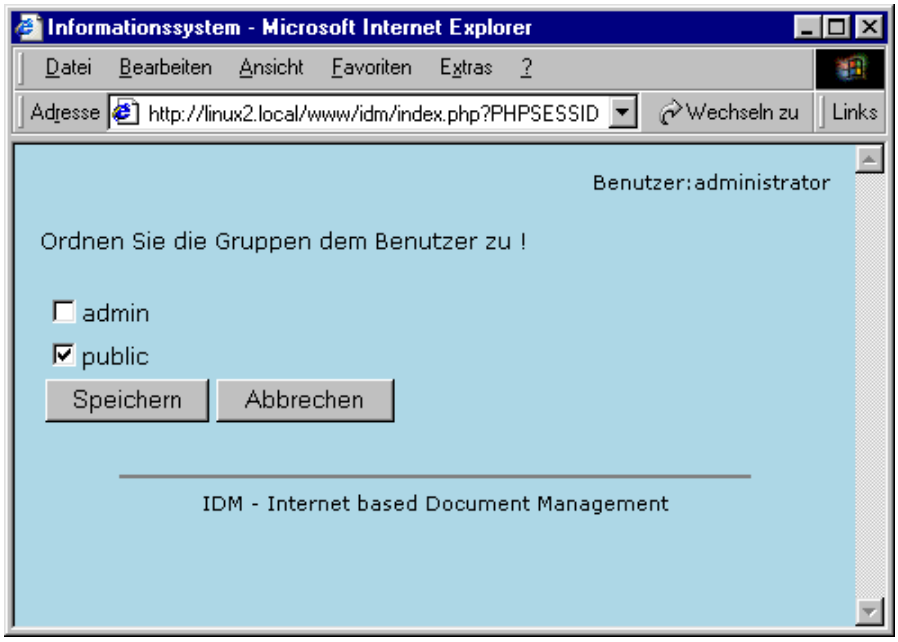


Abbildung 27 Benutzergruppen eines Benutzers

Wie bei der Beziehung zwischen den Benutzer und den Benutzergruppen kann man auch bei den Zugriffsrechten zwischen Benutzergruppen und Ordnern über zwei analoge Wege zum Ziel kommen:

a) Benutzergruppen auflisten, die Funktion ‚Ordner' auswählen (Abbildung 26 [4])
b) Struktur auflisten, neben den gewünschten Ordner die Funktion ‚Gruppen' auswählen

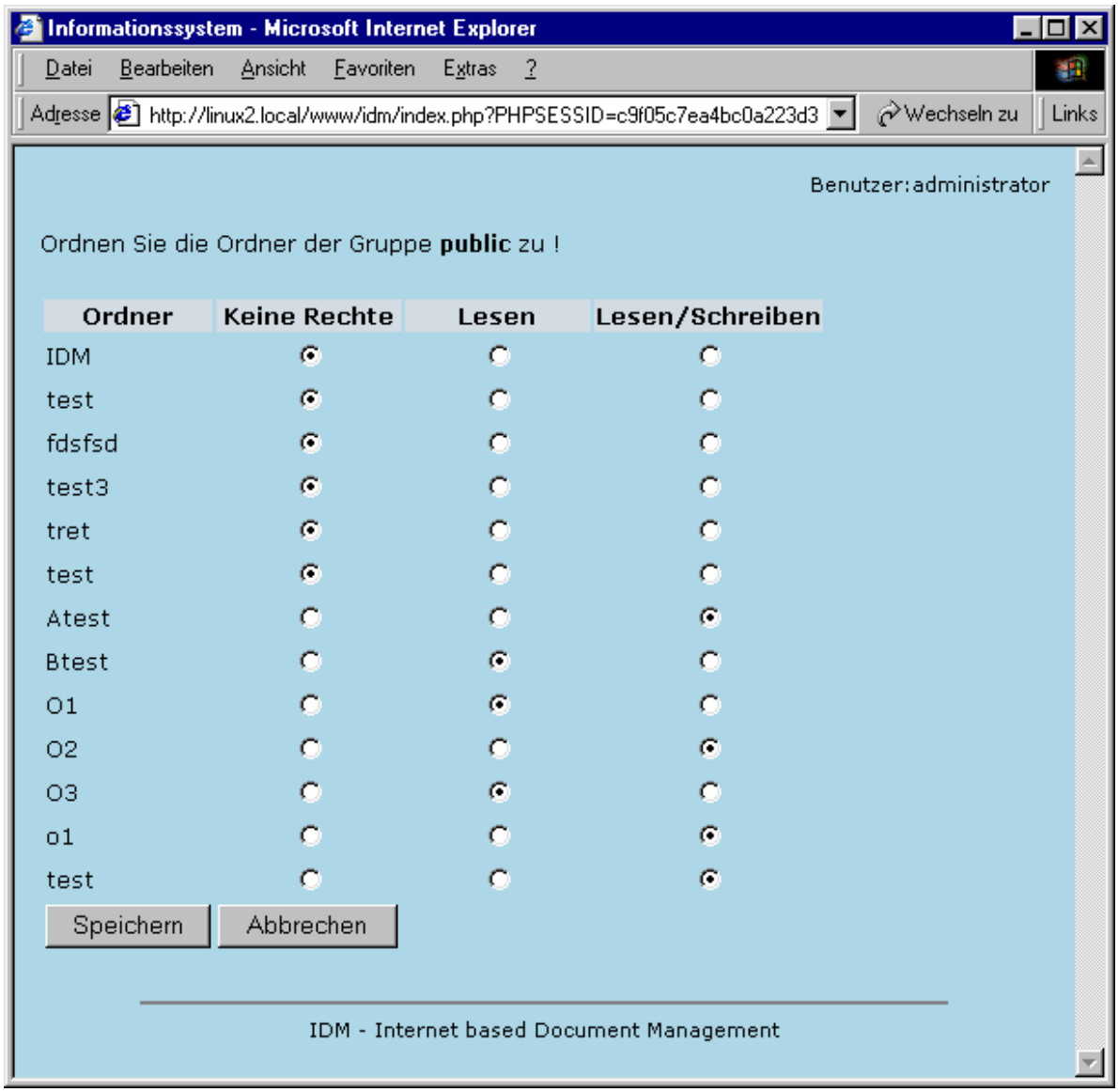


Abbildung 28 Berechtigungen für eine Benutzergruppe

6.14 E-Mail Versand *(Administratordesktop [8]; Standarddesktop [6])*

Das IDM ermöglicht eine Benachrichtigung der Benutzer, wenn ein neuer Informationseintrag im System hinzugekommen ist. Die Voraussetzungen dafür, daß der Benutzer ein E-Mail bekommt, sind folgende:

- Es muß eine gültige E-Mail Adresse des Benutzers im IDM eingetragen sein
- Die E-Mail-Option für den Benutzer muß eingeschaltet sein
- Der Ordner, in dem ein neuer Eintrag hinzugefügt wurde, muß ebenfalls die E-Mail-Option eingeschaltet haben
- Der Benutzer muß mindestens einen Lesezugriff auf den betreffenden Ordner haben
- Der neue Eintrag wurde noch nicht von diesem Benutzer geholt, bzw. gelesen

- Der Eintrag wurde nach dem gewählten Datum erstellt. Dieses ist ein Parameter im E-Mail-Versand, der systemübergreifend einstellbar ist

Ein zweiter Parameter ist die Email-Kopfzeile, die ebenfalls systemübergreifend gespeichert ist.

Wenn alle diese Kriterien erfüllt sind, bekommt der Benutzer eine E-Mail mit der Liste der neuen Einträge und den dazugehörenden Ordnern. Die Einträge und Ordner sind als Hyperlinks enthalten, der Benutzer kann automatisch aus seiner E-Mail-Applikation die Datei holen, bzw. den Ordner öffnen. Wenn er noch nicht an das IDM angemeldet ist, verlangt das System noch eine Anmeldung, und führt anschließend die gewünschte Funktion aus.

Ein zusätzliches Feature im E-Mail Versand ist die Vorschau-Funktion. Diese Funktion simuliert den gesamten Versand und liefert dem Administrator alle E-Mails, die zu dem gegebenem Moment und den eingestellten Parameter an die Benutzer verschickt gewesen wären. Mit der Vorschau kann man schnell und einfach den gesamten E-Mail-Versand testen.

Wenn die Vorschau oder der Versand gestartet werden, speichert das System die eingestellten Parameter in die Datenbank und führt mit diesen Parameter die eventuellen nächsten Versand-Jobs aus.

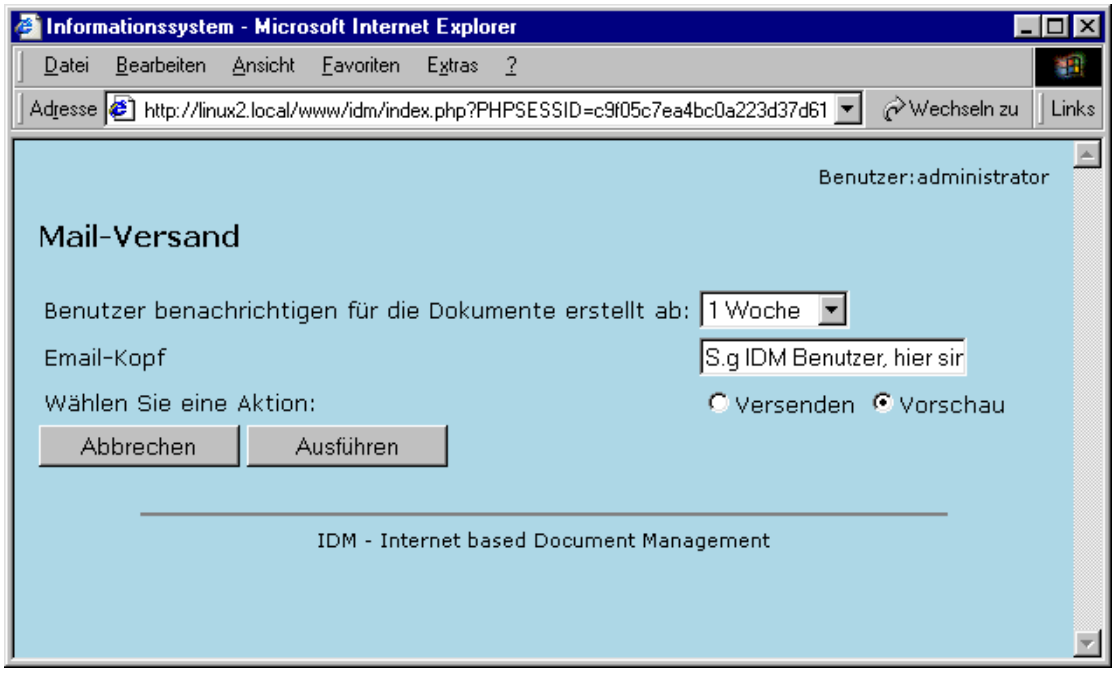


Abbildung 29 E-Mail-Versand

Die Systemvoraussetzung für E-Mail-Versand ist ein gültiger E-Mail-Client oder gar ein Server auf dem Rechner, auf welchem der Webserver mit der IDM-Applikation installiert ist.

6.15 Suche *(Administratordesktop [8]; Standarddesktop [6])*

Im IDM ist ein einfacher Suchmechanismus implementiert. Zur Zeit ist es möglich, über Ordner, Dateien und Hyperlinks eine Suche zu starten.

Als Suchbegriff ist ein einzelnes Wort möglich und dieses wird über gängige Eigenschaften der Informationseinträgen durchgesucht - über Name, Bezeichnung und eventuelle Kommentare.
Der Suchbegriff muß nicht vollständig bekannt sein – die unbekannte Teile können durch ‚*' ersetzt werden.

Zum Beispiel:

dat* _____ **dat**ei, **Dat**eien ...
unkt _____ P**unkt**, F**unkt**ion, ...

In der Standardumgebung läuft die Suche nur über jene Informationseinträge, zur welchen der aktuelle Benutzer einen Zugriff hat.

6.16 Desktop umschalten *(Administratordesktop [9]; Standarddesktop [7])*

Die Benutzer der Administrator-Gruppe haben Zugang zur beiden Benutzerumgebungen. Mit dieser Funktion können sie jederzeit zwischen diesen beiden Umgebungen wechseln.
Die anderen Benutzer haben diese Möglichkeit gar nicht in ihrem Menü eingeblendet.
Das IDM merkt sich die zuletzt verwendete Benutzerumgebung, und stellt sie bei einem erfolgreichen Neueinstieg dem Benutzer wieder zur Verfügung.

6.17 Abmelden *(Administratordesktop [10]; Standarddesktop [8])*

Wenn der Benutzer seinen Browser schließt, bedeutet es nicht unbedingt, daß die aktuelle Applikation beendet wurde. Der Session-Schlüssel und die temporären Daten sind noch immer am Server gespeichert, und wenn der Browser die ‚Cookie'-Unterstützung eingeschaltet hat, wird beim nächsten Aufruf der URL mit der IDM-Adresse die Applikation dort fortgesetzt, wo sie zuletzt verlassen wurde.
Erst durch die explizite Abmeldung aus dem IDM beendet man die Applikation, und die temporär abgespeicherte Variablen am Server werden endgültig gelöscht. Der Benutzer muß sich beim nächsten Aufruf der IDM-Seite wieder anmelden.

6.18 Grundsätzliches zu einer Internet-Applikation

Eine Web-basierte Applikation verhält sich anders als eine ‚normale' PC-Applikation. Daher sollte man sich einige Punkte vor Augen halten:

- Jede Interaktion wird mit einem einfachem Anklicken eines Hyperlinks, Buttons oder eines Bildes gestartet. In einer webbasierten Applikation gibt es keine ‚Doppelklicks'.

- Es gibt keine Kontextmenüs, d.h. Menüs, zur welchen man durchs Anklicken der rechten Maustaste gelingt. Die rechte Maustaste öffnet ein Browser-spezifisches Menü.

- Eine und dieselbe Interaktion mit dem Server kann unterschiedliche Antwortzeiten zur Folge haben. Das liegt von vielen Parametern ab, wie zum Beispiel: Auslastung des Servers, Geschwindigkeit der Verbindung, usw.

7 Schlußbemerkungen und Ausblick

IDM ist ein Beispiel für ein, mehrbenutzerfähiges, plattformübergreifendes Informationssystem auf der Basis freier Software. Durch die entsprechende Auswahl von Server-Software ist sogar eine Plattform-Unabhängigkeit auf der Serverseite möglich.

Durch eine zweischichtige Architektur ist auch ein Ansatz für die bessere Skalierbarkeit vorhanden. Man kann zum Beispiel den Webserver mit dem PHP-Interpreter auf einem Rechner und die relationale Datenbank an einem anderen Rechner betreiben.

Vor allem der Einsatz eines Frameworks und sein objektorientierter Aufbau ermöglichen eine sehr schnelle Weiterentwicklung bzw. Anpassung bestehender Applikationen.

Der Frameworkansatz wurde konsequent verfolgt, was aber noch lange nicht bedeutet, daß das Framework ausgereift wäre. Es wurden vor allem jene Stellen implementiert, die dann bei der Applikation selbst vorkamen.
Eine fehlender Punkt wäre die Abstrahierung der HTML-Objekte in PHP-Klassen. Diese wurde aber von vielen anderen PHP-Entwicklern in Angriff genommen, und könnte vielleicht von diesen übernommen bzw. integriert werden.

Zur Zeit beschäftigen sich noch zwei (dem Autor bekannte) Programmierer(-gruppen) in der Welt mit einem objektorientiertem PHP-Framework. Günstig wäre auch eine Zusammenlegung bzw. Erfahrungsaustausch dieser Gruppen untereinander, und/oder Zusammenlegung ihren Frameworks zu einem allgemeinen. Dies würde aber die Rahmen dieser Diplomarbeit eindeutig sprengen.

Ein weiterer fehlender Punkt ist der Mangel an PHP-Entwicklungsumgebungen, die den objektorientierten Ansatz unterstützen. Hoffnung auf solche machen PhpCoder (leider nur eine Windows-Applikation), und ZEND-Developer (Java basiert und kostenpflichtig).

PHP ist auf jeden Fall eine zukunftsorientierte Skriptsprache, die bestimmt noch viel Aufsehen erregen wird.

8 Glossar

In diesem Glossar sind einige in der Diplomarbeit vorkommende oder verwandte Begriffe kurz erklärt.

4GL
Heutige Programmiersprachen werden in Generationen eingeteilt. Sprachen der dritten Generation, wie C, Pascal, Cobol, etc. werden als imperativ bezeichnet, weil der Programmierer explizit angibt, wie etwas gemacht werden soll. Sprachen der vierten Generation sind im Gegensatz dazu eine Ansammlung von mächtigen Befehlen, die deklarativen Charakter aufweisen. Es wird lediglich definiert, was gemacht werden soll, nicht wie. Leider zeigen viele 4GL-Umgebungen, daß ihre Sichtweise zu eingeschränkt ist und nachträgliche triviale Änderungen schon mal nicht durchgeführt werden können (Problem der 4GL-Falle). Deshalb haben neuere Entwicklungsumgebungen Schnittstellen zu 3GL-Sprachen.

Abstrakte Klasse
Eine Klasse, die selbst keine Instanzen besitzt. Sie definiert das gemeinsame, allgemeine Verhalten einer Gruppe von Objekten. Erst ihre Unterklassen, die weitere Spezialisierungen vornehmen, besitzen Instanzen.

Aggregation
Mit Aggregation wird eine Beziehung zwischen Objekten bezeichnet, bei der ein Objekt in einem übergeordneten Objekt enthalten ist.

Analyse
Mit Analyse wird die Beschreibung eines Systems aus dem Blickwinkel des Anwendungsbereichs bezeichnet. Man definiert, was realisiert werden soll.

Assoziation
Mit Assoziation wird eine Beziehung zwischen Objekten bezeichnet, bei der ein Objekt ein anderes kennt und dessen Funktionalität nutzen kann.

Attribut
Attribute sind Datenelemente oder Variablen einer Klasse. Attribute halten den Zustand eines Objekts fest.

Basisklasse
Oberklasse

Bindung
Der Prozeß, bei dem der eine Prozedur aufrufende Programmteil die Einsprungadresse eines Unterprogramms ermittelt. In prozeduralen Sprachen wird diese Bindung nur auf eine Weise bewerkstelligt: Bereits bei der Compilierung setzt der Compiler ausgehend vom Namen des Unterprogramms die Startadresse. Diese Vorgehensweise wird frühe Bindung (early binding) genannt.
In objektorientierten Sprachen muß wegen der Vererbung noch eine andere Form der Bindung zugelassen werden. In manchen Fällen kann erst zur Laufzeit entschieden werden, welche Methodenimplementierung verwendet werden muß. Diese Art der Ermittlung der Einsprungadresse eines Unterprogramms wird späte Bindung (late binding) genannt.

Methoden, die durch frühe Bindung gebunden werden, heißen statische Methoden, jene mit später Bindung virtuelle Methoden.

Browser
Browser sind spezielle Fenster in einer Entwicklungsumgebung, die den Zugriff auf Klassendefinitionen der Bibliothek zulassen. Sie ermöglichen spezielle Sichtweisen auf Definitionen zu erzeugen, diese zu editieren oder neue Definitionen anzulegen. Sie wurden zuerst in der Smalltalk-Umgebung eingesetzt und sind heute in allen OO-Umgebungen zu finden.

Datenkapselung
Information Hiding

Design
Mit Design wird die Beschreibung eines Systems aus dem Blickwinkel des Lösungsbereichs bezeichnet. Man definiert, wie realisiert werden soll.

Design Pattern
Entwurfsmuster

Einfache Vererbung
Vererbung

Empfänger
Als Empfänger wird ein Objekt bezeichnet, das eine Nachricht erhält.

Entity Relationship Model
Das ER-Modell wurde 1976 von Chen entwickelt und wird eingesetzt zur Datenmodellierung. Es stellt Gegenstände (Entitäten) und die zwischen ihnen existierenden Beziehungen (Relationen) dar. Dabei ist eine Entität eine Abstraktion einer Gruppe gleichartiger Dinge, wie z.B. Mitarbeiter.

Framework
Ein Framework ist eine Menge von Klassen, die für ein bestimmtes Problemfeld eine Reihe von Diensten zur Verfügung stellt. Im Vergleich zu Entwurfsmustern beinhalten Frameworks zusätzlich eine gewisse Anzahl klar definierter Mechanismen, die die Interaktion der Klassen bewerkstelligen. Sie werden häufig für die Entwicklung von GUIs verwendet.

Garbage Collection
Im Laufe der Programmausführung sammeln sich Objekte im Speicher an, die diesen in kurzer Zeit völlig überlasten würden. Bei Bedarf wird deshalb vom System eine Bereinigung des Speichers von nicht mehr benutzten Objekten durchgeführt. Objekte, auf die am längsten nicht mehr zugegriffen wurde, werden zuerst gelöscht. Garbage Collection kann ein zeitaufwendiger Vorgang sein und die Performance eines Systems drücken. Jedoch hilft sie, die Stabilität des Systems zu erhöhen, da der Programmierer sich nicht selbst mit den komplizierten Speicherverwaltungsvorgängen beschäftigen muß, die durch den Einsatz von Zeigerprogrammierung oft zu schwer aufdeckbaren Fehlern führt (z.B. bei C++).

Generalisierung
Mit Generalisierung wird eine Beziehung zwischen Klassen bezeichnet, bei der gemeinsame

Eigenschaften in einer Basisklasse allgemein definiert werden. Generalisierung ist das abstrakte Konzept der Vererbung.

Identität
Jedes Objekt eines Systems ist eindeutig und existiert nur einmal. Es besitzt eine Identität.

Implementierung
Bei der Umsetzung eines Designs in ein ablauffähiges System spricht man von Implementierung.

Information Hiding
Information Hiding ist eine Technik zur Modularisierung von Softwaresystemen. Ideales Beispiel für sie sind Objekte, die ihre internen Details nach außen hin nicht bekanntgeben. Nur über die definierte Methodenschnittstelle ist ein Zugriff auf die gekapselten Daten (Attribut) möglich. Die Konstruktion von Objekten, auf die über Methoden zugegriffen wird, wird daher auch als Datenkapselung bezeichnet.

Instanz
Mit Instanz wird die konkrete Ausprägung einer Klasse bezeichnet. Dieser Begriff stammt in erster Linie aus der OOP, um veschiedene Betrachtungsweisen eines Objekts zu unterscheiden. ‚Objekt' wird manchmal als Synonym gebraucht.

Klasse
Objekte ähnlichen Verhaltens werden in Klassen zusammengefaßt.

Klassenhierarchie
Statische Beziehungen zwischen Klassen werden in einer Klassenhierarchie abgebildet. Vererbungsmechanismen ermöglichen eine Erweiterung der Hierarchie.

Klassenbibliothek
Eine Klassenbibliothek ist eine Sammlung von anwendungsneutralen Klassen, die allgemeingültig und wiederverwendbar sind. Sie können auf bestimmte Systemausschnitte bezogen sein (GUI, Container, etc.).

Message Passing
Unter Message Passing wird der Mechanismus verstanden, der eine Nachricht überträgt und die dazugehörige Methode findet. Ist die Methode nicht im adressierten Objekt zu finden, wird die Vererbungshierarchie nach der entsprechenden Methode durchsucht.

Methode
Methoden sind Funktionen, die ein Objekt ausführen kann.

Model-View-Controller
MVC ist ein Architekturmodell von Smalltalk. Hierbei werden alle GUI-spezifischen Klassen von den fachspezifischen Klassen getrennt. MVC ist die Grundlage für die Verteilbarkeit von Smalltalk-Objekten in Distributed Smalltalk.

Nachricht
Objekte kommunizieren über Nachrichten miteinander. Beim Empfang einer Nachricht wird die entsprechende Methode im Empfänger gesucht und ausgeführt.

Oberklasse
Oberklassen sind in der Klassenhierarchie oberhalb einer Klasse angesiedelt und vererben alle ihre Eigenschaften an diese.

Objekt
Ein Objekt ist eine in sich geschlossene Einheit, die einen Zustand und ein Verhalten besitzt und zudem immer eindeutig ist.

Objektorientierte Analyse
OOA ist eine methodische Vorgehensweise, bei der ein Problembereich im Form von Klassen und Objekten definiert wird. Mit Hilfe der Konzepte des Anwendungsbereichs wird beschrieben, was realisiert werden soll.

Objektorientiertes Design
OOD ist eine methodische Vorgehensweise, bei der der Lösungsbereich in Form von Klassen und Objekten definiert wird. Es wird festgelegt, wie ein System implementiert werden soll.

Objektorientierte Programmierung
OOP ist eine Art der Implementierung, bei der ein System mit Hilfe von Klassen implementiert wird. Bei der Programmausführung entstehen Objekte, die miteinander kommunizieren und die gewünschte Funktionalität bereitstellen.

Persistenz
Mit Persistenz wird die Dauerhaftigkeit von Objekten beschrieben. Beim Beenden eines Programms ist gewährleistet, daß beim erneuten Programmstart mit den gleichen Objekten weitergearbeitet werden kann.

Sender
Objekt, das eine Nachricht abschickt.

Unterklasse
Unterklassen sind in der Klassenhierarchie unterhalb einer Klasse angesiedelt und erben alle Eigenschaften der Klasse.

Use Case
Unter Use Case versteht man einen Anwendungsfall, also die Funktionalität, die das System einem Benutzer bieten muß.

Vererbung
Vererbung beschreibt die Weitergabe von Eigenschaften einer Oberklasse an eine Klasse. Gemeinsamkeiten von Klassen müssen so nur einmal definiert werden und können dann an andere Klassen weitergegeben werden. Erbt eine Klasse Eigenschaften von zwei oder mehr Oberklassen, so spricht man von mehrfacher Vererbung. Bei 1:1-Beziehungen wird von einfacher Vererbung gesprochen.

9 Literaturverzeichnis

E. Gamma, R. Helm, R. Johnson, J.Vlissides; **Design Patterns Elements of Reusable Object-Oriented Software**; Addison Wesley 1995

Ben Laurie, Peter Laurie; **Apache: The Definitive Guide Vital Information for Apache Programmers and Administrators**; O'Reilly 2nd Edition February 1999

Randy Jay Yarger, George Reese, Tim King; **MySQL & mSQL**; O'Reilly July 1999

Rasmus Lerdorf; **PHP Pocket Reference**; O'Reilly 1st Edition January 2000

Tobias Ratschiller, Till Gerken; **Webanwendungen mit PHP 4.0 entwickeln**; Addison-Wesley 2001

Milan Krizanek; **Dynamische Websites. Webserver, Datenbanken, Web Application Development**; Smart Books Publishing AG

Jakob Nielsen; **Usability Engineering**; Academic Press 1994

Kevin Mullet, Darrell Sano; **Designing Visual Interfaces**; Prentice Hall 1994

Scott Mann, Ellen Mitchell; **Linux System Security: The Administrator's Guide to Open Source Security Tools**; Prentice Hall 1999

Olaf Kirch, Terry Dawson; **Linux Network Administrator's Guide**; O'Reilly 2nd Edition June 2000

Webressourcen :

Apache HTTP Server Documentation Project : http://httpd.apache.org/docs-project/

Über das MVC Framework:
http://www.forst.uni-muenchen.de/EXT/LST/AWINF/LEHRE/OOP/UNTERLAGEN

PHP 4 Einführung: http://www.heise.de/ix/artikel/2000/07/052/01.shtml

NETCRAFT Statistik: http://www.netcraft.co.uk

Homeseite der PHP-Entwicklungssoftware: http://www.php.net/

Homeseite der relationalen Datenbank MySql: http://www.mysql.com/