

Prüfer: Prof. Dr.-Ing. Bernhard Mitschang
Prof. Dr.-Ing. Peter Göhner

Betreuerin: Priv. Doz. Dr. Anne Töpfer

Betreuer: Dipl.-Ing. Thorsten Strobel

begonnen am: 01.04.1999

beendet am: 28.10.1999

CR-Nummer: H.2.8, K.4.1, D.4.6, H.3.5, C.2.4, D.2

Diplomarbeit-Nr. 1761

**"Entwicklung einer
Datenbank-Anwendung zur
Optimierung von
Verwaltungsabläufen"**

Jürgen Rau

Fakultät Informatik
Institut für Parallele und Verteilte
Höchstleistungsrechner
Universität Stuttgart
Breitwiesenstr. 20-22
70565 Stuttgart

Inhaltsverzeichnis

INHALTSVERZEICHNIS	1
KAPITEL 1:	
EINLEITUNG.....	4
1 TEILAUFGABEN DER DIPLOMARBEIT.....	4
2 ÜBERBLICK	5
KAPITEL 2:	
ANFORDERUNGEN.....	7
1 ZIELBESTIMMUNG.....	7
1.1 <i>Mußkriterien</i>	7
1.2 <i>Wunschkriterien</i>	7
2 EINSATZ.....	8
2.1 <i>Anwendungsbereiche</i>	8
2.2 <i>Zielgruppen</i>	8
2.3 <i>Systemumgebung</i>	8
3 FUNKTIONALE ANFORDERUNGEN.....	9
3.1 <i>Datenbank</i>	9
3.2 <i>Listen und Statistiken</i>	15
3.3 <i>Use-Cases</i>	16
4 QUALITÄTS-ZIELBESTIMMUNG.....	17
KAPITEL 3:	
DATENSCHUTZ	18
1 ÜBERSICHT.....	18
2 ORGANISATORISCHE MAßNAHMEN.....	19
3 AUTHENTISIERUNG	19
4 ZUGRIFFSKONTROLLE	19
5 KRYPTOGRAPHIE	21
KAPITEL 4:	
SYSTEMARCHITEKTUR	23
1 UMGEBUNGS- UND RANDBEDINGUNGEN	23
2 GRUNDLEGENDE ENTWURFSENTSCHEIDUNGEN	24
2.1 <i>Datenbanksystem</i>	24
2.2 <i>Benutzerverwaltung</i>	25
2.3 <i>Datenschutz</i>	25
2.4 <i>Anwendungsstruktur</i>	26
3 DIAGRAMM DER SOFTWARE-SYSTEMARCHITEKTUR.....	27
4 SOFTWAREKOMPONENTEN	27
4.1 <i>DBMS</i>	27
4.2 <i>Transaktionsverwaltung</i>	28
4.3 <i>Relation-Objekt-Konverter</i>	28
4.4 <i>Sicherer Kanal</i>	28
4.5 <i>Anwendung</i>	28
4.6 <i>Oberfläche</i>	29
5 SCHNITTSTELLEDEFINITIONEN	29

5.1	<i>JDBC-Schnittstelle</i>	29
5.2	<i>Relation-Schnittstelle</i>	29
5.3	<i>Objekt-Schnittstelle</i>	29
5.4	<i>Netzwerk-Schnittstelle</i>	29
5.5	<i>Export-Schnittstelle</i>	30
5.6	<i>Interaktion-Schnittstelle</i>	30

KAPITEL 5:

IMPLEMENTIERUNG..... 31

1	KOMPONENTE DBMS.....	32
1.1	<i>Aufgabe</i>	32
1.2	<i>Bestandteile</i>	32
2	KOMPONENTE TRANSAKTIONSVERWALTUNG.....	32
2.1	<i>Aufgabe</i>	32
2.2	<i>Prinzip</i>	33
2.3	<i>Klassendiagramm</i>	33
3	KOMPONENTE RELATION-OBJEKT-KONVERTER.....	34
3.1	<i>Aufgabe</i>	34
3.2	<i>Prinzip</i>	34
3.3	<i>Klassendiagramm</i>	35
4	KOMPONENTE SICHERER KANAL.....	35
4.1	<i>Aufgabe</i>	35
4.2	<i>Prinzip</i>	35
4.3	<i>Klassendiagramm</i>	36
5	KOMPONENTE ANWENDUNG.....	36
5.1	<i>Aufgabe</i>	36
5.2	<i>Prinzip</i>	37
5.3	<i>Klassendiagramm</i>	38
6	KOMPONENTE OBERFLÄCHE.....	38
6.1	<i>Aufgabe</i>	38
6.2	<i>Bestandteile</i>	38
6.3	<i>Klassendiagramm</i>	40
7	EINBINDUNG IN WEBBROWSER.....	41

KAPITEL 6:

BENUTZUNGSANLEITUNG..... 43

1	INSTALLATION.....	43
2	INBETRIEBNAHME.....	43
3	ÜBERSICHT ÜBER DIE BEDIENELEMENTE.....	45
3.1	<i>Fensterstruktur</i>	45
3.2	<i>Anmeldefenster</i>	46
3.3	<i>Hauptfenster</i>	46
3.4	<i>Hilfsfenster</i>	48
3.5	<i>Sortier-Fenster</i>	50
3.6	<i>Filter-Fenster</i>	51
3.7	<i>Listen-Fenster</i>	51
3.8	<i>Statistik-Fenster</i>	52
4	FUNKTIONEN.....	53
4.1	<i>Übersicht</i>	53
4.2	<i>Benutzerverwaltung</i>	54
4.3	<i>Datensatzverwaltung</i>	56

4.4	Listen.....	59
4.5	Statistiken.....	60
KAPITEL 7:		
VERBESSERUNGS- UND ERWEITERUNGSVORSCHLÄGE.....		61
1	VERBESSERUNGEN.....	61
2	KONZEPTIONELLE ERWEITERUNGEN.....	61
KAPITEL 8:		
ERFAHRUNGEN.....		63
1	ERFAHRUNGEN.....	63
2	PROBLEME.....	65
ZUSAMMENFASSUNG / ABSTRACT.....		66
ANHANG A:		
DATENBANKMODELLE.....		67
ANHANG B:		
GLOSSAR.....		70
ANHANG C:		
LITERATURVERZEICHNIS.....		73
ANHANG D:		
ABBILDUNGSVERZEICHNIS.....		75

Kapitel 1:

Einleitung

Am Dekanat der Fakultät Elektrotechnik und Informationstechnik wird zur Optimierung der Verwaltungsabläufe und zur Reduzierung der Einarbeitungszeit für die wechselnden Dekane ein Qualitätsmanagementsystem eingeführt. Neben der Festschreibung der Abläufe in einem Qualitätsmanagementhandbuch und dessen Bereitstellung über Intranet auf allen Arbeitsplätzen am Dekanat, wird der Einsatz einer Datenbank-Anwendung zur Verwaltung der anfallenden Daten eine weitere tragende Säule des geplanten Qualitätsmanagementsystems sein. In Rahmen dieser Diplomarbeit sollte deshalb diese Datenbank-Anwendung konzeptioniert und implementiert werden.

Die Datenbank-Anwendung soll die Mitarbeiter des Dekanats bei ihren administrativen Aufgaben unterstützen und statistische Daten der gesamten Fakultät zur Verfügung stellen. Deshalb war es nötig, eine Datenbank zu erstellen, die alle für das Dekanat der Fakultät Elektrotechnik und Informationstechnik relevanten Daten aufnehmen kann. Um mit dieser Datenbank arbeiten zu können, sollte die Datenbank-Anwendung mit einem Frontend ausgestattet werden, welches die komfortable Interaktion zwischen dem Benutzer und der Datenbank ermöglichen sollte.

Als Architektur für die Datenbank-Anwendung sollte ein Client/Server-System gewählt werden. Auf Client-Seite sollte das Frontend als Java-Applet verwirklicht werden, das in einem Webbrowser lauffähig sein sollte. Diese zukunftsweisende Technologie, die ohne zusätzliche Installation von Software auf Client-Seite auskommt und die Administration des Systems auf die Server-Seite verschiebt, sollte es auch Benutzern, die im Umgang mit Computern nicht sehr erfahren sind, ermöglichen, die Datenbank-Anwendung zu bedienen und diese dauerhaft betriebsfähig zu halten.

Als Erkenntnis aus dieser Diplomarbeit sollte gewonnen werden, ob bzw. mit welchen Einschränkungen diese gewählte Systemarchitektur in Anwendungen zur Unterstützung von Verwaltungsabläufen eingesetzt werden kann und wo eventuelle Schwierigkeiten liegen könnten.

Im Rahmen des gesamten Projekt stellte die Einhaltung der Datenschutzbestimmungen eine wichtige Anforderung dar. Dies hatte sehr großen Einfluß auf die in der Datenbank aufzunehmenden Daten und auf den Entwurf der Datenbank-Anwendung. Unter anderem wurde deshalb eine Konzeption erarbeitet, welche die Authentisierung und Zugriffskontrolle mit kryptographischen Verfahren sicherstellen soll. Um die Einhaltung der Datenschutzbestimmungen zu verwirklichen, wurde eine weitreichende Abstimmung mit dem Datenschutzbeauftragten der Universität Stuttgart vorgenommen.

1 Teilaufgaben der Diplomarbeit

Um einen Einblick in den Ablauf der Diplomarbeit zu geben seien deren Teilaufgaben kurz vorgestellt. Die Diplomarbeit gliederte sich in folgende sechs Teilaufgaben:

Zuerst war eine Einarbeitung in die Programmiersprache Java ([CoHo97], [CoHo98] und [CaWa98]) und in die Grundlagen von Datenbanksystemen ([ReLy97], [Mits99] und [KeEi99]) verlangt. Mit Hilfe der am Institut für Automatisierungs- und Softwaretechnik

(IAS) bestehenden Datenbank-Anwendung und einer mit Microsoft Access erstellten Datenbank am Dekanat der Fakultät Elektrotechnik und Informationstechnik, sollte ein Überblick über die Problemstellung gewonnen werden. Außerdem sollte eine Einarbeitung in Verfahren der Softwaretechnik ([Balz96 und Göhn99]) erfolgen, um die Diplomarbeit mit deren Hilfe durchzuführen.

Als zweiter Aufgabenteil sollte analysiert werden, welche Daten bei den Verwaltungsabläufen des Dekanats anfallen und für die Fakultät von Bedeutung sind. Hierzu war eine umfangreiche Befragung der Mitarbeiter des Dekanats notwendig. Die gewonnenen Ergebnisse waren anschließend mit dem Datenschutzbeauftragten der Universität Stuttgart zu diskutieren und im Hinblick auf die Datenschutzbestimmungen zu modifizieren.

Darauf aufbauend sollte im dritten Aufgabenteil die Datenbank-Anwendung im Detail entworfen und spezifiziert werden. Auch hier wurde eine Abstimmung mit dem Datenschutzbeauftragten durchgeführt. Besonderes Augenmerk wurde hierbei auf die eingesetzten kryptographischen Verfahren ([Roth], [Baum98], [Tane97], [TiJe97] und [Pete99]) gelegt, in die ebenfalls eine umfangreiche Einarbeitung erfolgte.

Als vierter Aufgabenteil wurde die Datenbank-Anwendung nach der gewonnenen Spezifikation implementiert. Anschließend sollte ein Test der Datenbank-Anwendung erfolgen und das System installiert werden.

Im fünften Aufgabenteil sollte eine schriftliche Ausarbeitung erstellt werden und abschließend im sechsten Aufgabenteil war eine multimediale Präsentation zu erstellen. Letztere sollte im Rahmen des IAS Kolloquiums und des Anwendersoftware Kolloquiums vorgetragen werden, um die Ergebnisse der Diplomarbeit einem interessierten Publikum vorzustellen.

2 Überblick

Im folgenden wird eine Übersicht über die verschiedenen Teile und die Struktur dieser Ausarbeitung gegeben.

Im Anschluß an diese Einleitung werden in Kapitel 2 zunächst die Anforderungen an die Datenbank-Anwendung vorgestellt. Daran schließt sich in Kapitel 3 eine Vorstellung der zur Einhaltung des Datenschutzes eingesetzten grundlegenden Konzepte an. Hierbei wird soweit möglich nicht näher auf konkrete Implementierungsdetails eingegangen, sondern vielmehr Prinzipien erläutert. Außerdem sollen Rückwirkungen auf die ermittelten Anforderungen erläutert werden.

Danach folgt eine detaillierte Beschreibung der erstellten Datenbank-Anwendung. Dazu wird in Kapitel 4 die Systemarchitektur vorgestellt. Zunächst werden die einzelnen Systembestandteile und die Schnittstellen zwischen ihnen erläutert. Im nachfolgenden Kapitel 5 wird dann dargelegt, wie die einzelnen Systembestandteile implementiert wurden.

In Kapitel 6 wird erläutert, welche Schritte für eine erfolgreiche Installation des Systems notwendig sind und wie es zu bedienen ist. Die Beschreibung geht von der graphischen Oberfläche aus und stellt die wesentlichen Bedienungskonzepte vor. Hierbei wird außerdem auf die Administration des Systems eingegangen.

Abschließend wird in Kapitel 7 auf möglich Verbesserungs- und konzeptionelle Erweiterungsvorschläge und in Kapitel 8 auf im Rahmen dieser Diplomarbeit gewonnene Erfahrungen eingegangen.

Nachdem nun einführend die Ziele der Diplomarbeit und der Aufbau dieser Ausarbeitung vorgestellt wurden, kann nun zur detaillierten Beschreibung der Datenbank-Anwendung übergegangen werden. Dazu werden nachfolgend zunächst die Anforderungen an die Datenbank-Anwendung vorgestellt, wobei eine Fokussierung auf die zu erstellende Datenbank und auf die von der Datenbank-Anwendung geforderten Funktionalität erfolgt.

Kapitel 2: Anforderungen

Die an die zu implementierende Datenbank-Anwendung gestellten Anforderungen werden ausführlich in diesem Kapitel vorgestellt. Hierbei wird der Schwerpunkt auf die Datenbank gelegt. Nur am Rande werden in diesem Kapitel Anforderungen behandelt, die von Seiten des Datenschutzes gestellt wurden. Für die Erläuterung der durchgeführten Datenschutzmaßnahmen ist eigens das nachfolgende Kapitel vorgesehen.

1 Zielbestimmung

Im Rahmen der Einführung eines Qualitätsmanagementsystems für das Dekanat der Fakultät Elektrotechnik und Informationstechnik sollte eine Datenbank-Anwendung entwickelt werden, welche die Angehörigen des Dekanats bei ihrer Arbeit unterstützt und deshalb die Speicherung der anfallenden Daten über ein Client/Server-Datenbanksystem ermöglicht.

1.1 Mußkriterien

Die Datenbank muß alle für die Verwaltung der Fakultät Elektrotechnik und Informationstechnik durch das Dekanat relevanten Daten aufnehmen können.

Mit Hilfe der Datenbank-Anwendung muß die einfache Erfassung, Recherche und Löschung von Datensätzen ermöglicht werden. Dabei ist die Einhaltung der Datenschutzrichtlinien zu unterstützen. Dies bedeutet, daß die Datenbank vor unberechtigtem Zugriff geschützt werden muß und daß Datensätze nach vorgegebenen Fristen gelöscht werden müssen. Außerdem ist eine Benutzerverwaltung vorzusehen.

Durch die Datenbank-Anwendung sollen die Dekanatsangehörigen außerdem bei der Durchführung von häufig vorkommenden Tätigkeiten unterstützt werden. Hierbei ist vor allem an die automatische Erstellung von Listen und Statistiken gedacht. Um diese auch außerhalb der Datenbank-Anwendung nutzen zu können, ist der Datenaustausch mit anderen im Dekanat eingesetzten Programmen zu unterstützen. Hierbei ist vor allem an Microsoft Word gedacht.

Um die Mitarbeiter des Dekanats von administrativen Aufgaben bezüglich der Datenbanksoftware zu entlasten, muß die Datenbank-Anwendung als Applet in einem Standard-Webbrowser implementiert werden und der Zugriff auf die Datenbank soll, unter Zuhilfenahme eines SQL-Servers, über Intranet bzw. Internet erfolgen. Die Administration der Datenbank muß - soweit sinnvoll - ebenfalls über die als Applet ausgeführte Datenbank-Anwendung erfolgen.

1.2 Wunschkriterien

Der Datenaustausch mit anderen Datenbank-Systemen soll durch die Bereitstellung geeigneter Schnittstellen ermöglicht werden. Hierbei ist vor allem an eine noch in der Planung befindliche Universitätsdatenbank gedacht.

Da die in der Datenbank zu speichernden Daten, und damit auch die Anforderungen an die Datenbank-Anwendung, noch Änderungen unterliegen werden, soll das System so entworfen

werden, daß ohne größeren Aufwand Anpassungen vorgenommen werden können. Dies bedeutet, daß das System stark modular aufgebaut werden soll.

Um in der Datenbank enthaltene Daten auch ohne den Umweg über eine zusätzliches Programm, wie etwa Microsoft Word ausdrucken zu können, wäre die Implementierung einer Druckfunktion wünschenswert.

2 Einsatz

Die Datenbank-Anwendung unterstützt die Mitarbeiter des Dekanats der Fakultät Elektrotechnik und Informationstechnik bei ihren administrativen Aufgaben. Ein wichtiger Punkt ist dabei die Erhebung von statistischen Daten, die auch anderen autorisierten Angehörigen der Fakultät zugänglich sein sollten.

Die Administration der Datenbank bzw. der Datenbank-Anwendung (Benutzer anlegen, Zugriffsrechte vergeben usw.) wird von einem Administrator durchgeführt. Die Datenbank-Anwendung ist in den Standard-Webbrowser als Applet lauffähig.

2.1 Anwendungsbereiche

Der Hauptanwendungsbereich ist die Verwaltung der für das Dekanat der Fakultät Elektrotechnik und Informationstechnik relevanten Daten mit Hilfe eines Standard-Webrowsers via Intranet/Internet. Außerdem sollen häufig vorkommende Tätigkeiten besonders unterstützt werden. Hierbei ist vor allem die Erfassung und Recherche von Daten, die Erstellung von Listen und Statistiken, sowie die Vernichtung von Daten nach den Datenschutzrichtlinien zu erwähnen. Der Zugriff auf die Datenbank soll innerhalb der Fakultät Elektrotechnik und Informationstechnik für autorisierte Personen ständig möglich sein.

2.2 Zielgruppen

Im Rahmen der Analyse wurden die folgenden drei Benutzerkategorien identifiziert:

- Dekan, Prodekan, Studiendekan, Prüfungsausschußvorsitzender
- DekanatsmitarbeiterInnen
- Systemadministrator

Diese Benutzerkategorien unterscheiden sich im wesentlichen durch die Rechte, die sie für den Zugriff auf die Datenbank haben. Die Vertreter der ersten Kategorie sollten möglichst nur lesend zugreifen können. Die DekanatsmitarbeiterInnen sollten auf bestimmte Teile der Datenbank lesend und schreibend zugreifen können. Im Gegensatz zu den vorherigen Benutzerkategorien sollten Systemadministratoren vollen Zugriff auf die Datenbank haben.

2.3 Systemumgebung

Bei der Analyse der Anforderungen ist eine Betrachtung des Systemumfeldes sehr wichtig, um schon beim Entwurf eine Anpassung an dabei aufgedeckte Besonderheiten zu ermöglichen.

In den beiden nachfolgenden Tabellen ist die Softwareumgebung auf Server- und auf Client-Seite aufgeführt.

Server	
Betriebssystem	Windows-NT oder Windows-95/98.
Datenbank-Server	Sybase SQL Anywhere Server (ab der Version 6.0).
Datenbank	Für die Entwicklung wird das Programm Sybase PowerDesigner verwendet.
Netzwerk	Netzwerk-Software für TCP/IP.

Clients	
Betriebssystem	Windows-NT-Workstation oder Windows-95/98 (oder mit Rücksicht auf nachfolgenden Punkt auch beliebig).
Web-Browser	Microsoft Internet-Explorer oder Netscape Navigator (jeweils ab Version 4.0).
Netzwerk	Netzwerk-Software für TCP/IP.

Daten müssen mit Hilfe von Text- bzw. HTML-Dateien und über die Zwischenablage exportiert werden können, hierbei ist vor allem auf eine Kompatibilität mit Microsoft Word zu achten. Außerdem ist eine Schnittstelle zu anderen Datenbanken, wie z.B. der des Prüfungsamtes, zu konzipieren und soweit möglich auch zu implementieren.

3 Funktionale Anforderungen

In den folgenden Abschnitten sollen die funktionalen Anforderungen, die an die Datenbank-Anwendung gestellt werden, detailliert erläutert werden. Zunächst werden die in die Datenbank aufzunehmenden Datensätze vorgestellt. Dies geschieht im wesentlichen auf graphischem Wege.

Anschließend werden die durch die Datenbank-Anwendung zu erstellenden Listen und Statistiken vorgestellt. Hierbei wird eine genaue Aufführung der in der Analyse ermittelten Details gegeben.

Zum Ende wird kurz auf die ermittelten Use Cases eingegangen. Use Cases dienen zur Beschreibungen von Funktionen, die durch Interaktionen mit dem System ausgelöst bzw. ausgeführt werden. Wie man in späteren Kapiteln noch sehen wird, sind Use Cases textuelle Beschreibungen, die tabellarisch strukturiert sind.

3.1 Datenbank

Die benötigte Datenbasis sollte mittels Entity-Relationship-Modellierung entworfen werden. Mit Hilfe des Werkzeugs PowerDesigner DataArchitect sollte aus der modellierten Datenbasis die Datenbank erstellt werden. Im Anhang A werden die dabei erstellten Datenbankmodelle vorgestellt.

Nachfolgend sind die in die Datenbank aufzunehmenden Daten der Übersichtlichkeit wegen graphisch strukturiert dargestellt. Dabei sind die verschiedenen Datenklassen in fetter Schrift notiert, die jeweiligen Attribute in normaler Schrift. Unterstrichene Attribute sind an anderer Stelle näher definiert. Eine Zahl bzw. ein Stern an einem Attribut gibt dessen Häufigkeit an, wobei ein Stern beliebig oft bedeutet.

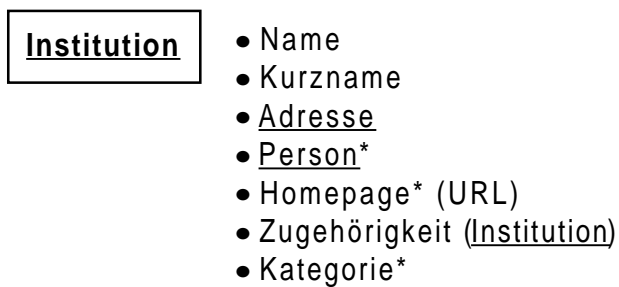


Abbildung 2.1: Institutionen

In Abbildung 2.1 stellt das Attribut Homepage einen Verweis auf eine Webseite (URL) dar. In der erstellten Datenbank-Anwendung soll es möglich sein, durch das Anklicken eines Datenfeldes das eine URL enthält, ein neues Browser-Fenster zu öffnen, in dem die jeweilige Homepage geladen wird.

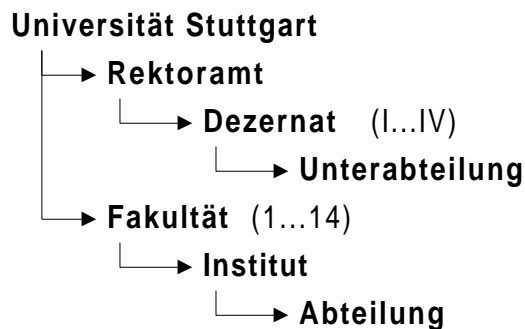


Abbildung 2.2: Institutionsbaum

Der vorstehende Institutionsbaum der Universität Stuttgart in Abbildung 2.2 soll als Vorgabe in die Datenbank eingetragen werden, da dieser für das Dekanat von Bedeutung ist. Selbstverständlich kann dieser später erweitert und an veränderte Bedürfnisse angepaßt werden.

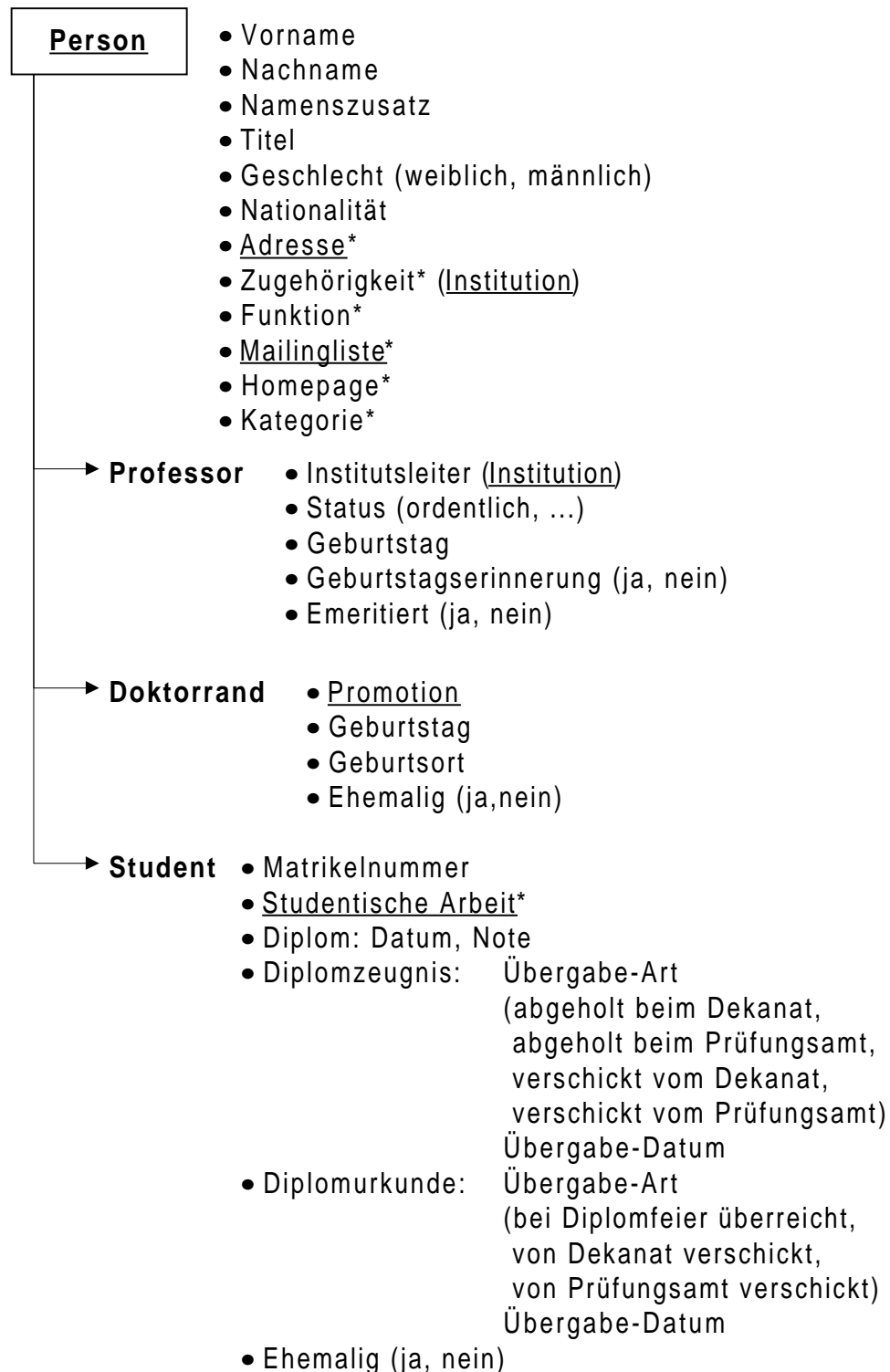


Abbildung 2.3: Personen

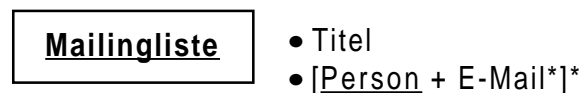


Abbildung 2.4: Mailinglisten



Abbildung 2.5: Adressen

Einzelne Anschriften von Personen bzw. Institutionen sollen per Zwischenablage exportiert werden können. Für die Erstellung von Serienbriefen siehe den Abschnitt über die Erstellung von Listen.

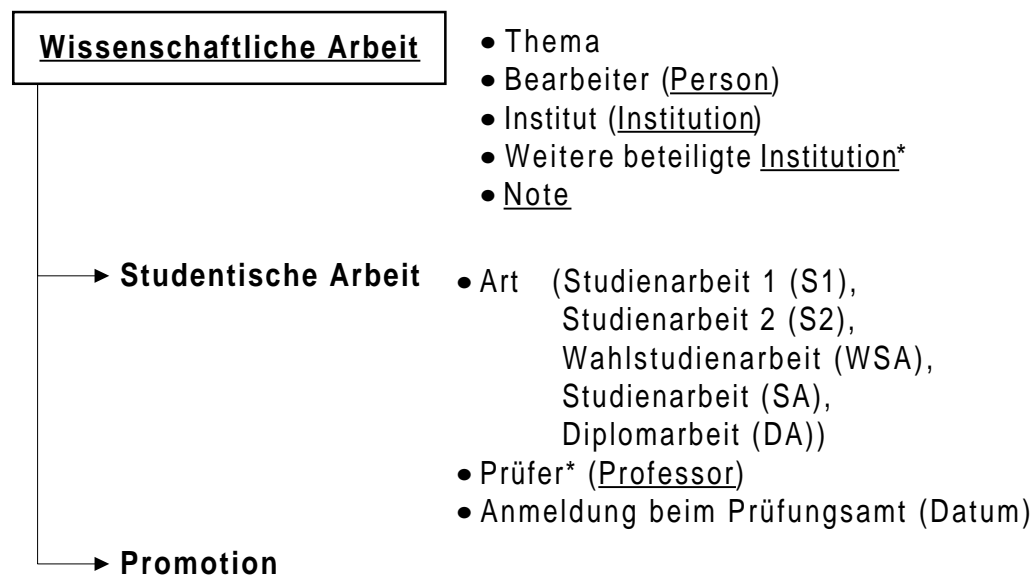


Abbildung 2.6: Wissenschaftliche Arbeiten

Note

- Text
- Zahl mit Nachkommastelle

Abbildung 2.7: Noten

Promotion

- Promotionsgesuch (Datum)
- Eingereicht: Datum,
Anzahl der Beilagen
- Abiturzeugnis (o.ä.) beigefügt (ja, nein)
- Diplom-Note
- Vorgelegt in Promotionsausschußsitzung (Datum)
- Ankunft Hauptbericht (Datum)
- Ankunft Mitbericht* (Mitberichter + Datum)
- Erinnerungsfrist
- Promotionsprüfungsausschuß:
Vorsitzender (Professor),
Hauptberichter (Professor),
Mitberichter* (Professor),
Mitprüfer (Professor),
Kleiner Umlauf⁶ (Professor),
- Mündliche Prüfung (Datum)
- Ausgelegt im Dekanat: von (Datum),
bis (Datum)
- Registraturnummer
- Akten an Prüfungsamt verschickt (Datum)
- Doktorurkunde: Übergabe-Art
(persönlich,
per Einschreiben)
- Pflichtexemplare eingereicht (Datum)
- Freigabe zur Veröffentlichung (Datum)
- Aufzuheben bis (Datum)
- Vorschlägen für Preis (ja , nein)
- Vorschlagsfrist

Abbildung 2.8: Promotionen

Nach einer variabel einstellbaren Frist (Vorgabewert 3 Monate) nach der Vorlage im Promotionsprüfungsausschuß soll eine Erinnerung erfolgen, wenn noch nicht alle Berichte, also Haupt- und Mitberichte, angekommen sind.

Die Registraturnummer soll automatisch vergeben werden können. Sie besteht aus der zweistelligen Jahresangabe des Datums der mündlichen Prüfung, einem darauffolgenden Schrägstrich und schließlich einer fortlaufenden Nummer. Die fortlaufende Nummer richtet

sich nach der Reihenfolge der mündlichen Doktorprüfungen. Die fortlaufende Nummer soll durch den Benutzer dauerhaft verändert werden können, wobei eine Konsistenz-Prüfung mit anderen Datensätzen erfolgt.

Der erste Eintrag in das Feld für den Vorschlag zum „Anton- und Klara-Rösner-Preis“ soll automatisch erfolgen, eine dauerhafte manuelle Änderung ist darauffolgend möglich. Die Bedingungen für den Vorschlag zur Preisvergabe sind derzeit:

1. Dissertation mit „Auszeichnung“,
2. Diplomprüfung mit „Sehr Gut“ oder „Auszeichnung“.

Aus der Vergabe des Preises ist die Art und die Frist für die Aufbewahrung der Dissertation abzuleiten. Wird ein Preis vergeben, ist der Bearbeiter, der Titel, die Registraturnummer, die Diplom-Note und die Note des Datensatzes bis zum Oktober des jeweiligen Jahres aufzuheben. Wird kein Preis vergeben, kann die Dissertation nach Ablauf der normalen Löschrfrist zum Löschen vorgeschlagen werden. Die Löschrfristen können an veränderte Bedürfnisse variabel angepaßt werden.

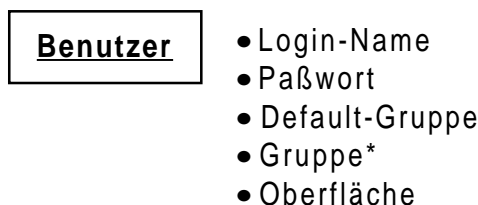


Abbildung 2.9: Benutzer

Um die in der Datenbank gespeicherten Daten vor unberechtigtem Zugriff zu schützen, wird eine Zugangsprüfung mit Benutzernamen und Paßwörtern durchgeführt. Verschiedenen Benutzern und Benutzergruppen werden verschiedene Zugriffs- und Modifikationsrechte zugeteilt. Das Anlegen von Benutzern und Benutzergruppen, sowie die Vergabe von Rechten, erfolgt durch den Administrator.

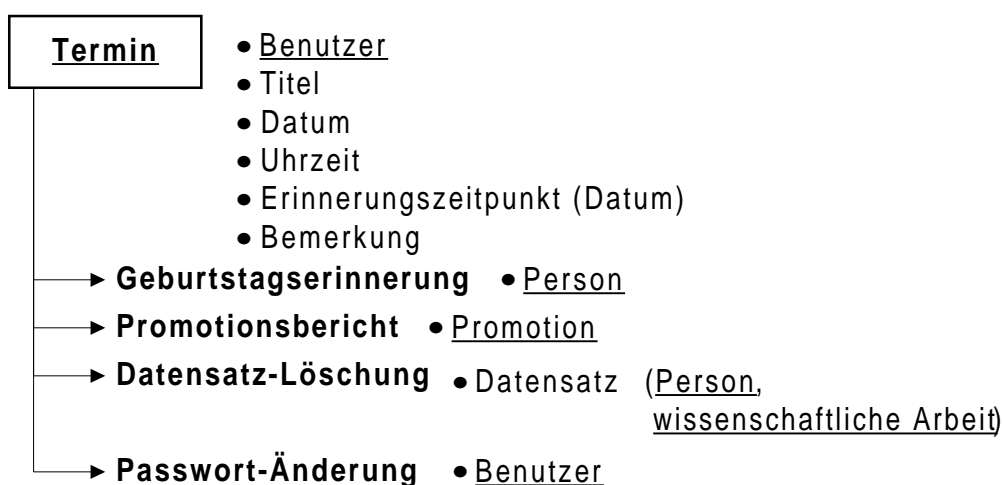


Abbildung 2.10: Termine

Da nach den Datenschutzrichtlinien in der Datenbank gespeicherte Daten nicht beliebig lang gespeichert bleiben dürfen, sofern sie personenbezogene Daten enthalten, ist eine Löschung dieser Datensätze vorzusehen, sobald sie nicht mehr benötigt werden. Da dies nicht automatisch erfolgen soll, wird dafür eine Erinnerung eingeführt, die zur manuellen Löschung auffordert. Dies ist für alle Professoren, Doktoranden und Studenten an der Fakultät Elektrotechnik und Informationstechnik der Universität Stuttgart, sowie für sonstige in der Datenbank erfaßte Personen vorzusehen. Das Löschen einer Person zieht auch das Löschen der von ihr verfaßten wissenschaftlichen Arbeiten nach sich. Analoges gilt auch für den umgekehrten Fall, allerdings wird hier halbautomatisch geprüft, ob die Person nicht mehr gebraucht wird. Die Frist für die Erinnerung ist für die verschiedenen Datensatzklassen separat einstellbar vorzusehen.

Anzumerken bleibt noch, daß durch das Löschen der Datensätze die nachfolgend beschriebenen statistischen Auswertungen nicht verfälscht bzw. eingeschränkt werden dürfen.

Die Erinnerung an das Löschen von Datensätzen soll zu einer Erinnerungsfunktion an beliebige Termine erweitert werden. Dabei ist vor allem die Erinnerung an Geburtstage von Professoren zu erwähnen. Die Geburtstagserinnerungen sollen zwei Wochen vor dem Geburtstag erfolgen, ist dies ein Samstag bzw. Sonntag oder ein Feiertag, dann soll die Erinnerung am darauffolgenden Werktag erfolgen.

3.2 Listen und Statistiken

Aus den in der Datenbank gespeicherten Daten müssen diverse Statistiken erstellt werden können, dabei ist ein Generator für Statistiken vorzusehen, der durch Programmierung beliebige Statistiken erstellen kann.

Es sollen folgende statistische Angaben ermittelt werden:

- Ausländische Studenten nach Nationalität.
- Ausländische Doktoranden nach Nationalität.
- Zahl der Diplome mit Notenverteilung pro Studienjahr und Institut.
- Zahl der Promotionen mit Notenverteilung und Preisen pro Jahr und Institut.
- Zahl der studentischen Arbeiten und Promotionen pro Institut.
- Dauer des Promotionsverfahrens nach Jahr verteilt.
- Absolventenzahl pro Studienjahr. Mit erfolgreicher/erfolgloser Abschlußprüfung (DA).
- Doktorandenzahl pro Jahr und Institut. Mit erfolgreicher/erfolgloser Prüfung.
- Zahl der betreuten studentischen Arbeiten je Prüfer (Professor).
- Zahl der betreuten Promotionen je Professor und Institut.
- Frauenanteil bei Abschluß des Studiums.
- Frauenanteil bei Promotionen.

Die ermittelten Statistiken sollen in einer optisch ansprechenden Art dargestellt werden, was auch eine sinnvolle graphische Darstellung impliziert. Außerdem soll der Export der gewonnenen Statistiken per Datei bzw. Zwischenablage möglich sein.

Es sollen mit Hilfe der Datenbank-Anwendung mehrere Tätigkeiten in der Verwaltung automatisiert werden. Dazu zählen neben der Erstellung des Stammblasses „Promotionsverfahren“ auch die Erstellung verschiedener Listen:

- Liste der Doktorprüfungen des letzten Jahres, des laufenden Jahres und allgemein. Die Liste soll folgende Spalten umfassen:
 1. Angaben zur Doktorprüfung mit folgenden Feldern:
 - Titel + Vorname + Nachname (fett, Nachname unterstrichen)
 - Titel der Arbeit (kursiv)
 - Bericht (Haupt- und Mitbericht): Status + Titel + Vorname + Nachname (+ die Angabe „emeritiert“ in Klammern, wenn dies der Fall ist).
 2. Datum
 3. Note
 4. Registraturnummer
- Liste der Doktorprüfungen des letzten Jahres für den deutschen Fakultätentag, wie vorige Liste aber ohne Note, ohne laufende Nummer und ohne Angabe des Tages im Datum.
- Liste der angemeldete Arbeiten. Die Liste soll folgende Spalten umfassen:
 1. Anmeldedatum beim Prüfungsamt
 2. Name
 3. Matrikelnummer
 4. Art der Arbeit (WSA, SA, DA)
 5. Prüfer
- Datum der Anmeldungen von Diplomarbeiten beim Prüfungsamt (analoge Liste auch für Wahl- und Pflichtstudienarbeiten).
- Liste der Diplomarbeiten für die Verwaltung von Zeugnissen und Urkunden. Ein Datensatz in der Liste sollte folgende Felder umfassen:
 1. Name
 2. Adresse
 3. Diplom (Datum)
 4. Institut
 5. Thema
 6. Art der Übergabe von Zeugnis und Urkunde, jeweils mit Datum
- Adreßlisten sollen mit vom Benutzer auswählbaren Angaben zu erzeugen sein.
- Mailinglisten

Die erstellten Listen bzw. auch das Stammbblatt „Promotionsverfahren“ sollen auf übersichtliche Weise dargestellt werden. Ein Export per Datei soll vorgesehen werden. Außerdem soll ein direkten Ausdruck aus der Datenbank-Anwendung heraus möglich sein.

3.3 Use-Cases

Use-Cases sind strukturierte Beschreibungen der wichtigsten Funktionalitäten einer Anwendung. In der nachfolgenden Tabelle sind alle Use-Cases aufgelistet, die für die Datenbank-Anwendung von wesentlicher Bedeutung sind. Die einzelnen Use-Cases werden im Kapitel Bedienungsanleitung im Detail behandelt. Der Übersicht halber seien sie an dieser Stelle bereits aufgeführt.

Dekanatsdatenbank	
Use-Case 1	Benutzer anlegen
Use-Case 2	Benutzer aktualisieren
Use-Case 3	Benutzer löschen
Use-Case 4	Benutzer anmelden
Use-Case 5	Benutzer erinnern
Use-Case 6	Datensatz anlegen

Dekanatsdatenbank	
Use-Case 7	Datensatz suchen
Use-Case 8	Datensatz aktualisieren
Use-Case 9	Datensatz-Zugriffsrechte ändern
Use-Case 10	Datensatz-Hompage aufrufen
Use-Case 11	Datensatz löschen
Use-Case 12	Datensatz exportieren
Use-Case 13	Datensätze sortieren
Use-Case 14	Datensätze filtern
Use-Case 15	Liste erzeugen
Use-Case 16	Liste exportieren
Use-Case 17	Liste drucken
Use-Case 18	Statistik erzeugen
Use-Case 19	Statistik exportieren
Use-Case 20	Statistik drucken

4 Qualitäts-Zielbestimmung

Abschließend sei eine Übersicht über die Qualitätsziele gegeben. Anhand der nachstehenden Tabelle soll ein Eindruck von der Gewichtung verschiedener Produktqualitäten gegeben werden.

Produktqualität	Sehr hoch	Hoch	Normal	Nicht relevant
Funktionalität		X		
Richtigkeit		X		
Sicherheit	X			
Zuverlässigkeit	X			
Reife		X		
Fehlertoleranz	X			
Wiederherstellbarkeit	X			
Benutzbarkeit		X		
Verständlichkeit		X		
Erlernbarkeit		X		
Bedienbarkeit		X		
Effizienz			X	
Zeitverhalten			X	
Verbrauchsverhalten			X	
Änderbarkeit			X	
Analysierbarkeit			X	
Modifizierbarkeit		X		
Übertragbarkeit			X	

Kapitel 3: Datenschutz

Nachdem im vorigen Kapitel die Anforderungen an das System im Bezug auf Funktionalität und Benutzung beschrieben wurden, soll in diesem Kapitel auf die Komponente des Datenschutzes eingegangen werden. Wie bereits angedeutet wurde, spielte der Datenschutz im Rahmen des ganzen Projektes eine wesentliche Rolle und hatte auch weitgehenden Einfluß auf die gewählte Systemarchitektur. Somit soll ihm ein ganzes Kapitel gewidmet werden, um eine kompakte Betrachtung aller Aspekte zu ermöglichen.

1 Übersicht

Bevor auf Details eingegangen werden kann, ist es nötig einen Überblick über die verschiedenen Bereiche des Datenschutzes vorzustellen. Nach [KeEi99] gliedert sich der Datenschutz in mehrere Ebenen, diese sind in der Abbildung 3.1 dargestellt. Die im Rahmen der Erstellung einer Datenbank-Anwendung wichtigen und somit auch im Rahmen dieser Arbeit implementierten Ebenen, sind die Ebene der Authentisierung, die Ebene der Zugriffskontrolle und die Ebene der Kryptographie. Die Problematik der Zugriffskontrolle wurde schon im vorigen Abschnitt bei der die Benutzerverwaltung diskutiert und soll hier noch ergänzend behandelt werden.

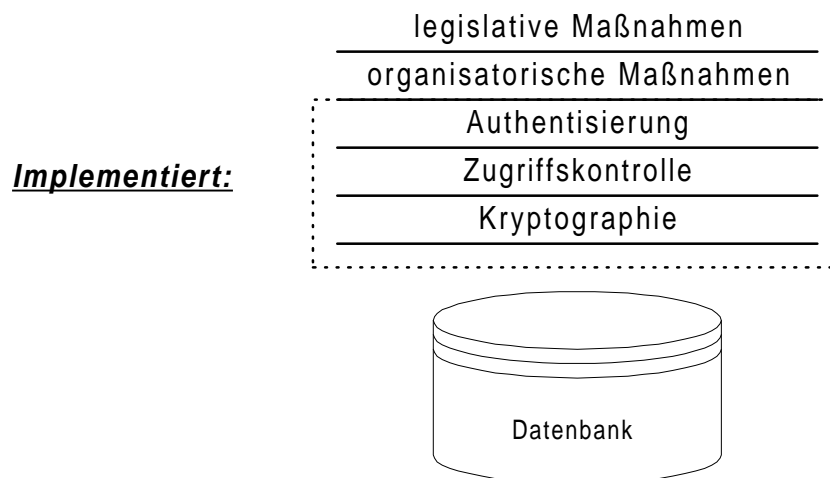


Abbildung 3.1: Ebenen des Datenschutzes

Zusätzlich zu den genannten Ebenen werden noch die organisatorischen Maßnahmen in diesem Kapitel angesprochen. Der Grund dafür ist, daß neben den Maßnahmen bei der Implementierung der Datenbank-Anwendung auch die organisatorischen Maßnahmen im Rahmen dieses Projektes beeinflußt werden können. Dies steht im Gegensatz zu den gegebenen legislativen Maßnahmen, die hier deshalb auch nicht näher behandelt werden sollen.

2 Organisatorische Maßnahmen

Als organisatorische Maßnahmen kann die Aufstellung des Datenbankservers in einem abschließbaren Raum betrachtet werden, zu dem nur befugte Personen Zugang haben. Außerdem ist hierbei die Vergabe der Paßwörter und der Zugriffsrechte zu nennen, z.B. dürfen nicht alle Benutzer als Administrator eingetragen sein und es muß dafür gesorgt werden, daß das System ordentlich betreut wird.

Wichtig ist aber auch die Auswahl der aufzunehmenden Daten in die Datenbank. Bei den Gesprächen mit dem Datenschutz-Beauftragten der Universität Stuttgart trat die hierbei anzuwendende Vorgehensweise deutlich zu Tage. Bei jedem Typ von Daten ist zu hinterfragen, ob dieser für die Arbeitsabläufe der betreffenden Personen überhaupt notwendig ist. Ist dies nicht eindeutig mit ja zu beantworten, wird der Datensatz nicht aufgenommen, eventuell spätere Gegebenheiten, die dessen Aufnahme rechtfertigen würden, sind keine ausreichende Begründung dafür.

Datensätze müssen so früh wie möglich gelöscht werden, die Verwaltung hat sich dem unterzuordnen. Dies bedeutet, daß Daten z.B. nicht lediglich für die Erhebung von Statistiken aufgehoben werden dürfen. Für diesen Zweck muß also eine Anonymisierung durchgeführt werden. Dies bedeutet, daß alle personenbezogenen Informationen entfernt werden müssen, außerdem darf aus den verbleibenden Daten nicht auf die bzw. Teile der gelöschten Daten geschlossen werden können.

Anzumerken bleibt noch, das sich die gemachten Ausführungen nicht nur auf elektronisch erfaßte Daten beziehen, sondern auch auf andere Arten der Datenaufbewahrung, wie etwa auf Papier.

3 Authentisierung

Ein wichtiger Gesichtspunkt im Rahmen des Datenschutzes stellt die Authentisierung dar. Unter Authentisierung versteht man den Nachweis, daß jemand oder etwas seine Identität zweifelsfrei nachweist. Im Bezug auf die Datenbank-Anwendung bedeutet dies, daß sich die Benutzer der Datenbank-Anwendung zweifelsfrei identifizieren müssen, um den Zugang zur Datenbank-Anwendung zu bekommen der ihnen zusteht.

In der implementierten Datenbank-Anwendung erfolgt dies über Kombinationen aus Benutzername und Paßwort, die nur der jeweilige Benutzer kennt. Damit auf lange Sicht, diese Kombinationen nicht doch einem Dritten bekannt werden, wird eine regelmäßige Änderung des Paßwortes von dem Benutzer verlangt. Dabei wird darauf geachtet, daß der Benutzer keine trivialen Paßwörter gebraucht, indem jedes Paßwort mindestens acht Zeichen lang sein und zwei Sonderzeichen enthalten muß.

Wichtig ist noch zu erwähnen, daß die Authentisierung beim Starten des Servers und beim einloggen in den Client erfolgt. Die hierbei vergebenen Kombinationen aus Benutzername und Paßwort haben im Normalfall nichts miteinander zu tun.

4 Zugriffskontrolle

In diesem Abschnitt soll nun die im vorigen Kapitel nur gestreifte Benutzerverwaltung genauer vorgestellt werden, über welche die Zugriffskontrolle verwirklicht wird. Wie erwähnt, ist das Ziel, nur Datensätze bzw. Datensatzteile dem jeweiligen Benutzer zugänglich zu machen, die auch für ihn bestimmt sind.

Da die in der Datenbank gespeicherten Daten den Datenschutzrichtlinien unterliegen, haben nur autorisierte Benutzer Zugriff auf die Datenbank. Die Benutzer werden von einem Administrator verwaltet, dem außerdem die Pflege der Datenbank unterliegt. Deshalb hat er als einziger Benutzer uneingeschränkten Zugriff auf alle Daten. Die Verwaltung der Zugriffsrechte auf einzelne Datensätze wird den Zugriffsrechten auf Dateien in Unix-Dateisystemen nachempfunden, d.h. es gibt verschiedene Benutzer und Benutzergruppen, die unterschiedliche Lese- und Modifikationsrechte haben. Hier ist aber auch sogleich der wesentliche Unterschied erkennbar. Es gibt eine Rechte, die allgemein vergeben werden können, wie es in Unix der Fall ist.

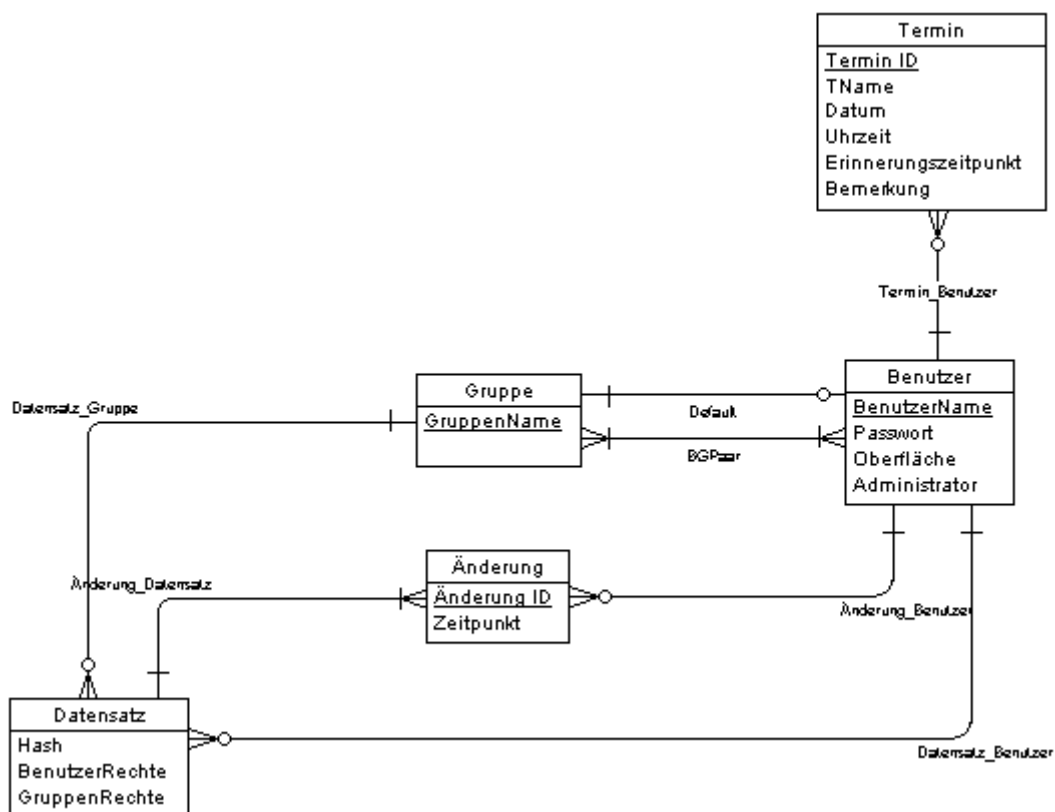


Abbildung 3.2: Benutzerverwaltung

In Abbildung 3.2 ist die gewählte Implementierung als Conceptual Data Model (CDM) zu sehen. Zur Erklärung der grafischen Notation sie auf [PDUG97] verwiesen. Es sei lediglich angemerkt, daß die Darstellung im wesentlichen einem Entity-Relationship-Model entspricht.

Jedem Benutzer (bzw. auch jeder Benutzergruppe) kann im Rahmen einer vorgegebenen Grundstruktur eine für ihn zugeschnittene Benutzungsoberfläche bereitgestellt werden. Damit kann erreicht werden, daß er eine für ihn optimale Benutzerführung erhält und außerdem können damit nicht relevante Teile der Datenbank vor ihm verborgen werden.

Durch die Kombination aus Zugriffsrechten auf Datensätze und der Selektion der sichtbaren Attribute eines Datensatzes mit Hilfe der Oberfläche, ist es möglich eine individuelle Zugriffskontrolle für jedes Attribute eines jeden Datensatzes vorzunehmen

Aus Sicht des Datenschutzes sind in Abbildung 3.2 noch drei wichtige Punkte zu erkennen. Erstens kann jedem Benutzer verschiedene Termine haben. Auf diese Art kann für alle Benutzer jeweils ein Zeitpunkt für die Paßwortänderung festgelegt werden. Auch die

Aufforderung zur Löschung von Datensätzen kann auf diese Weise verwirklicht werden. Womit sich schon der zweite Punkt andeutet. Beim Löschen der Datensätze müssen diese anonymisiert werden. Damit aber keine (wesentlichen) Verfälschungen bei statistischen Auswertungen auftreten (z.B. doppeltes Löschen), wird beim Anonymisieren ein Hash-Wert gesetzt, der den Datensatz eindeutig identifiziert, aber trotzdem keinen Rückschluß auf dessen ursprünglichen Inhalt erlaubt. Auf das verwendete Hashverfahren wird im nächsten Abschnitt näher eingegangen. Als dritter Punkt ist abschließend das Protokollieren (Logging) von Änderungen an Datensätzen zu nennen.

5 Kryptographie

Da sich mit dem zu Verfügung stehenden Datenbank-Server der Firma Sybase nur eine rudimentäre Sicherheit der Datenübertragung gewährleisten läßt, wird dieser nur lokal eingesetzt und ist somit nicht von außen über das Netzwerk zugänglich. Für die Kommunikation mit dem Applet wird deshalb ein Server implementiert. Dieser Server baut zusammen mit dem Applet eine verschlüsselte Verbindung auf, über welche die gesamte weitere Kommunikation stattfindet. Die notwendigen Algorithmen hierfür sind Thema dieses Abschnitts.

Auf die kryptographischen Grundlagen soll hier allerdings nicht eingegangen werden, da dies den Rahmen dieser Ausarbeitung sprengen würde und es außerdem genügend gute Literatur zu diesem Thema gibt. Eine Auswahl ist im Literaturverzeichnis zu finden, z.B. [Baum98], [TiJe97] oder [Pete99]. Ziel dieses Abschnitts ist die Darlegung des gewählten Verfahrens zum Aufbau der verschlüsselten Verbindung, zu dem die Authentifikation anhand öffentlicher Verschlüsselung aus [Tane97] Pate gestanden hat.

Um den Datenaustausch zwischen Client und Server zu sichern, wird mit Hilfe kryptographischer Verfahren ein sicherer Kanal aufgebaut. Diese bedeutet, daß die Daten zwischen Client und Server verschlüsselt ausgetauscht werden. Zu diesem Zweck muß man sich deshalb auf ein Verschlüsselungsverfahren einigen, daß mit einer Geschwindigkeit arbeitet, die noch einen annehmbaren Betrieb der Datenbank-Anwendung ermöglicht, d.h. das keine wesentlichen Verzögerungen bei der Bedienung auftreten.

Für die Einhaltung dieser Anforderung kommt somit nur ein symmetrisches Verschlüsselungsverfahren in Frage. Hierfür wurde der IDEA Algorithmus gewählt, da dieser eine ausreichend starke Verschlüsselung ermöglicht. Im Gegensatz zum verbreiteten DES Algorithmus, der durch eine zu geringe Schlüssellänge nicht sicher verschlüsseln kann, werden im IDEA Algorithmus Schlüssel mit einer Länge von 128 Bit eingesetzt, was eine sichere Verschlüsselung gewährleistet.

Symmetrische Verschlüsselungsverfahren benötigen einen geheimen Schlüssel zwischen den Kommunikationspartnern. Für diesen Zweck wird ein Sitzungsschlüssel erstellt, der dann über asymmetrische Verfahren ausgetauscht wird. Als asymmetrischer Verschlüsselungsalgorithmus kommt RSA mit 1024 Bit langen Schlüsseln zum Einsatz. Zu erwähnen bleibt noch, daß außerdem ein sicheres Hashverfahren benötigt wird, wobei SHA (160 Bit langer Hash-Wert) verwendet wird.

Für die Datenbank-Anwendung ist nunmehr eine Implementierung dieser Algorithmen nötig. Da durch die in den USA geltenden Exportbeschränkungen für kryptographie Produkte (diese fallen dort unter das Kriegswaffenexportgesetz) die Java Cryptography Extension (JCE), die von SUN selbst angebotene wird, nicht legal beziehbar ist, kann diese nicht eingesetzt

werden. SUN verweist aber auf seiner Homepage selbst auf Drittfirmen bzw. Institutionen in aller Welt, die eigene Implementierungen für die von SUN vorgegebene Schnittstelle anbieten. Für dieses Projekt wurde die Implementierung von IAIK in Österreich gewählt, da sie eine sehr umfangreiche Sammlung von Algorithmen implementiert hat und vor allem, weil das Softwarepaket für wissenschaftliche Zwecke frei verfügbar ist.

Nach diesen langen Vorreden kann nun zum eigentlichen Algorithmus für den Austausch des Sitzungsschlüssels gekommen werden. Der Algorithmus läuft in zwei Stufen ab. Zunächst wird zwischen Server und Client ein Sitzungsschlüssel auf eine Art ausgetauscht, die verhindert, daß ein Dritter sich mit einer Bucket-Bridge-Attacke zwischen die beiden Kommunikationspartner schalten kann. Zu diesem Zeitpunkt ist es immer noch möglich das einer der Kommunikationspartner ein Eindringling ist. Deshalb authentifizieren sich Server und Client in der zweiten Stufe gegenseitig.

Der Ablauf in der ersten Stufe ist vollkommen symmetrisch. Server und Client tauschen zunächst ihre öffentlichen Schlüssel (asymmetrisches Verschlüsselungsverfahren) direkt ohne Verschlüsselung über das Netz aus. Der Client schickt danach die geraden Bits, einer mit dem öffentlichen Schlüssel des Servers verschlüsselte Zufallszahl an den Server. Der Server antwortet darauf mit den geraden Bits, einer von ihm erstellten mit dem öffentlichen Schlüssel des Clients verschlüsselte Zufallszahl. Danach werden die ungeraden Bits ausgetauscht. Darauf folgt ein zweiter Durchgang, in dem allerdings die jeweils entschlüsselte Zufallszahl, jeweils eine neu erstellte Zufallszahl und jeweils ein Vorschlag für einen Sitzungsschlüssel verschickt werden. Konnten darauf beide ihre im ersten Durchlauf verschickte Zufallszahlen wieder erkennen, dann gilt der vom Server erstellte Sitzungsschlüssel. An dieser Stelle sei angemerkt, daß das Verfahren mit dem Versenden der Teilnachrichten als Interlock-Protokoll bezeichnet wird.

In der zweiten Stufe authentifizieren sich die beiden Kommunikationspartner gegenseitig über den mit dem Sitzungsschlüssel verschlüsselten Kanal (symmetrisches Verschlüsselungsverfahren). Zunächst schickt der Client seinen Benutzernamen, eine Zufallszahl und den Hash-Wert aus Paßwort und Zufallszahl an den Server. Kommt dieser mit der Zufallszahl und dem ihm bekannten Paßwort zum selben Hash-Wert, dann ist der Client authentifiziert. In diesem Fall antwortet der Server mit einer neuen Zufallszahl und dem Hash-Wert aus Paßwort und neuer Zufallszahl. Kommt der Client mit der neuen Zufallszahl und seinem Paßwort auf den gleichen Hash-Wert, wie der vom Server empfangene, dann ist auch der Server gegenüber dem Client authentifiziert, und die Verbindung steht.

Abschließend sei noch auf einen allgemeinen Schwachpunkt dieser und ähnlicher Lösungen hingewiesen. Um dem Benutzer der Datenbank-Anwendung zu gewährleisten, daß er es auch mit dem echte Applet zu tun hat, müßte eine Authentisierung des Applets stattfinden. Da die in Java vorgesehenen Methoden zur Authentisierung [CoHo98] von den derzeitigen Standard-Webbrowsern nicht unterstützt werden, können diese nicht verwendet werden und es muß versucht werden eine Authentisierung mit Hilfe des WWW-Servers durchzuführen.

Kapitel 4: Systemarchitektur

Nachdem in den vorherigen Kapitel die Anforderungen im Bezug auf Funktionalität und Datenschutz ausführlich erläutert worden sind, kann nun auf die Entwurf der Datenbank-Anwendung eingegangen werden. Zu diesem Zweck wird in diesem Kapitel zunächst die gewählte Systemarchitektur vorgestellt. Es werden hierbei sowohl grundlegende Entwurfsentscheidungen, als auch die benötigten Systemkomponenten und die Schnittstellen zwischen ihnen erläutert. Beim Erläutern der grundlegenden Entwurfsentscheidungen werden wichtige Punkte des vorhergehenden Kapitels nochmals kurz zusammen gefaßt.

1 Umgebungs- und Randbedingungen

In Abbildung 4.1 ist eine Übersicht des Gesamtsystem dargestellt. Die grau unterlegten Symbole heben die im Rahmen dieser Diplomarbeit zu implementierenden Systembestandteile hervor.

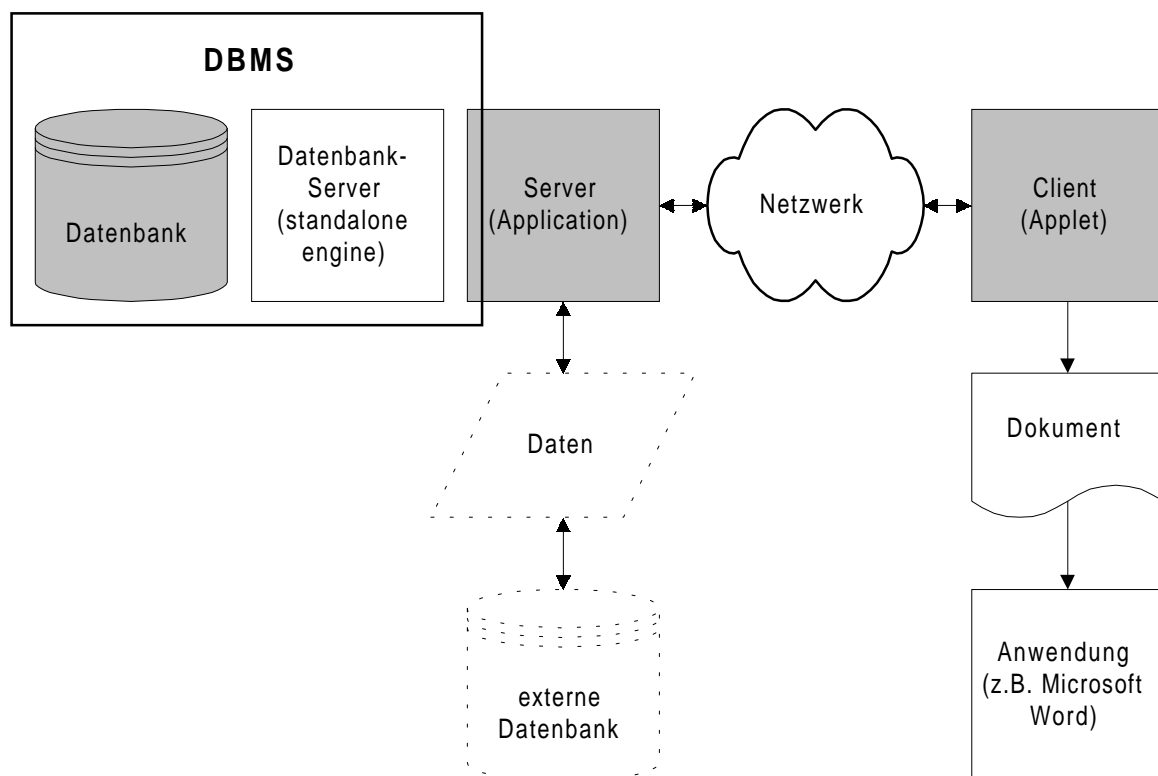


Abbildung 4.1: Systemübersicht

Wie zu erkennen ist, zerfällt das System in eine Server- und Client-Seite, die über ein Netzwerk miteinander kommunizieren können. Als Clients dienen Applets, die in Webbrowsern ausgeführt werden und für die Interaktion mit den Benutzern der Datenbank-Anwendung zuständig sind. Der Server hingegen ist für die Speicherung der Daten und für die Koordinierung der Zugriffe der Clients auf die Daten zuständig. Er zerfällt im Gegensatz zum

Client in mehrere Teilsysteme. Wie in Abbildung 4.1 zu sehen ist, sind dies die eigentliche Datenbank, der Datenbank-Server, der Zugriffe auf die Datenbank erst möglich macht, und der Server, der die Anbindung an das Netz vollzieht.

Weiterhin ist in Abbildung 4.1 zu sehen, das ein Datenaustausch mit externen Datenbanken über die Server-Seite erfolgen kann. Da allerdings derzeit noch keine dafür verwendeten Datenbanken existieren, sind auch noch keine zu unterstützende Datenformate bekannt. Dies hat zur Folge, daß die hierfür nötigen direkten Schnittstellen im Rahmen dieser Arbeit noch nicht implementiert werden konnten, deshalb wird lediglich der Import und Export mit Hilfe von SQL-Skripten vorgesehen. Vollständig unterstützt wird hingegen der Export von Daten über Text- und HTML-Dateien für die Weiterverarbeitung in Standard-Softwarepaketen. Außerdem ist der Datenexport per Zwischenablage möglich.

2 Grundlegende Entwurfsentscheidungen

2.1 Datenbanksystem

Es wurde die Client/Server-Architektur „Verteilte Funktion“ nach [Mits99] für das Datenbanksystem gewählt. Kennzeichnend für diese Architektur ist die Verteilung der Komponenten der Anwendungsfunktionen auf verschiedene Rechner (distributed function model). Das Prinzip dieser Architektur ist in Abbildung 4.2 zu sehen.

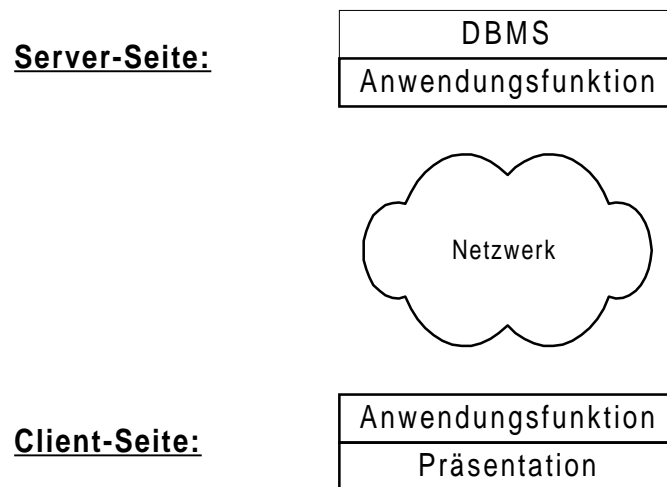


Abbildung 4.2: Datenbanksystem

Da die Aufteilung der Komponenten leistungskritisch ist, muß auf die Art der Datenzugriffe und auf Lokalität bei wiederholten Datenreferenzen geachtet werden. Nach [Mits99] empfiehlt sich bei herkömmlichen kurzen Transaktionen folgende Aufteilung:

Server-Seite:

- gesamte Datenhaltung
- transaktionsbezogener Teil der Anwendung (gesamte Datenmanipulation, Commit)

Client-Seite:

- lokale Anwendungsverarbeitung
- Steuerung der graphischen Benutzungsoberfläche

Wie bereits erläutert wurde, bestehen die Clients aus Applets, die in einem Webbrowser ablaufen. Der eigentliche Server wird ergänzt durch einen Datenbank-Server und natürlich der Datenbank selbst. Außerdem muß auf Server-Seite noch ein WWW-Server eingerichtet sein, der die HTML-Seite mit dem Applet bereitstellt. Der WWW-Server darf nicht auf einem anderen Rechner beheimatet sein, weil sonst das Applet auf Client-Seite nicht mit dem Server kommunizieren kann, da Applets in Webbrowsern lediglich den Zugriff auf den Rechner erlauben, von dem sie herunter geladen worden sind.

In der im Rahmen dieser Diplomarbeit zu erstellenden Datenbank-Anwendung, ist die eigentliche Datenbank auf dem Rechner beheimatet, auf dem auch der WWW-Server eingerichtet ist. Ein spätere Weiterentwicklung könnte allerdings eine Verteilung der Datenbank auf mehrere Rechner erfordern, deshalb wird darauf geachtet, daß in der für den Server vorgesehen Schichtenarchitektur eine Schicht vorgesehen wird, welche die tatsächliche Struktur der Datenbank verbergen kann.

2.2 Benutzerverwaltung

Um die Identifizierung von rechtmäßigen Benutzern der Datenbank-Anwendung zu ermöglichen, wird beim Einloggen der Benutzername und das Paßwort abgefragt, erst dann wird die eigentliche Anwendung geladen.

Die Benutzungsoberfläche der Datenbank-Anwendung ist für jeden Benutzer anders und auf seine Bedürfnisse zugeschnitten, allerdings wird dabei immer die gleiche Struktur eingehalten, was später noch genauer erläutert werden wird. Deshalb werden außer den für die Benutzerverwaltung benötigten Informationen auch jeweils die benutzerspezifischen Java-Komponenten in der Datenbank gespeichert. Nach erfolgtem Einloggen werden diese aus der Datenbank in das Applet geladen. Sowohl die benutzerspezifischen Java-Komponenten, wie auch allgemein gültigen Java-Komponenten werden jeweils in einer JAR-Dateien (komprimiert) abgelegt und können somit auf einmal geladen werden, womit eine schnellere Übertragung gewährleistet wird.

Jeder Datensatz gehört einem Benutzer und ist außerdem einer Benutzergruppe zugeordnet. Ein Benutzer bzw. eine Benutzergruppe kann nur dann auf einen Datensatz zugreifen, wenn die Berechtigung dazu vorliegt, es gibt Schreib- und Leserechte. Wie im vorherigen Kapitel schon beschrieben wurde, ist dieses Konzept an das Unix-Dateisystem angelehnt. Zu erwähnen bleibt noch, daß auch Teile eines Datensatzes vor einem Benutzer bzw. einer Benutzergruppe verborgen werden können, in dem eine Sicht auf die Datenbank erzeugt wird, in der nur ein Teil der Attribute eines Datensatzes enthalten sind.

2.3 Datenschutz

Wie im Kapitel über den Datenschutz bereits beschrieben wurde, gliedert sich der Datenschutz in mehrere Ebenen. Die im Rahmen der Erstellung einer Datenbank-Anwendung wichtigen und somit auch im Rahmen dieser Arbeit implementierten Ebenen sind die Ebene der Authentisierung, die Ebene der Zugriffskontrolle und die Ebene der Kryptographie. Die Problematik der Zugriffskontrolle wurde schon im vorigen Abschnitt über die Benutzerverwaltung diskutiert.

Um dem Benutzer zu gewährleisten, daß er es auch mit dem echte Applet zu tun hat, findet eine Authentisierung des Applets statt. Da die in Java vorgesehenen Methoden zur

Authentisierung [CoHo98] von den derzeitigen Standard-Webbrowsern nicht unterstützt werden, wird die Authentisierung mit Hilfe des WWW-Serves durchgeführt.

Da sich mit dem zu Verfügung stehenden Datenbank-Server der Firma Sybase nur eine rudimentäre Sicherheit der Daten gewährleisten läßt, wird dieser nur lokal eingesetzt und ist somit nicht von außen über das Netzwerk zugänglich. Für die Kommunikation mit dem Applet wird deshalb ein Server implementiert. Dieser Server baut zusammen mit dem Applet eine verschlüsselte Verbindung auf, über welche die gesamte weitere Kommunikation stattfindet.

2.4 Anwendungsstruktur

Wie bereits erwähnt wurde, stellt sich die grundlegende Struktur der Datenbank-Anwendung für alle Benutzer gleich dar. Über ein Hauptfenster sind die jeweiligen Datensatzklassen zu erreichen. Von dort aus können über Buttons bzw. Menüleiste Funktionen auf den Datensätzen ausgeführt werden. Bei Bedarf werden weitere Fenster geöffnet, wie z.B. für Filter, Listen und Statistiken.

Da dieses Grundgerüst für alle Benutzer gleich ist, können Komponenten erstellt werden, die durch diverse Parameter an die jeweiligen Benutzer angepaßt werden können. In Fällen, wie etwa der Darstellung von Datensätzen im Hauptfenster, in denen keine allgemein gültige und dynamisch anpaßbare Struktur erzielt werden kann, wird zumindest ein Skelett für die jeweilige manuelle Erstellung von Komponenten vorgegeben.

Die graphische Darstellung der Benutzungsoberfläche wird von der reinen Anwendung getrennt (vergleiche Abbildung 4.3). Zu diesem Zweck wird auf der Client-Seite eine zusätzliche Schicht eingeführt, welche die Steuerung der graphischen Komponenten vornimmt. In den Objekten der Benutzungsoberfläche darf also keine Logik enthalten sein, die mit der Anwendung selbst zutun hat. Die Kommunikation zwischen diesen beiden Schichten erfolgt eventbasiert.

Diese Konzeption ist ein erster Schritt zu einem durchgehenden „Model-View-Control“-Paradigma [GHJV95]. Es fehlt eigentlich nur eine „Model“-Schicht, auf der die Datenbank-Anwendung lediglich spezifiziert werden muß und die gewünschten Abläufe in der Grafik- bzw. der Steuerungsschicht können daraus automatisch abgeleitet werden. Da hierfür derzeit allerdings eine geeignete Konzeption fehlt, ist dies im Rahmen dieser Arbeit nicht zu verwirklichen.

Unter der Steuerungsschicht befindet sich deshalb die Datenschicht. Die Datenschicht stellt die verschiedenen Datenobjekte zu Verfügung, auf der die Steuerungsschicht operiert. Die Datenschicht ist auf Client und Server verteilt. Die relationale Datenhaltung in der Datenbank ist auf Server-Seite verwirklicht. Aus den Daten die in der Datenbank gespeichert sind, werden Objekte erzeugt. Diese Objekte werden auf Client-Seite der steuernden Schicht zur Verfügung gestellt.

3 Diagramm der Software-Systemarchitektur

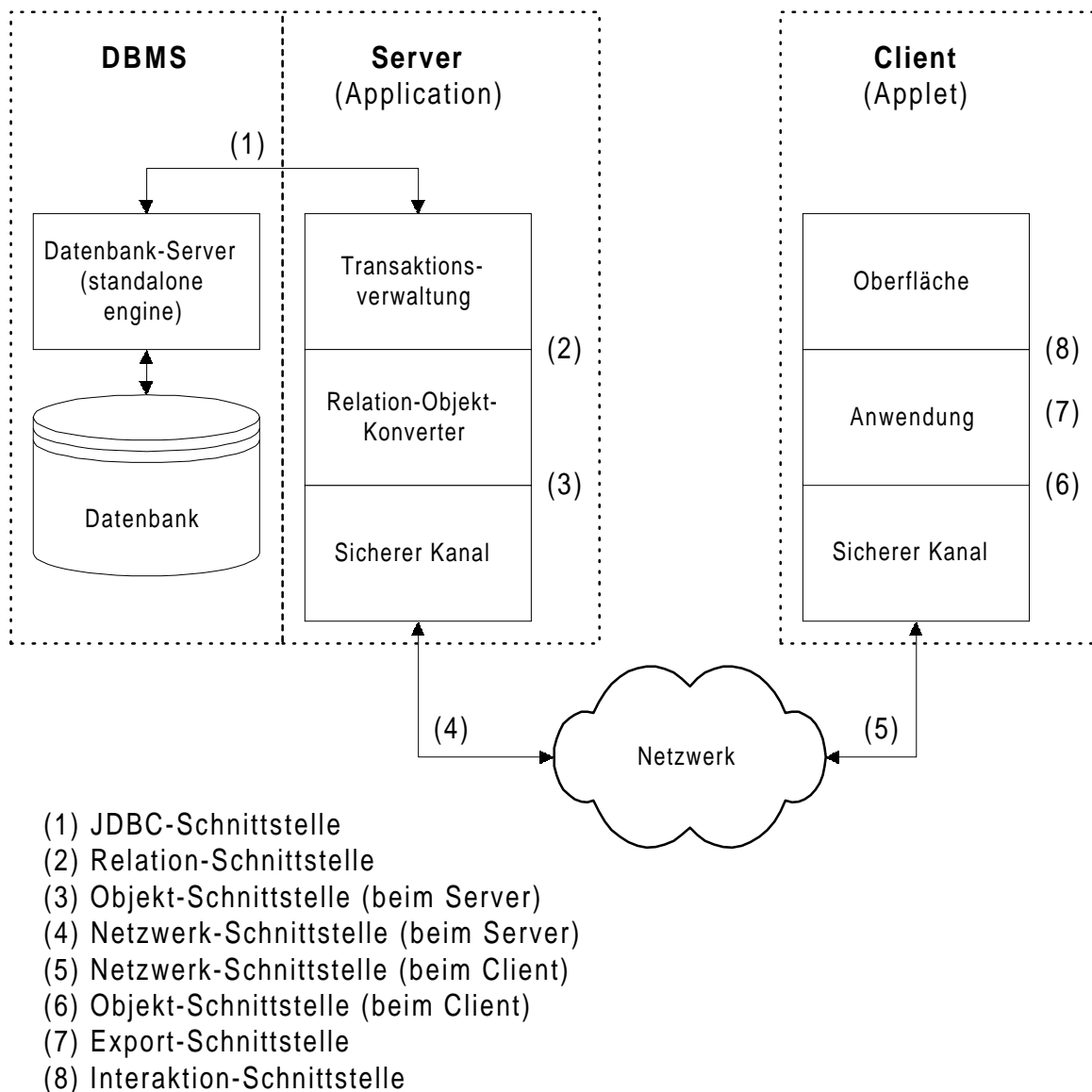


Abbildung 4.3: Software-Systemarchitektur

4 Softwarekomponenten

4.1 DBMS

Das DBMS besteht aus der erstellten relationalen Datenbank, die alle für das Dekanat der Fakultät Elektrotechnik und Informationstechnik relevanten Daten enthält, und der „standalone“ Engine des Datenbank-Servers von Sybase, der Zugriffe auf die Datenbank per SQL ermöglicht. Über das DBMS wird außerdem die Benutzerverwaltung abgewickelt. Dafür werden die Benutzerinformationen zusammen mit den benutzerspezifischen Komponenten der Datenbank-Anwendung gespeichert.

Jeder Datensatz in der Datenbank hat seine eigenen Zugriffsrechte. Durch die vergebenen Zugriffsrechte in Kombination mit speziellem Sichten auf die Datenbank für die

verschiedenen Benutzer, kann für jeden Benutzer exakt die ihm zur Verfügung stehende Informationsmenge spezifiziert werden.

4.2 Transaktionsverwaltung

Die Transaktionsverwaltung stellt sicher, daß bei Mehrbenutzerbetrieb keine Inkonsistenzen in der Datenbank entstehen können. Für diesen Zweck wird das Zwei-Phasen-Sperrprotokoll eingesetzt, wie es in [KeEi99] beschrieben wird. Zur Vermeidung von Verklemmungen werden Zeitstempel eingesetzt. Außerdem werden dynamisch vergebene Schreib- und Leserechte für die verschiedenen Datensätze verwendet. Die Verwirklichung dieser Funktionalitäten erfolgt mit Hilfe geeigneter Einstellung des Datenbank-Servers.

Die Transaktionsverwaltung könnte die Möglichkeit bieten, die Datenhaltung auf mehreren Datenbanken zu verteilen. Diese Datenbanken müßten dabei nicht auf dem gleichen Rechner beheimatet sein. Außerdem ist über sie die Anbindung einer externer Datenbanken denkbar. Beide Einsatzmöglichkeiten konnten im Rahmen dieser Arbeit nicht implementiert werden, da das zukünftige Systemumfeld noch nicht in allen Details bekannt ist.

4.3 Relation-Objekt-Konverter

Der Relation-Objekt-Konverter erstellt aus einer Relation ein Objekt bzw. extrahiert aus einem Objekt eine Relation. Der Relation-Objekt-Konverter stellt also die Schnittstelle zwischen dem relationenorientierten SQL auf der Datenbankseite und dem objektorientierten Java auf Anwendungsseite dar. Der Relation-Objekt-Konverter ist die einzige Komponente, die eine genauere Kenntnisse von der relationalen Datenstruktur in der Datenbank haben sollte.

Der Relation-Objekt-Konverter ist des weiteren unverzichtbarer Bestandteil der Systemarchitektur, da er für die eigentliche Anpassung an die JDBC-Schnittstelle verantwortlich ist, z.B. muß er Ergebnisse von Datenbankabfragen für die Client-Seite geeignet umsetzen.

4.4 Sicherer Kanal

Der Sichere Kanal besteht aus einer Client- und einer Server-Komponente. Diese bauen nach einem festen Protokoll eine verschlüsselte Verbindung auf (siehe Kapitel zum Datenschutz), über die danach der Datenaustausch stattfindet. Da mehrere Clients gleichzeitig mit dem Server verbunden sein können müssen, kommt auf Server-Seite Multithreading zum Einsatz. Aus diesem Grund folgt auch die Notwendigkeit der Transaktionsverwaltung.

4.5 Anwendung

Die Anwendung ist der Kern des gesamten Systems. Von ihr aus wird die Benutzungsoberfläche gesteuert und von ihr werden Datenabfragen bzw. Datenmanipulationen angestoßen. Sie ist also hauptsächlich für die Steuerung des Systems verantwortlich. In Richtung der Datenhaltung kommuniziert sie durch das Versenden und Empfangen von Datenobjekte, mit der Oberfläche werden hingegen Events ausgetauscht.

Als weitere Aufgabe fällt der Anwendung der Export von Daten auf der Client-Seite per Datei oder Zwischenablage zu.

4.6 Oberfläche

Die Oberfläche besteht aus einer Reihe von Komponenten, die zur Interaktion mit dem Benutzer dienen. Die Komponenten stellen Informationen für den Benutzer dar und nehmen Eingaben von ihm entgegen. Die Kommunikation mit der Anwendungsschicht erfolgt ausschließlich über Events. Die Oberfläche enthält lediglich Logik, die direkt mit der grafischen Darstellung zu tun hat, nicht aber mit der Anwendung selbst.

Die Komponenten der Oberfläche wurden so erstellt, daß sie möglichst allgemein gehalten sind und somit über Parameter auf ihren jeweiligen Einsatz eingestellt werden können. Ein Beispiel dafür ist etwa eine Statistik-Komponente die beliebige Daten auf verschiedene Arten als eindimensionales Balkendiagramm darstellen kann.

5 Schnittstellendefinitionen

5.1 JDBC-Schnittstelle

Die JDBC-Schnittstelle dient zur Übergabe von SQL-Statements durch die Transaktionsverwaltung an den Datenbank-Server. Außerdem werden über sie die Ergebnisse von Datenbankabfragen zurückgeliefert.

Da JDBC eine spezifizierte Schnittstelle ist, möchte ich hier nicht näher auf deren Bestandteile eingehen und verweise statt dessen auf die einschlägige Literatur, wie etwa [CoHo98] oder [CWHT98].

5.2 Relation-Schnittstelle

Die Relation-Schnittstelle übergibt SQL-Befehle der verschiedenen Benutzerprozesse an die Transaktionsverarbeitung. Als Ergebnis einer solchen Anfrage wird ein Tabelle zurückgeliefert, sofern der SQL-Befehl ein Ergebnis erzeugt.

5.3 Objekt-Schnittstelle

Die Objekt-Schnittstelle besteht aus einem Objekt-Input- und einem Objekt-Output-Stream auf Client- und auf Server-Seite. Über die Streams werden Objekte der Klasse „Object“ ausgetauscht, womit die Übertragung fast beliebiger Objekte, also auch von Jar-Dateien, ermöglicht wird. Einzige Bedingung ist, daß die Objekte serialisierbar sind.

Die Objekt-Schnittstelle garantiert eine sichere Übertragung zwischen Client und Server über einen verschlüsselten Kanal.

5.4 Netzwerk-Schnittstelle

Die Netzwerk-Schnittstelle besteht aus je einem Byte-Stream zwischen der Client-Seite und einem dynamisch einstellbaren Port auf Server-Seite und. Somit wird die Übertragung von vollkommen beliebigen Daten zwischen Client und Server ermöglicht. Da mehrere Clients gleichzeitig mit dem Server kommunizieren können sollen, ist der Port auf der Server-Seite multiplex-fähig.

5.5 Export-Schnittstelle

Über die Export-Schnittstelle können per Text- bzw. HTML-Datei oder auch per Zwischenablage Daten in Standardanwendungen, wie etwa Microsoft Word, exportiert werden.

Für den Export über die Zwischenablage werden die in Java allgemein verwendbaren Schnittstellen eingesetzt. Soll der Export allerdings per Datei erfolgen, ist dies für Applets, die in einem Webbrowser ausgeführt werden, nicht ohne weiteres möglich, da es diesen normalerweise nicht gestattet ist, auf die lokalen Speichermedien zuzugreifen. Um diese Funktionalität trotzdem zu ermöglichen, müssen dem Applet zusätzliche Rechte erteilt werden, was später noch erläutert werden wird.

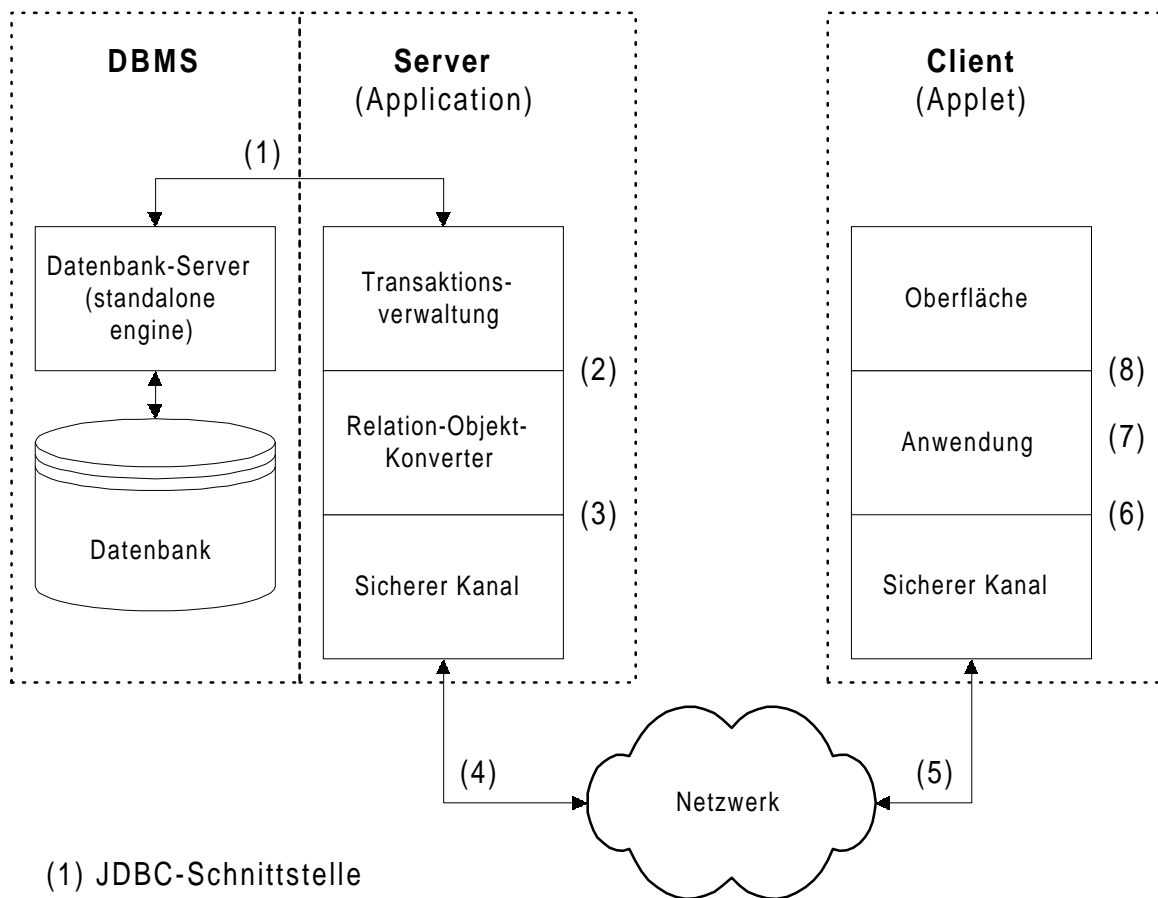
5.6 Interaktion-Schnittstelle

Die Interaktion-Schnittstelle ist rein eventbasiert. Von der Anwendung werden die einzelnen Komponenten der Oberfläche gesteuert und eingestellt. Dabei kommen neben neu erstellten Events auch Standard-Events zum Einsatz, da die Komponente der Oberfläche von in Java enthaltenen Komponenten abgeleitet werden.

Interagiert der Benutzer mit den Komponenten der Oberfläche, dann werden die Aktionen des Benutzer bzw. dessen Eingaben per Event an die Anwendung geliefert.

Kapitel 5: Implementierung

Im folgenden soll die Funktion und der Aufbau aller Komponenten detailliert beschrieben werden, aus denen die Datenbank-Anwendung besteht. In Abbildung 5.1 ist nochmals die Übersicht über die Systemarchitektur zu sehen, in der die Beziehungen unter den Komponenten veranschaulicht werden. Abschließend wird die Einbindung des Applets in die Standard-Webbrowser beschrieben.



- (1) JDBC-Schnittstelle
- (2) Relation-Schnittstelle
- (3) Objekt-Schnittstelle (beim Server)
- (4) Netzwerk-Schnittstelle (beim Server)
- (5) Netzwerk-Schnittstelle (beim Client)
- (6) Objekt-Schnittstelle (beim Client)
- (7) Export-Schnittstelle
- (8) Interaktion-Schnittstelle

Abbildung 5.1: Software-Systemarchitektur

1 Komponente DBMS

1.1 Aufgabe

Das DBMS beinhaltet die relationale Datenbank, die alle für das Dekanat der Fakultät für Elektrotechnik und Informationstechnik relevanten Daten enthält, und ermöglicht die Verwaltung dieser Datenbank per SQL.

Über das DBMS wird außerdem die Benutzerverwaltung abgewickelt. Dafür werden die Benutzerinformationen zusammen mit den benutzerspezifischen Komponenten der Datenbank-Anwendung gespeichert. Die benutzerspezifischen Komponenten werden in einer JAR-Datei mit jeweils allen Klassen abgespeichert. Es sei an dieser Stelle angemerkt, daß auch die allgemeinen Bestandteile des Applet ebenfalls als JAR-Datei auf dem Web-Server abgelegt werden

Jeder Datensatz in der Datenbank hat seine eigenen Zugriffsrechte. Durch die vergebenen Zugriffsrechte in Kombination mit speziellem Sichten auf die Datenbank für die verschiedenen Benutzer, kann für jeden Benutzer exakt die ihm zur Verfügung stehende Informationsmenge spezifiziert werden.

1.2 Bestandteile

Wie in der Abbildung 5.1 zu sehen ist, besteht das DBMS aus zwei Bestandteilen, einer (relationalen) Datenbank und einem Datenbank-Server.

Die relationale Datenbank ist in der Lage, alle für das Dekanat relevanten Daten aufzunehmen. Die Datenbank wird automatisch aus einer relationalen Beschreibung dieser Daten mit dem Werkzeug PowerDesigner erstellt. Da alle personenbezogenen Datensätze nach bestimmten Fristen gelöscht werden sollen, werden die für statistische Auswertungen nötigen Daten in anonymisierter Form dauerhaft gespeichert. Damit hierbei keine die Statistik verfälschende Doppelseinträge entstehen können, wird zu jedem anonymisierten Datensatz ein eindeutiger Hashwert gespeichert. Dieser Hashwert wird vor dem Anonymisieren aus dem ursprünglichen Datensatz ermittelt, er ermöglicht aber nicht dessen Rekonstruktion.

Da die Datenbank selbst keine SQL-Schnittstelle besitzt, ist ein Datenbank-Server nötig, der Zugriffe auf die Datenbank per SQL ermöglicht. Da dieser nicht den Anforderungen entsprechende Sicherungsfunktionen besitzt, wird dessen „standalone“ Engine eingesetzt. Er ist somit nicht mehr direkt über das Netz zugänglich.

Der Datenbank-Server bietet außerdem eine weitgehende Verwaltung der Datenbank per SQL an. Neben der Abfrage und der Manipulation von Datensätzen stellt er zum Beispiel die Möglichkeit zur Verfügung, Transaktionen durchzuführen und Datensätze zeitweilig für verschiedene Zugriffsarten zu sperren.

2 Komponente Transaktionsverwaltung

2.1 Aufgabe

Die Transaktionsverwaltung ist dafür zuständig, daß die Zugriffe von verschiedenen Clients auf die Datenbank koordiniert ablaufen. Dies bedeutet vor allem, daß die Konsistenz der Daten gewährleistet bleibt. Die Clients können somit so vorgehen, als ob sie allein auf die Datenbank zugreifen würden.

2.2 Prinzip

Die Transaktionsverwaltung wird im wesentlichen über Funktionen realisiert, die durch das DBMS zur Verfügung gestellt werden. Die Transaktionsverwaltung selbst muß diese lediglich mit geeigneten SQL-Befehlen aufrufen. Beispielsweise wird das Zwei-Phasen-Sperrprotokoll schon vom DBMS bereitgestellt und eine Blockadeerkennung wird ebenfalls geboten.

Die wichtigste Aufgabe der Transaktionsverwaltung ist das Erstellen von Transaktionen aus den SQL-Aufrufen die von den verschiedenen Clients kommen. Die Clients gehen allerdings davon aus, daß sie allein auf die Datenbank zugreifen und wissen deshalb auch nichts von anderen Clients. Die Transaktionsverwaltung führt eine Serialisierung der verschieden Anfragen der Clients durch. Zu diesem Zweck werden Transaktionen erstellt und das DBMS so gesteuert, daß Locks gesetzt bzw. zurückgenommen und entstandene Verklemmungen erkannt werden. Weiterhin obliegt der Transaktionsverwaltung die Ergebnisse von SQL-Befehlen zurück zu liefern.

Neben Anfragen von Clients ist die Transaktionsverwaltung auch in der Lage, den Im- und Export von Daten durchzuführen. Dazu kann der Administrator entsprechende SQL-Befehle direkt an den Datenbank-Server schicken. Folgende Dateiformate werden dabei unterstützt:

- ASCII
- DBase II und DBase III
- Data Interchange Format (DIF)
- FIXED
- FoxPro
- Lotus Workspace
- WATFile

Da der Datenbank-Server normalerweise nicht auf Dateien zugreifen kann, die auf Client-Seite abgelegt sind, wird ein Mechanismus zur Verfügung gestellt, der Dateien zwischen der Client-Seite und einem temporären Verzeichnis auf der Server-Seite transferieren kann.

2.3 Klassendiagramm



Legende: ———▶ erbt
 - - -▶ verwendet

Abbildung 5.2: Klassendiagramm der Transaktionsverwaltung

3 Komponente Relation-Objekt-Konverter

3.1 Aufgabe

Der Relation-Objekt-Konverter ist dafür zuständig, daß aus Datenobjekten die von Client-Seite zum Server geschickt werden, SQL-Befehle erstellt werden. Umgekehrt müssen aus den Ergebnissen von SQL-Befehlen Datenobjekte erstellt werden, mit denen auf Client-Seite gearbeitet werden kann.

3.2 Prinzip

Der Relation-Objekt-Konverter erhält vom Client über den Sicheren Kanal ein Datenobjekt zugesandt. Aus diesem extrahiert er einen SQL-Befehl, den er dann an die Transaktionsverwaltung weiterleitet. Welche Funktion der Transaktionsverwaltung er dabei aufruft, hängt davon ab, ob der SQL-Befehl als Ergebnis Werte zurück liefert oder nur Operationen auf der Datenbasis ausführt.

Auf jedes Datenobjekt vom Client antwortet der Relation-Objekt-Konverter mit einem Datenobjekt als Ergebnis. In diesem Objekt ist neben eventuell ermittelten Daten auf jeden Fall eine Information darüber enthalten ob der SQL-Befehl ausgeführt werden konnte bzw. warum er fehlgeschlagen ist. Aus diesem Grund muß der Server auch die Datenobjekt-Klasse kennen, auf welcher der Relation-Objekt-Konverter arbeitet, d.h. die Datenobjekt-Klasse muß als einzige Klasse auf Server- und Client-Seite zwingend identisch sein.

Um aus einem Datenobjekt die von ihm geforderte Operation ermitteln zu können, enthält jedes Datenobjekt eine Angabe über seinen Typ. Beispielsweise stehen folgende Typen zur Verfügung: *getColumnData*, *executeSQL*, *transferFile*, *logout*, *error* usw. Wie man sieht, gibt es Typen für die unterschiedlichsten Aufgaben.

Als Erläuterung soll die Verwendung von *executeSQL* und *transferFile* näher beschrieben werden. Um etwa Daten in die Datenbank zu importieren zu können, wird mit *transferFile* eine Datei in das temporäre Verzeichnis beim Server transferiert. Mit *executeSQL* kann dann ein SQL-Befehl ausgeführt werden, der den Import durchführt. Die Datei bzw. der SQL-Befehl kann als "Nutzlast" bei der Übertragung des Datenobjekts betrachtet werden.

Bei näherer Betrachtung der ablaufenden Kommunikation könnte man an den Aufruf einer Funktion mit Parametern und der darauffolgenden Rückgabe von Werten erinnert werden. Auf diesen Umstand wird im Kapitel über Verbesserungs- und Erweiterungsvorschläge näher eingegangen.

3.3 Klassendiagramm

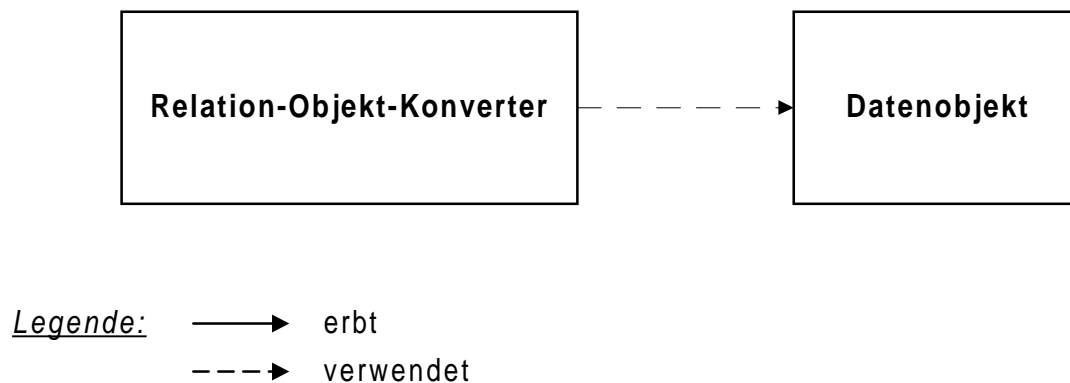


Abbildung 5.3: Klassendiagramm des Relation-Objekt-Konverters

4 Komponente Sicherer Kanal

4.1 Aufgabe

Der sichere Kanal ist für die verschlüsselte Übertragung von Daten bzw. Objekten über das Netz verantwortlich. Deshalb stellt er auch die Verbindung des Servers und des Clients zum Netz dar. Er ist außerdem für die korrekte Durchführung des Anmeldeprotokolls von Clients zuständig.

4.2 Prinzip

Die Komponente sicherer Kanal ist auf Server und Client Seite nicht identisch. Zwischen ihnen wird allerdings ein festes Protokoll abgearbeitet, z.B. für die Durchführung der Anmeldeprozedur. Dazu gibt es auf der Schicht des Sicheren Kanals zwei Verbindungen zwischen Server und Client. Eine für die Übertragung vom Client zum Server und die andere zur Übertragung vom Server zum Client. Eine Kommunikation erfolgt immer dadurch, daß vom Client ein Datenobjekt gesendet wird, worauf der Server mit einem Datenobjekt als Ergebnis oder Statusanzeige antwortet

Auf Server-Seite wird Multithreading eingesetzt, d.h. einen Thread für die Kommunikation für jeden Client. Jeder Thread und somit auch jeder Client wird durch eine Nummer eindeutig identifiziert. Für jeden Client werden auf Server-Seite Informationen zur Überwachung der Kommunikation durch einen Administrator ausgegeben. Durch die Nummer können dabei die Meldungen für verschiedenen Clients unterschieden werden.

Der Server ist über einen Port ansprechbar. Die Portnummern kann durch den Administrator dynamisch vergeben werden. Der Administrator kann die Portnummern beim Start der Server-Applikation eingeben. Deshalb wird beim Start der Server-Applikation ein Fenster geöffnet, die Portnummern eingetragen werden müssen. Außerdem muß dort auch der Name der ODBC-Verbindung angegeben werden, damit der Datenbank-Server automatisch gestartet werden und eine Verbindung zu ihm aufgebaut werden kann. Damit nicht jedesmal die gleichen Wert wieder eingegeben werden müssen, werden diese in einer ini-Datei abgelegt und beim nächsten Start als Default-Werte angeboten.

Um die Datenbank auch auf Server-Seite vor unberechtigtem Zugriff zu schützen, muß beim Start des Servers ebenfalls ein Benutzername und ein Paßwort eingegeben werden. Diese haben aber keine Gemeinsamkeit mit den Benutzernamen und Paßwörtern auf der Client-Seite.

4.3 Klassendiagramm

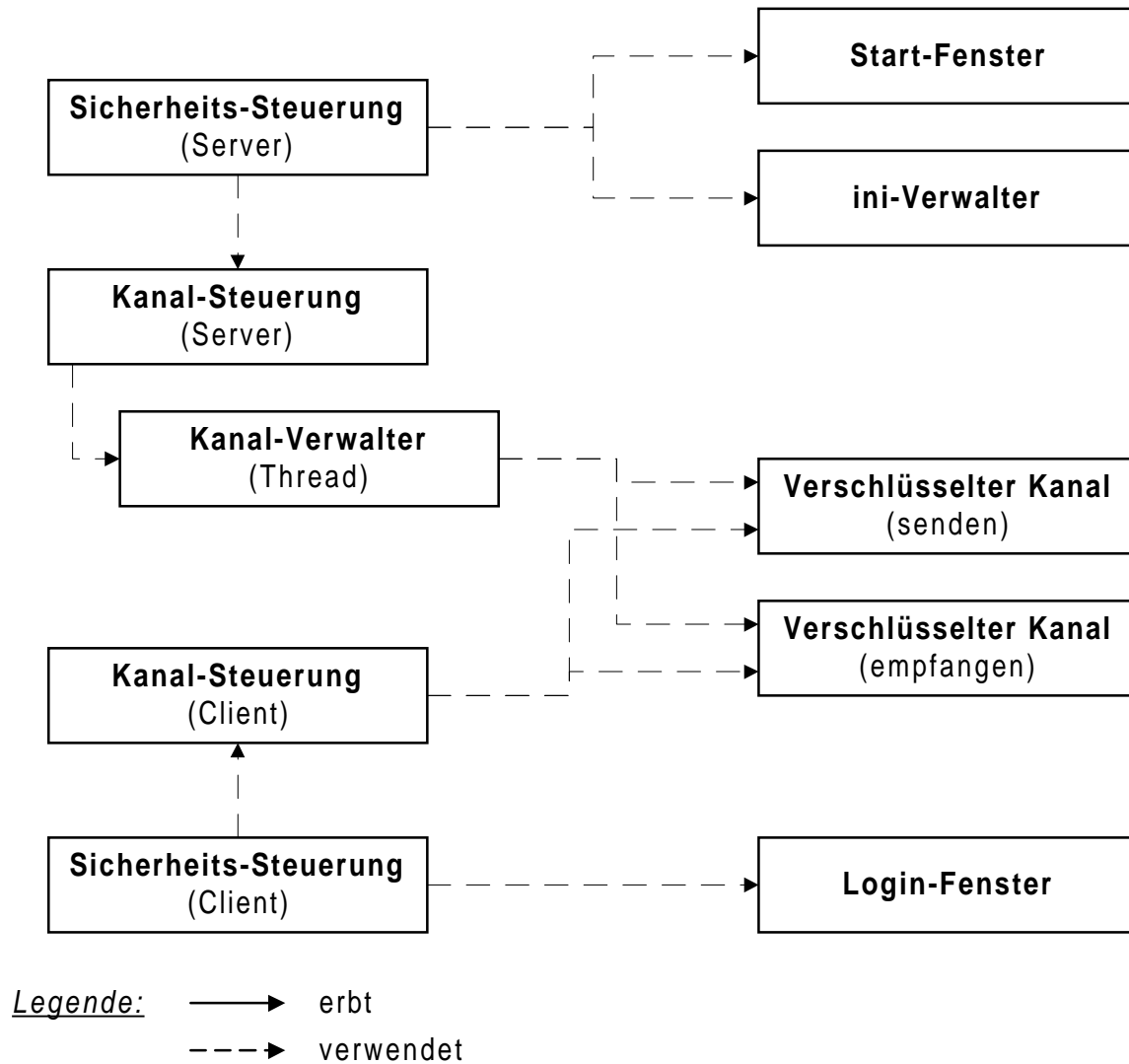


Abbildung 5.4: Klassendiagramm des Sicheren Kanals

5 Komponente Anwendung

5.1 Aufgabe

Die Komponente Anwendung ist für die Steuerung der Datenbank-Anwendung auf Client-Seite zuständig. Sie übernimmt dort die Verwaltung der lokalen Daten und interagiert über die Benutzungsschnittstelle, die sie aus Komponenten der Oberflächen-Schicht zusammenstellt, mit dem Benutzer. Außerdem ist sie für die Durchführung von Datenexporten per Text- bzw. HTML-Datei oder über die Zwischenablage verantwortlich.

5.2 Prinzip

In Abbildung 5.5 ist der prinzipielle Aufbau der Anwendungsschicht dargestellt. Wie man sehen kann, besteht die Anwendungsschicht im wesentlichen aus einer Anwendungssteuerung, die von diversen Hilfsobjekten unterstützt wird.

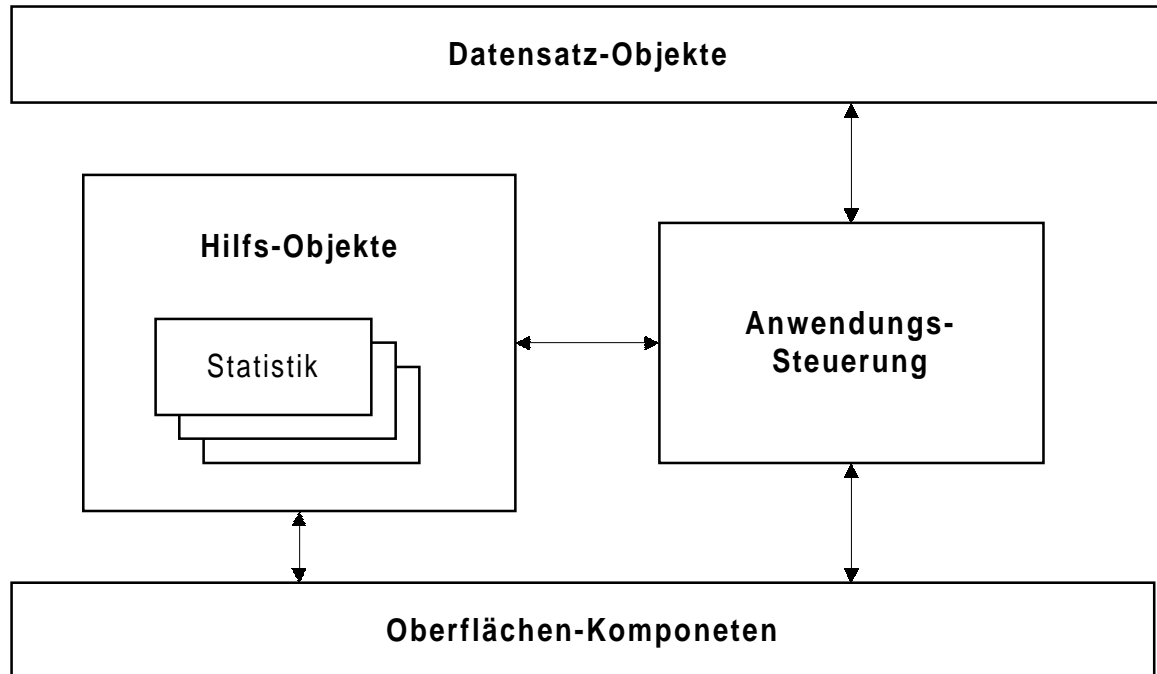
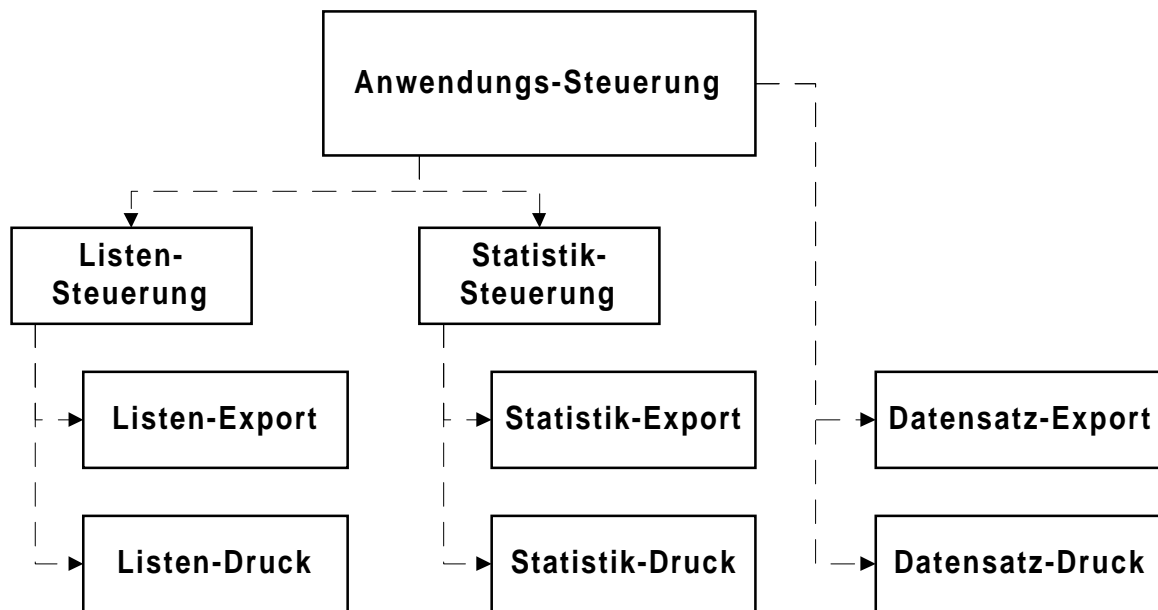


Abbildung 5.5: Anwendungsschicht

Die Anwendungssteuerung ist hauptsächlich für die Kommunikation mit den Datenobjekten verantwortlich, die vom Relation-Objekt-Konverter verarbeitet werden und ist somit auch für die Steuerung des Hauptfensters verantwortlich. Letzteres erfolgt über den Austausch von Events.

Für Aufgaben, die eine komplexe Funktionalität aufweisen, stehen der Anwendungssteuerung Hilfsobjekt zur Verfügung, die für sie diese Aufgaben erledigen. Hier sind vor allem die Erstellung von Listen und Statistiken, aber auch der Datenexport zu nennen. Selbstverständlich übernehmen die Hilfsobjekt die Steuerung der von ihnen benötigten Komponenten der Oberfläche selbst und verbergen diese Problematik somit vor der Anwendungssteuerung.

5.3 Klassendiagramm



Legende: ———▶ erbt
 - - - -▶ verwendet

Abbildung 5.6: Klassendiagramm der Anwendung

6 Komponente Oberfläche

6.1 Aufgabe

Zur Kommunikation mit dem Benutzer dienen die Komponenten der Oberfläche. Sie stellen Informationen dar und nehmen Befehle vom Benutzer entgegen, die sie weitestgehend an die Anwendungsschicht weiterleiten. Nur Befehle, die lediglich die grafische Repräsentation betreffen, werden von ihr direkt ausgeführt. Dies bedeutet, daß die Komponenten der Oberfläche nur Logik im Bezug auf die Steuerung ihrer eigenen Darstellung enthalten, nicht aber Logik die mit der Steuerung der Datenbank-Anwendung zutun hat. Letztere ist Bestandteil und Zweck der Anwendungsschicht.

6.2 Bestandteile

Die Struktur der Fenster für die Benutzerinteraktion auf Client-Seite ist in Abbildung 5.7 zu sehen. Die Beschreibung des Startfensters auf Server-Seite ist in der Beschreibung der Komponente Sicherer Kanal in Kapitel 4 zu finden, da in diesem Kapitel ausschließlich die Oberfläche der Datenbank-Anwendung auf Client-Seite erläutert wird, siehe dazu auch Abbildung 5.4.

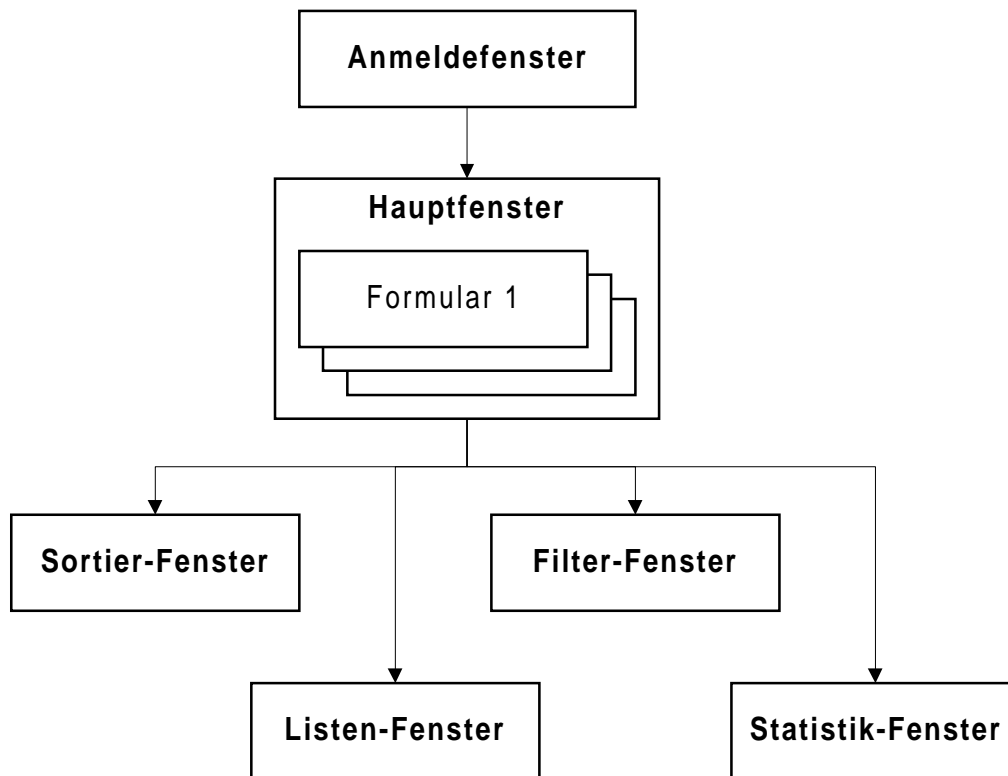


Abbildung 5.7: Fensterstruktur

Nachfolgend werden die in Abbildung 5.7 aufgeführten Fenster kurz beschrieben:

- *Anmeldefenster:*
Das Anmeldefenster ermöglicht es den Benutzern sich durch Eingabe ihres Benutzernames und des dazu gehörigen Paßworts sich gegenüber der Datenbank-Anwendung zu identifizieren und sich in diese einzuloggen.
- *Hauptfenster:*
Das Hauptfenster besteht aus einer Menüleiste und einem Register in dem für die verschiedenen Datensatzklassen jeweils ein Formular zu Verfügung steht. Da die Formulare des Registers für jeden Benutzer auf verschiedene Weise dargestellt werden, wird jeweils der Name des Formulars und dessen Inhalt aus der Datenbank geladen.
- *Sortier-Fenster:*
Im Sortier-Fenster kann der Benutzer spezifizieren, nach welchem Attribut eines Datensatzes sortiert jeweils die Menge der Datensätze aufgelistet werden soll. Die Sortierung wird für jedes Formular des Registers individuell eingestellt. Es kann jeweils eine Sortierhierarchie angegeben werden.
- *Filter-Fenster:*
Im Filter-Fenster kann angegeben werden, welche Datensätze einer Datensatzklasse angezeigt werden sollen. Dazu werden Kriterien auf den Attributen der jeweiligen Datensätze spezifiziert. Nur Datensätze die diese Kriterien erfüllen werden daraufhin noch aufgelistet.

- *Listen-Fenster:*
Das Listen-Fenster besteht aus zwei nacheinander angezeigten Fenstern. Im ersten Fenster wird spezifiziert, welche Attribute der Datensätze in die Liste aufgenommen werden sollen und wie diese angeordnet (z.B. nebeneinander) bzw. dargestellt (z.B. fett) werden sollen. Im zweiten Fenster erscheint dann die erzeugte Liste. Es werden schon einige Listen vordefiniert angeboten.
- *Statistik-Fenster:*
Das Statistik-Fenster besteht wie das Listen-Fenster aus zwei nacheinander angezeigten Fenstern. Im ersten Fenster wird ausgewählt, welche Daten auf welche Art ausgewertet werden sollen. Außerdem wird spezifiziert, wie die Darstellung des Ergebnisses erfolgen soll, z.B. rein textuell, als 2D- oder 3D-Grafik, gemischt usw. Im zweiten Fenster erscheint dann die erzeugte Statistik. Es werden schon einige Statistiken vordefiniert angeboten.

6.3 Klassendiagramm

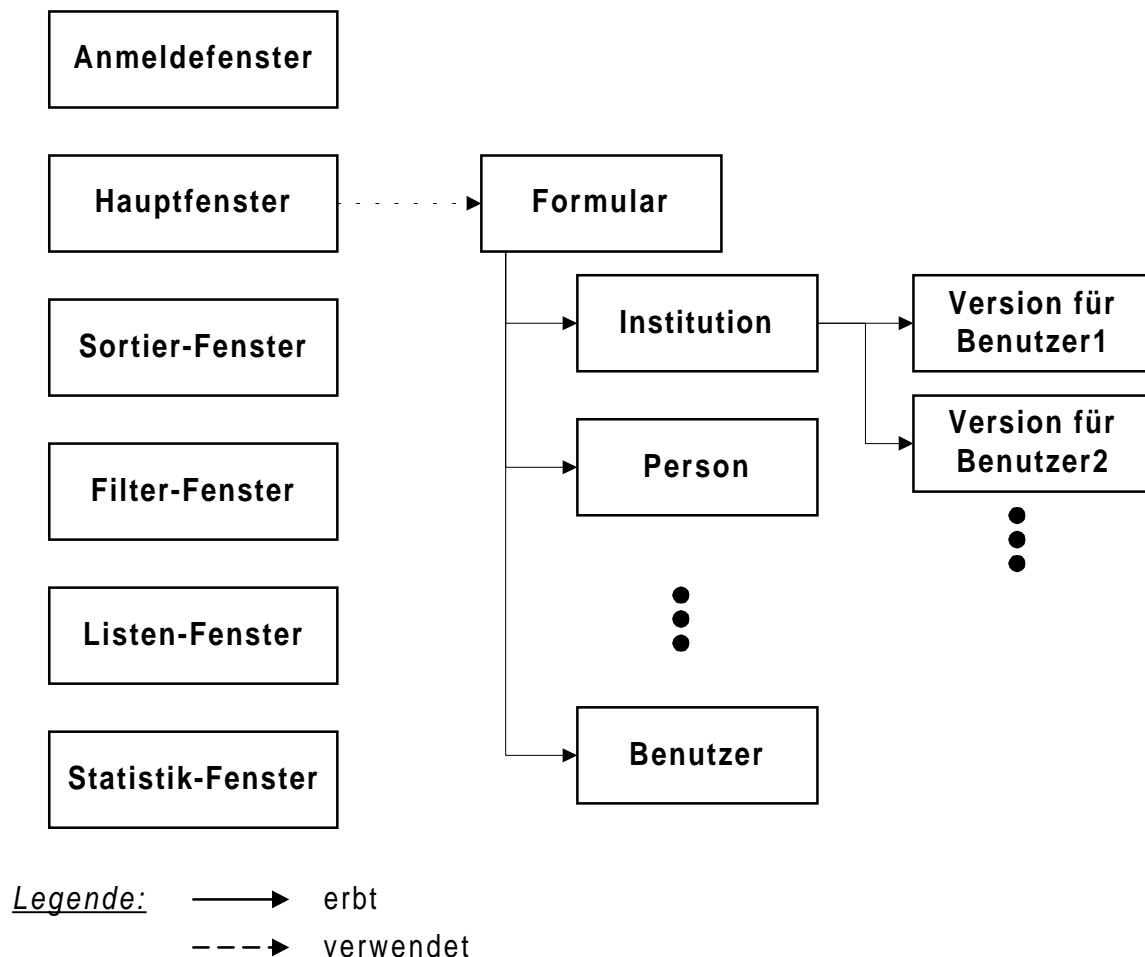


Abbildung 5.8: Klassendiagramm der Oberfläche

7 Einbindung in Webbrowser

Zum Abschluß der Implementationsbeschreibung soll noch die Einbindung des Applets in die Standard-Webbrowser erläutert werden. Die derzeit gängigsten Webbrowser sind der Netscape Navigator und der Microsoft Internet Explorer. Leider verwenden diese nicht den gleichen Umfang von HTML, der Beschreibungssprache für Webseiten. Hinzu kommt, daß sie zueinander inkompatible Erweiterungen an HTML vornehmen.

Da das Applet aber in Java 2 geschrieben ist und dies von den Browsern noch nicht unterstützt wird, ist es nötig ein Plug-In zu installieren und jeweils beim Start des Applets aufzurufen. Aus diesem Grund kann die normale Methode Applets in Webseiten einzubinden nicht verwendet werde.

Nachfolgend ist der hierfür notwendige Code zu sehen, der in beiden Browsern lauffähig ist. Zur näheren Erläuterung von HTML möchte ich auf die Literatur [TiJe97] verweisen, da eine eingehende Erklärung den Rahmen dieser Ausarbeitung sprengen würde.

```
<OBJECT classid="clsid:8AD9C840-044E-11D1-B3E9-00805F499D93"
  width="100%"
  height="100%"
  align="baseline"
  codebase="http://java.sun.com/products/plugin/1.2.2/
            jinstall-1_2_2-win.cab#Version=1,2,2,0">
<PARAM NAME="code" VALUE="ddb.client.DDBClient.class">
<PARAM NAME="archive" VALUE="DDBClient.jar">
<PARAM NAME="type" VALUE="application/
                        x-java-applet;version=1.2.2">
<COMMENT>
  <EMBED type="application/x-java-applet;version=1.2.2"
    width="100%"
    height="100%"
    align="baseline"
    code="ddb.client.DDBClient.class"
    archive="DDBClient.jar"
    pluginspage="http://java.sun.com/products/plugin/
                1.2/plugin-install.html">
  <NOEMBED>
</COMMENT>
  No JDK 1.2 support for APPLLET!!
</NOEMBED>
</EMBED>
</OBJECT>
```

Die <OBJECT>-Tags werden vom Internet Explorer verstanden, die <EMBED>-Tags hingegen vom Netscape Navigator. Wie man sehen kann, sind die Tags an der Grau unterlegten Stelle nicht korrekt geklammert. Dies würde normalerweise zu Problemen führen, wenn jeweils alle Tags bekannt wären. Da dies aber nicht der Fall ist und unbekannte Tags einfach ignoriert werden, ergibt dies in beiden Fällen einen korrekt erstellte HTML-Code.

Wie zuvor schon erwähnt wurde, ist es nötig dem Applet zusätzliche Rechte einzuräumen, um z.B. für den Datenexport auf lokale Speichermedien zugreifen zu können. Dies ist durch das Ablegen einer Policy-Datei (.java.policy) im Benutzerverzeichnis des jeweiligen Client-Systems möglich, mit der eine Änderung der Rechtevergabe für bestimmte Applets eingestellt werden kann. Das Benutzerverzeichnis hängt vom jeweiligen System ab, für Windows95/98

ist dies etwa das Windows-Verzeichnis. Auf der erstellten Webseite, von der die Policy-Datei heruntergeladen werden kann, ist eine Beschreibung für verschiedene Systeme vorhanden.

Wie man erkennt, widerspricht der letzte Punkt gegen die Anforderung auf eine explizite Installation von Software auf der Client-Seite zu verzichten. Dieses Problem wurde bei der Wahl dieser Lösung durchaus bedacht. Eine Diskussion, der Gründe für die Wahl dieser Lösung, erfolgt im Kapitel über die im Rahmen dieser Diplomarbeit gemachten Erfahrungen.

Kapitel 6:

Benutzungsanleitung

In den vorhergehenden Kapitel wurde das System aus der Sicht der Implementierung beschrieben, in diesem Kapitel soll nunmehr auf die Sicht des Benutzer eingegangen werden. Zu diesem Zweck wird zunächst die Installation des Systems und die darauf folgende Inbetriebnahme erläutert. Daraufaufgehend werden die wichtigsten Komponenten der Benutzerschnittstellen vorgestellt. Abschließend werden die Hauptfunktionen der Datenbank-Anwendung im Detail beschrieben.

1 Installation

Die Installation auf Server-Seite gestaltet sich verhältnismäßig einfach. Nach erfolgter Installation des Datenbank-Servers von Sybase und des JDK (bzw. auch nur der JRE). Muß eine ODBC-Verbindung eingerichtet werden. Dort wird die Datei *ddb.db* als Datenbank und die "standalone" Engine *dbeng6* (-x *NONE* als Parameter schaltet zusätzlich Protokolle ab) des Datenbank-Server als auszuführendes Programm angegeben, natürlich beides mit korrekter Pfadangabe versehen. Sollen die beigefügten Batch-Dateien zum Start der Server-Applikation verwendet werden, dann muß dort noch die Variable *JAVA_HOME* auf die den Pfad des JDK (bzw. des JRE) gesetzt werden. Ist dies erfolgt, kann der Server durch das Ausführen der Datei *StartDDBServer.bat* gestartet werden.

Für die Installation des Client-Applets auf dem Server ist es lediglich erforderlich, daß die Datei *DDBClient.jar* mittels des Webservers über das Netz zugänglich ist. Dabei ist zu beachten, daß der Host des Servers und des zu ladenden Applets identisch sein müssen.

2 Inbetriebnahme

Die Inbetriebnahme erfolgt durch Start des Servers, wie es im vorherigen Abschnitt beschrieben wurde. Bevor allerdings der Server über das Netz zugänglich wird, muß zunächst der Name der ODBC-Verbindung zur Datenbank und der Port für den Zugang zum Netz angegeben werden, außerdem erfolgt eine Abfrage des Benutzernamen und des Paßworts. In Abbildung 6.1 ist das dafür verwendete Fenster zu sehen. Vorgabewerte für die ODBC-Verbindung und den Port werden aus einer ini-Datei geladen, die immer die Werte enthält, die beim letzten Start verwendet wurden.

Der Benutzername ist auf *dba* und das Paßwort zunächst auf *SQL* eingestellt. Durch Drücken des Buttons "Passwort ändern" erscheint ein Dialog, mit dessen Hilfe ein neues Paßwort gesetzt werden kann. Dies sollte beim ersten Start erfolgen. Das neue Paßwort muß mindestens acht Zeichen lang sein und muß zwei Sonderzeichen enthalten. Dies gilt ebenfalls für die Vergabe von Paßwörtern für das Einloggen in die Datenbank-Anwendung auf Client-Seite. Beide Paßwörter haben aber keine gemeinsame Verwendung, da man sich mit dem Benutzernamen und dem Paßwort für den Start des Servers normalerweise nicht beim Client einloggen kann.



Abbildung 6.1: Start-Fenster

Nachdem die korrekte Kombination aus Benutzername und Paßwort angegeben wurde, erfolgt der Start des Datenbank-Servers und der Server wird über das Netz zugänglich gemacht. War beides erfolgreich, dann erscheint das in Abbildung 6.2 zu sehende Statusfenster. In diesem Statusfenster werden Informationen über die Aufnahme bzw. die Beendigung (oder auch der Abbruch) der Kommunikation mit den verschiedenen Clients ausgegeben. Außerdem ist es hier möglich den Server wieder vom Netz zu nehmen und zu beenden.

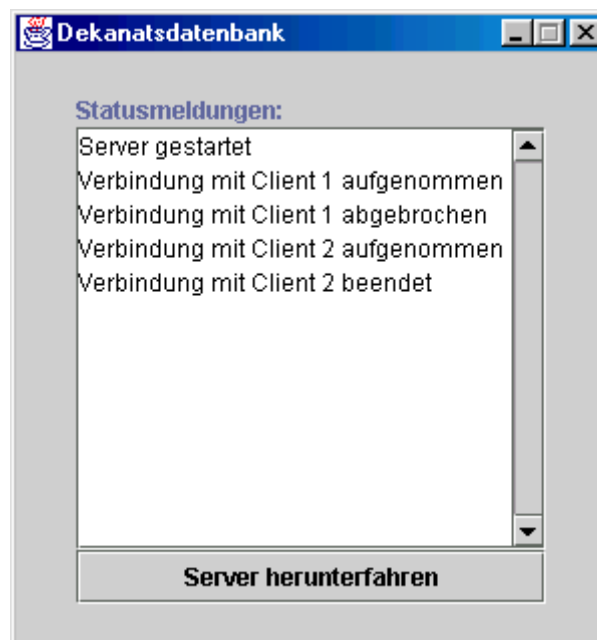


Abbildung 6.2: Statusfenster

3 Übersicht über die Bedienelemente

3.1 Fensterstruktur

In der nachfolgenden Abbildung 6.3 ist nochmals die grundlegende Fensterstruktur der Datenbank-Anwendung zu sehen.

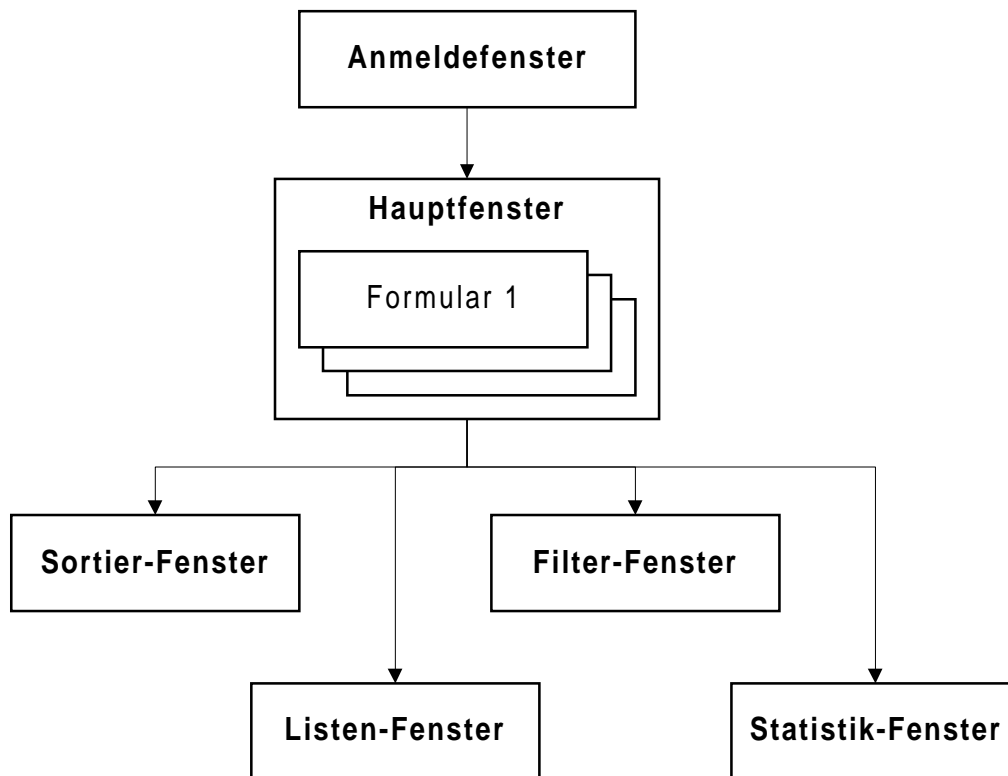


Abbildung 6.3: Fensterstruktur

Nach erfolgtem Login in die Datenbank-Anwendung gliedert sich die Benutzungsschnittstelle in ein Hauptfenster, das verschiedene Formulare beinhaltet. Jedes dieser Formulare kann eine Klasse von Datensätzen darstellen. Pro Formular wird jeweils ein Datensatz angezeigt. Der Benutzer kann zwischen verschiedenen Datensätzen hin und her blättern.

Zu Beginn sind alle für den Benutzer zugänglichen Datensätze direkt in einer vorgegebener Reihenfolge zugreifbar. Da dies sehr viele Datensätze sein können, hat der Benutzer die Möglichkeit die Datensätze zu sortieren und zu filtern. Diese Funktion ist vor allem für die Erstellung von Listen und Statistiken wichtig. Das Erstellen und Anzeigen von Listen und Statistiken erfolgt über ein separates Fenster, aus dem diese exportiert bzw. gedruckt werden können.

Neben den Fenstern zum Erstellen von Listen und Statistiken bzw. zum Sortieren und Filtern, gibt es noch weitere Hilfsfenster, die für den Dialog mit dem Benutzer zuständig sind, um besondere Operationen durchführen zu können.

3.2 Anmeldefenster

Das Anmeldefenster in Abbildung 6.4 erscheint beim Login in die Datenbank-Anwendung. Hier muß der Benutzername und das Paßwort angegeben werden. Beim ersten Einloggen in die Datenbank-Anwendung ist nur ein Benutzer mit dem Benutzernamen *dba* und dem Paßwort *SQL* vorhanden. Dieser Benutzer hat Administratorrechte. Er sollte gleich einen neuen Benutzer mit Administratorrechten einrichten und sich dann selbst löschen.



Abbildung 6.4: Anmeldefenster

Im Anmeldefenster kann zusätzlich noch der Host und der Port des Server eingestellt werden. Wie man in Abbildung 6.4 sehen kann, muß in einem Applet keine Angabe für den Host gemacht werden, da er nur eine Verbindung mit dem Host aufgebaut werden kann, von dem das Applet geladen wurde. Für den Port wird immer der Wert aus Abbildung 6.4 vorgegeben.

Beim ersten Start der Datenbank-Anwendung auf einem Rechner kann es vorkommen, daß nicht gleich das Anmeldefenster erscheint, sondern auf eine Webseite verzweigt wird, von der ein Plug-In geladen werden kann. Dies passiert in solchen Fällen, in denen der Webbrowser nicht Java 2 fähig ist. Wird das Plug-In nicht installiert, dann kann die Datenbank-Anwendung nicht gestartet werden. Dies ist übrigens auch der Fall, wenn die *.java.policy* Datei nicht in das Benutzerverzeichnis installiert wurde, wie es bereits beschrieben wurde.

3.3 Hauptfenster

Über das Hauptfenster werden die verschiedenen Datensätze verwaltet und weitergehende Funktionen aufgerufen. Jedem Benutzer wird eine für seine Bedürfnisse angepaßte Darstellung des Hauptfensters geboten, indem jeweils individuelle Formulare geladen werden.

In Abbildung 6.5 ist das Hauptfenster zu sehen, wenn es nicht in einem Webbrowser als Applet, sondern über die JRE als Applikation ausgeführt wird. Dies ist möglich, wenn die JRE auf dem Client-Rechner installiert ist und die Jar-Datei über das Internet heruntergeladen wurde, dann genügt für den Start ein Doppelklick auf die Jar-Datei. Bei dieser Lösung muß außerdem keine Policy-Datei installiert werden.

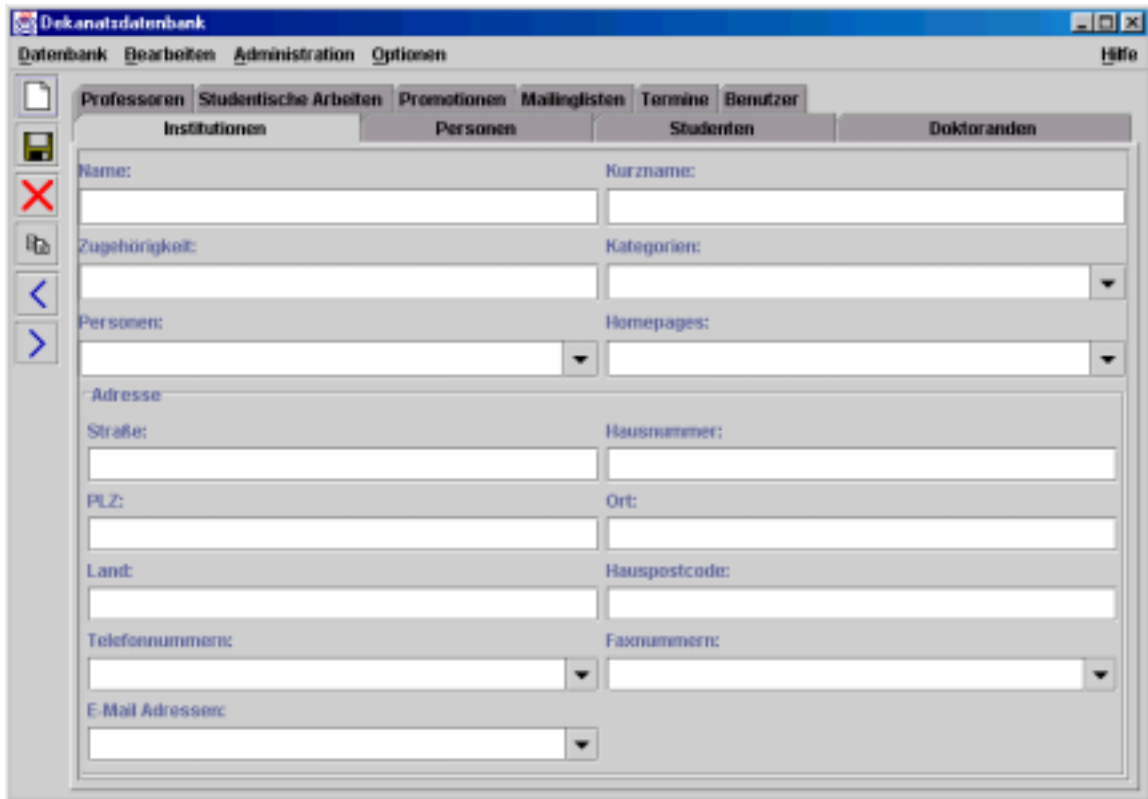


Abbildung 6.5: Hauptfenster (1)

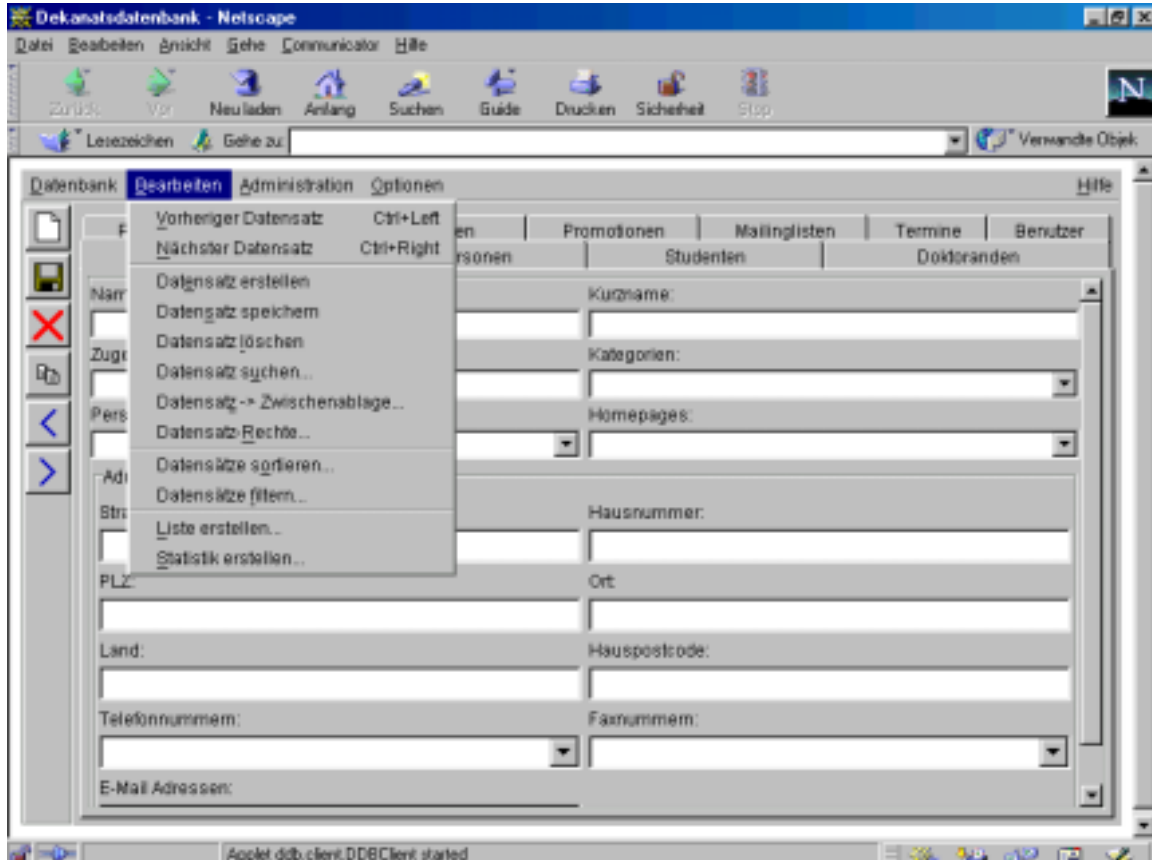


Abbildung 6.6: Hauptfenster (2)

Wie Abbildung 6.5 zeigt, besteht das Hauptfenster aus einer Menüleiste, eine Toolbar und einem Register mit den verschiedenen Formularen. Über die Menüleiste werden alle global verfügbaren Funktionen aufgerufen. Einige häufig verwendete Funktionen werden zusätzlich noch mal in der Toolbar angeboten. Das Hauptfenster in Abbildung 6.5 wird mit dem in Java üblichen "Lock and Feel" dargestellt.

In Abbildung 6.6 wird das Hauptfenster mit dem in Windows verwendeten "Look and Feel" angezeigt. Über das Optionen-Menü kann das zu verwendende "Look and Feel" ausgewählt werden. Außerdem kann dort die Anzeige von Tooltips für die Toolbar an- und abgeschaltet werden. Im Bearbeiten-Menü, das in Abbildung 6.6 sichtbar ist, können die wesentlichen Funktionen der Datenbank-Anwendung aufgerufen werden.

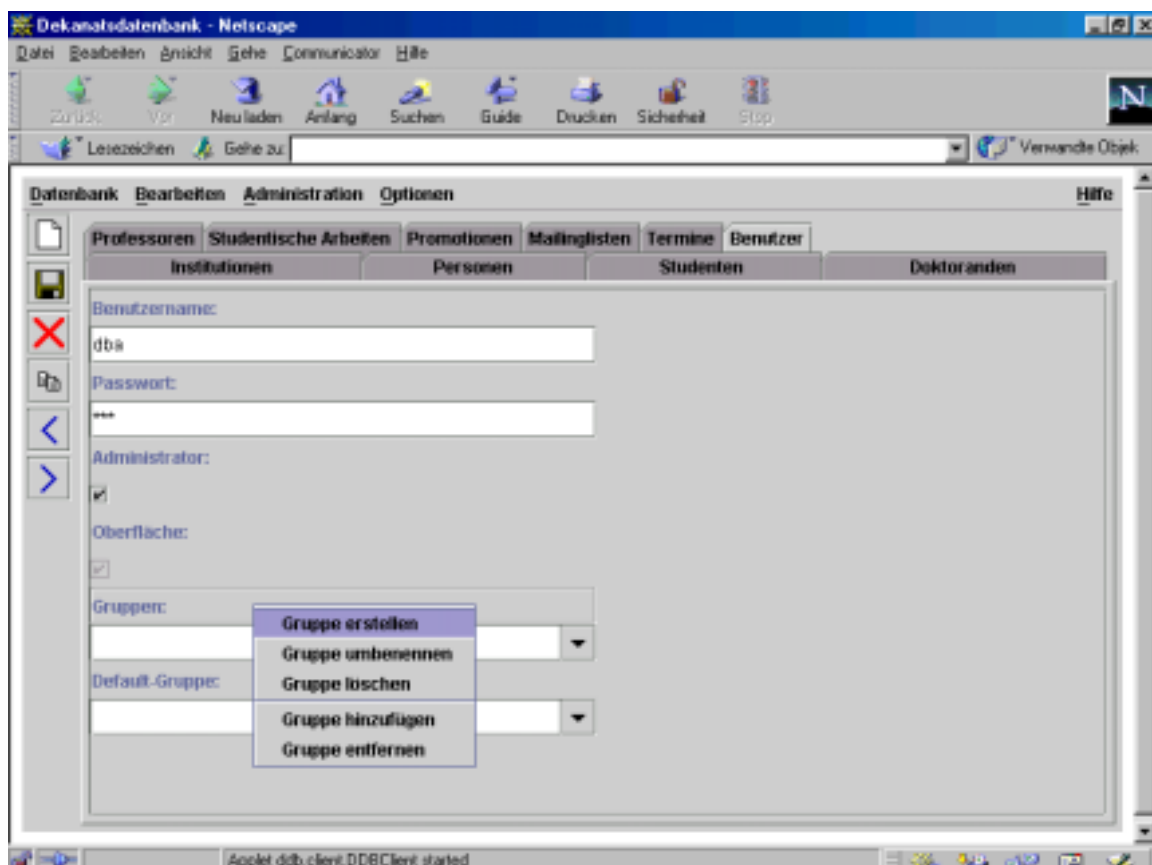


Abbildung 6.7: Hauptfenster (3)

Neben den über die Menüleiste zugänglichen globalen verfügbaren Funktionen, können durch anklicken des Labels zu einem bestimmten Feld eines Datensatzes spezielle Funktionen aufgerufen werden. In Abbildung 6.7 ist dies anhand des Gruppen-Feldes zu sehen, wo ein Popup-Menü mit Funktionen zur Verwaltung von Gruppe erschienen ist.

3.4 Hilfsfenster

Für die Durchführung von komplexeren Interaktionen mit dem Benutzer, werden zusätzlich zum Hauptfenster diverse Hilfsfenster verwendet. In diesem Abschnitt sollen deshalb Beispiele für Hilfsfenster vorgestellt werden. In Abbildung 6.8 ist zunächst das Paßwort-Fenster zu sehen, das über das Datenbank-Menü aufgerufen werden kann. Über das Paßwort-Fenster kann der jeweilige Benutzer sein Paßwort ändern. Es gelten dabei die gleichen Regeln, wie sie zuvor beim Server beschrieben worden sind.

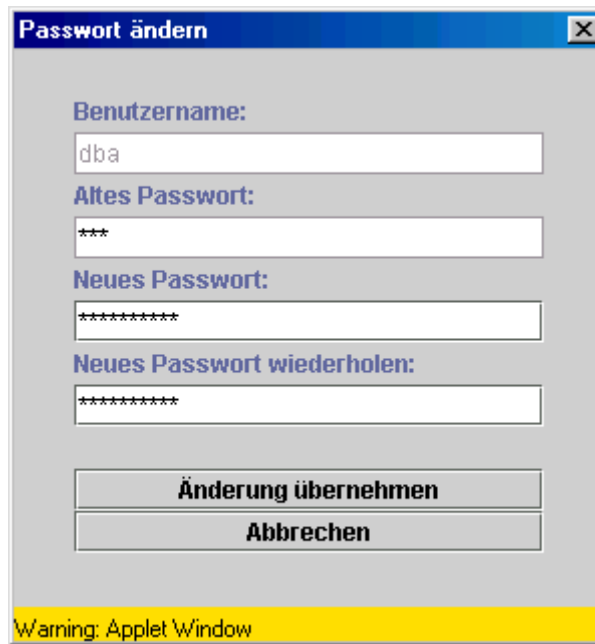


Abbildung 6.8: Paßwort-Fenster

Als zweites Beispiel sei der SQL-Dialog in Abbildung 6.9 vorgestellt. Über den SQL-Dialog können direkt SQL-Befehle auf der Datenbank ausgeführt werden. Da dies eine sehr mächtige Funktion ist, steht sie nur Benutzern mit Administratorrechten zur Verfügung.

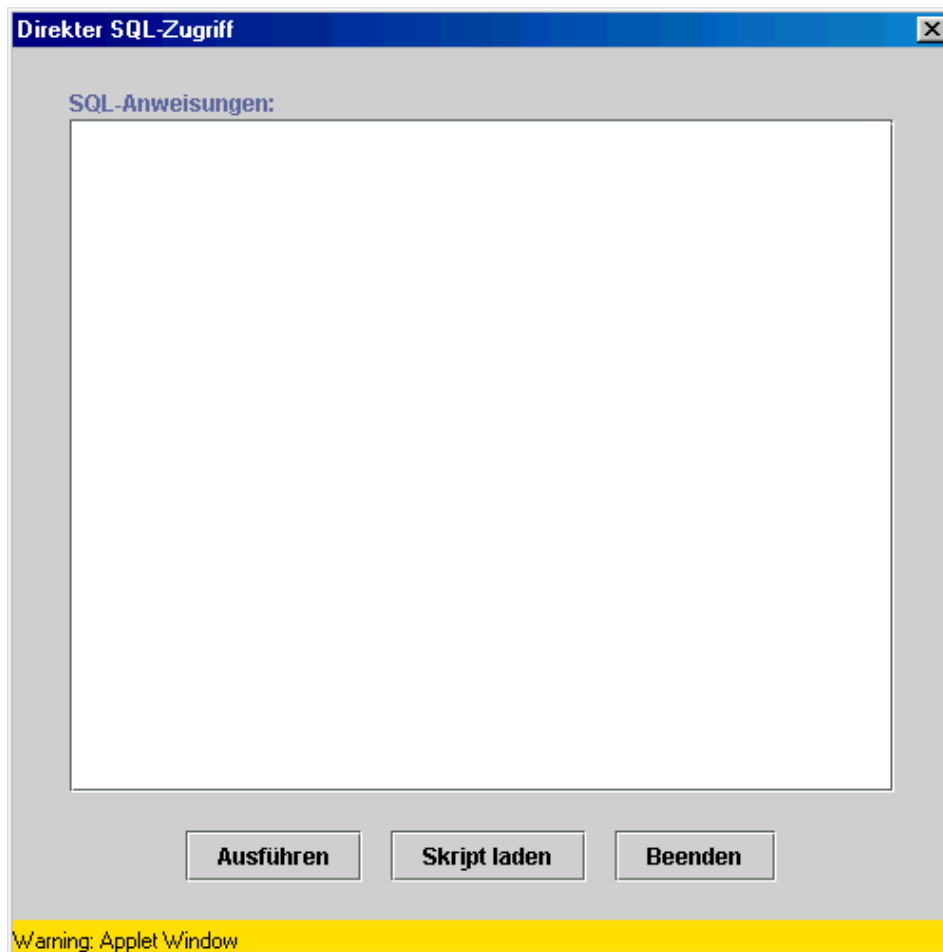


Abbildung 6.9: SQL-Dialog

Alle Funktionen die lediglich für Administratoren bestimmt sind, werden im Administration-Menü versammelt. Dieses ist nur bei Benutzern mit Administratorrechten sichtbar, damit die Datenbank vor unberechtigten Zugriffen geschützt ist und Fehlbedienungen vermieden werden.

Ein wichtige Operation die über den SQL-Dialog ausgeführt werden kann, ist der Im- und Export von Daten. Zunächst muß die gewünschte Datei in das temporäre Verzeichnis des Server transferiert werden. Dies ist über eine weitere Funktion des Administration-Menü möglich. Dann kann mit Hilfe des SQL-Dialogs ein entsprechender SQL-Befehl an den Datenbank-Server geschickt werden.

3.5 *Sortier-Fenster*

Neben dem Hauptfenster mit seinen Hilfsfenstern gibt es noch Fenster, die komplexe Dialoge und Anzeigen ausführen. Von diesen Fenstern handeln dieser und die folgenden Abschnitte. Im Rahmen dieser Abschnitte möchte ich hauptsächlich die eingesetzten Dialoge vorstellen, da sich die danach erstellten Anzeigen nicht prinzipiell von denen aus anderen Programmen bekannten unterscheiden.

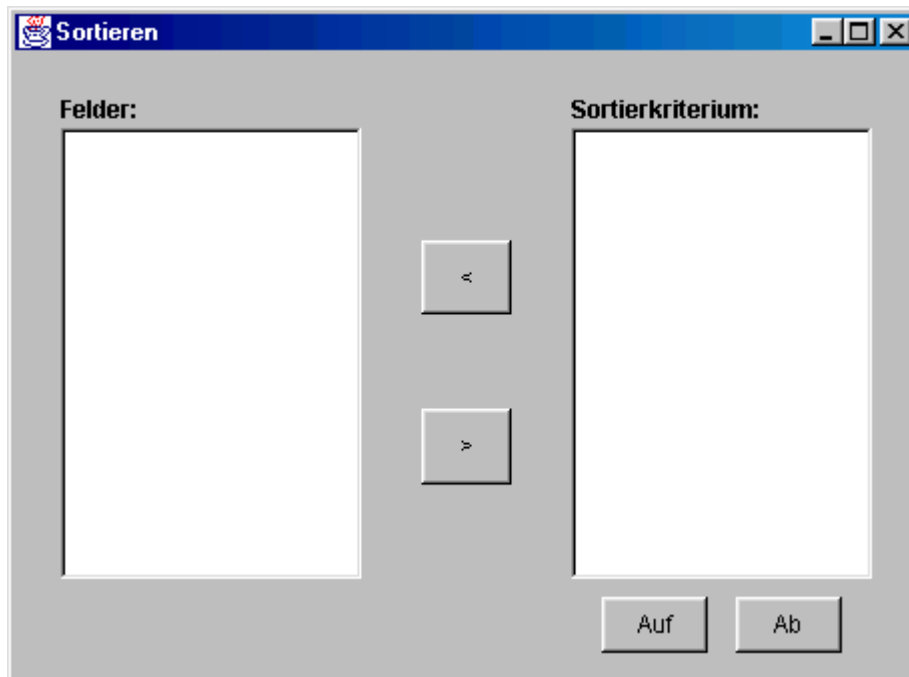


Abbildung 6.10: Sortier-Fenster

In Abbildung 6.10 ist der Dialog zur Sortierung von Datensätzen zu sehen. Auf der linken Seite werden die verfügbaren Felder der Datensätze aufgelistet. Aus diesen können mit den Pfeiltasten in der Mitte des Fensters die Felder ausgewählt werden, nach denen sortiert werden soll. Diese Felder werden dann auf der rechten Seite aufgelistet.

Mit den Buttons unter den Sortierkriterien können diese nach ihrer Gewichtung sortiert werden. Das Sortierkriterium an der obersten Stelle hat das höchste Gewicht, was bedeutet, daß nach ihm als letztes sortiert wird.

3.6 Filter-Fenster

Im Filter-Fenster kann festgelegt werden, welcher Teil der Datensätze momentan zugänglich sein soll. Wie in Abbildung 6.11 zu sehen ist, können verschiedene Filterkriterien festgelegt werden. Diese werden untereinander logisch mit „UND“ verknüpft. Ist eine kompliziertere Verknüpfung gewünscht, kann alternativ zwischen den Filterkriterien auch ein beliebiger Boolescher Ausdruck definiert werden.



Abbildung 6.11: Filter-Fenster

Ein Filterkriterium besteht aus dem Feld des Datensatzes auf den es angewandt werden soll und aus dem geforderten Inhalt. Das Feld kann aus einer Liste der zur Verfügung stehenden Felder ausgewählt werden. Unter Inhalt können eine oder mehrere Bedingungen angegeben werden, die das ausgewählte Feld des jeweiligen Datensatzes erfüllen muß.

3.7 Listen-Fenster

Zur Definition einer zu erstellenden Liste dient das Listen-Fenster in Abbildung 6.12. Der Aufbau der zu erzeugenden Liste kann auf analoge Weise wie beim Sortier-Fenster bestimmt werden. Im Unterschied zum Sortier-Fenster, legt hier die Reihenfolge der ausgewählten Felder die Reihenfolge der Spalten in der Liste fest.

Durch das Anklicken eines ausgewählten Feldes kann dessen Schriftart in der Liste eingestellt werden. Hier kann unter anderem die Schriftgröße festgelegt werden und ob Schrift Fett bzw. Kursiv sein soll.

Die Definition einer Liste kann abgespeichert werden, wobei eine Eintragung in eine ini-Datei in der Jar-Datei des jeweiligen Benutzers abgelegt wird. Anstatt selbst eine Liste zu erstellen, kann auch gleich eine der auf die beschriebene Art vordefinierten Listen ausgewählt werden. Die in den Anforderungen aufgeführten Listen sind für die entsprechenden Benutzer vom Administrator zu Beginn einzurichten.

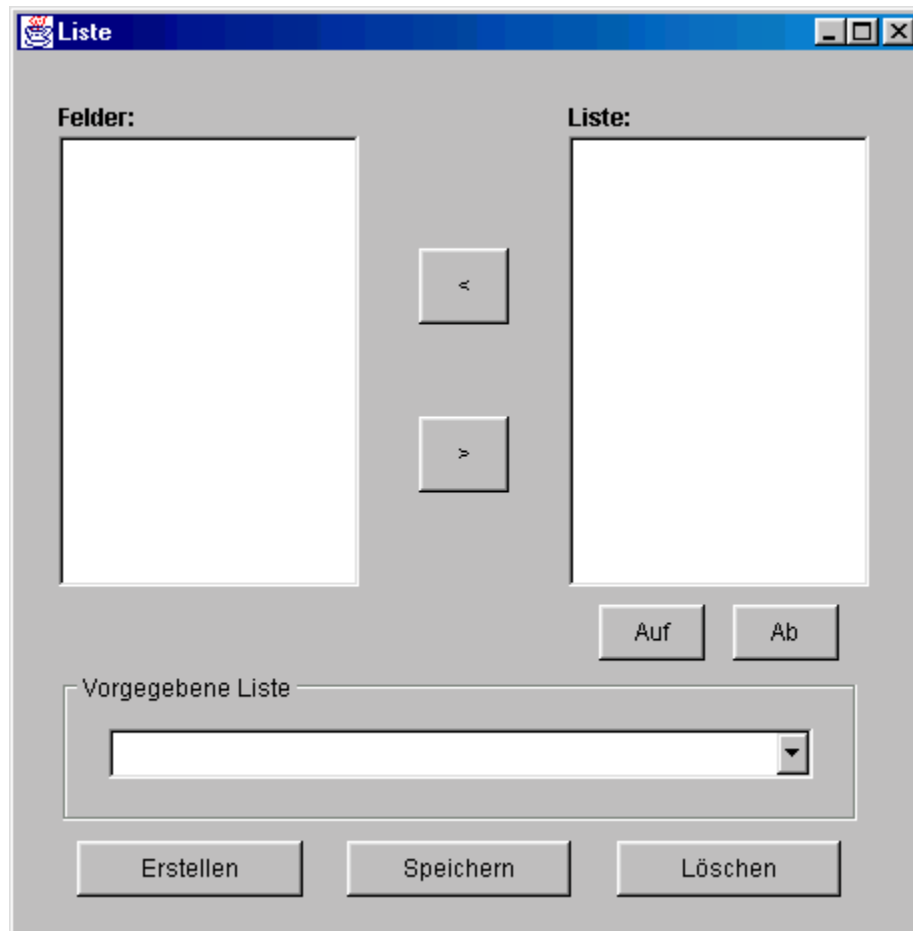


Abbildung 6.12: Listen-Fenster

3.8 Statistik-Fenster

Abschließend wird in diesem Abschnitt das Statistik-Fenster aus Abbildung 6.13 beschrieben. Analog zum Listen-Fenster kann hier eine zu erstellende Statistik definiert werden, wobei auch hier Statistiken gespeichert und wieder aufgerufen werden können. Die Speicherung erfolgt auch hier über eine ini-Datei in der Jar-Datei des jeweiligen Benutzers.

Die Vorgabe von Statistiken durch den Administrator ist wesentlich wichtiger, als die von Listen, da die Definition der Statistiken nicht so intuitiv anzuwenden ist. Um eine Statistik zu definieren muß, die Gesamtheit und der Anteil bestimmt werden. Aus diesen beiden Werten, wird die Statistik ermittelt. Zur Definition von Gesamtheit und Anteil wird ein SQL-Befehl verwendet, der eine Zahl als Ergebnis haben muß.

Um als Statistik nicht nur einen Wert zu erhalten, können in Gesamtheit und Anteil ein oder auch mehrere Parameter angegeben werden. Für die dann der Reihe nach die möglichen bzw. vorhandenen Werte eingesetzt werden. In diesem Zusammenhang macht die Auswahl der Darstellung der Statistik einen Sinn.

Zur Darstellung von Statistiken sind mehrere Varianten für verschiedene Anzahl an Parametern fest in der Datenbank-Anwendung verankert. Beispielsweise ist bietet sich eine Auflistung von Werten oder ein Balkendiagramm für die Darstellungen bei Statistiken mit einem Parameter an.

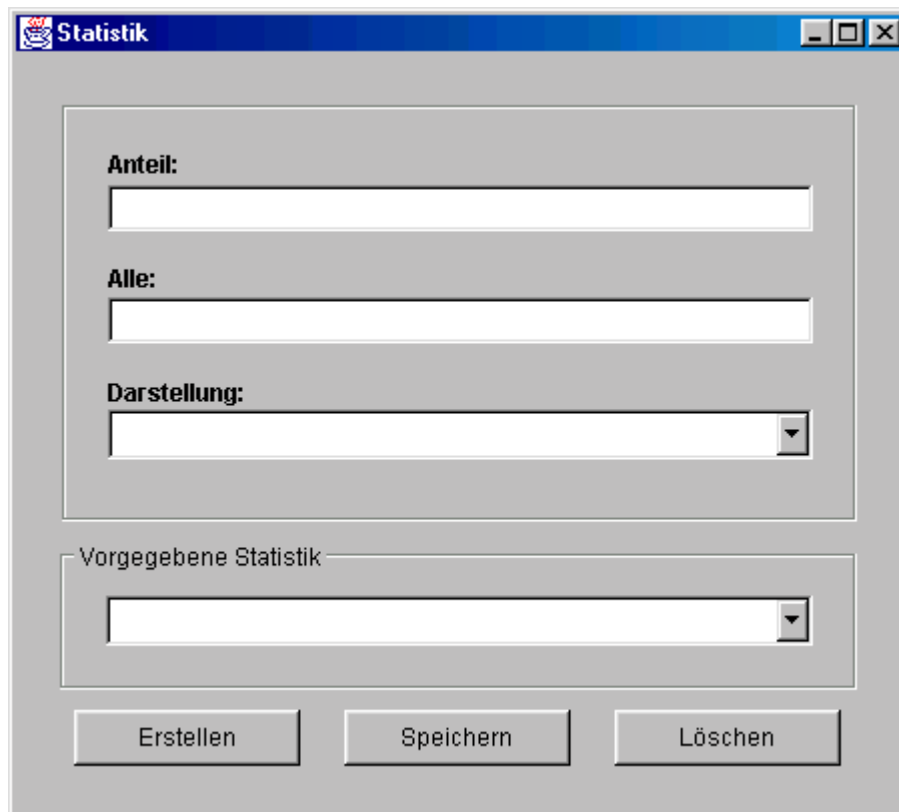


Abbildung 6.13: Statistik-Fenster

4 Funktionen

4.1 Übersicht

In den nachfolgenden Abschnitten wird die Bedienung der wichtigsten Funktionen der Datenbank-Anwendung beschrieben. Zu diesem Zweck werden, wie im Kapitel über die Anforderungen bereits erwähnt wurde, vor allem Use Cases verwendet. Nachfolgend sind nochmals die erstellten Use Cases aufgelistet.

Benutzerverwaltung:

1. Benutzer anlegen
2. Benutzer aktualisieren
3. Benutzer löschen
4. Benutzer anmelden
5. Benutzer erinnern

Datensatzverwaltung:

6. Datensatz anlegen
7. Datensatz suchen
8. Datensatz aktualisieren
9. Datensatz-Zugriffsrechte ändern
10. Datensatz-Hompage aufrufen
11. Datensatz löschen
12. Datensatz exportieren
13. Datensätze sortieren
14. Datensätze filtern

Listen:

- 15. Liste erzeugen
- 16. Liste exportieren
- 17. Liste drucken

Statistiken:

- 18. Statistik erzeugen
- 19. Statistik exportieren
- 20. Statistik drucken

Use Cases haben sich als übersichtliche und kompakte Art der Darstellung bewährt, weshalb sich die folgenden Abschnitte auch zum schnellen Nachschlagen eignen sollten. In Verbindung mit der Beschreibung der Bedienelemente in den vorherigen Abschnitten, dürfte in den Use Cases alle wesentliche Information enthalten sein, um die Datenbank-Anwendung bedienen zu können.

4.2 Benutzerverwaltung

Eine wesentliche Komponente der erstellten Datenbank-Anwendung ist die Benutzerverwaltung. Neben den Gründen des Datenschutzes rechtfertigt vor allem die Anpassung der Oberfläche an die Bedürfnisse der unterschiedlichen Benutzer die aufwendige Benutzerverwaltung.

Wie man an den folgenden Use Cases erkennen kann, wurde die Benutzerverwaltung harmonisch in die normale Handhabung der Datenbank-Anwendung eingepaßt, in dem Benutzer ebenfalls als Datensätze verwaltet werden. Ziel war es hierbei den Einarbeitungsaufwand für die Benutzer so gering wie möglich zu halten.

Use Case 1:	Benutzer anlegen
Ziel:	Ein neuer Benutzer soll angelegt und mit Zugriffsrechten ausgestattet werden. Außerdem wird seine Benutzungsoberfläche eingerichtet.
Externe Aktoren:	Administrator.
Vorbedingungen:	Der Benutzer muß als Administrator angemeldet sein (Administratorrechte).
Beschreibung:	Der Benutzer befindet sich im Formular „Benutzer“ des Hauptfensters, in dem er durch hinzufügen eines neuen Datensatzes einen neuen Benutzer anlegt. Es muß der Benutzername, das Paßwort und die Zugehörigkeit zu einer Default-Gruppe angegeben werden. Die Benutzungsoberfläche wird mit Hilfe eines Dateidialogs geladen, wobei die entsprechende Jar-Datei ausgewählt werden muß.
Alternative Abläufe:	Keine.

Anhand des vorstehenden Use Case 1 soll hier kurz der Aufbau eines Use Cases erläutert werden. Ein Use Case besteht immer aus einem Titel und darauffolgenden Beschreibungen des verfolgten Ziels, der externen Aktoren, der Vorbedingungen, des prinzipiellen Ablaufs und alternativer Abläufe.

Unter externen Aktoren werden hierbei oft mit dem System interagierende Personen verstanden, beispielsweise sind aber auch andere Systeme hierfür denkbar. Die beschriebenen Abläufe gliedern sich in den normalen (prinzipiellen) Ablauf und einer Menge alternativen Abläufen. Die Einteilung unterliegt der Sichtweise des Use Cases.

Use Case 2:	Benutzer aktualisieren
Ziel:	Der Benutzername, das Paßwort oder die Benutzungsoberfläche eines Benutzers sollen geändert werden.
Externe Aktoren:	Administrator.
Vorbedingungen:	Der Benutzer muß als Administrator angemeldet sein (Administratorrechte).
Beschreibung:	Der Benutzer befindet sich im Formular „Benutzer“ des Hauptfensters, in dem er durch verändern des Datensatzes des jeweiligen Benutzers die Anpassungen vornimmt. Es kann der Benutzername, das Paßwort und die Zugehörigkeit zur Benutzergruppe verändert werden. Die Benutzungsoberfläche kann mit Hilfe eines Dateidialogs geladen, wobei die entsprechende Jar-Datei ausgewählt werden muß.
Alternative Abläufe:	Der Benutzer kann sein Paßwort selber verändern. Dazu muß er im im Datenbank-Menü des Hauptfenster den Menüeintrag mit der Beschriftung „Passwort ändern“ auswählen und in dem darauf geöffneten Fenster sein altes und sein neues Paßwort eingeben.

Use Case 3:	Benutzer löschen
Ziel:	Ein Benutzer soll aus der Datenbank entfernt werden.
Externe Aktoren:	Administrator.
Vorbedingungen:	Der Benutzer muß als Administrator angemeldet sein (Administratorrechte).
Beschreibung:	Der Benutzer wechselt in das Formular „Benutzer“ des Hauptfensters und sucht den Datensatz des jeweiligen Benutzers. Dieser Datensatz kann dann gelöscht werden. Alle dem Benutzer zugeordneten Datensätze werden dem Administrator zugeordnet. Sollen die Datensätze einem anderen Benutzer zugeordnet werden, sollte dies zuvor über die Eingabe von SQL-Befehlen im „SQL-Dialog“ geschehen, das nur dem Administrator zu Verfügung steht.
Alternative Abläufe:	Keine.

Use Case 4:	Benutzer anmelden
Ziel:	Einem autorisierten Benutzer den Zugriff auf die Datenbank ermöglichen.
Externe Aktoren:	Benutzer (beliebig).
Vorbedingungen:	Die Webseite, die das Applet der Datenbank-Anwendung enthält, ist in einem Webrowsers der Java unterstützt geladen worden.
Beschreibung:	Vom Anmeldefenster aus wird nach Eingabe des Benutzernamens und des Paßwortes durch Anklicken des Login-Buttons der Login-Vorgang ausgelöst. Bei Eingabe des Benutzerkennworts werden anstatt Buchstaben nur Sternchen ausgegeben.
Alternative Abläufe:	Keine.

Wie Anhand von Use Case 5 deutlich wird, bringt die Unterscheidung von verschiedenen Benutzern, neben der Einhaltung des Datenschutzes und der angepaßten Oberflächengestaltung, die Möglichkeit weiterer benutzerspezifischer Funktionalität mit sich.

Use Case 5:	Benutzer erinnern
Ziel:	Den Benutzer an Termine erinnern.
Externe Aktoren:	Benutzer (beliebig).
Vorbedingungen:	Der Login-Vorgang wurde soeben beendet.
Beschreibung:	Nach Beendigung des Login-Vorgangs erscheint ein Dialogfenster, in dem alle bevorstehenden Termine aufgelistet sind, deren Benachrichtigungszeitpunkt in der Vergangenheit lag. Das Dialogfenster ist modal, somit kann der Benutzer erst seine Arbeit nach Schließen des Fensters fortsetzen.
Alternative Abläufe:	Keine.

4.3 Datensatzverwaltung

Die Datensatzverwaltung ist die Hauptaufgabe und das eigentliche Ziel der Datenbank-Anwendung. Neben einfachen Operationen, wie das Erstellen und Löschen von Datensätzen, werden auch komplexere Funktionen angeboten, wie etwa das Filtern und das Sortieren von Datensätzen.

Die beiden genannten Funktionen, haben eine wesentliche Bedeutung im Zusammenhang mit dem Erstellen von Listen und Statistiken, da sie auf den gefilterten und sortierten Datensätzen operieren. Man kann daraus erkennen, daß diese Funktionen zusammen eine Einheit bilden, was auch deren Heraushebung durch Abbildung 6.3 rechtfertigt.

Use Case 6:	Datensatz anlegen
Ziel:	Ein Datensatz einer beliebigen Kategorie soll neu angelegt werden.
Externe Aktoren:	Benutzer (beliebig).
Vorbedingungen:	Der Benutzer befindet sich im zum anzulegenden Datensatz passenden Formular des Hauptfensters.
Beschreibung:	Der Benutzer füllt den momentan geladenen Datensatz mit neuen Werten aus. Durch Auswahl des Menüpunktes „Datensatz erstellen“ im Bearbeiten-Menü des Hauptfensters wird daraus ein neuer Datensatz erstellt. Der ursprüngliche Datensatz bleibt dabei unverändert.
Alternative Abläufe:	Keine.

Use Case 7:	Datensatz suchen
Ziel:	Ein Datensatz soll gesucht werden.
Externe Aktoren:	Benutzer (beliebig).
Vorbedingungen:	Der Benutzer befindet sich im passenden Formular des Hauptfensters.
Beschreibung:	Durch Auswahl des Menüpunktes „Datensatz suchen“ im Bearbeiten-Menü des Hauptfensters, wird ein Fenster geöffnet in dem der zu suchende Datensatz spezifiziert werden kann. Werden daraufhin mehrere passende Datensätze gefunden, kann der Benutzer aus einer Auflistung dieser Datensätze auswählen, die in einem separaten Fenster angezeigt wird. Wird kein passender Datensatz gefunden, dann erscheint ein leerer Datensatz.
Alternative Abläufe:	Über das Filtern kann auch eine Datensatzsuche durchgeführt werden.

Use Case 8:	Datensatz aktualisieren
Ziel:	Die Angaben eines Datensatzes sollen verändert werden.
Externe Aktoren:	Benutzer (beliebig).
Vorbedingungen:	Der Benutzer befindet sich im passenden Formular des Hauptfensters.
Beschreibung:	Nachdem der zu ändernde Datensatz im Formular aufgerufen wurde, können die verschiedenen Felder des Datensatzes geändert werden. Nach betätigen des Menüpunktes „Datensatz speichern“ im Bearbeiten-Menü des Hauptfensters werden die geänderten Daten in die Datenbank übernommen.
Alternative Abläufe:	Keine.

Use Case 9:	Datensatz-Zugriffsrechte ändern
Ziel:	Der Datensatz soll für andere Benutzer zugänglich gemacht werden.
Externe Aktoren:	Benutzer (beliebig).
Vorbedingungen:	Der Benutzer befindet sich im passenden Formular des Hauptfensters.
Beschreibung:	Durch Aufrufen der Datensatz-Rechte des momentanen Datensatzes über das Bearbeiten-Menü wird unter anderem dessen Benutzer und Gruppenzuordnung sichtbar. Diese Angaben können teilweise geändert werden, womit der Datensatz anderen Benutzern zugänglich gemacht werden kann.
Alternative Abläufe:	Der Administrator kann alle Datensatz-Rechte jedes beliebigen Datensatzes ändern.

Use Case 10:	Datensatz-Homepage aufrufen
Ziel:	Zu einer URL in einem Datenfeld soll die entsprechende Webseite aufgerufen werden.
Externe Aktoren:	Benutzer (beliebig).
Vorbedingungen:	Im Formular das der Benutzer ausgewählt hat, befindet sich ein Datenfeld, das eine URL enthält.
Beschreibung:	Durch ausführen eines Doppelklicks auf das Datenfeld mit der URL, wird ein (neues) Browserfenster geöffnet, das die zur URL gehörende Homepage enthält.
Alternative Abläufe:	Keine.

Use Case 11:	Datensatz löschen
Ziel:	Ein Datensatz soll aus der Datenbank entfernt werden.
Externe Aktoren:	Benutzer (beliebig).
Vorbedingungen:	Der Benutzer befindet sich im passenden Formular des Hauptfensters.
Beschreibung:	Nachdem der zu löschende Datensatz im Formular aufgerufen wurde, kann dieser durch Auswahl des Menüpunktes „Datensatz löschen“ im Bearbeiten-Menü des Hauptfensters aus der Datenbank gelöscht werden.
Alternative Abläufe:	Der Administrator kann jeden beliebigen Datensatzes löschen.

Use Case 12:	Datensatz exportieren
Ziel:	Ein Datensatz der Datenbank soll in eine andere Anwendung exportiert werden.
Externe Aktoren:	Benutzer (beliebig).
Vorbedingungen:	Der Benutzer befindet sich im passenden Formular des Hauptfensters.
Beschreibung:	Nachdem der zu exportierende Datensatz im Formular aufgerufen wurde, kann dieser durch betätigen des entsprechenden Menüpunktes im Bearbeiten-Menü des Hauptfensters in die Zwischenablage exportiert werden, von wo er in andere Anwendungen übernommen werden kann. Beim Export kann eingestellt werden, welche Felder des Datensatzes exportiert werden sollen. Dies erfolgt in einem separaten Fenster, das analog zum Sortier-Fenster bedient werden kann. An dieser Stelle kann der Datensatz auch auf einem Drucker ausgegeben werden.
Alternative Abläufe:	Keine.

Use Case 13:	Datensätze sortieren
Ziel:	Die Datensätze einer bestimmten Kategorie sollen sortiert werden.
Externe Aktoren:	Benutzer (beliebig).
Vorbedingungen:	Der Benutzer befindet sich im passenden Formular des Hauptfensters.
Beschreibung:	Durch Auswahl des Menüpunktes „Datensätze sortieren“ im Bearbeiten-Menü des Hauptfensters gelangt der Benutzer in das Sortier-Fenster. Dort kann er die Attribute des Datensatzes auswählen, nach denen sortiert werden soll. Durch die festgelegte Reihenfolge der gewählten Attribute wird die Reihenfolge festgelegt, in der die Sortierdurchgänge ausgeführt werden.
Alternative Abläufe:	Keine.

Use Case 14:	Datensätze filtern
Ziel:	Die Datensätze einer bestimmten Kategorie sollen gefiltert werden.
Externe Aktoren:	Benutzer (beliebig).
Vorbedingungen:	Der Benutzer befindet sich im passenden Formular des Hauptfensters.
Beschreibung:	Durch Auswahl des Menüpunktes „Datensätze filtern“ im Bearbeiten-Menü des Hauptfensters gelangt der Benutzer in das Filter-Fenster. Dort können verschiedene Filterkriterien festgelegt werden, diese werden logisch mit „UND“ verknüpft. Ist eine kompliziertere Verknüpfung gewünscht, kann zwischen den Filterkriterien ein Boolescher Ausdruck definiert werden. Nach erfolgtem Filtern, gelangt der Benutzer wieder in das Hauptfenster. Dort sind ihm aber nur noch die Datensätze zugänglich, die den Filterkriterien genügen. Möchte er wieder auf alle Datensätze zugreifen können, muß er die Filterkriterien zurücksetzen.
Alternative Abläufe:	Keine.

4.4 Listen

Wie bereits erwähnt wurde, stellt die Erstellung von Listen eine nicht unwesentliche Aufgabe für die Mitarbeiter des Dekanats der Fakultät Elektrotechnik und Informationstechnik dar. Aus diesem Grund ist eine umfangreiche Unterstützung dieser Tätigkeit durch die Datenbank-Anwendung nötig.

Wichtig ist des weiteren, daß ein leichter Datenaustausch mit anderen Programmen wie etwa Microsoft Word möglich ist. Da sich in letzter Zeit immer mehr Programme mit einer Unterstützung von HTML hervortun, wurde dieses als bevorzugtes Austauschformat gewählt.

Um aber auch mit Programmen, die nicht HTML unterstützen, Daten austauschen zu können, wird noch das Format einer ganz normalen Textdatei angeboten. Weitere nötige Datenformate mögen sich mit der Zeit ergeben und sollten, von der Implementierung der Datenbank-Anwendung her, leicht ergänzbar sein.

Use Case 15:	Liste erzeugen
Ziel:	Es soll eine Liste erstellt werden.
Externe Aktoren:	Benutzer (beliebig).
Vorbedingungen:	Der Benutzer befindet sich im Hauptfenster.
Beschreibung:	Durch Auswahl des Menüpunktes „Liste erstellen“ im Bearbeiten-Menü des Hauptfensters gelangt der Benutzer in das Listen-Fenster. Dort kann der Benutzer eine vordefinierte Liste auswählen und durch Drücken des „Erstellen“ Buttons erstellen. Der Aufbau der zu erzeugenden Liste kann alternativ auch durch Auswahl der in der Liste aufzunehmenden Felder bestimmt werden. Die erstellte Liste wird mit allen Formatierungen in einem neuen Fenster dargestellt.
Alternative Abläufe:	Keine.

Use Case 16:	Liste exportieren
Ziel:	Eine Liste soll in andere Anwendungen exportiert werden.
Externe Aktoren:	Benutzer (beliebig).
Vorbedingungen:	Der Benutzer befindet sich im Fenster, das die erzeugte Liste enthält.
Beschreibung:	Durch Auswahl des entsprechenden Menüpunkts im Fenster, das die erstellte Liste enthält, kann die Liste exportiert werden. Es gibt zwei Möglichkeiten des Exports. Dies sind der Export in eine Textdatei und in eine HTML-Datei.
Alternative Abläufe:	Keine.

Use Case 17:	Liste drucken
Ziel:	Eine Liste soll gedruckt werden.
Externe Aktoren:	Benutzer (beliebig).
Vorbedingungen:	Der Benutzer befindet sich im Fenster, das die erzeugte Liste enthält.
Beschreibung:	Durch Auswahl des entsprechenden Menüpunkts im Fenster, das die erstellte Liste enthält, kann die Liste gedruckt werden.
Alternative Abläufe:	Keine.

4.5 Statistiken

Die Festlegung der zu erstellenden Statistiken war während der Erhebung der Anforderungen ein viel diskutiertes Thema. Eine vollständige Klärung dieses Punktes wurde nicht erreicht, deshalb ist die hier vorgeschlagene Lösung eher als konzeptioneller Vorschlag denn als perfektes Verfahren zu betrachten.

Use Case 18:	Statistik erzeugen
Ziel:	Es soll eine Statistik erstellt werden.
Externe Aktoren:	Benutzer (beliebig).
Vorbedingungen:	Der Benutzer befindet sich im Hauptfenster.
Beschreibung:	Durch Auswahl des Menüpunktes „Statistik erstellen“ im Bearbeiten-Menü des Hauptfensters gelangt der Benutzer in das Statistik-Fenster. Dort kann der Benutzer eine vordefinierte Statistik auswählen und diese kann durch Drücken des „Erstellen“ Buttons erstellt werden. Außerdem ist die Erstellung von weitgehend beliebigen Statistik in bestimmten vorgegebenen Darstellungsformen durch den Benutzer möglich. Die erstellte Statistik wird dann in der gewählten Darstellungsart in einem neuen Fenster angezeigt.
Alternative Abläufe:	Keine.

Use Case 19:	Statistik exportieren
Ziel:	Eine Statistik soll in andere Anwendungen exportiert werden.
Externe Aktoren:	Benutzer (beliebig).
Vorbedingungen:	Der Benutzer befindet sich im Fenster, das die erzeugte Statistik enthält.
Beschreibung:	Durch Auswahl des entsprechenden Menüpunkts im Fenster, das die erstellte Statistik enthält, kann die Statistik exportiert werden. Es gibt zwei Möglichkeiten des Exports. Dies sind der Export in eine Textdatei und in eine HTML-Datei. Beim Export gehen eventuell erstellte Diagramme verloren, sofern sie nicht in eine Gif-Datei umgewandelt werden können.
Alternative Abläufe:	Keine.

Use Case 20:	Statistik drucken
Ziel:	Eine Statistik soll gedruckt werden.
Externe Aktoren:	Benutzer (beliebig).
Vorbedingungen:	Der Benutzer befindet sich im Fenster, das die erzeugte Statistik enthält.
Beschreibung:	Durch Auswahl des entsprechenden Menüpunkts im Fenster, das die erstellte Statistik enthält, kann die Statistik gedruckt werden.
Alternative Abläufe:	Keine.

Kapitel 7:

Verbesserungs- und Erweiterungsvorschläge

1 Verbesserungen

Im Verlauf der Diplomarbeit hat sich die Einbindung der SQL-Befehle in den Programmcode als sehr mühsam erwiesen. Sollen nun Änderungen an den verwendeten SQL-Befehlen vorgenommen werden, z.B. wenn eine Anpassung an eine geänderte Struktur der Datenbank nötig wird, dann müssen die entsprechenden SQL-Anweisungen im Programmcode gesucht und verändert werden. Aus dieser Erkenntnis heraus würde es sich anbieten, wenn ein Editor für die SQL-Befehle in das Programm integriert würde. Dies würde es außerdem dem Administrator erlauben, Änderungen vorzunehmen, ohne daß das Programm neu übersetzt werden müßte.

Für die Unterstützung eines SQL-Editors würde sich eine Umstellung auf JDBC 2 anbieten. In JDBC 2 wird eine weitergehende Abstraktion von der reinen Programmierung in SQL ermöglicht, womit deutlich weniger SQL-Anweisungen nötig sind. Vor allem wird eine wesentlich verbesserte Handhabung der Ergebnisse von Abfragen ermöglicht. In JDBC 2 können z.B. Operationen wie Löschungen und Änderungen direkt auf dem Ergebnis einer Abfrage durchgeführt werden. Außerdem kann zeilenweise durch das Ergebnis einer Abfrage "gescrollt" werden. Die Kommunikation zwischen Client und Server könnte somit vereinfacht und besser strukturiert werden.

Im der erstellten Datenbank-Anwendung sind die Schnittstellen für JDBC 2 bereits implementiert worden. Allerdings konnte JDBC 2 noch nicht eingesetzt werden, da es für die "standalone" Engine des Datenbank-Server von Sybase noch keinen Treiber gibt.

Eine wichtige Verbesserung für Benutzer der Datenbank-Anwendung wäre die Erstellung einer Online-Hilfe. Hierbei wäre es vorteilhaft, diese kontextsensitiv auszuführen, damit für alle Bedienelemente die passenden Informationen direkt aufgerufen werden können.

2 Konzeptionelle Erweiterungen

Wie im Kapitel über die Implementierung bereits erwähnt, erinnert bei näherer Betrachtung die ablaufende Kommunikation zwischen Client und Server an den Aufruf von Funktion mit Parametern und der darauffolgenden Rückgabe von Werten. Für die Implementierung einer solchen Kommunikation über das Netzwerk ist in Java RMI vorgesehen, es wird mittlerweile aber auch das von Java unabhängige CORBA unterstützt.

Die Kommunikation über RMI würde, neben einer abstrakteren Schnittstelle, vor allem eine Erleichterung bei der Implementierung bedeuten. Bei einer Lösung mit CORBA hätte man zusätzlich noch den Vorteil, daß keine Festlegung getroffen wird, in welcher Programmiersprache der Client und vor allem auch der Server implementiert wurde.

Beim Entwurf der Datenbank-Anwendung wurde auf eine Lösung mit RMI verzichtet, da die hohen Anforderungen im Bezug auf Datenschutz zum damaligen Zeitpunkt nicht erfüllbar schienen. Mittlerweile sind aber geeignete Funktionen in Java vorgesehen worden, so daß eine Umstellung auf RMI oder auch CORBA in Betracht gezogen werden kann.

Wie bereits im vorigen Abschnitt beschrieben wurde, bringt der Zugriff auf die Datenbank per SQL einige Probleme bei Implementieren mit sich. Nicht zuletzt, da ein Strukturbruch zwischen relationaler Datenbank und objektorientierter Datenbank-Anwendung vorliegt und außerdem die Datenbankstruktur bis ins Detail bekannt sein muß. Ein automatisches Auslesen der Datenbankstruktur und damit auch ein automatisches Anpassen der Datenbank-Anwendung wäre wünschenswert. Eine Umstellung auf eine objektorientierte Datenbank könnte dem Erreichen dieses Ziels voraussichtlich behilflich sein.

Im Rahmen des zuvor gesagten würde sich auch die Einbeziehung einer verteilten Datenbank anbieten. Dies würde z.B. die Anbindung der Datenbank des Prüfungsamtes oder verschiedener Institute ermöglichen, womit der bislang sehr mühsame und fehleranfällige Datenaustausch automatisiert werden könnte. Dieser Punkt könnte auch im Bezug auf die geplante Universitätsdatenbank von Interesse sein.

Kapitel 8: Erfahrungen

Wie bereits erwähnt wurde, sollte im Rahmen dieser Diplomarbeit die Erkenntnis gewonnen werden, ob bzw. mit welchen Einschränkungen die in den vorherigen Kapiteln vorgestellte Systemarchitektur in Anwendungen zur Unterstützung von Verwaltungsabläufen eingesetzt werden kann. Außerdem sollten dabei auftretende Schwierigkeiten aufgezeigt werden.

In diesem Kapitel sollen nun zunächst die gewonnen Erkenntnisse vorgestellt werden. Außerdem werden darüber hinausgehend gemachte Erfahrungen während des Projekts erläutert. Im zweiten Abschnitt dieses Kapitels werden aufgetretene Probleme beschrieben, um diese in nachfolgenden Projekten vermeiden zu können.

1 Erfahrungen

Das Projekt hat gezeigt, daß Anwendungen, die per Webbrowser mit dem Benutzer interagieren, sehr vorteilhaft in der Verwaltung eingesetzt werden können. Allerdings gibt es derzeit noch einen großen Schwachpunkt im Bezug auf Benutzerfreundlichkeit, wenn es um den Datenaustausch mit anderen Anwendungen geht. Der Grund dafür ist, daß ein im Webbrowser laufendes Applet nur sehr eingeschränkte Rechte hat, was die Kommunikation mit der "Außenwelt" anbelangt, z.B. darf nur eine Netzwerkverbindung mit dem Host aufgebaut werden von dem das Applet geladen wurde und vor allem darf von der lokalen Platte weder gelesen noch darf auf sie geschrieben werden. Lediglich ein Datenaustausch über die Zwischenablage ist möglich. Dies verhindert somit die Erstellung von Anwendungen, die auf komfortable Art und Weise mit lokalen Anwendungen umfangreiche Daten austauschen sollen.

Deshalb mußte nach Lösungsansätzen für dieses Problem gesucht werden, wobei sich zwei verschiedene Klassen herauskristallisierten. Die erste geht zu Lasten des Bedienkomforts, die zweite verstößt gegen den Verzicht auf eine explizite Softwareinstallation auf Client-Seite.

Ein Vertreter der ersten Klasse wäre die Lösung, daß auszutauschende Daten jeweils vom Benutzer per FTP zum Host, von dem das Applet geladen wurde, transferiert bzw. von dort geholt werden müssen. Dies wurde als für den Benutzer unzumutbar verworfen. Außerdem wären bei dieser Lösung Daten, die eventuell dem Datenschutz unterliegen, ungeschützt über das Netz übertragen worden.

Aus diesem Grund wurde eine Lösung der zweiten Klasse gewählt. Der Benutzer muß einmalig eine "policy"-Datei herunterladen und in einem bestimmten (vom jeweiligen System abhängigen) Verzeichnis ablegen. Aufgrund dieser Datei werden dem Applet zusätzliche Rechte eingeräumt. Im Extremfall können dem Applet auf diese Weise Rechte erteilt werden, wie sie normale Anwendungen auf dem lokalen Rechner haben. Bei dieser Lösung muß der Benutzer allerdings selbst überprüfen, ob er dem Applet vertraut und ob er die erweiterten Rechte wirklich vergeben will.

Eine weitere überraschende Erkenntnis war, daß im Gegensatz zu in HTML-Seiten eingebetteten Java-Scripts es Java-Applets nicht möglich ist, den Inhalt des Browser-Fensters dynamisch zu verändern. Applets ist es nur gestattet neue HTML-Seite aus dem Internet zu laden.

Wesentlich besser als der Datenaustausch ist das Drucken aus Java-Applets heraus gelöst. Hier muß der Anwender lediglich eine Sicherheitsabfrage in Kauf nehmen, ob er das Drucken dem Applet gestatten will oder nicht. Als Fazit kann daraus gezogen werden, daß Anwendungen die den Datenaustausch mit anderen Anwendungen benötigen, bei den derzeitigen drastischen Schutzmaßnahmen nicht allgemein befriedigend implementiert werden können. Deshalb wäre eine Umstellung auf eine ähnliche Konzeption wünschenswert, wie sie für das Drucken verwirklicht wurde. Dies könnte z.B. heißen, daß nach Genehmigung des Benutzers, ein Applet auf ein dafür vorgesehenes Verzeichnis lesend bzw. schreibend zugreifen darf, ähnlich wie bei Cookies, die in einem speziell Verzeichnis abgelegt und von dort auch wieder gelesen werden dürfen.

Im Verlauf dieses Projektes ließ sich erkennen, daß ein Projekt um so komplexer wird, je mehr verschiedene Produkte (von verschiedenen Herstellern) zur Erstellung eines Systems eingesetzt werden sollen. In diesem Projekt wurde DataArchitekt, Adaptive Server und PowerJ von Sybase, Java Development Kit (JDK) von Sun und die Implementierung der Java Cryptography Extension (JCE) von IAIK eingesetzt. Die Probleme die sich bei der Zusammenarbeit zwischen den verschiedenen Produkten ergeben, sind in der Regel um so größer je umfangreicher die Schnittstelle zwischen ihnen ist. Dies läßt sich z.B. gut zwischen PowerJ und JDK erkennen, wo die Schnittstelle praktisch die komplette Programmiersprache Java ist. Zwischen JDK und JCE ist es hingegen lediglich ein Java-Interface.

Sehr gut bewährt hat sich der Einsatz des Tools DataArchitekt, daß aus einer Beschreibung einer Datenbasis mit Entity-Relationship-Modellen beinahe automatisch eine entsprechende relationale Datenbank erstellt. Der einzige kleine Schwachpunkt, der sich herausgestellt hat, ist daß die grafische Darstellung der erstellten Modelle nicht automatisch übersichtlich ausgerichtet angezeigt werden kann. Dies hat zur Folge, daß man z.B. bei einem automatisch erstellten physikalischen Datenbankmodellen dieses von Hand ausrichten muß, was bei größeren Modellen schnell schwierig wird.

Relationale Datenbanken und objektorientierte Programme sind konzeptionell nur schwer miteinander zu vereinen, dies kann im Extremfall dazu führen, daß die Vorteile der objektorientierten Programmierung verloren gehen. Dies ist zuverlässig nur zu verhindern, wenn man eine Konvertierungsschicht zwischen die relationale Datenbank und die objektorientiert programmierte Anwendung legt. Dies bedeutet allerdings einen deutlichen Mehraufwand. Aus diesem Blickwinkel wäre somit der Einsatz von objektorientierten Datenbanken wünschenswert, damit dieser Bruch in der Systemstruktur vermieden wird.

Das Vorgehensmodell des IAS hat sich in diesem Projekt gut bewährt. Vorallem die frühzeitige detaillierte Beschäftigung mit den Anforderungen hat sich günstig bemerkbar gemacht, da später keine wesentlichen Änderungen mehr notwendig waren. Auch die Beschreibung des Systemmodells mit Use Cases war vorteilhaft, da aus ihnen quasi direkt die Testfälle für das System abgeleitet werden konnten.

Ein Verbesserungsmöglichkeit könnte es allerdings in Java-Projekten bei der Spezifikation der Softwarekomponenten geben. Um eine detailliertere Spezifikation zu ermöglichen könnte vorteilhaft JavaDoc als Ergänzung eingesetzt werden. Damit würde zugleich doppelte Arbeit durch das Schreiben von Word-Dateien und späteres Überführen in Java-Quelldateien vermieden. Der Übergang zwischen Entwurf und Implementierung könnte damit fließender verlaufen. Außerdem könnte so schon der erste Schritt für die spätere Dokumentation der Quelldateien gemacht werden.

2 Probleme

Im Laufe des Projektes stellte sich die Datenschutzkomponente als sehr wichtig heraus, was zu neuen Anforderungen führte, die zu Beginn des Projektes noch nicht in dieser Form absehbar waren. Aus diesem Grund gab es während der Erhebung der Anforderungen wesentliche Verzögerungen. Für spätere Projekte wäre es wünschenswert, wenn im Voraus eine detaillierte Klärung über derart grundlegende Anforderungen erfolgen würde.

Beim Einsatz von PowerJ 2.5 stellte sich heraus, daß es nicht kompatibel zu Java 2.0 (und auch nicht zu Swing) ist. Nachdem sich bei Versuchen mit älteren Java Versionen weitere wesentliche Schwächen ergaben, wurde auf dessen Einsatz verzichtet, da durch PowerJ keine Arbeitserleichterungen zu erwarten war. Dies zog allerdings nach sich, daß die Benutzerschnittstelle umständlich von Hand erstellt werden mußte.

Um die Schwachstellen von PowerJ zu verdeutlichen, möchte ich im folgenden ein paar exemplarische Beispiele vorstellen, wobei eine weitere Aufzählung ohne weiteres möglich wäre.

Eine wesentliche Schwachstellen von PowerJ ist, daß es sich nicht an die in Java üblichen Konventionen hält, z.B. werden die Quelldateien nicht hierarchisch nach "packages" strukturiert abgelegt. Außerdem legt es für jede erstellte Klasse (entspricht einer Quelldatei) eine zusätzliche Datei an, die Informationen über die Formatierung der Quelldatei enthält, beispielsweise wo die benutzerdefinierten Konstanten beginnen. Dies hat zur Folge, daß der Austausch von Quelldateien mit anderen Anwendungen beinahe unmöglich wird. Bezeichnend ist, daß PowerJ nicht einmal selbst in der Lage ist, von ihm erstellte Dateien wieder korrekt zu importieren.

Das Arbeiten mit PowerJ wird durch seine mangelhafte Fähigkeit, nötige Konfigurationseinstellungen automatisch vorzunehmen, sehr erschwert. Beispielsweise ist es nicht möglich, Applets zu starten, wenn eine Namensänderung der Hauptklasse vorgenommen wurde, da PowerJ die von ihm erstellte HTML-Datei nicht anpaßt. Außerdem ist der von PowerJ erstellte Programmcode für Menschen nur sehr schwer zu lesen, da es die Klassennamen mit allen "package"-Angaben versieht und auch von ihm automatisch vergebene Namen extrem lang sind. Will man dies von Hand korrigieren, dann versagt PowerJ in Extremfällen sogar den Dienst.

Als überraschend problematisch stellte sich die Kommunikation zwischen dem SQL-Server und der Java-Anwendung heraus. Da man in Java nur wenige Statusmeldungen über den Stand des Verbindungsaufbau bekommt, war es sehr schwierig, den Beginn der Datenbankabfragen durch die Anwendung so lange zu verzögern, daß es nicht zu Abstürzen der Anwendung kommt. Die einzige wenn auch unschöne Lösung, die einigermaßen zuverlässig funktionierte, war aktives Warten. Es wäre schön, wenn in späteren Java Versionen zuverlässige Statusmeldungen per Event gegeben würden.

Zusammenfassung

Im Rahmen dieser Diplomarbeit wurde eine Datenbank erstellt, die alle für das Dekanat der Fakultät Elektrotechnik und Informationstechnik relevanten Daten aufnehmen kann. Um mit dieser Datenbank arbeiten zu können, wurde eine Datenbank-Anwendung implementiert, welche die komfortable Interaktion zwischen dem Benutzer und der Datenbank ermöglichen soll. Als Systemarchitektur wurde ein Client/Server-System gewählt, wobei die Interaktion mit dem Benutzer über ein in einem Webbrowser laufendes Java-Applet auf Client-Seite verwirklicht wird. Diese zukunftsweisende Technologie kommt ohne explizite Installation von Software auf der Client-Seite aus. Dadurch wird der Einsatz in einem heterogenen Systemumfeld ermöglicht, ohne das zusätzlicher Entwicklungsaufwand nötig wird, um die Datenbank-Anwendung auf die verschiedenen Systeme anzupassen.

Für das ganze Projekt stellte die Einhaltung der Datenschutzbestimmungen eine wichtige Anforderung dar. Dies hatte sehr großen Einfluß auf die in der Datenbank aufzunehmenden Daten und auf den Entwurf der Datenbank-Anwendung. Unter anderem wurde deshalb eine Konzeption erarbeitet, welche die Authentisierung und Zugriffskontrolle mit kryptographischen Verfahren sicherstellt.

Als Erkenntnis aus diesem Projekt sollte gewonnen werden, ob bzw. mit welchen Einschränkungen die gewählte Systemarchitektur in Anwendungen zur Unterstützung von Verwaltungsabläufen eingesetzt werden kann. Deshalb sollten auch übliche Vorgänge wie das Erstellen von Listen und Statistiken durch die Datenbank-Anwendung unterstützt werden.

Abstract

Within the scope of this thesis a database was realized, which can take all relevant data for the dean's office of the faculty for electrical engineering and information technology. In order to work with this database, a database-application was implemented, which allows the comfortable interaction between the user and the database. A client/server-system was chosen as system architecture, whereby the interaction with the user was realized over a Java applet in a web browser on the client-side. This promising technology gets along without special software installation on the client-side. Because of this, the use in an heterogeneous system area will be possible without the need of additional expense in development to adapt the database application on different systems

For the whole project was the observance of the rules of data security an important requirement. This had huge influence on the data to hold in the database and the design of the database-application. Because of this a conception was developed, which secures the authentication and access control with cryptographic methods.

As cognition from this project should be gained, if resp. by which restrictions the chosen system architecture can be used in applications for the support of administration processes. Because of this, usual processes like the creation of lists and statistics should be supported from the database-application too.

Anhang A: Datenbankmodelle

Auf den folgenden beiden Seiten ist in den Abbildung A.1 und Abbildung A.2 das vollständige konzeptionelle Datenbankmodell (Conceptual Data Model, CDM) der implementierten Datenbank zu sehen, das mit dem Programm PowerDesigner DataArchitect der Firma Sybase erstellt wurde. Das konzeptionelle Datenbankmodell entspricht im wesentlichen einem Entity Relationship Model und dient dazu, die Datenbasis auf übersichtliche Art zu spezifizieren und von der eigentlichen Implementierung, als Menge von in Beziehung stehenden Relationen, zu abstrahieren.

Aus dem konzeptionellen Datenbankmodell konnte mit dem Programm DataArchitect praktisch automatisch, das für die Implementierung maßgeblich physikalische Datenbankmodell (Physical Data Model, PDM) erstellt werden. Dies ist möglich, da es feststehende Transformationsregeln gibt, die ein Entity Relationship Model in eine relationale Darstellung überführen. Auf diese Transformationsregeln wird hier nicht näher eingegangen und es sei deshalb auf die Literatur verwiesen, z.B. auf die Dokumentation des Programm PowerDesigner DataArchitect [PDGS97], [PDUG97] oder auf das Buch [Balz96], wo eine detaillierte Beschreibung zu finden ist.

Wie in der Darstellung des konzeptionellen Datenbankmodells zu sehen ist, wurde viel mit Vererbungen gearbeitet, um eine übersichtliche Darstellung zu erzielen. Beispielsweise erbt "Studentische Arbeit" und "Promotion" von Wissenschaftliche Arbeit". Dies hatte zur Folge, daß das konzeptionellen Datenbankmodells übersichtlich auf zwei Seiten untergebracht werden konnte.

Bei der Darstellung des aus dem Konzeptionellen Datenbankmodell ermittelten physikalische Datenbankmodell, konnte aufgrund der fehlenden Möglichkeit zur Abstraktion und Strukturierung durch Vererbung ein solch übersichtliche und kompakte Darstellung nicht erreicht werden. Außerdem wurden die gewonnen Relationen jeweils sehr umfangreich, da zahlreiche Fremdschlüssel aufzunehmen waren. Aus diesem Grund wird in dieser Ausarbeitung auf die Darstellung des ermittelte physikalische Datenbankmodell verzichtet. Eine Darstellung des physikalische Datenbankmodells hätte mindestens 6 Seiten erfordert und es wäre dennoch keine ausreichende Übersichtlichkeit erreicht worden.

Anhang B: Glossar

<i>Assoziation</i>	Eine Assoziation beschreibt eine fachliche Beziehung zwischen Entitäten.
<i>Attribut</i>	Ein Attribut ist eine Eigenschaft einer Entität oder einer Beziehung.
<i>CDM</i>	Als <u>C</u> onceptual <u>D</u> ata <u>M</u> odel wird im Programm DataArchitect der Firma Sybase eine auf Entity Relationship Modellen basierende grafische Beschreibung der konzeptionellen Datenstruktur einer Datenbank bezeichnet.
<i>CORBA</i>	Die <u>C</u> ommon <u>O</u> bject <u>R</u> equest <u>B</u> roker <u>A</u> rchitecture ist mit RMI vergleichbar, allerdings ist keine Festlegung auf die Programmiersprache Java gegeben. CORBA ist somit von System und Programmiersprache unabhängig verwendbar.
<i>Datenbasis</i>	Die Gesamtheit der gespeicherten Daten und ihre Struktur wird Datenbasis genannt.
<i>Datensatz</i>	Ein Datensatz ist eine Zeile in einer Tabelle. Werden mehrere Tabellen miteinander verknüpft (z.B. über Fremdschlüssel), stellt ein Datensatz alle Attribute von zusammengehörenden Zeilen dar.
<i>DBMS</i>	Das <u>D</u> aten <u>b</u> an <u>k</u> manag <u>e</u> ment <u>s</u> ystem verwaltet die Datenbasis.
<i>DBS</i>	Datenbasis und Datenbankmanagementsystem zusammen werden als <u>D</u> aten <u>b</u> an <u>k</u> system bezeichnet.
<i>Embedded SQL</i>	Werden in einer Programmiersprache (Wirtssprache) eingebettete SQL-Anweisungen verwendet, spricht man von Embedded SQL.
<i>Entität</i>	Eine Entität ist ein identifizierbares Element aus der realen oder der Vorstellungswelt
<i>Fremdschlüssel</i>	Dieses Schlüsselattribut eines Datensatzes verweist auf einen Primärschlüssel eines anderen Datensatzes. Somit lassen sich Beziehung zwischen Datensätzen herstellen.
<i>HTML</i>	Die <u>H</u> ypertext <u>M</u> arkup <u>L</u> anguage dient zur Programmierung von Seiten im World Wide Web, die in Webbrowsern dargestellt werden können. In diesem Rahmen dienen sie auch zur Einbindung von Java-Applets.

<i>IAIK</i>	Die für die Erstellung der Datenbank-Anwendung verwendete Implementierung der JCE, wurde vom Institut für angewandte Informationsverarbeitung und Kommunikation der Technischen Universität Graz für Forschungszwecke frei zur Verfügung gestellt.
<i>IAS</i>	Die Durchführung der Diplomarbeit erfolgte am Institut für Automatisierung- und Softwaretechnik der Fakultät für Elektrotechnik und Informationstechnik
<i>JCE</i>	Die Java Cryptography Extension ist ein Softwarepaket, das eine von der Firma Sun spezifizierte Schnittstelle einhält und zur Bereitstellung umfangreicher Sammlungen an kryptographischen Algorithmen in Java dient. Aufgrund der in den USA geltenden Beschränkungen beim Waffenexport, unter die auch sichere Verschlüsselungsprogramme (sog. "Harte Kryptographie") fallen, ist die Implementierung von Sun außerhalb der USA nicht erhältlich, es gibt aber zahlreiche frei zugängliche Implementierungen von Drittanbietern (siehe IAIK).
<i>JDBC</i>	Unter der Java Database Connectivity versteht man das Programmpaket, das es in Java möglich macht, auf Datenbanken per SQL zuzugreifen. Die Kommunikation erfolgt per ODBC oder durch einen Treiber, der von einem Drittanbieter bereit gestellt wird.
<i>JDK</i>	Das Java Development Kit ist das von der Firma Sun frei zur Verfügung gestellte Softwarepaket zur Implementierung von Anwendungen in Java.
<i>JRE</i>	Um in den derzeit gängigen Webbrowsern Applets ausführen zu können, die Java 2 geschrieben sind, ist die Installation der Java Runtime Environment nötig.
<i>ODBC</i>	Open Database Connectivity ist eine standardisierte Schnittstelle für Anwendungsprogramme zum herstellerunabhängigen Zugriff auf Datenbanksysteme. ODBC enthält eine Bibliothek mit Funktionsaufrufen, die einer Anwendung erlauben, eine Verbindung zu einem DBMS aufzubauen, SQL-Anweisungen auszuführen und Ergebnisse zu ermitteln.
<i>PDM</i>	Als Physical Data Model wird, im Programm DataArchitect der Firma Sybase, eine auf Relationen basierende grafische Beschreibung der Datenstruktur einer Datenbank. Sie kann praktisch automatisch aus dem CDM gewonnen werden.
<i>Primärschlüssel</i>	Das zur eindeutigen Identifikation einer Zeile in einer Tabelle verwendete Attribut (evtl. mehrere) wird Primärschlüssel genannt.

<i>Relation</i>	Eine Relation ist eine Teilmenge eines Kartesisches Produktes über den Wertebereich der Attribute einer Entitäts- oder Beziehungsmenge.
<i>Relationenschema</i>	Die Beschreibung des Aufbaus einer Relation, besteht aus dem Relationsnamen und den Attributnamen, wird Relationenschema genannt,
<i>RMI</i>	Unter <u>R</u> emote <u>M</u> ethod <u>I</u> nvocation versteht man die Kommunikation zwischen einem in Java implementierten Client und Server, in dem der Client Methoden aufruft, die der Server über eine Schnittstellendefinition zur Verfügung stellt.
<i>SQL</i>	Die <u>S</u> tructured <u>Q</u> uery <u>L</u> anguage ist eine Datenbanksprache zur Abfrage und Manipulation von Daten und ihrer Struktur.
<i>Tabelle</i>	Die tabellarische Darstellung eines Relationenschemas (Tabellenkopf) mit Relationen (Tabelleninhalt) wird als Tabelle bezeichnet.
<i>Wirtssprache</i>	Die Programmiersprache (hier: Java), welche Anweisungen einer anderen Programmiersprache (hier: SQL) enthält, wird Wirtssprache genannt.

Anhang C: Literaturverzeichnis

- [Balz96] Helmut Balzert
Lehrbuch der Software-Technik
Band 1: Software-Entwicklung
Spektrum Akademischer Verlag, 1996
- [Baum98] Dr. Joachim Baumann
Vorlesungsskript: Rechnernetze II
Universität Stuttgart, 1998
- [CaWa98] Mary Campione, Kathy Walrath
The Java Tutorial Second Edition
Object-Oriented Programming for the Internet
Addison-Wesley Publishing Company, 1998
- [CoHo97] Gary Cornell, Cay S. Horstmann
Core Java 1.1
Volume I-Fundamentals
Sun Microsystems Press / Prentice Hall, 1997
- [CoHo98] Gary Cornell, Cay S. Horstmann
Core Java 1.1
Volume II-Advanced Features
Sun Microsystems Press / Prentice Hall, 1998
- [CWHT98] Mary Campione, Kathy Walrath, Alison Huml, Tutorial Team
The Java Tutorial Continued
The Rest of the JDK
Addison-Wesley Publishing Company, 1998
- [GHJV95] Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides
Design Patterns
Elements of Reusable Object-Oriented Software
Addison-Wesley Publishers Ltd., 1995
- [Göhn99] Prof. Dr. Peter Göhner
Vorlesungsskript: Softwaretechnik II
Universität Stuttgart, 1999
- [KeEi99] Alfons Kemper, André Eickler
Datenbanksysteme
Eine Einführung
R. Oldenbourg Verlag, 1999

- [Mits99] Prof. Dr. Bernhard Mitschang
**Vorlesungsskript: Datenbankbasierte Anwendungen
Modellierungs-, Entwicklungs- und Verarbeitungskonzepte**
Universität Stuttgart, 1999
- [Pete99] Dr. Holger Petersen
Vorlesungsskript: Kryptographische Verfahren
Universität Stuttgart, 1999
- [PDUG97] **PowerDesigner DataArchitect
User's Guide**
Powersoft, 1997
- [PDGS97] **PowerDesigner DataArchitect
Getting Started**
Powersoft, 1997
- [ReLy97] Prof. Dr. Andreas Reuter, Dr. Frank Leymann
Vorlesungsskript: Grundlagen der Datenbank und Transaktionssysteme
Universität Stuttgart, 1997
- [Roth97] Prof. Dr. Kurt Rothermel
Vorlesungsskript: Grundlagen der Verteilten Systeme
Universität Stuttgart, 1997
- [SASA98] **SQL Anywhere Studio
Adaptive Server Anywhere
User's Guide**
Sybase Inc., 1998
- [SASF98] **SQL Anywhere Studio
First Guide to SQL Anywhere Studio**
Sybase Inc., 1998
- [Tane97] Andrew S. Tanenbaum
Computernetzwerke
Prentice Hall, 1997
- [TiJe97] Michael Tischer, Bruno Jennrich
**Internet intern
Technik und Programmierung**
Data Becker, 1997
- [WaCa99] Kathy Walrath, Mary Campione
**The JFC Swing Tutorial
A Guide to Constructing GUIs**
Addison-Wesley Publishing Company, 1999

Anhang D: Abbildungsverzeichnis

KAPITEL 2: ANFORDERUNGEN

ABBILDUNG 2.1: INSTITUTIONEN.....	10
ABBILDUNG 2.2: INSTITUTIONSBAUM.....	10
ABBILDUNG 2.3: PERSONEN.....	11
ABBILDUNG 2.4: MAILINGLISTEN.....	12
ABBILDUNG 2.5: ADRESSEN.....	12
ABBILDUNG 2.6: WISSENSCHAFTLICHE ARBEITEN	12
ABBILDUNG 2.7: NOTEN	13
ABBILDUNG 2.8: PROMOTIONEN	13
ABBILDUNG 2.9: BENUTZER.....	14
ABBILDUNG 2.10:TERMINE.....	14

KAPITEL 3: DATENSCHUTZ

ABBILDUNG 3.1: EBENEN DES DATENSCHUTZES	18
ABBILDUNG 3.2: BENUTZERVERWALTUNG	20

KAPITEL 4: SYSTEMARCHITEKTUR

ABBILDUNG 4.1: SYSTEMÜBERSICHT.....	23
ABBILDUNG 4.2: DATENBANKSYSTEM.....	24
ABBILDUNG 4.3: SOFTWARE-SYSTEMARCHITEKTUR	27

KAPITEL 5: IMPLEMENTIERUNG

ABBILDUNG 5.2: SOFTWARE-SYSTEMARCHITEKTUR	31
ABBILDUNG 5.2: KLASSENDIAGRAMM DER TRANSAKTIONSVERWALTUNG.....	33
ABBILDUNG 5.3: KLASSENDIAGRAMM DES RELATION-OBJEKT-KONVERTERS	35
ABBILDUNG 5.4: KLASSENDIAGRAMM DES SICHEREN KANALS	36
ABBILDUNG 5.5: ANWENDUNGSSCHICHT.....	37
ABBILDUNG 5.6: KLASSENDIAGRAMM DER ANWENDUNG	38
ABBILDUNG 5.7: FENSTERSTRUKTUR	39
ABBILDUNG 5.8: KLASSENDIAGRAMM DER OBERFLÄCHE	40

KAPITEL 6: BENUTZUNGSANLEITUNG

ABBILDUNG 6.1: START-FENSTER	44
ABBILDUNG 6.2: STATUSFENSTER.....	44
ABBILDUNG 6.3: FENSTERSTRUKTUR	45
ABBILDUNG 6.4: ANMELDEFENSTER	46
ABBILDUNG 6.5: HAUPTFENSTER (1).....	47
ABBILDUNG 6.6: HAUPTFENSTER (2).....	47
ABBILDUNG 6.7: HAUPTFENSTER (3).....	48
ABBILDUNG 6.8: PAßWORT-FENSTER	49
ABBILDUNG 6.9: SQL-DIALOG	49
ABBILDUNG 6.10: SORTIER-FENSTER	50
ABBILDUNG 6.11: FILTER-FENSTER	51
ABBILDUNG 6.12: LISTEN-FENSTER	52
ABBILDUNG 6.13: STATISTIK-FENSTER	53

ANHANG A: DATENBANKMODELLE

ABBILDUNG A.1: CONCEPTUAL DATA MODEL (TEIL 1) 68
ABBILDUNG A.2: CONCEPTUAL DATA MODEL (TEIL 2) 69

Erklärung

Ich versichere, daß ich diese Arbeit selbständig verfaßt und nur die angegebenen Hilfsmittel verwendet habe.

Ostfildern, den

(Jürgen Rau)

Ein Musterexemplar liegt bei der die Arbeit entgegennehmenden Stelle zur Einsicht aus.