Universität des Saarlandes
Lehrstuhl für Programmiersysteme
Gebäude 45
Postfach 15 11 50
66041 Saarbrücken Germany

Konzeption und Realisierung eines Bibliotheksverwaltungssystem

Diplomarbeit an der
Universität des Saarlandes

Dag Kröper
Juli 2002

Eidesstattliche Erklärung

Hiermit erkläre ich an Eides statt, dass ich die vorliegende Diplomarbeit selbstständig verfasst und dazu keine anderen als die angegebenen Hilfsmittel benutzt habe.

Saarbrücken, im Juli 2002

Dag Kröper

Danke!

Herrn Prof. Dr. Gert Smolka für die Vergabe dieses interessanten Themas und für die kompetente Betreuung.

Dr. Christian Schulte für die kompetente Betreuung beim Erstellen meiner Diplomarbeit.

Simone Schulze und Anja Becker für ihre Unterstützung bei der Planung des Projektes.

Meiner Frau Kerstin und meiner Familie für ihre Unterstützung, Geduld und ihr Verständnis während meines Studiums.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Kapitel 1

Einleitung

1.1 Motivation

Das vorhandene Bibliotheks- und Verwaltungssystems der Fachrichtungsbibliothek Informatik ist Ende der 80-er Jahre entstanden. Es basiert zum großen Teil auf csh-Skripten, die im Laufe der Jahre ständig geändert und angepasst wurden. Das so gewachsene System wurde immer unübersichtlicher und wartungsunfreundlicher.

Des Weiteren wird der Datenbestand in ASCII-Dateien gehalten, wodurch der Zugriff mit wachsender Datenmenge immer langsamer wird. Dieses Problem wurde durch die Aufnahme der Daten der Fachrichtungsbibliothek der Angewandten Mathematik forciert.

Es stellt sich nun die Frage, das vorhandene System auf die heutigen Bedürfnisse anzupassen, ein Bibliothekssystem zu kaufen oder ein neues Bibliothekssystem zu entwerfen. Das Anpassen des bestehenden Systems käme einer Neuimplementierung gleich, da die gesamte Datenstruktur geändert werden müsste. Ein extern erworbenes System hat den Nachteil, dass man es nicht selbst erweitern kann. Es bleibt nur noch die letzte Möglichkeit übrig, ein System selbst zu entwickeln.

1.2 Problemstellung und Ziele

Um ein neues Bibliothekssystem zu entwickeln, sind verschiedene Aspekte zu beachten. Diese Aspekte werden in [Bal96a] beschrieben.

Ein Teil der Arbeit besteht darin, die Anforderungen an das neue System abzustecken. Es muss den Benutzer bei der täglichen Routinearbeit in der Bibliothek unterstützen. Des Weiteren soll es die Administration erleichtern und es soll wartungsfreundlich sein. Es dürfen auch keine Einbußen in der Funktionalität des bestehenden Systems entstehen. Ein weiterer Punkt ist die Implementierung. Eine der wichtigen Fragen ist, welche Programmiersprache und welche Datenhaltung werden verwendet, um den Anforderungen gerecht zu werden.

1.3 Aufbau der Arbeit

In Kapitel 2 werden Grundlagen besprochen, die für das System notwendig sind. Es werden die Abläufe einer Bibliothek aus verschiedenen Blickpunkten gezeigt. In diesem Kapitel wird das alte Bibliothekssystem der Fachrichtungsbibliothek vorgestellt. Der Abschluss bildet die Beschreibung des Maschinellen Austauschformates für Bibliotheken (MAB2) [Die99]. Dieses Format bildet die Struktur der neuen Datenbank.

Kapitel 3 beschäftigt sich mit der Entwicklung von BiVIS (**Bi**bliotheks- und **V**erwaltungssystem der Fachrichtungsbibliothek **I**nformatik **S**aarbrücken). Es werden die Anforderungen an BiVIS spezifiziert und die Auswahl des zugrundeliegenden Prozessmodells diskutiert. Die Architektur und der Datenbankentwurf bilden den Abschluss des Kapitels.

Die Implementierung wird in Kapitel 4 beschrieben. Es werden die wichtigsten Module von BiVIS vorgestellt und ihre Implementierung beschrieben.

Eine Zusammenfassung in Kapitel 5 bildet den Abschluss der Arbeit.

Kapitel 2

Grundlagen

In diesem Kapitel werden Grundlagen zum Verständnis dieser Arbeit geschaffen. In Abschnitt 2.1 werden Arbeitsabläufe in der Bibliothek aus der Sicht des Kunden und des Mitarbeiters beschrieben. Das bestehende System und dessen Datenstruktur wird in den Abschnitten 2.2 und 2.3 beschrieben. Den Abschluss diese Kapitels bildet eine kurze Beschreibung des MAB2-Datenformats in Abschnitt 2.4.

2.1 Arbeitsabläufe in der Bibliothek

Wir betrachten hier die Funktionalität, die BiVIS aus der Sicht der Anwender und Kunden mitbringt. Dies kann man am anschaulichsten darstellen, indem man die Abläufe in der Fachrichtungsbibliothek beschreibt.

2.1.1 Bestellung und Schenkung von Büchern und Zeitschriften

In diesem Abschnitt wird die Frage betrachtet, wie eine Bibliothek ihr Literatur bezieht. Es gibt hier grundsätzlich zwei Wege: die Bücher werden bestellt oder der Bibliothek werden die Bücher geschenkt. Der wichtigste Unterschied ist, dass es bei der Bestellung eine Rechnung gibt, die bezahlt werden muss. Also muss bei einer Bestellung ein Rechnungsdatensatz erzeugt werden.

Bestellung von Büchern

Der Bestellvorgang wird in Abbildung 2.1 vorgestellt. Bei einer Bestellung gibt der Anwender das zu bestellende Buch mit einigen Zusatzinformationen in das System ein. Diese Zusatzinformationen sind zum Beispiel der Lieferant, voraussichtlicher Preis oder Anzahl der Exemplare. Aus den eingegebenen Daten erstellt das System automatisch die entsprechende Einträge in der Lokaldatenbank. Das System druckt nun auf Anweisung des Benutzers einen Bestellbrief für einen Lieferanten aus oder versendet die Bestellung per E-Mail.

Zu diesem Zeitpunkt wurde von dem System ein Literaturstammdatensatz für jeden bestellten Titel und ein Lokaldatensatz und ein Rechnungsdatensatz für jedes bestelle Exemplar dieses Titels angelegt. Der Status jedes bestellten Exemplares wird auf 'bestellt' gesetzt.

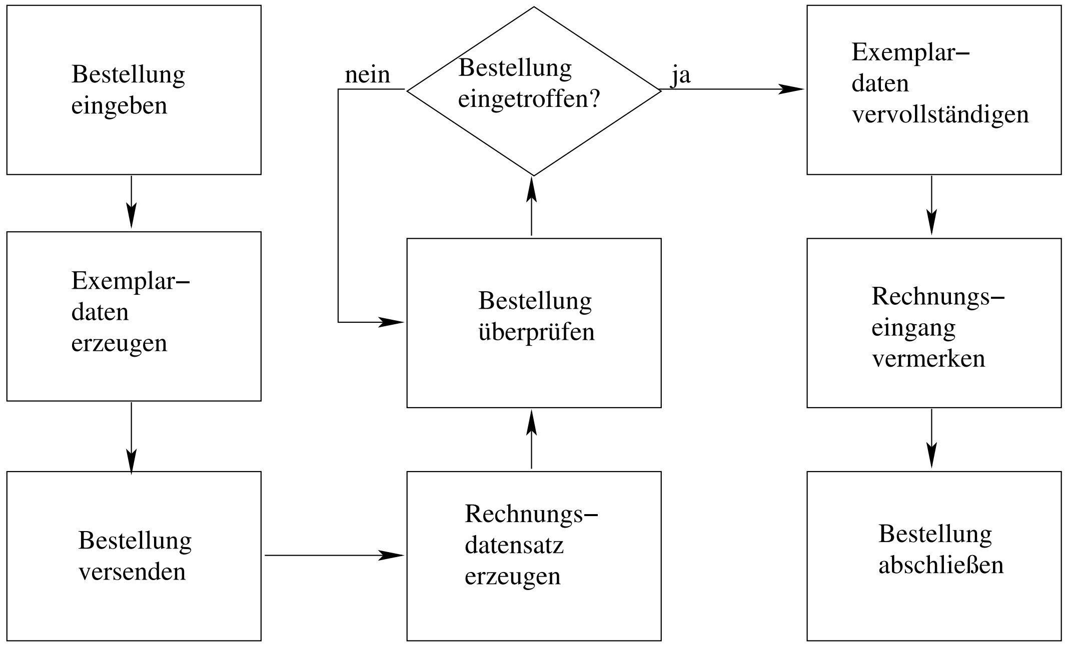

Abbildung 2.1: Ablaufdiagramm der Bestellung von Büchern

Der Biblitotheksmitarbeiter muss nun die Lieferung der Bücher überwachen. Das System unterstützt ihn dabei, indem es dem Anwender mitteilt, welche Bestellungen, nach einem voreingestellten Zeitraum, noch nicht eingetroffen sind.

Trifft ein Buch ein, so wird der Status dieses Exemplars gesetzt und die Signatur zum Druck freigegeben. Der Rechnungsdatensatz kann jetzt aktualisiert werden. Das bestellte Buch ist nun für die Kunden der Bibliothek verfügbar.

Schenkung

Wird ein Buch als einen Schenkung inventarisiert, dann wird nur ein Stammdatensatz (falls noch nicht vorhanden) und ein Lokaldatensatz erzeugt. Da das Buch im allgemeinen schon vorliegt, kann man direkt den Status auf einen entsprechenden Wert setzen und die Signatur zum drucken freigeben.

Zeitschriften

Die Bestellung ist bei Zeitschriften identisch zu den Büchern. Es unterscheidet sich nur die Überwachung der Lieferung. Da Zeitschriften in bestimmten Zeitabständen erscheinen, muss der Eingang jedes Heftes überprüft werden. Wenn ein Jahrgang komplett ist wird er zum Binden freigegeben und in das Zeitschriftenarchiv eingestellt.

2.1.2 Kunden und Ausleihe

Kunden

Für jeden Kunden der Bibliothek wird ein Kundendatensatz angelegt, der Angaben zur Person und deren Ausleihen speichert. Damit eine Person ausleihberechtigt wird, muss sie einen schriftlichen Antrag an die Bibliothek stellen. Nach dem Erstellen des Kundendatensatzes werden die Kundenangaben auf eine Kundenkarteikarte gedruckt. Der Kunde erhält daraufhin eine Begrüssungs-E-Mail zur Kontrolle seiner Angaben und ist ab diesem Zeitpunkt ausleihberechtigt in der Bibliothek für die Dauer eines vorher festgelegten Zeitraums.

Ausleihe

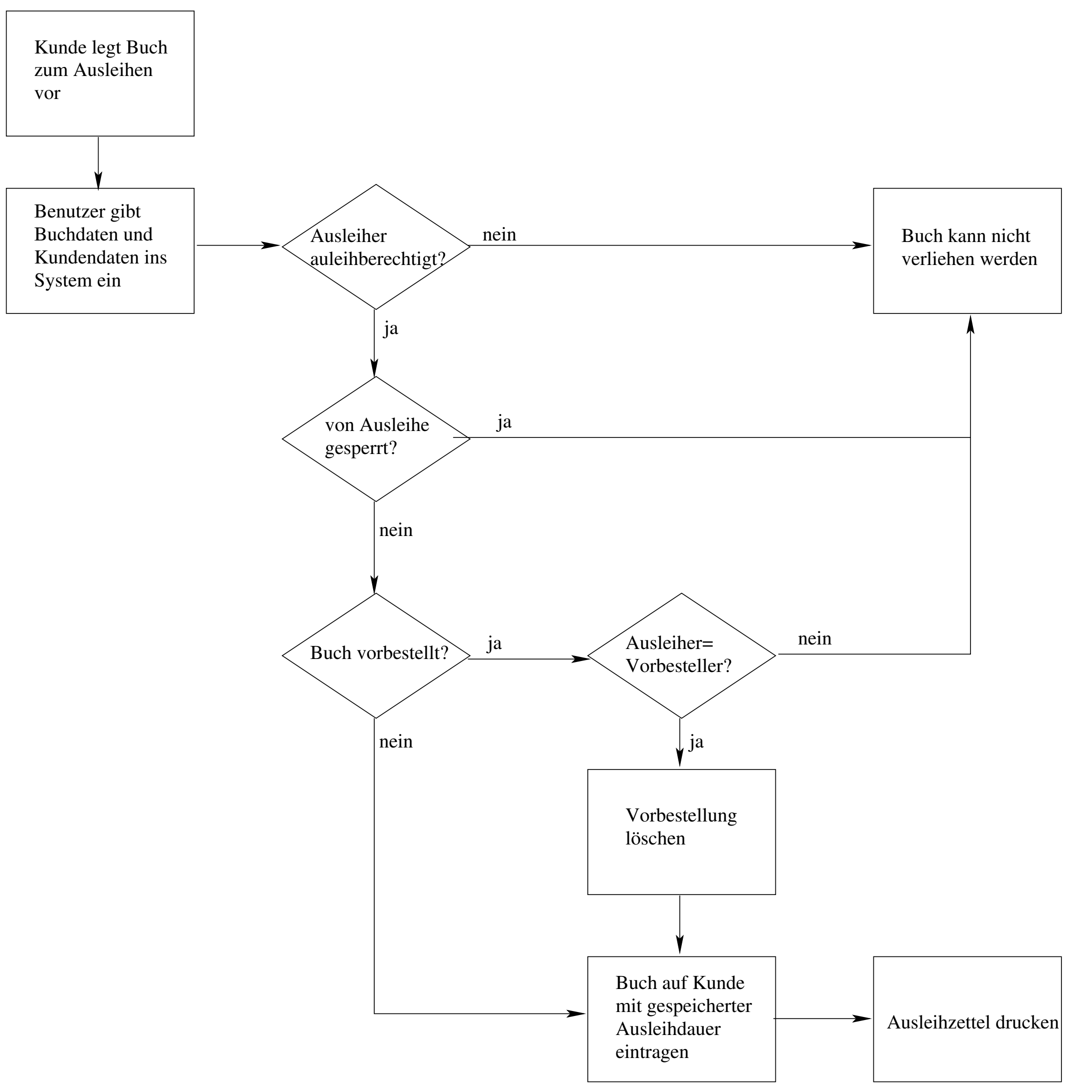

Abbildung 2.2: Ablaufdiagramm der Ausleihe

Ein ausleihberechtigter Kunde kann Bücher, die nicht von der Ausleihe gesperrt sind, für einen festgelegten Zeitraum ausleihen. Die Bibliothek erlaubt keine Selbstverbuchung, d.h. der Kunde muss das Buch von der Aufsicht als Ausleihe verbuchen lassen. Zur Ausleihverbuchung gibt der Anwender die Kundennummer des Ausleiher und die Signaturen der Bücher ein. Das System speichert die Ausleihe, das Ausleihdatum und das Ausleihende und druckt für jedes ausgeliehene Buch einen Ausleihzettel. Der Ausleiher hat die Möglichkeit, nach Ablauf der Leihfrist das Buch um die eingestellte Leihfrist verlängern zu lassen. Der Ablauf der Ausleihe ist in der Abbildung 2.2 graphisch dargestellt. Bei Rückgabe der Bücher wird das Rückgabedatum gespeichert, der Datensatz anonymisiert und für statistische Zwecke erhalten. Die Ausleihdaten werden regelmäßig von dem Bibliothekssystem automatisch überprüft. Sollte eine Leihfrist überschritten werden mahnt das System automatisch die ausstehenden Bücher bei dem Ausleiher an. Dies erfolgt per E-Mail, die erste und zweite Mahnung, und als Brief, die dritte Mahnstufe.

Vorbestellung

Falls ein Buch verliehen sein sollte kann ein Kunde dieses Buch vorbestellen. Der Ausleiher kann es dann nicht mehr verlängern lassen und muss es nach Ablauf der Leihfrist zurückbringen. Nach Rückgabe des Buches wird der Vorbesteller automatisch benachrichtigt. Nur der Vorbesteller kann jetzt dieses Buch ausleihen. Nach einem festgelegten Zeitraum wird der Vorbestellungsvermerk gelöscht und das Buch steht wieder für alle Ausleiher bereit.

2.2 Beschreibung des bestehenden Systems

Das bestehende Bibliotheksverwaltungssystem IBBIB wurde Ende der 80er Anfang der 90er Jahre implementiert. Es besteht aus einer Ansammlung von *csh*-, *awk* und *perl*-Skripten. Diese Skripte wurden ständig an die Bedürfnisse der Biblithek angepasst. Durch diese ständigen Änderungen, die nicht immer kommentiert wurden, ist das System unübersichtlich geworden.

Funktional ist IBBIB auch mit den Jahren gewachsen. Zum heutigen Zeitpunkt bietet es folgende Funktionalität:

- Literaturverwaltung
  - Katalogisieren von Büchern, Reports und Proceedings
  - Bestellung von Büchern mit Überwachung der Lieferung
  - Zeitschriftenarchivierung, Hefteingangskontrolle
- Benutzerverwaltung
  - Kontrolle der Zugriffe auf die einzelnen Skripte
- Ausleihverbuchung
  - Erfassen von Ausleihen
  - Überprüfen der Ausleihen
  - Automatisches Anmahnen ausstehender Ausleihen
- Kundenverwaltung

  - Aufnehmen der Kundendaten
  - Überprüfung der Ausleihberechtigung
- Buchhaltung
  - Erstellen von Rechnungsdatensätzen
  - Erstellen von vordefinierten Statistiken

2.3 Beschreibung des bestehenden Datenformats

Das Datenformat von IBBIB ist textbasiert. Es besteht aus ASCII-Dateien, die in einer Verzeichnisstruktur abgelegt sind.

2.3.1 Aufbau der Dateien

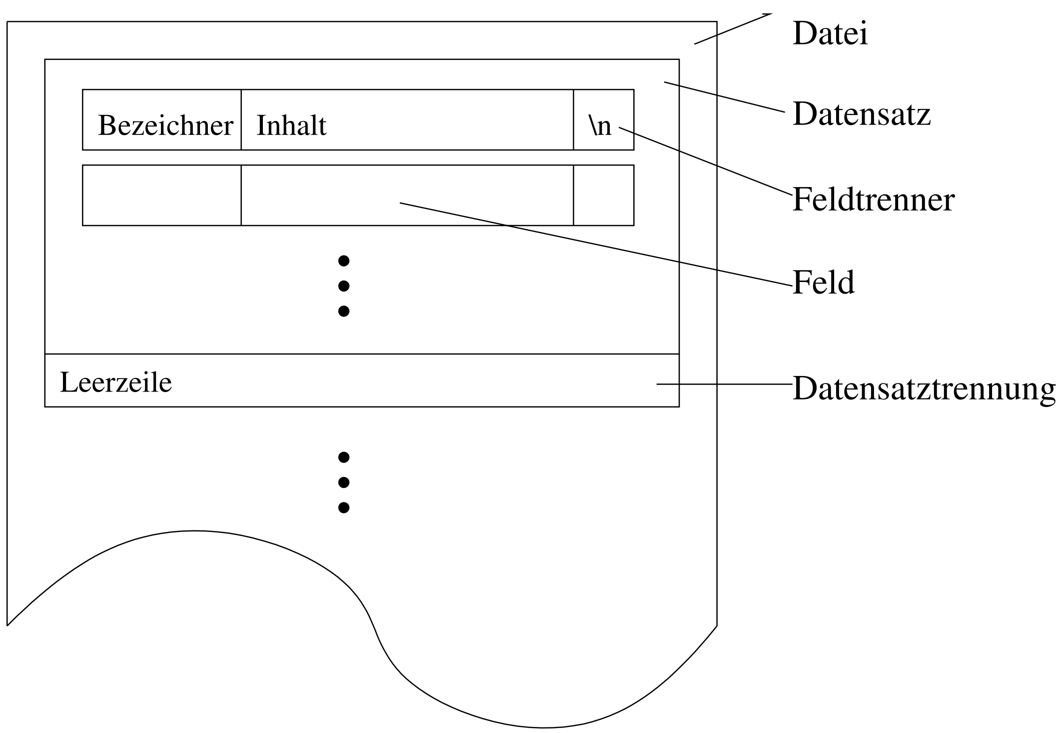

Abbildung 2.3: Aufbau der Datendateien von IBBIB

Die einzelnen Dateien von IBBIB haben alle den gleichen Aufbau. Dieser Aufbau ist in Abbildung 2.3 dargestellt. Eine Datei enthält Datensätze, die durch eine Leerzeile getrennt sind. Die Datensätze bestehen aus Datenfelder, die aus dem Feldbezeichner, dem Inhalt und dem Trennsymbol $\backslash n$ aufgebaut sind.

Die Feldbezeichner bestehen aus einem Prozent-Zeichen und einem oder zwei Buchstaben. Sie haben in den verschiedenen Dateien unterschiedliche Bedeutungen. In den Kundendaten bezeichnet z.B. *%H* die Kundennummer und in den Literaturdaten die Inventarnummer. In den Tabellen 2.1, 2.2, 2.3 und 2.4 wird die Bedeutung der Feldbezeichner aufgelistet.

Bezeichner	**Bedeutung**
%H	Inventarnummer
%A	Autor
%E	Herausgeber (Editor)
%T	Titel
%t	Untertitel
%S	Serie
%G	Lehrstuhl (Diplomarbeiten oder Dissertationen)
%I	Verlag
%C	Verlagsort/Konferenzort
%O	ISBN/ISSN
%D	Erscheinungsjahr
%Y	Besteller
%L	Lieferant
%W	Standort
%R	Signatur

Tabelle 2.1: Feldbezeichner der Literaturdaten

Bezeichner	**Bedeutung**
%H	Kundennummer
%N	Nachname
%V	Vorname
%G	Geschlecht
%W	Wohnort
%S	Straße
%T	Telefon privat
%Z	Anschrift dienstlich
%D	Telefon dienstlich
%Q	Lehrstuhl
%A	Dauer der Ausleihberechtigung
%E	E-Mail Adresse
%C	Nummer der Zugangskarte
%J	Information über Hefteingang dieser Zeitschrift
%Y	Ausleihen

Tabelle 2.2: Feldbezeichner der Kundendaten

Bezeichner	**Bedeutung**
%H	Inventarnummer
%Y	Besteller
%L	Lieferant
%D1	Bestelldatum
%P1	Vorraussichtlicher Preis und Währung
%D2	Lieferdatum
%P2	Preis laut Rechnung und Währung
%D3	Datum der Bezahlung
%P3	Endgültiger Preis und Währung
%K	Kommentar
%N	Rechnungsnummer

Tabelle 2.3: Feldbezeichner der Rechnungsdaten

Bezeichner	**Bedeutung**
%R	Zeitschriftennummer
%T	Titel
%t	Kurztitel
%I	Verlag
%O	ISSN
%U	Erscheinungsweise
%B	Bestelldatum
%D	Bestandsnachweis
%L	Lieferant
%Y	Besteller
%K	Kommentar
%W	Standort
%u	URL zur Zeitschrift
%M	Hefteingang
%F	fehlendes Heft

Tabelle 2.4: Feldbezeichner der Zeitschriftendaten

2.3.2 Verzeichnisstruktur der Daten in IBBIB

Die Daten von IBBIB sind in drei Gruppen aufgeteilt. Jeder Gruppe ist ein Verzeichnis zugeordnet, das noch weiter unterteilt sein kann. Die Gruppen entsprechen der funktionalen Zuordnung der Daten. Der Verzeichnisbaum ist in Abbildung 2.4 beschrieben. Die Knoten des Baums entsprechen den einzelnen Verzeichnissen, während die Blätter Dateien repräsentieren.

Im Verzeichnis `invent` sind die Inventar- und Rechnungsdaten untergebracht. Dieses Verzeichnis ist noch weiter untergliedert nach den Jahren, in denen die Werke in der Bibliothek erfasst wurden. Somit stehen im Verzeichnis `1990` die Bücher die im Jahre 1990 erfasst wurden. Es gibt zwei Besonderheiten in dieser Gliederung: das Verzeichnis `1989` enthält alle Bücher die 1989

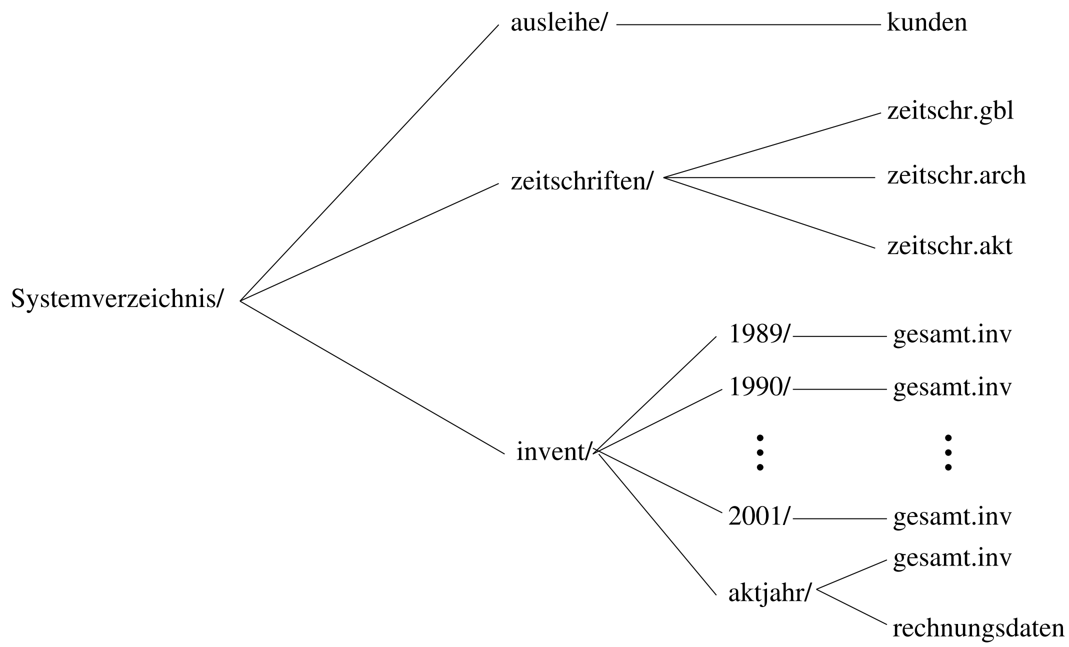

Abbildung 2.4: Verzeichnisbaum der Datendateien von IBBIB

und vorher inventarisiert wurden und das Verzeichnis `aktjahr` enthält neben den Büchern des aktuellen Jahres auch die gesamten Rechnungsdaten.

Das Verzeichnis `ausleihe` enthält die Kunden- und Ausleihdaten in der Datei `kunden`.

Die Daten der Zeitschriften beherbergt das Verzeichnis `zeitschriften`. Die Daten sind in drei Dateien aufgeteilt. Die Datei `zeitschr.akt` enthält alle laufenden Zeitschriften und deren Heftnachweis. Die archivierten Zeitschriften sind in der Datei `zeitschr.arch` gespeichert. Die Datei `zeitschr.gbl` enthält alle in der Bibliothek enthaltene Zeitschriften mit deren Bestandsangaben.

2.4 Das MAB2 Datenformat

Die Deutsche Bibliothek Berlin hat im Jahre 1973 ein **M**aschinelles **A**ustauschformat für **B**iblitheken (MAB) veröffentlicht. Dieses Austauschformat hat sich bis heute bei den deutschen Bibliotheken durchgesetzt. In der Zeit von 1993 bis 1995 wurde dieses Format grundlegend überarbeitet. Das Ergebnis dieser Überarbeitung ist das **M**aschinelles **A**ustauschformat für **B**iblitheken Version **2** (MAB2). Im MAB2-Format werden Satzaufbau, Feldkennung, Feldstruktur, Feldinhalt, Beziehungen zwischen Datensätzen und Zeichensätze festgelegt.

Das MAB2-Format besteht aus mehreren Teilen. Die literarischen bzw. bibliographischen Stammdaten sind alle Angaben über ein literarisches Werk. Die Datenstruktur der Stammdatensätze wird im *MAB-Format für Titeldaten (MAB-TITEL)* festgelegt. Um die Einträge konsitent zu halten beinhaltet das MAB-Format mehrere Normdateien. Diese Normdateien sind:

- Personennamendatei (PND)

- Gemeinsame Körperschaftendatei (GKD)
- Schlagwortnormdatei (SWD)

Das MAB-Format legt auch das Format dieser Dateien fest.

Alle Daten die sich auf ein Exemplar eines literarischen Werkes beziehen, werden in der Lokaldatendatei gespeichert. Das Format für diese Datei ist im *MAB-Format für Lokaldaten (MAB-LOKAL)* festgelegt. Solche exemplarspezifischen Daten sind z.B. die bibliotheksinterne Inventarisierungsnummer, die Signatur, usw.

2.4.1 Struktur des MAB2-Formates

Jeder MAB2-Datensatz besteht aus einer Satzkennung, mehreren variablen Datenfeldern und einem Satzendezeichen.

Die Satzkennung ist 24 Zeichen lang. An Position 0-4 steht die Satzlänge. Sie errechnet sich aus der Anzahl der Zeichen des Datensatzes. An Position 5 steht der Satzstatus. Er gibt den Bearbeitungszustand (neu, korrigiert, gelöscht, ...) an. Die MAB-Versionsangabe ist ein konstanter, vierstelliger Wert (M2.0) der ab der 6. Position in der Satzkennung steht. An Position 10 findet man die Indikatorlänge als konstanter Wert 1. An 11. Stelle steht die Teilfeldkennungslänge. Dieser Wert ist auf 2 festgelegt. Die Datenanfangsadresse ist auf 00024 festgelegt und steht an Position 12-16 der Satzkennung. Die Positionen 17-22 sind in der Satzkennung nicht benutzt und werden mit Leerzeichen aufgefüllt. An 23. Stelle steht der Satztyp. Er beschreibt die Funktion des Datensatzes. Eine Auflistung der Satzstatusarten und der Satztypen befindet sich in [Die99]. In Abbildung 2.5 wird der Aufbau der Satzkennung anhand eines Beispieles erläutert. Dieser Datensatz besteht aus 240 Zeichen, wurde neu angelegt und ist ein Titeldatensatz.

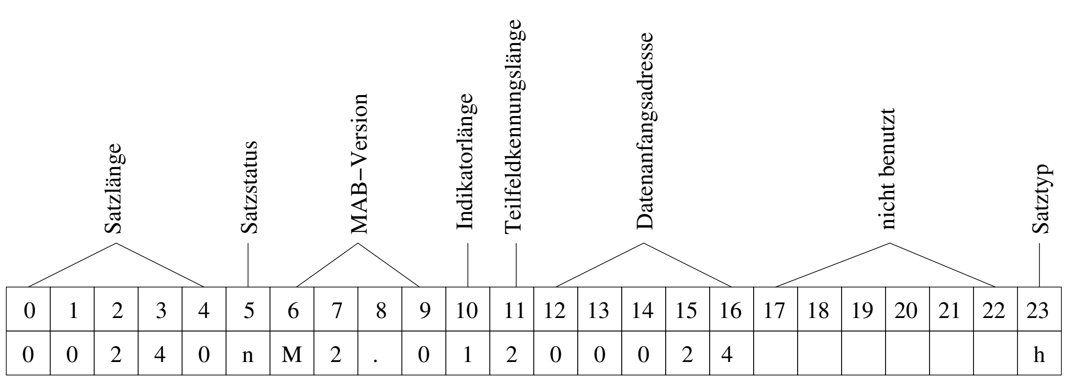

Abbildung 2.5: Beispiel für eine MAB2-Satzkennung

Der Aufbau der variablen Datenfelder ist für jedes oben beschriebene Datenformat identisch. In Tabelle 2.5 ist die Struktur dargestellt.

Feldnummer (3 byte)	Indikator (1 byte)	Daten (variabel)	Feldende (1 byte)

Tabelle 2.5: Aufbau der Datenfelder im MAB2-Format

Die Feldnummer definiert das Feld eindeutig in dem jeweiligen Kontext, d.h. das Datenfeld mit der Feldnummer 100 bezeichnet zum Beispiel im MAB-LOKAL die Signatur eines Buches und im MAB-TITEL den Name der 1. Person in Ansetzungsform. Im Kontext *MAB-TITEL* ist jede Feldnummer eindeutig einer Bedeutung zugeordnet. Die Definitionen der einzelnen Feldnummern sind in [Die99] aufgelistet. Der Indikator ist als ein einzelner kleiner Buchstabe definiert. Er soll den Feldinhalt näher charakterisieren. Das Datenfeld beinhaltet die durch die Feldnummer und Indikator spezifizierten Daten. Das Feldendesymbol ist in der MAB-Zeichentabelle [Die99] definiert und kennzeichnet das Ende eines Datenfeldes.

Als letztes Element eines MAB-Datensatzes steht das Satzendesymbol. Dieses Zeichen ist wie das Feldendesymbol in der MAB-Zeichentabelle definiert.

Kapitel 3

Entwurf von BiVIS

In diesem Kapitel wird der Aufbau von BiVIS beschrieben. Der Abschnitt 3.1 beschreibt im Überblick die Anforderungen an BiVIS. Eine detaillierte Beschreibung der Anforderungen befindet sich im Pflichtenheft (Anhang A). In Abschnitt 3.2 werden zwei Prozessmodelle beschrieben, die zur Auswahl für dieses Projekt standen. Es wird zudem begründet, welches Prozessmodell verwendet wird. Abschnitt 3.3 beschreibt grundlegende Architekturmuster von BiVIS. Die Vorgaben für die Einsatzumgebung des Produktes werden in Abschnitt 3.4 zusammengefasst. Im Abschnitt 3.5 wird der Aufbau der Datenbank beschrieben. Die funktionale Gliederung wird im Abschnitt 3.6 dargestellt. Mit der Aufteilung der funktionalen Blöcke in einzelnen Module und deren Struktur beschäftigt sich Abschnitt 3.7. Im Abschnitt 3.5 wird der Aufbau der Datenbank beschrieben.

3.1 Anforderungen an BiVIS

Mit BiVIS soll ein neues Bibliothekssystem entstehen, dass wartungsfreundlicher als das alte System ist. Durch die ständig wachsende Datenmenge muss die Datenhaltung an die neuen Bedürfnisse angepasst werden. Hierfür bietet es sich an ein Datenbanksystem zu verwenden. Die Migration der Daten sollte recht einfach sein und zum grossen Teil automatisch ablaufen. Für BiVIS ist eine graphische Benutzungsschnittstelle vorgesehen. BiVIS soll auch auf der „antiquierten“Hardware der Bibliothek laufen.

Die Funktionalität soll der des alten Systems weitestgehend entsprechen. Im Folgenden werden die funktionalen Anforderungen an BiVIS kurz beschrieben.

Titelaufnahme Sie ermöglicht die manuelle Aufnahme der Literaturdaten. Dies ist erforderlich, damit literarische Werke, die nicht bestellt wurden, inventarisiert werden können.

Datenabgleich um Daten konsistent zu halten. Autoren, Serien, Körperschaften und Verlage sind Daten, die bei unterschiedlichen Werken identisch sein können. Zur Vermeidung von Eingabefehlern soll während der Titelaufnahme ein Abgleich mit schon vorhandenen Daten erfolgen.

Bestellungen erfassen und drucken.

Lieferung der bestellten Werke überwachen und gegebenenfalls ausstehende Lieferungen anmahnen.

Zeitschriften bestellen und verwalten.

Kunden eintragen, bearbeiten und löschen.

Ausleihen erfassen, verlängern, austragen und ausstehende Ausleihen automatisch anmahnen.

Vorbestellungen zur Ausleihe aufnehmen und verwalten.

Datenexport nach MAB2 und ins alte Datenformat. Der Export ins alte Datenformat wird benötigt, um den Datenaustausch mit der Suchmaschine der Informatikbibliotheken in Saarbrücken (IBIS) zu gewährleisten.

Buchhaltung sie sollte Konten verwalten, Budgets für bestimmte Konten einrichten und überprüfen und vordefinierte Statistiken erstellen.

Mehrbenutzersystem Das gemeinsame Arbeiten an dem System mit mehreren Benutzern über ein Netzwerk soll möglich sein.

Zugangskontrolle Da nicht jeder Benutzer die gleichen Aufgaben in der Bibliothek hat, soll der Zugriff auf die einzelnen Daten und Programmteile beschränkt werden. Des Weiteren muss das System gegen unerlaubte Zugriffe aus dem Netzwerk geschützt werden.

Die detaillierten Anforderungen an BiVIS sind im Pflichtenheft (Anhang A) festgehalten.

3.2 Auswahl des Prozessmodells

In [Bal96b] und [Zel02] werden verschiedene Prozessmodelle zur Softwareentwicklung besprochen. Für BiVIS kamen zwei Prozessmodelle in Frage das Wasserfallmodell oder das inkrementelle Modell. Im Folgenden werden kurz die beiden Modelle beschrieben.

3.2.1 Das Wasserfallmodell

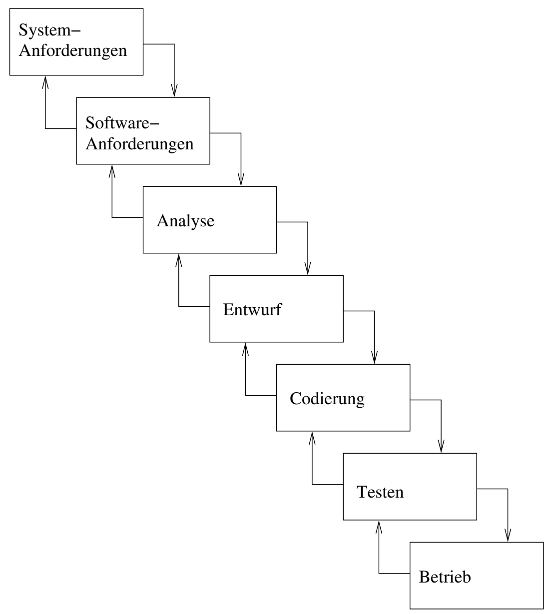

Abbildung 3.1: Das Wasserfallmodell mit Rückkopplung

Das Wasserfallmodell ist ein strikt sequentielles Modell, d.h. die einzelnen Phasen werden hintereinander ausgeführt. Eine Phase muss abgeschlossen sein, bevor die nächste Phase beginnen kann. Es gibt im Wasserfallmodell die Möglichkeit der Rückkopplung, d.h. wenn man in einer späteren Phase einen Fehler in der vorherigen Phase feststellt, kann man in die vorherige Phase zurückgehen und den Fehler korrigieren. Man muss dann allerdings die vorherige Phase wieder komplett abschliessen. In der Wirtschaft sind solche Rückkopplungen teuer. Generell gilt je weiter man im Wasserfall zurückgehen muss, desto teurer wird die Änderung. Dieses Modell ist in Abbildung 3.1 graphisch dargestellt. Das Wasserfallmodell bietet, ausser der Rückkopplung, keine Möglichkeit auf Änderungen der Anforderungen einzugehen. Diese Änderungen sind in der Softwaretechnik unter dem Schlagwort *Antizipation des Wandels* bekannt.

Am Ende einer jeden Phase im Wasserfallmodell wird ein Dokument erstellt. Dies kann dazu führen, dass die Dokumentation wichtiger ist, als das fertige Produkt.

Die Interaktion des Kunden erfolgt im Wasserfallmodell eigentlich nur in der Definitionsphase. Das setzt voraus, dass der Kunde genau weiß, was er will.

3.2.2 Das inkrementelle Modell

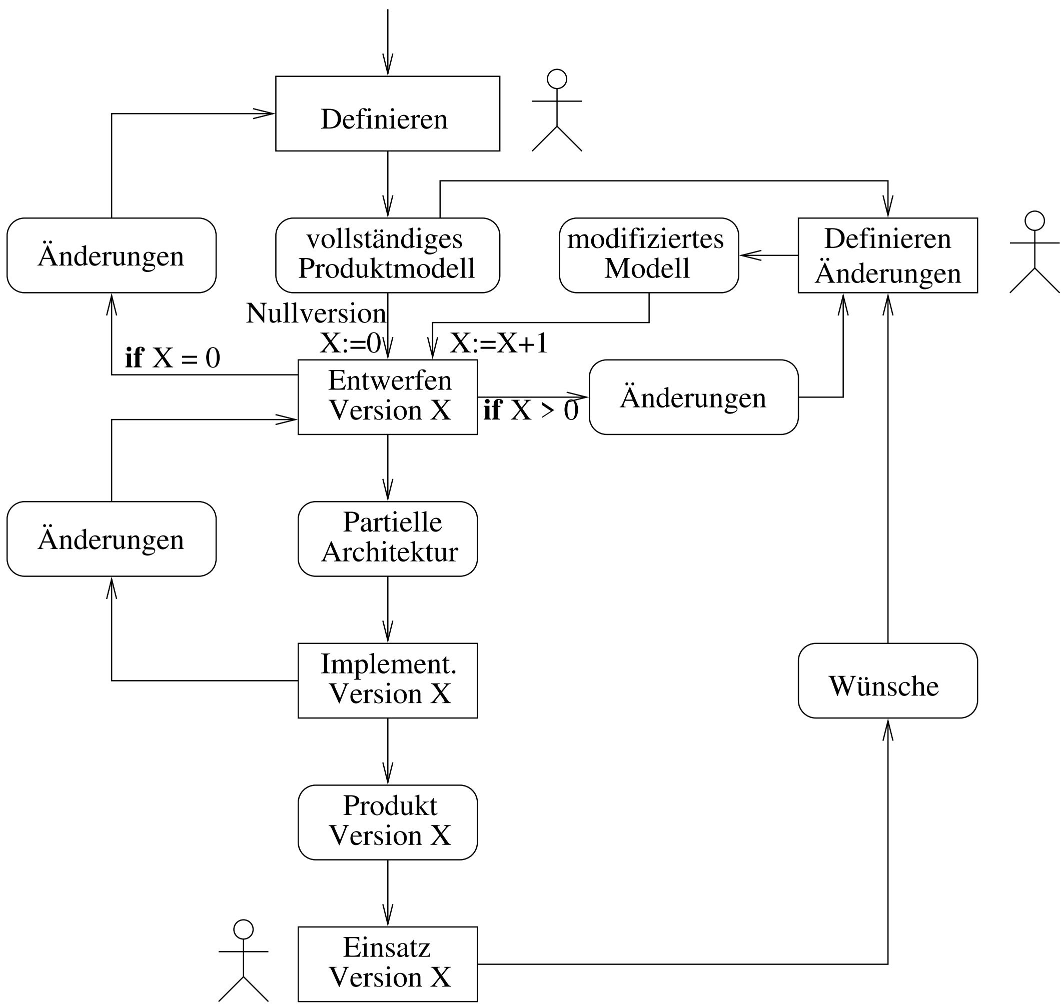

Abbildung 3.2: Das inkrementelle Modell

Im Gegensatz zum Wasserfallmodell wird beim inkrementellen Modell das Produkt nicht in gesamter Breite entwickelt. Es wird nur auf die Musskriterien des Kunden eingegangen und daraus eine sogenannte Nullversion entwickelt und implementiert. Diese Nullversion kommt dann beim Kunden zum Einsatz. Während dem Einstaz der Nullversion hat der Kunde die Möglichkeit Erfahrungen zu sammeln und aus diesen Erfahrungen Änderungen der Anforderungen für eine neue Version einzureichen. Die Anforderungen an das Produkt müssen trotzdem möglichst vollständig sein, damit man später nicht aufgrund einer Änderung der Anforderung das Kernsystem neu entwickeln muss. Die Abläufe und die Kunden-Interaktion in diesem Modell werden in Abbildung 3.2 verdeutlicht.

3.2.3 Verwendetes Modell in BiVIS

Auf Grund der Tatsache, dass das alte Bibliothekssystem IBBIB mit den Jahren gewachsen ist und ständig die Anforderungen an das System geändert wurden, habe ich mich für das inkrementelle Modell entschieden. Durch diese Änderungen an dem alten System wurde es sehr unübersichtlich und wartungsunfreundlich, das neben den Performance-Gründen der Hauptgrund war ein neues System zu entwickeln. Ein weitere Punkt für das inkrementelle Modell ist die Kundeninteraktion. In einem so grossen Projekt ist die kundennahe Entwicklung sehr wichtig.

3.3 Architekturmuster von BiVIS

Die Datenhaltung in einem Datenbanksystem und das Arbeiten über ein Netzwerk sind zwei wichtige Anforderungen, die das Design von BiVIS prägen. Aus diesen Anforderungen wird klar, dass eigentlich nur ein Architekturmuster in Frage kommt: das Client/Server Konzept.

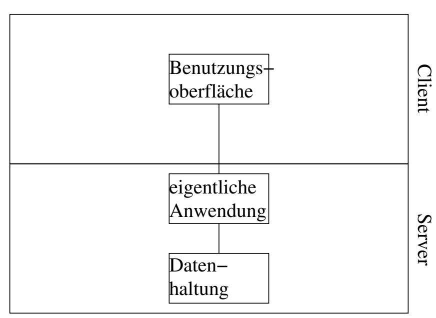

Abbildung 3.3: Client/Server Konzept von BiVIS

Ein Problem der Client/Server-Architektur ist die Aufteilung der Aufgaben auf den Client und den Server. Da der Client auf verschiedenen Plattformen mit unterschiedlicher Leistung laufen soll, kann man ihm nicht zu viele Aufgaben zuweisen. In BiVIS bekommt der Client nur die Aufgabe der Benutzungsoberfläche. Dieses Konzept nennt man *Entfernte Benutzungsoberfläche*.

Der Server bekommt somit die Aufgabe der eigentlichen Anwendung und der Datenhaltung. In Abbildung 3.3 wird dieses Konzept verdeutlicht. In Abschnitt 3.6 wird die funktionale Gliederung von BiVIS beschrieben. Auf Grund dieser Gliederung kann man das System in Module aufteilen. Als Architekturmuster der Module wird das *3-Schichten-Modell* verwendet. Dieses Modell bietet den Vorteil, dass man die einzelnen Schichten austauschen kann. Wenn z.B. später ein anderes Datenbanksystem (DBMS) eingesetzt werden soll, muss man nur die Datenhaltungsschicht auf das neue DBMS anpassen. Wie man aus Abbildung 3.4 ersehen kann, ist oberste Schicht die Darstellungsebene. Dort wird die Benutzungsschnittstelle realisiert. Die mittlere Schicht ist die Anwendungs- bzw. Logikschicht. Die unterste Schicht, die Datenhaltungsschicht, bildet die Schnittstelle zu den persistent gespeicherten Daten. Zwischen den einzelnen Schichten existieren Schnittstellen, über die die einzelnen Schichten kommunizieren.

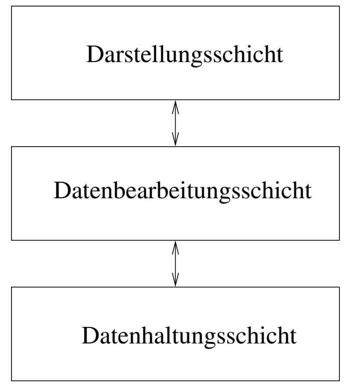

Abbildung 3.4: Das 3-Schichten-Modell der Module

3.4 Designvorgaben

Ein Software-Entwurf soll unabhängig von der verwendeten Programmiersprache und Hardware sein. Es ist allerdings notwendig die spätere Einsatzumgebung der Software zu spezifizieren. In diesem Abschnitt wird diese Einsatzumgebung festgelegt.

In Abschnitt 3.3 wird als Architekturmuster eine Client-Server-Architektur vorgeschlagen. Dieses Architekturmuster macht eine Aufteilung der Einsatzumgebung in einen Client-Teil und einen Server-Teil notwendig. Die Anforderung, dass das Arbeiten über ein Netzwerk möglich sein muss, erfordert eine Protokollspezifikation zwischen Client und Server auf der Anwendungsebene des OSI-Schichtenmodells. Es stellt sich nun die Frage, ob ein eigenes Protokoll entwickelt wird oder ob man auf ein bestehendes Protokoll zurückgreift. Die Entwicklung eines eigenen Protokolls ist sehr zeitaufwendig und fehleranfällig, wobei der Zeitfaktor die größere Rolle spielt. Als bestehendes Protokoll bietet sich das Hypertext-Transfer-Protokoll an, da es dafür schon fertig implementierte Clients und Server gibt.

Da über die Netzwerkverbindung zwischen Client und Server auch schützenswerte Daten, wie Passwörter der Benutzer, übertragen werden, sollte der Datenverkehr verschlüsselt werden. Die heutzutage verfügbaren Web-Server unterstützen alle das SSL-Protokoll zur sicheren Datenübertragung. SSL ist sehr weit verbreitet und bekannt. Es gibt kaum einen Browser der dieses Protokoll nicht unterstützt. Aus diesem Grund wird für BiVIS auch das SSL-Protokoll verwendet.

Um den Client so einfach wie möglich zu halten wird hier ein handelsüblicher Web-Browser verwendet. Somit besteht die Benutzungsschnittstelle aus Web-Seiten in HTML 3.0-Code.

Serverseitig wird ein Standard-Web-Server eingesetzt. Um die Anwendung zu programmieren bietet sich nun eine Serverside-Scripting-Language an.

Um hohe Lizenzkosten zu vermeiden und die vorhandene Hardware auszunutzen wird auf der Server-Seite ein UNIX-Betriebssystem vorausgesetz.

Das Speichern der persistenten Daten ist der nächste Aspekt, der die Einsatzumgebung mitbeschreibt. Das ständig steigende Datenvolumen erfordert den Einsatz eines Datenbanksystems. Dadurch muss der Server eine Datenbankschnittstelle bereitstellen. Zur Zeit gibt es zwei bekanntere Datenbankschnittstellen – SQL und ODBC. In der UNIX-Welt ist SQL weiter verbreitet als ODBC. Aus diesem Grund wird für BiVIS SQL als Datenbankschnittstelle verwendet.

Die Einsatzumgebung für BiVIS sieht also wie folgt aus:

- Software
  - Server
    - UNIX-Betriebssystem
    - Web-Server
    - Serverside-Scripting-Language
    - Datenbankschnittstelle (SQL)
  - Client
    - Web-Browser der HTML Version 3.0 interpretiert.
- Hardware
  - vorhandene Hardware (SUN Workstations)
  - günstige Standardhardware (z.B. PC's)

Im Kapitel 4 wird auf die Auswahl der verwendeten Software eingegangen.

3.5 Datenbankentwurf

In diesem Abschnitt werden der Aufbau und die Zusammenhänge der Datenbank erörtert und beschrieben. Die Zusammenhänge in der Datenbank werden mit Hilfe des Entity-Relationship-Modells (ER-Modell) erläutert. Eine Beschreibung zu diesem Modell findet man in dem Buch [AS00]. Die Struktur der einzelnen Tabellen wird erst im Kapitel 4 besprochen, da dafür das verwendete Database Management System (DBMS) festgelegt sein muss.

Zuerst muss erfasst werden, welche Daten das System persistent speichert. Aus den Anforderungen geht hervor, dass Daten über literarische Werke, Bestellungen, Benutzer, Kunden, Ausleihen und Konfiguration entstehen.

3.5.1 Literaturdaten

Die Daten für literarische Werke kann man in einen exemplarspezifischen und einen allgemeinen Teil aufteilen. Der exemplarspezifische Teil beinhaltet Angaben über Signatur, Inventarisierungsnummer, Besteller, Verfügbarkeitsstatus, usw. Das sind alles Angaben, die bei zwei Exemplaren desselben Buches unterschiedlich sein können bzw. müssen. Im allgemeinen Teil sind die Angaben über Titel, Autoren bzw. Editoren, Verlag, Erscheinungsjahr, ISBN, usw. gespeichert. Im Hinblick auf die Verbundfähigkeit des Systems wird der allgemeine Teil der Daten, die sogenannten *Stammdaten*, in einer gesonderte Datenbank gespeichert. Durch diese Trennung auf Datenbankebene kann man später die Stammdaten verteilt speichern. Der exemplarspezifische Teil, die *Exemplar- bzw. Lokaldaten*, der für jede Bibiliothek unterschiedlich ist, wird in einer lokalen Datenbank gehalten.

Der Datenabgleich von Autoren, Körperschaften, Serien und Verlagen verlangt eine Erweiterung der Stammdaten. Für jeden Typ des Datenabgleiches muss eine Tabelle angelegt werden, die die Daten speichert. In der eigentlichen Literaturstammdatentabelle wird nur noch eine Referenz auf die Datensätze dieser Tabellen gespeichert. Somit ist die Konsistenz der Daten gesichert.

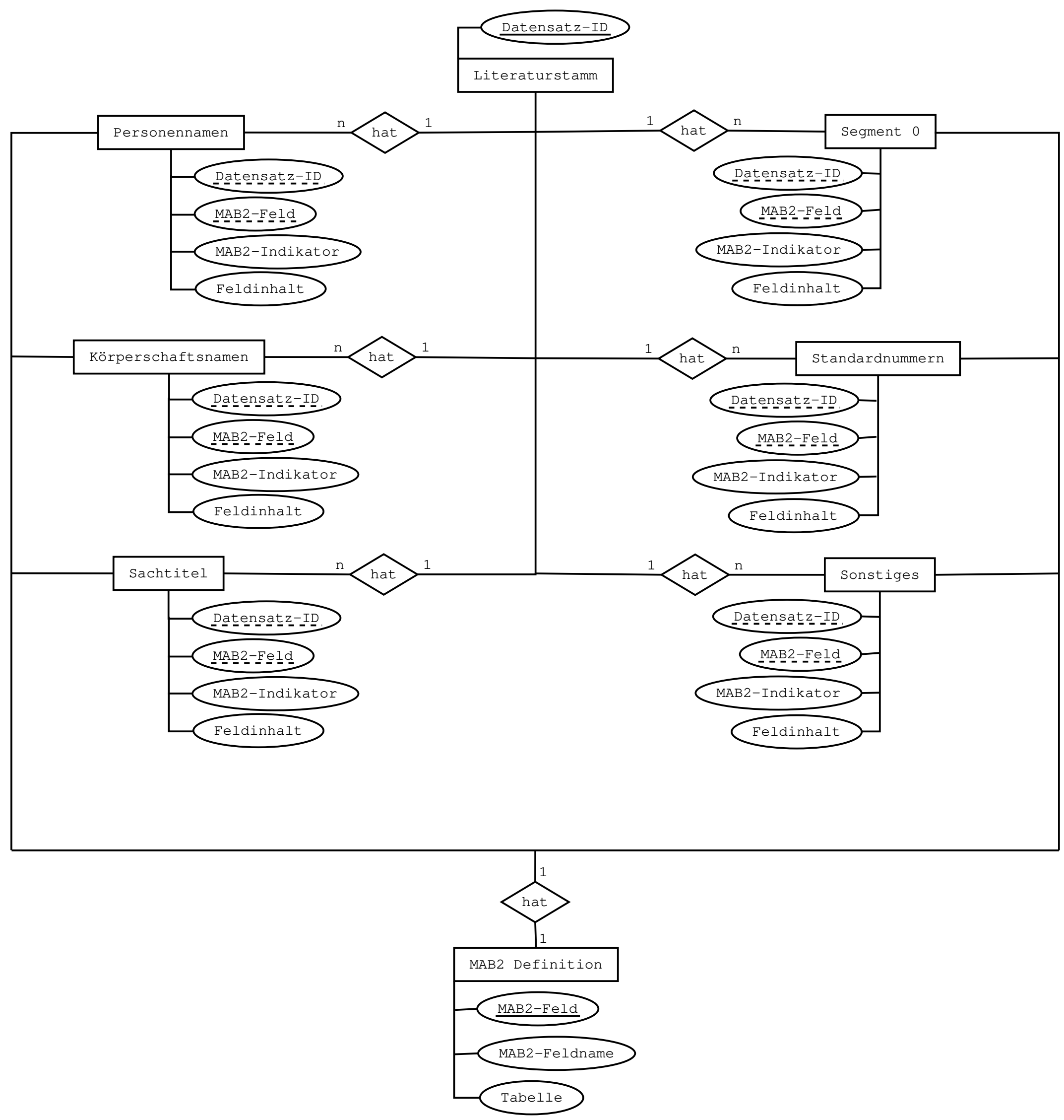

Abbildung 3.5: ER-Modell der Literaturstammdaten

Durch die Anforderung den Datenbestand als MAB2-Datei zu exportieren bietet es sich an die Literaturstammdaten im MAB2-Format zu speichern. Das Format für die Literaturstammdaten wird im Abschnitt MAB-TITEL des MAB2-Formates [Die99] beschrieben. Um zu vermeiden, dass bei einer Indizierung der Daten eine sehr große Indexmenge entsteht werden die Daten auf einzelne *Indizierungstabellen* verteilt. Diese Verteilung folgt aus dem Inhalt der einzelnen Datenfelder. So werden z.B. die Personennamen in eine eigene Tabelle geschrieben, die Titelangben in eine andere usw. Durch die Gliederung des MAB-TITEL-Formates ist die Aufteilung vorgegeben. Das Aufteilen der Daten auf Indizierungstabellen erleichtert auch die Suche nach

Büchern. Sucht man z.B. nach dem Autor eines Buches muss man nicht den gesamten Datenbestand durchsuchen, sondern es reicht aus die Tabelle mit den Personennamen zu durchsuchen. Durch die Einführung der Indizierungstabellen kann man leichter im laufenden Betrieb neue Indizierungstabellen hinzufügen und eigene Datenfelder definieren. Diese Zusammenhänge werden nochmals in Abblidung 3.5 verdeutlicht.

Die Aufteilung hat auch einen Nachteil. Um ein Buch eindeutig zu identifizieren muss der zugehörige Datensatz eine eindeutige Identifikationsnummer besitzen, die jedem Datenfeld zugeordnet sein muss.

Wie aus Abbildung 3.5 hervorgeht ist die Struktur der einzelnen Indizierungstabellen identisch. Dadurch entsteht ein weiteres Problem. Das Datenfeld `Feldinhalt` kann sowohl Text enthalten, als auch eine Referenz auf einen Autoren-, Verlag-, Körperschaft- oder Seriendatensatz. Zur Wahrung der referenziellen Integrität kann man deshalb keine Fremdschlüssel verwenden. Die referenzielle Integrität muss durch die Anwendung gewährleistet werden.

Für die Lokaldaten ist die referenzielle Integrität zu den Literaturstammdaten relativ wichtig. Um eine manuelle Auflösung der referenziellen Integrität zu vermeiden, wird nicht das MAB-LOKAL-Format als Datenstruktur verwandt. Für ein Exemplar eines Buches müssen folgende Daten gespeichert werden:

- eine eindeutige Identifikationsnummer (ID) `ExemplarID`
- die Signatur `Signatur`
- der Standort `Standort`
- die Inventarisierungsnummer `Invnummer`
- der Besteller `BestellerID`
- ein Kommentarfeld `Kommentar`
- der Lieferant `LieferantID`
- der Verfügbarkeitsstatus `Status`
- der Verweis auf den Literaturstammdatensatz `StammID`

Um Redundanzen in den Daten zu vermeiden, werden Besteller und Lieferant als Referenz gespeichert. Das erfordert zwei zusätzliche Tabellen für Lieferanten und Besteller. In der Lieferantentabelle werden Angaben über

- den Namen
- die Anschrift
- den Ansprechpartner
- die Sprache

gespeichert. Als Angaben über den Besteller genügen der Name und der Lehrstuhl. Die Zusammenhänge der drei Tabellen werden im ER-Diagramm (Abbildung 3.6) verdeutlicht.

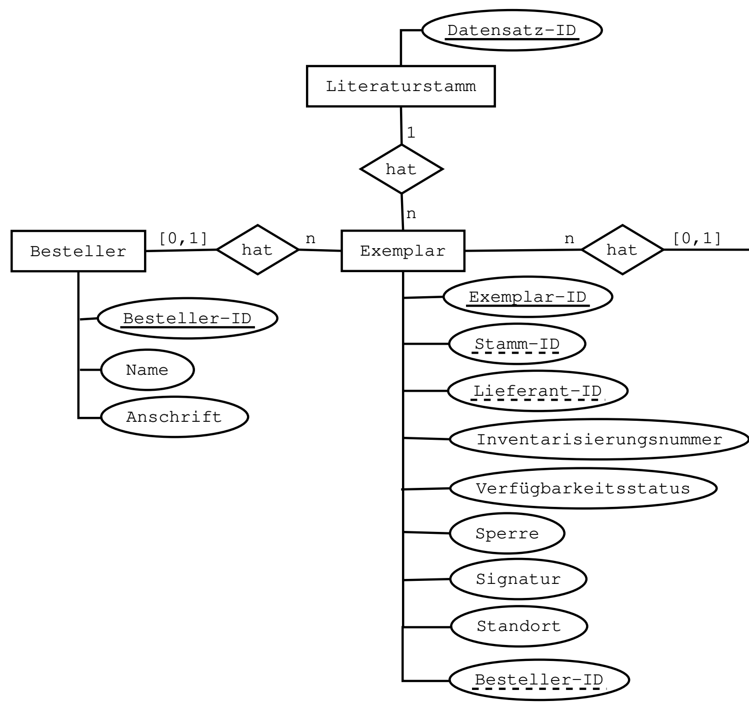

Abbildung 3.6: ER-Modell der Exemplardaten

Die bei der Bestellung von Büchern und Zeitschriften anfallenden Daten sind:

- Referenz zu einem Exemplardatensatz
- Bestelldatum
- Lieferdatum
- Mahndatum

Die Angaben zum Lieferanten und Besteller sind in den Exemplardaten gespeichert. Das Zusammenspiel der Tabellen ist in Abbildung 3.7 dargestellt.

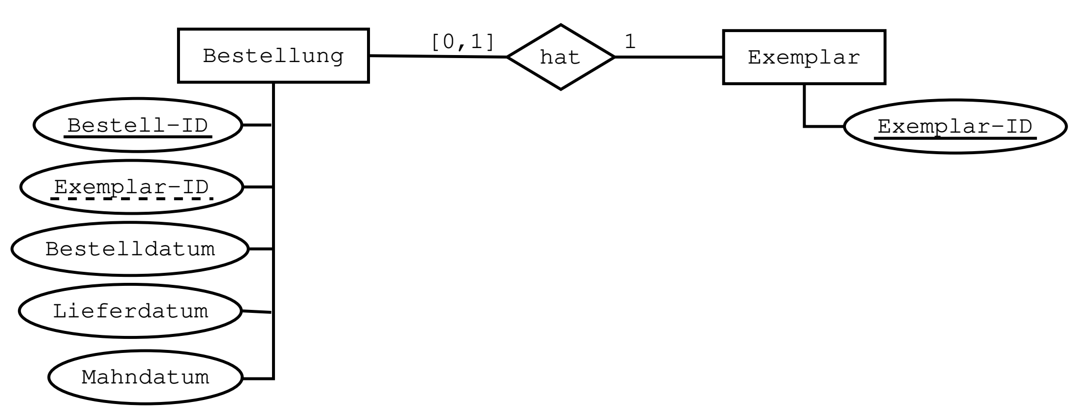

Abbildung 3.7: ER-Modell der Bestellungen

Die Zeitschriften werden in BiVIS gesondert behandelt. Die Stammdaten einer Zeitschrift werden in den Literaturstammdaten gespeichert. Für gebundenen Hefte einer laufenden Zeitschrift wird pro Band ein Exemplardatensatz erzeugt. Zusätzlich wird für den aktuellen Band ein Datensatz erzeugt, indem fehlende Hefte und der Hefteingang erfasst werden. Neben den Literaturstammdaten und den Exemplardaten fallen folgende Daten an, die in einer Tabelle gespeichert werden.

- fehlende Hefte
- neue Hefte (Hefteingang)
- Bestandsnachweis

In den Literaturstammdaten muss zusätzlich die Erscheinungsweise gespeichert werden. Dafür ist allerdings keine Änderung der Literaturstammdaten notwendig, da das MAB2-TITEL-Format dies unterstützt. Der Zusammenhang von Zeitschriften, Exemplardate und Literaturstammdaten wird in Abbildung 3.8 gezeigt.

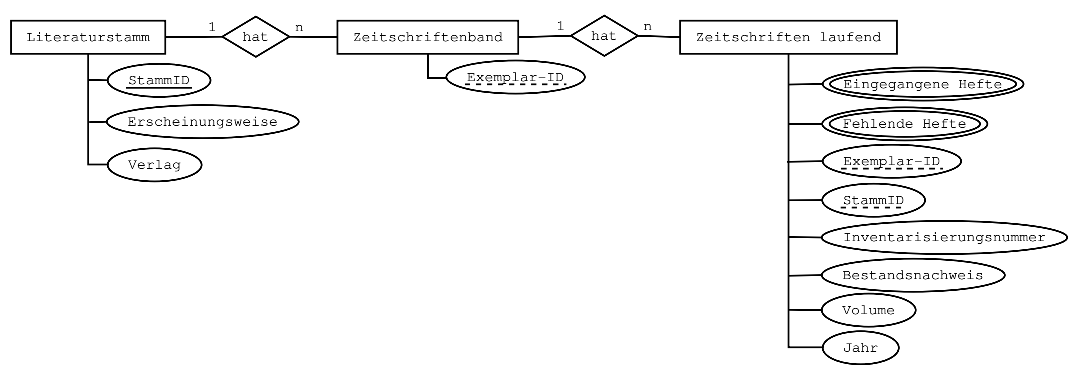

Abbildung 3.8: ER-Modell der Zeitschriften

3.5.2 Kunden und Ausleihe

Nachdem die anfallende Daten der Literaturverwaltung und deren Zusammenhänge gezeigt wurden, wird in diesem Abschnitt die Ausleihverbuchung untersucht. Für die Ausleihverbuchung werden Daten über die Kunden der Bibliothek, die Ausleihen und die Vorbestellungen benötigt. Die Daten über die Kunden umfassen:

- die Kundenummer
- der Name
- das Geschlecht
- die Titel
- die Anrede
- die Anschrift
- die Telfonnummern (privat, dienstlich)

- die Dienstanschrift
- eine E-Mail-Adresse (zum automatischen Anmahnen)
- die Zugangskartennummer
- die Ausleihdauer
- die Gültigkeit der Ausleihberechtigung (Enddatum und/oder Ausleihberechtigungsdauer)

In der Tabelle für die Ausleihen werden

- Referenz auf eine Buchexemplar
- Referenz auf einen Kunden
- Ausleihdatum
- Ausleihendedatum
- Mahnstufe
- Mahndatum

gespeichert. Es wird für jedes verliehene Buch ein Datensatz in dieser Tabelle erzeugt.

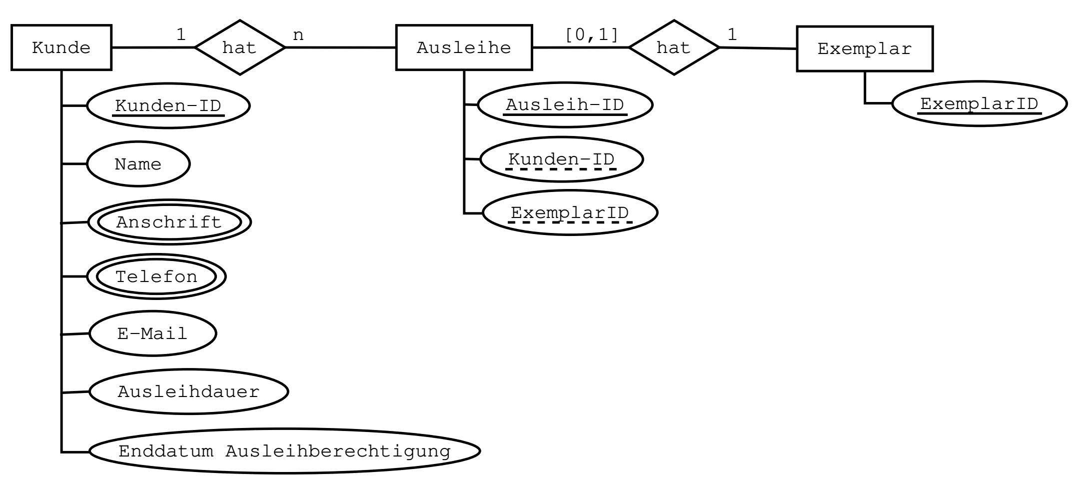

Abbildung 3.9: ER-Modell der Ausleihen

Sollte ein Kunde sich für ein Buch interessieren, dass aber verliehen ist, so kann er das Buch vorbestellen. Bei dieser Vorbestellung fallen wiederum Daten an, die gespeichert werden müssen:

- Referenz auf ein Buchexemplar
- Referenz auf einen Kunden
- Datum der Vorbestellung

- Benachrichtigungsdatum

Das Benachrichtigungsdatum wird bei der Vorbestellung auf einen NULL-Wert gesetzt. Erst wenn der Kunde über die Rückgabe des Buches benachrichtigt wurde, wird dort ein gültiges Datum eingetragen. Ein vorbestelltes Buch kann nur an den Kunden, der das Buch vorbestellt hat, ausgeliehen werden.

3.5.3 Buchhaltung

Aus den Anforderungen an die Buchhaltung (siehe 3.1 und A.5.4) geht hervor, dass Hauptkonten, Unterkonten und Buchungen als Daten gespeichert werden. Es müssen nun zuerst die Bedeutung der Begriffe *Hauptkonto* und *Unterkonto* festgelegt werden. Ein `Hauptkont` ist ein Konto, dass keine übergeordneten Konten besitzt. Ein *Unterkonto* ist ein Konto, dass zur Gruppierung von Buchungen verwendet wird und einem übergeordneten Konto zugeordnet ist. Diese übergeordnete Konto kann ein anderes Unterkonto sein oder ein Hauptkonto. Diese hierarchische Struktur kann man als Baum darstellen. Das Hauptkonto bildet die Wurzel des Baumes. Die Unterkonten sind die Knoten und an den Blättern residieren die Buchungen. Die Wurzel eines Baumes ist im Allgemeinen ein Knoten ohne Vaterknoten. Aus dieser Überlegung heraus folgt, dass man in der Datenstruktur zwischen Hauptkonto und Unterkonto keine Unterscheidung benötigt. In der Datenbank von BiVIS wird ein Hauptkonto als Konto definiert, das kein übergeordnetes Konto hat.

Für die Konten müssen folgende Daten gespeichert werden:

- Kontonummer
- Kontoname
- Referenz auf übergeordnetes Konto (leer falls Hauptkonto)
- Saldo
- Budget
- Buchungsstelle

Die Buchungen sind entweder einer Literaturbestellung zugeordnet oder stammen von Materialkäufen. Dies macht aber keinen großen Unterschied. Es muss lediglich beachtet werden, dass die Möglichkeit besteht einer Buchung einen Bestelldatensatz zuzuordnen. Die Daten die für einen Buchung gespeichert werden müssen sind:

- Verweis auf zugehöriges Konto
- Wert der Buchung
- Zuordnung zu einem Exemplardatensatz
- Zuordnung zu einer Bestellung
- Rechnungsnummer

- Buchungsnummer
- Beschreibung
- Kommentar
- Typ (Einnahme/Ausgabe)
- Abgleich-Flag
- Datum der Erstellung
- Datum des Rechnungseingangs
- Datum der Überweisung/Abgleichung

Im ER-Model der Buchhaltung (Abbildiung 3.10) werden nochmals die Zusammenhänge zwischen Buchung und Konto verdeutlicht.

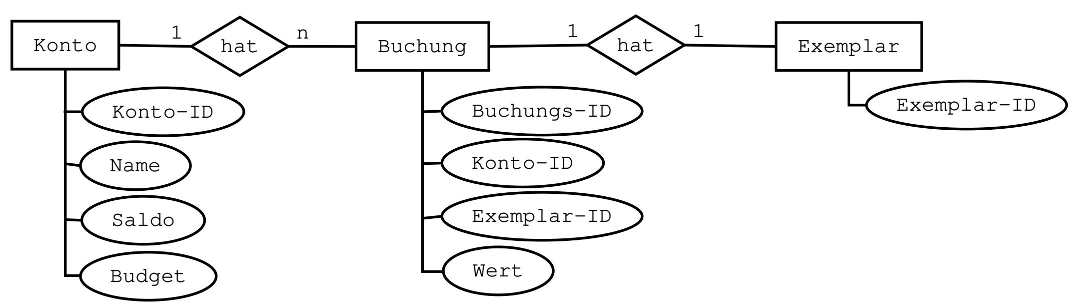

Abbildung 3.10: ER-Modell der Buchhaltung

3.5.4 Konfiguration/Administration

Bei der Konfiguration fallen Daten an, die eigentlich keinen Einfluss auf das Zusammenspiel der Tabellen nehmen. Eine Ausnahme ist die Beschreibung der MAB2-Felder. In dieser Tabelle werden die MAB2-Feldnummern auf die Indizierungstabellen abgebildet. Aus dieser Aufgabe geht direkt hervor, welche Daten in einem solchen Datensatz gespeichert werden:

- MAB2-Feldnummer
- Kurzbeschreibung
- Name der Indizierungstabelle

Da die MAB2-Beschreibungstabelle nur von den Literaturstammdaten benötigt wird, wird sie zusammen mit den Literaturstammdaten in der Stammdatenbank gespeichert. Für die Mehrbenutzerfähigkeit von BiVIS müssen auch Benutzer angelegt werden. Die minimalen Anforderungen an die gespeicherten Daten in einem Benutzerdatensatz sind:

- Login-Name

- Benutzername
- Passwort

Darüber hinaus werden noch weitere persönliche Angaben wie Telefonnummer, Anschrift usw. gespeichert. Die Zugriffssteuerung der einzelnen Benutzer erfordert eine Tabelle in der Datenbank, in der Paare von Funktionalitäten und Benutzer gespeichert werden. Nur wenn für einen Benutzer und eine Funktionalität ein solches Paar existiert, darf der Benutzer diese Funktionalität benutzen.

In einer separaten Konfigurationstabelle werden die Systemeinstellungen gespeichert. Darunter fallen z.B.

- der Druckbefehl
- Pfad zum temporären Verzeichnis
- usw.

3.5.5 Zusammenfassung

In diesem Abschnitt wird der oben diskutierte Entwurf zusammengefasst. Der Entwurf umfasst zwei Datenbanken, eine Stammdatenbank und eine Lokaldatenbank. In der Stammdatenbank werden die Literaturstammdaten und die MAB2-Beschreibungstabelle gespeichert. In der Lokaldatenbank werden die bibliotheksspezifischen Daten gespeichert. Die Verteilung der Tabellen wird in der folgenden Aufzählung aufgelistet.

- Stammdatenbank
  - Literaturstammdaten
    - Indizierungstabellen
    - Autoren
    - Verlage
    - Körperschaften
    - Serientitel
  - MAB2-Beschreibungstabelle
- Lokaldatenbank
  - Exemplardaten
  - Zeitschriften
  - Kunden
  - Ausleihe
  - Vorbestellung
  - Konten
  - Buchungen
  - Benutzer
  - Zugriffssteuerung
  - Konfiguration

3.6 Funktionale Gliederung von BiVIS

Man kann BiVIS anhand der Funktionalität in verschiedene Teile gliedern. Diese Aufteilung ermöglicht einen modularen Aufbau des Systems. Die funktionalen Anforderungen kann man in fünf Module zusammenfassen. Alle Anforderungen, die mit der Erfassung und Verwaltung von Büchern, Zeitschriften und anderen literarischen Werken zu tun haben, kann man in dem ersten Modul, die *Literaturverwaltung* zusammenfassen. Das Erfassen von Kunden und das Verwalten von Kundendaten wird in dem zweiten Modul, der *Kundenverwaltung*, gekapselt. Das dritte Modul, die *Ausleihverbuchung* umfasst die Ausleihe und die Vorbestellung von Büchern. Die *Buchhaltung* bildet das vierte Modul. Die Konfiguration des Systems und die Benutzerverwaltung werden in einem fünften Modul zusammengefasst. Dieses fünfte Modul bildet das *Basismodul* von BiVIS.

3.7 Struktur der Module

Die Grundstruktur ist für jedes Modul gleich, das 3-Schichten Modell. Aus den Vorgaben (Abschnitt 3.4) ist ersichtlich, dass in der Darstellungsschicht die Web-Seiten erzeugt werden. Die Datenhaltungsschicht erzeugt die SQL-Queries an das Datenbanksystem. Zusammen mit dem Client-Server-Konzept wird diese Struktur in Abbildung 3.11 verdeutlicht.

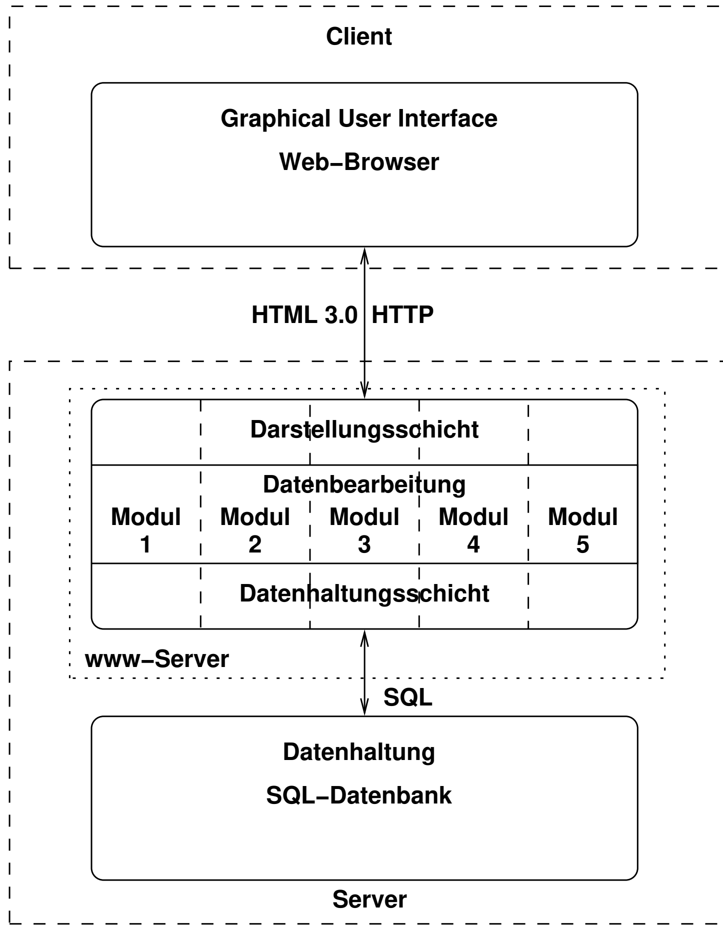

Abbildung 3.11: Die Grundstruktur von BiVIS: Client-Server mit entfernter Benutzungsschnittstelle und 3-Schichten-Modell im Anwendungsbereich

In den folgenden Abschnitten wird der Aufbau der einzelnen Schichten der Module gezeigt. Die Schnittstelle zwischen der Datenhaltungschicht und der Datenbearbeitungsschicht wird als *UML-Diagramm* dargestellt. In der Datenhaltungsschicht und in der Datenbearbeitungsschicht werden die Daten als Objekte behandelt. Die Datenbearbeitungsschicht stellt der Darstellungsschicht eine rein funktionale Schnittstelle zur Verfügung. Um die Vollständigkeit der Schnittstellen zu zeigen werden die Funktionnummern (**/Fxxxx/**) aus dem Pflichtenheft (Anhang A) den Methoden und Funktionen der Schichten zugeordnet.

3.7.1 Modul Literaturverwaltung

Die Literaturverwaltung umfasst die Titelaufnahme, die Zeitschriftenverwaltung, den Datenexport und die Literatursuche. Die Titelaufnahme ist unterteilt in Schenkung, Bestellung, Stammdatenbearbeitung und Exemplardatenbearbeitung. Die Objekte der Datenhaltungsschicht definieren sich direkt aus den persistent zu speichernden Daten.

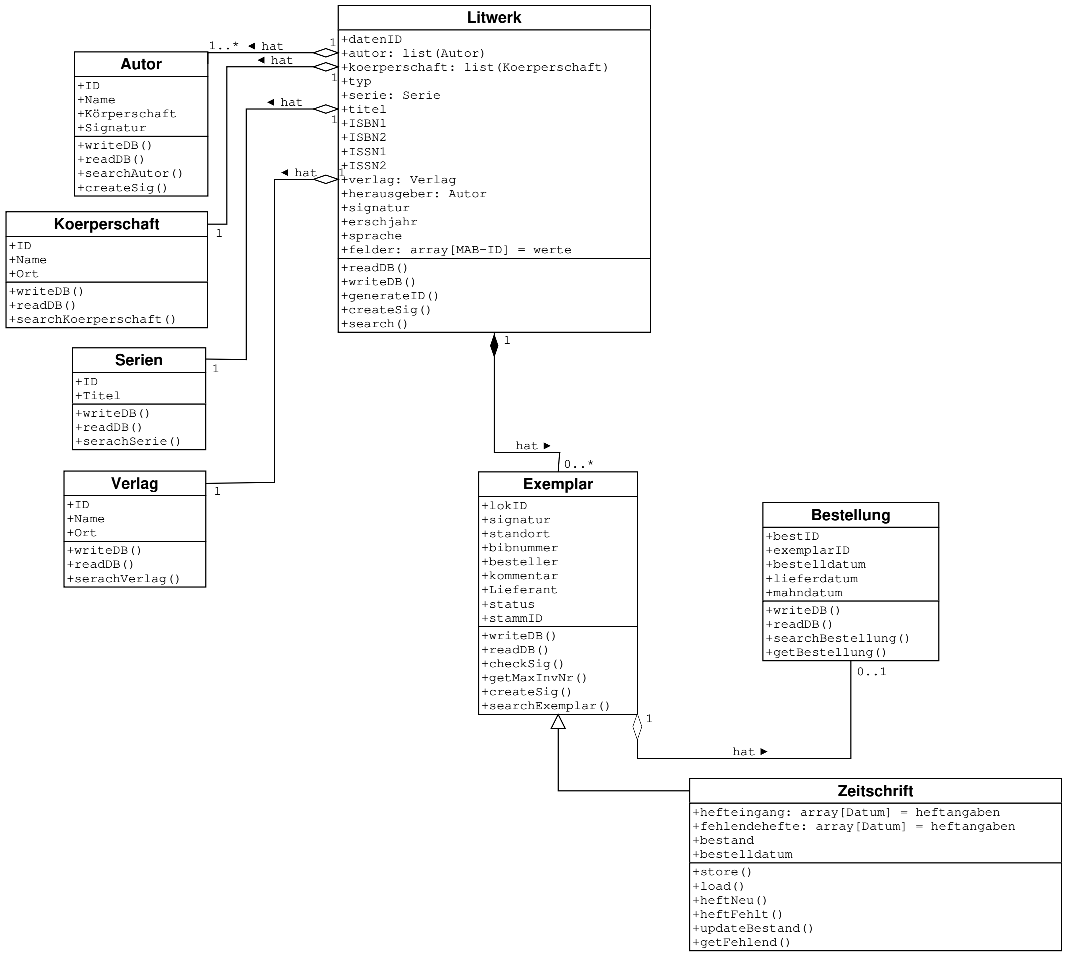

Abbildung 3.12: UML-Diagramm der Literaturverwaltung

Wie aus Abbildung 3.12 hervorgeht existieren zwischen den Klassen der Datenhaltungsschicht Abhängigkeiten. Als zentrale Klasse steht die Klasse `Litwerk`. Sie beinhaltet die literarischen Stammdaten. Jedes Objekt dieser Klasse kann auf ein oder mehrere Objekte der Klasse `Autor` verweisen. Das wird im UML-Diagramm durch die Aggregation zwischen den Klassen `Litwerk` und `Autor` dargestellt. Des Weiteren verweist ein Objekt der Klasse `Litwerk` auf maximal ein Objekt der Klasse `Koerperschaft`, auf maximal ein Objekt der Klasse `Verlag` und auf maximal ein Objekt der Klasse `Serie`. Über diese Struktur wird der geforderte Datenabgleich durchgeführt.

Die Komposition zwischen den Klassen `Litwerk` und `Exemplar` besagt, dass zu jedem Objekt der Klasse `Litwerk` mehrere Objekte der Klasse `Exemplar` auftreten können. Jedoch darf zu einem Objekt der Klasse `Exemplar` nur genau ein Objekt der Klasse `Litwerk` in Verbindung stehen. Diese Beschränkung wird direkt klar, wenn man sich die semantische Bedeutung dieser Komposition überlegt: zu jedem literarischen Stammdatensatz können mehrerer Exemplare existieren, jedoch hat jedes Exemplar nur genau einen Stammdatensatz zu dem es gehört. Ein Exemplar kann also nicht ohne einen Stammdatensatz existieren, jedoch umgekehrt ist es möglich.

Die Klasse `Zeitschriften` ist eine Spezialisierung der Klasse `Exemplar`. Sie erweitert die Exemplarklasse um die für die Zeitschriftenverwaltung nötigen Attribute und Methoden.

Die Bestellung eines Buches ist exemplarbezogen, d.h. zu jeder Bestellung gibt es ein zugehöriges Exemplar. Dies wird im UML-Digramm durch eine 1:1-Komposition zwischen Objekten der Klasse `Bestellung` und den Objekt der Klasse `Exemplar` dargestellt. Allerding muss nicht zu jedem Exemplar eine Bestellung existieren, da das Exemplar auch eine Schenkung an die Bibliothek sein kann.

Die Schnittstelle zwischen Datenhaltungsschicht und Datenbearbeitungsschicht sind die Klassen der Datenhaltungsschicht mit ihren Methoden. Die Methoden sind aus Abbildung 3.12 ersichtlich.

Die Datenbearbeitungsschicht bildet die funktionale Schnittstelle zur Darstellungsschicht. In der Datenbearbeitungsschicht sind folgende Schnittstellenfunktionen definiert:

`create_stamm()` erzeugt einen neuen Stammdatensatz und erledigt den Datenabgleich. Diese Funktion wird auch verwendet um geänderte Stammdaten zu speichern. (/F1100/, /F1120/, /F1140/, /F1180/, /F1220/, /F1260/)

`suche_stamm()` ermöglicht die Suche nach Stammdaten. Als Suchpattern wird die ISBN oder der Autor und der Titel des Buches verwendet. Sie realisiert /F1130/

`select_stamm()` wählt den Stammdatensatz zu der übergebenen StammID aus.

`delete_stamm()` löscht den übergebenen Stammdatensatz, falls Integrität nicht verletzt (/F1110/)

`create_exemplar()` realisiert /F1300/ und /F1320/.

`select_exemplar()` wählt den Exemplardatensatz zu der übergebenen ExemplarID aus.

`delete_exemplar()` löscht das übergebene Exemplar, falls Integrität nicht verletzt (/F1310/)

`set_status()` setzt den Verfügbarkeitsstatus des übergebenen Exemplares. (/F1350/)

`create_ex_sig()` generiert die Exemplar-Signatur /F1330/

`create_autor()` erzeugt und ändert Autor-Stammdaten (/F1150/, /F1170/)

`delete_autor()` löscht Autorstammdaten falls Integrität nicht verletzt (/F1160/)

`get_autor()` liest einen Autor-Stamm-Datensatz

`create_koerper()` erzeugt und ändert Körperschaftsstammdaten (/F1190/, /F1210/)

`delete_koerper()` löscht Körperschaftsstammdaten falls Integrität nicht verletzt(/F1200/)

`get_koerper()` liest Körperschaftsstammdaten

`create_verlag()` erzeugt und ändert Verlagsstammdaten (/F1230/, /F1250/)

`delete_verlag()` löscht Verlagsstammdaten, falls Integrität nicht verletzt(/F1240/)

`get_verlag()` liest Verlagsstammdaten

`create_serie()` erzeugt und ändert Serienstammdaten (/F1270/, /F1290/)

`delete_serie()` löscht Serienstammdaten, falls Integrität nicht verletzt(/F1280/)

`get_serie()` liest Serienstammdaten

`export_MAB()` exportiert die Literaturdaten nach MAB2 (/F1360/)

`print_sigetikett()` druckt Signatur Etiketten (/F1331/)

`print_bcetikett()` druckt Barcode Etiketten (/F1331/)

`check_zeitlief()` überprüft ausstehende Lieferungen von Zeitschriften (/F1500/)

`mahn_zeitschr()` druckt eine Liste der fehlenden Zeitschriften (/F1510/)

`archiv_zeitschr()` archiviert komplette Zeitschriftenbände (/F1520/)

`best_buch()` bestellt ein Buch (/F1700/)

`best_pruef()` überprüft ausstehende Lieferungen und gibt eine Liste zurück (/F1340/, /F1730/, /F1740/)

`drucke_best()` druckt Bestellungen (/F1710/)

3.7.2 Kundenverwaltung

In der Datenhaltungsschicht wird nur eine einzige Klasse benötigt, die Klasse `Kunde`. Sie repräsentiert einen Datensatz der Kundentabelle und stellt Methoden zur Manipulation der Daten bereit. Die Struktur der Klasse wird aus dem UML-Diagramm zur Ausleihverbuchung (Abbildung 3.13) ersichtlich.

Die Datenbearbeitungsschicht stellt der Darstellungschicht folgende Schnittstellenfunktionen bereit:

`store_kunde()` speichert und ändert Kundendaten (/F2100/, /F2120/, /F2130/, /F2140/)

`delete_kunde()` löscht Kundendaten (/F2110/)

`get_kunde()` liest Kundendaten

`suche_kunde()` sucht Kundendaten

`drucke_kukart()` druckt Karteikarte (/F2151/)

`get_zugang()` gibt Zugangsliste zurück (/F2152/)

3.7.3 Ausleihverbuchung

Das Modul der Ausleihverbuchung ermöglicht die Ausleihen eines Kunden zu verwalten. Dazu erfasst es die ausgeliehenen Bücher eines Kunden. Dieses Modul bietet zusätzlich noch die Möglichkeit verliehene Bücher vorzubestellen. Die Daten, die von diesem Modul gespeichert werden liefern direkt die beiden Klassen der Datenhaltungsschicht. Es wird eine Klasse für die Ausleihen und eine für die Vorbestellungen benötigt. Die Klasse für die Kunden wird aus der Datenhaltungsschicht der Kundenverwaltung importiert. Die Klasse der Exemplare wird aus der

Literaturverwaltung übernommen. Hieraus wird deutlich, dass das Ausleihmodul nicht ohne das Kunden- und Literaturverwaltungsmodul benutzt werden kann.

Jeder Ausleihe wird genau ein Kunde und ein Buch zugeordnet. Kunden und Bücher können alleine existieren, Ausleihen jedoch nicht. Um dieses zu verdeutlichen wird im UML-Diagramm (Abbildung 3.13) eine Komposition verwendet. Das gleiche gilt auch für die Vorbestellungen. In dem Diagramm ist die Exemplarklasse nicht mehr in der vollständigen Struktur dargestellt. Die Attribute und Methoden der Klasse `Ausleihe` und `Vorbestellung` sind aus dem UML-Diagramm ersichtlich.

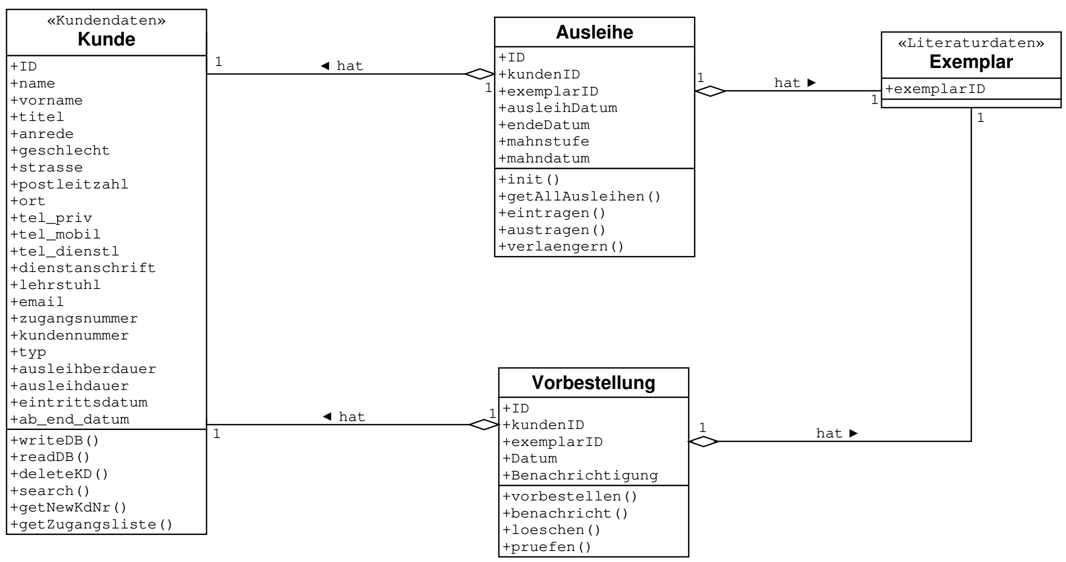

Abbildung 3.13: UML-Diagramm der Ausleihverbuchung

Folgende Schnittstellenfunktionen stellt die Datenbearbeitungsschicht der Darstellungsschicht bereit:

`buch_eintragen()` ermöglicht es, Bücher in die Ausleihe einzutragen. Diese Funktion wird auch benutzt um Kurzausleihen einzutragen. (/F3100/, /F3130/)

`drucke_karte()` druckt die Ausleihzettel, die in die Kundenkartei eingestellt werden (/F3101/)

`verlaenger()` verlängert ein Buch, falls es nicht vorbestellt ist (/F3110/)

`buch_austragen()` schliesst eine Ausleihe bei Rückgabe des Buches ab. (/F3120/, /F3140/, /F3160/)

`vorbest_eintragen()` trägt eine Vorbestellung ein (/F3150/)

`vorbest_loeschen()` löscht eine Vorbestellung (/F3170/)

`ben_kunde()` benachrichtigt einen Kunden, dass die Vorbestellung eingetroffen ist (/F3160/)

`calc_enddat()` berechnet das Ausleihendedatum anhand der Kundenangabe und den Öffnungstagen der Bibliothek `teste_datum()` überprüft, ob das übergebenes Datum ein Öffnungstag ist

`schliess_tage()` liefert eine Liste der Schliesstage (Feiertage werden automatisch berechnet)

`get_ausleihe()` liest eine Ausleihe anhand der Exemplar-Signatur

`get_ausleihen()` liest alle Ausleihen eines Kunden

`check_ende()` überprüft Ausleihende

`check_buch()` überprüft, ob ein Buch zur Ausleihe zur Verfügung steht

`check_buch_vb()` überprüft, ob ein Buch zur Vorbestellung zur Verfügung steht

`check_kunde()` überprüft, ob der Kunde ausleihberechtigt ist

`check_vb()` überprüft, ob ein Buch vorbestellt ist

`get_vorbestellung()` liest eine Vorbestellung anhand der Exemplar-Signatur

`get_vorbestellungen()` liest sämtliche Vorbestellungen eines Kunden

3.7.4 Buchhaltung

Die Buchhaltung ist teilweise abhängig von der Literaturverwaltung. Sie benötigt für die Abrechnung der bestellten Bücher Angaben der Bestellung und des Buches. Es können allerdings auch Buchungen existieren, die nicht von Buchbestellungen kommen. Das sind unter anderem Rechnungen von Materialbestellungen oder Spenden an die Bibliothek. Dieser Zusammenhang wird in Abbildung 3.14 durch eine Aggregation dargestellt. Um später eine statistische Auswertung der Buchungen zu machen, müssen diese gegliedert werden. Diese Gliederung wird durch Konten realisiert. Konten können ohne Buchungen existieren, jedoch muss jede Buchung genau einem Konto zugeordnet sein. In Abbildung 3.14 wird das durch eine Komposition der Klasse `Konto` mit der Klasse `Buchung` dargestellt. Die Struktur der Klassen in der Datenhaltungsschicht mit ihren Attributen und Methoden ist in Abbildung 3.14 beschrieben. Die Klassen `Exemplar` und `Bestellung` stammen aus der Literaturverwaltung.

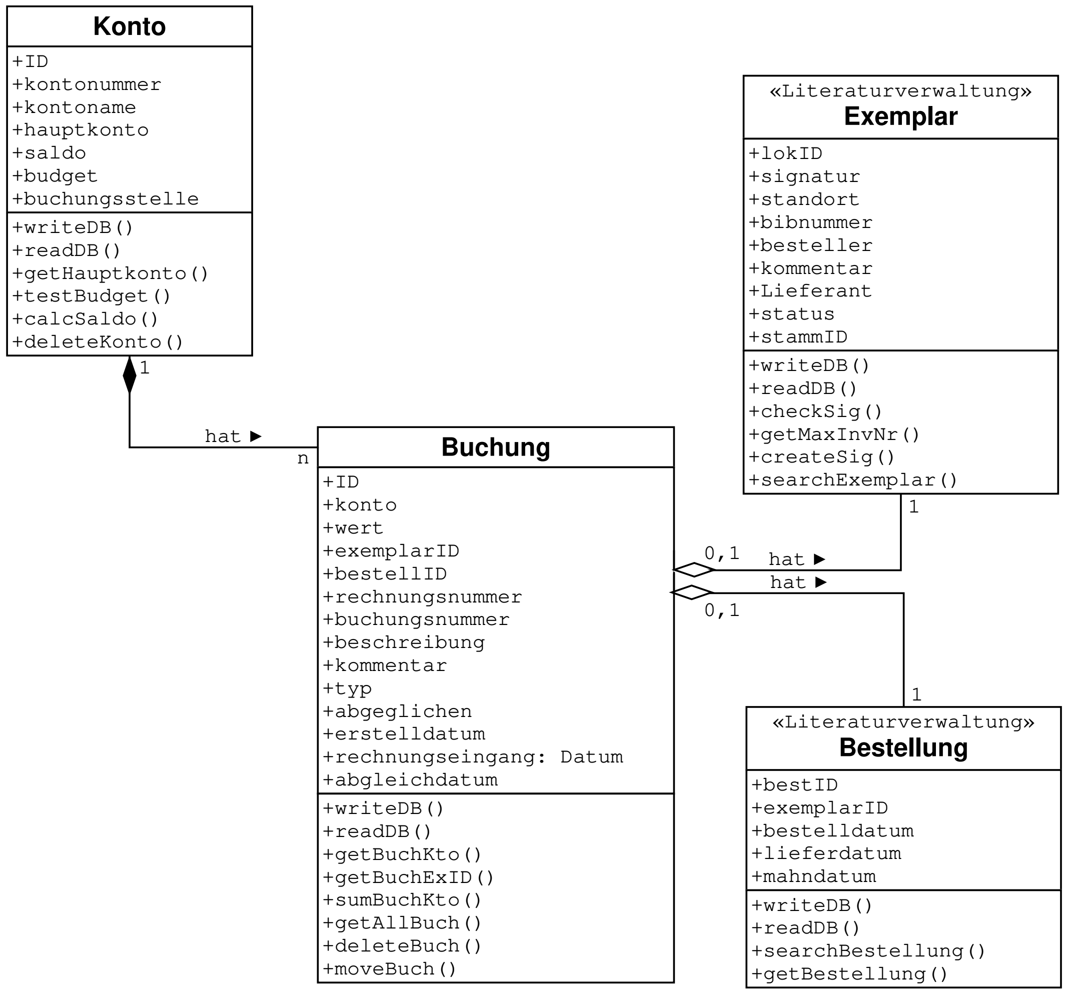

Abbildung 3.14: UML-Diagramm der Buchhaltung

Die Datenbearbeitungsschicht stellt der Darstellungsschicht folgende Funktionen zur Verfügung:

`konto_neu()` legt ein neues Haupt- oder Unterkonto an. (/F4100/, /F4110/)

`konto_aendern()` ändert Kontodaten (/F4150/)

`konto_lesen()` liest ein Konto

`konto_loeschen()` löscht ein Konto und verschiebt die enthaltene Buchungen in das übergeordnete Konto. Wenn ein Hauptkonto noch Buchungen enthält, kann es nicht gelöscht werden. (/F4120/)

`konto_abgeleich()` abgleichen der Buchungen mit einem Kontoauszug (/F4141/)

`buchung_neu()` erzeugt eine neue Buchung (/F4130/)

`buchung_storno()` storniert eine Buchung (/F4140/)

`hole_hauptkonten()` liefert die Hauptkonten

kontoauszug() erstellt einen Kontoauszug für alle Konten (/F4160/)

3.7.5 Konfiguration

Als letztes Modul wird die Konfiguration betrachtet. Die Datenhaltungsschicht besteht hauptsächlich aus den Klassen `User` und `Lieferanten`. Diese Klassen sind unabhängig voneinander, sodass ein UML-Diagramm nicht notwendig wird. Die Klasse /tt User repräsentiert die Anwender des System. Als Attribute enthält sie unter anderem den Login-Name, das Passwort und den wirklichen Namen. Sie stellt Methoden bereit um die Daten zu speichern, auszulesen und zu manipulieren.

Die Lieferantendaten bilden die Grundlage der Klasse `Lieferanten`. Die Attribute der Klasse sind eine Abbildung der persistent gespeicherten Daten. Die Methoden manipulieren, speichern und lesen die Daten.

Die Konfiguration von BiVIS wird über eine funktionale Schnittstelle aus der Datenbank gelesen und geschrieben. Da die Konfiguration sich während der Implementierung noch verändert, werden die Funktionen erst im Kapitel 4 beschrieben.

Die Schnittstelle zwischen Datenbearbeitungsschicht und Darstellungsschicht wird durch folgende Funktionen repräsentiert:

addUser() fügt einen neuen Benutzer hinzu (/F5100/)

changeUser() ändert die Benutzerdaten (/F5110/)

deleteUser() löscht die Benutzerdaten (/F5130/)

setzeRechte() setzt die Ausführungsrechte eines Benutzers (/F5130/)

storeLieferant() speichert Daten zu Lieferanten (/F5170/, /F5180/)

delete_lieferant() löscht einen Lieferanten (/F5190/)

newIndextabelle() legt eine neue Indizierungstabelle an (/F6120/)

deleteIndextabelle() löscht eine Indizierungstabelle, die von einem Benutzer angelegt wurde (/F6130/)

anmelden() meldet einen Benutzer an das System an (/F6100/)

Kapitel 4

Implementierung

In diesem Kapitel wird auf die Implementierung von BiVIS eingegangen. In Abschnitt 4.1 wird die Auswahl der Programmiersprache und des Datenbanksystems diskutiert. Ausserdem werden die darüber hinaus benötigten Softwarepakete aufgelistet. Abschnitt 4.2 beschreibt die Implementierung der einzelnen Module von BiVIS. Der Abschluss dieses Kapitels bildet die Beschreibung der Benutzungsschnittstelle in Abschnitt 4.3

4.1 Auswahl der Programmiersprache und Software

In diesem Abschnitt wird auf die Auswahl der verwendeten Softwarepakete eingegangen.

4.1.1 MySQL als Datenbanksystem

Es gibt heute einige Datenbanksysteme auf dem Markt. Da eine Anforderung an BiVIS war so kostengünstig wie möglich zu entwickeln, werden die kommerziellen Datenbanksysteme (ORACLE, SYBASE, INFORMIX, u.a.) hier nicht betrachtet. Die freien Datenbanksysteme (DBMS) wie MySQL, mSQL oder Postgres fallen somit in die nähere Auswahl. Vergleicht man diese Systeme sticht einem direkt MySQL wegen der guten Performance ins Auge. Durch die gute Performance muss man allerdings auf eine Funktionalität verzichten: die referenzielle Integrität. Die referenzielle Integrität muss folglich in der Anwendung behandelt werden. MySQL hat ausserdem den Vorteil, dass es in der Web-Gemeinde sehr weit verbreitet ist. In BiVIS wird MySQL als DBMS eingesetzt.

4.1.2 PHP als Programmiersprache

Durch die Definition der Einsatzumgebung im Abschnitt 3.4 werden Programmiersprachen wie JAVA oder C++ ausgeschlossen. Es bleiben als mögliche Sprachen nur die Skriptsprachen wie PHP oder Perl. Perl ist eine sehr mächtige Skriptsprache. Sie bietet die Möglichkeit über das DBI-Interface verschiedene Datenbanken zu kontaktieren. Allerdings müssten die erzeugten Skripte als CGI-Skripte (Common Gateway Interface) in den Webserver eingebaut werden. Die Erzeugung von HTML-Code zur Darstellung wird in Perl auch nicht direkt unterstützt. Perl bietet auch in der Grundversion keine Objekt-Orientierte Programmierung.

PHP (**P**HP **H**ypertext **P**reprozessor) ist eine Serverside-Scripting-Language zum dynamischen Erstellen von Webseiten. Sie wird also direkt über einen Interpreter vom Webserver ausgeführt. Diesen Interpreter gibt es für viele Webserver als Modul. Durch diese weite Verbreitung läuft PHP auf vielen Plattformen (Windows, verschieden Unix-Derivate). PHP bietet seit Version 4 auch die Möglichkeit des Objekt-Orientierten-Programmierens. PHP unterstützt die Erzeugung von HTML-Code als Ausgabe. Die Mächtigkeit von PHP ist ähnlich der von Perl. PHP kann als Plattformunabhängig bezeichnet werden, da die Ausgabe mittels HTML in jedem Web-Browser dargestellt werden kann.

BiVIS wird aus den oben genannte Gründen in PHP realisiert.

4.1.3 Sonstige verwendete Software

Der Web-Server wird nicht festgelegt. Es wird nur verlangt, dass er PHP4 interpretieren kann.

Für die zeitlich gesteuerten Programmteile wird ein Scheduler wie `cron` benötigt. Die Skripte für diese Programmteile werden in Perl geschrieben, da keine HTML-Ausgabe nötig ist.

Mit BiVIS können Bestellbriefe und Mahnbriefe erzeugt werden. Für die Erstellung dieser Dokumente wird LaTeXverwendet.

Zum Drucken der Ausleihzettel wird die Textbeschreibungssprache Postscript verwendet.

4.2 Implementierung von BiVIS

Um ein wartungsfreundliches System zu erhalten, muss der Speicherort der Skripte gut gegliedert sein. Wie aus Abbildung 4.1 ersichtlich wird, hat das BiVIS-Wurzelverzeichnis `bivis` drei Unterverzeichnisse, `common`, `pictures` und `modules`. Dieses Verzeichnis enthält noch die Datei `index.html` die den Einstiegspunkt in das System darstellt. Das Verzeichnis `common` enthält Dateien, die jedes Modul benötigt. Die Bilder und Schaltflächen, die von BiVIS benötigt werden sind in dem Verzeichnis `pictures` und dessen Unterverzeichnis `buttons` abgelegt. Die eigentliche Module sind im `modules`-Verzeichnis untergebracht. Jedes Modul hat sein eigenes Verzeichnis und eine PHP-Datei mit dem selben Namen wie das Verzeichnis. Diese Datei stellt die Startseite des Modules dar. Diese Anforderung wurde benötigt, damit man neue Module automatisch in das System einbinden kann. Die einzige Ausnahme zu diesem System bildet das Basismodul. Dieses Modul muss von Anfang an installiert sein, da es den Mechanismus mitbringt weitere Module einzubinden.

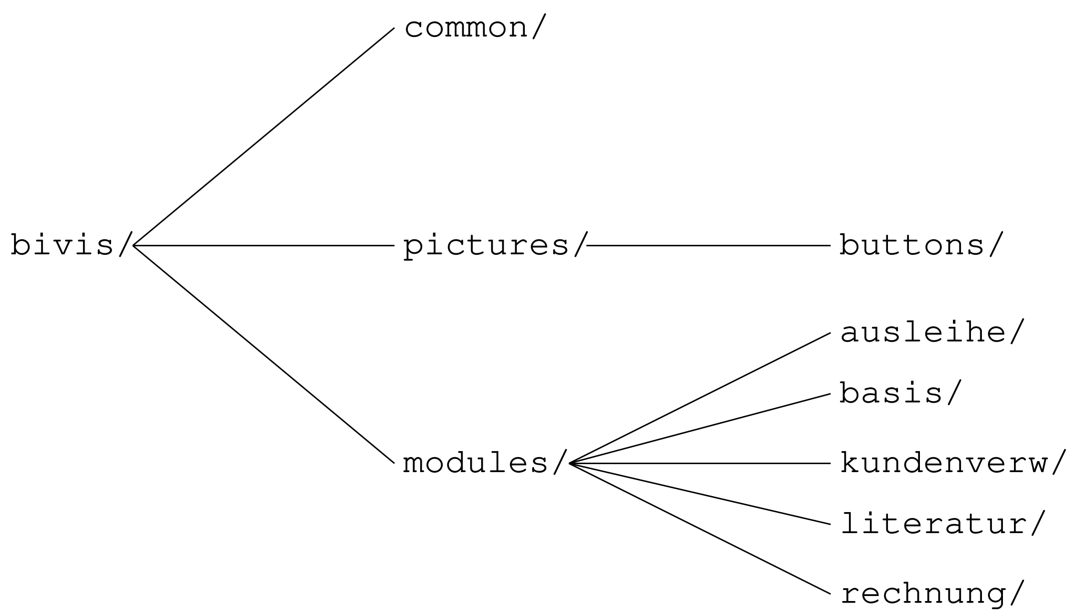

Abbildung 4.1: Die Verzeichnisstruktur von BiVIS

Die einzelnen Module sind nach einem bestimmten Schema aufgebaut. Wie oben schon erwähnt, haben sie eine Einsprungdatei. Diese beinhaltet das Hauptmenü des Modules. Zusätzlich gibt es noch drei Dateien, die bei jedem Modul denselben Namen und Bedeutung haben. Diese Dateien repräsentieren die einzelnen Schichten der Module. Die Datenhaltungsschicht mit ihrer Schnittstelle zur Datenbearbeitungsschicht wird von der Datei `datenschnittstelle.php` realisiert. Die Datei `logik.php` implementiert die Datenbearbeitungsschicht mit der funktionalen Schnittstelle zur Darstellungsschicht. Die Formulare der Benutzungsschnittstelle sind in der Datei `formulare.php` implementiert. Die übrigen Dateinen der Module steuern den Ablauf und die Menüführung der Benutzungsschnittstelle.

Jedes Modul erweitert auch die Datenbank. Deshalb existiert zu jedem Modul auch eine gleichnamige SQL-Datei. Dort ist die Tabellenstruktur der Datenbanken gespeichert.

Die Anforderung des sicheren Netzverkehrs wird durch den Webserver realisiert. Er muss so konfiguriert werden, dass die BiVIS-Verzeichnisse nur über eine SSL-Verbindung angesprochen werden können.

In dem folgenden Abschnitt wird auf die Implementierung der einzelnen Teile der Module eingegangen. Der Ablauf in den einzelnen Abschnitten ist identisch. Zuerst wird die Tabellenstruktur behandelt. Danach werden die Klassen und Funktionen der Datenhaltungsschicht besprochen. Zum Abschluss wird die Implementierung der Datenbearbeitungsschicht erläutert. Auf die Implementierung der Benutzungsschnittstelle wird in Abschnitt 4.3 eingegangen.

4.2.1 Literaturverwaltung

Tabellenstruktur

Wie aus Kapitel 3 hervorgeht, müssen für dieses Modul die Tabellen der Literaturstammdaten, der Exemplardaten und der Bestelldaten erzeugt werden.

Die Literaturstammdaten werden in den Indizierungstabellen gespeichert. Ein Datensatz der Indizierungstabelle besteht aus einem Feld für die Datensatz-ID. Aus der Anforderung heraus, dass die Datenmenge nur durch die Kapazität des Servers beschränkt sein soll, stellt sich nun die Frage welcher Datentyp für dieses Feld verwendet werden soll. Ein ganzzahliger Wert soll es auf jeden Fall sein, damit man die Generierung dieser Werte recht einfach halten kann. Zur Auswahl steht entweder `INT` oder `BIGINT`. Zur Speicherung eines `INT`-Wertes werden 4 Bytes benötigt. Mit einem `INT`-Datentyp kann man $2^{31} - 1$ verschiedene Zahlen erzeugen, das sind etwa 2 Mrd Stammdatensätze die gespeichert werden können. Beim Datentyp `BIGINT`, der 8 Byte Platz pro Wert benötigt, könnten $2^{63} - 1$ (9.223.372.036.854.775.807) Stammdatensätze gespeichert werden. Wenn man als Datentyp `BIGINT` nehmen würde und alle möglichen Datensatz-IDs einmal abspeichern würde, werden ca. 2^{66} Byte (2^{36} GByte) Plattenplatz benötigt. Solange man unter den 2 Milliarden Literaturstammdatensätze bleibt benötigt das `INT`-Datenformat nur die Hälfte des Plattenplatzes. Bei 2^{32} Einträgen wären das 2^{34} Byte (16 GByte). Aus diesen Gründen wird für die Datensatz-ID der Datentyp `INT` verwendet. Da die Datensatz-ID einen Literaturstammdatensatz eindeutig idnetifiziert, darf dieser Wert nicht leer bleiben.

Das nächste Datenfeld ist die MAB2-Feldnummer. Diese Feldnummer ist immer drei-stellig. Um die benutzerdefinierten Felder von den vordefinierten MAB2-Felder zu unterscheiden, kann man als Feldbezeichner Buchstabenkombinationen verwenden. Durch die Verwendung von Buchstaben wird als Datentyp für den MAB2-Feldbezeichner `VARCHAR(3)` eingesetzt.

Das Indikator-Feld ist ein alphanummerisches Feld der maximalen Länge 2. Deshalb wird für dieses Feld der Datentyp `VARCHAR(2)` eingesetzt.

Für das Inhaltfeld ist die Entscheidung welchen Datentyp man verwendet etwas schwieriger. Es muss zuerst überlegt werden, welche Eingabe die längste sein kann. Die längste Eingabe, die in den Indizierungstabellen gespeichert wird, ist der Titel. Aus Abbildung 4.2 wird ersichtlich, dass die meisten Titel eine Länge zwischen 30 und 90 Zeichen haben. Die Untersuchung von ca. 3800 Titeln ergab, dass der längste Titel eine Länge von 254 Zeichen hatte. Die durchschnittliche Titellänge beträgt 67 Zeichen. MySQL bietet als string-Datentyp auch einen `TEXT`-Datentyp an. Dieser hat allerdings den Nachteil, dass er nicht indiziert werden kann. Aus diesen Gründen wird in BiVIS als Datentyp für das Inhalt-Feld `VARCHAR(255)` verwendet. Das ist die maximale Länge des `VARCHAR`-Datentyps.

Die Struktur der Tabelle wird im Zusammenhang am besten durch den verwendeten SQL-Befehl deutlich. Als Beispiel wird hier die Struktur der Indizierungstabelle `personennamen` beschrieben.

```
CREATE TABLE personennamen (DID INT NOT NULL,
                            MabID varchar(3) NOT NULL,
                            Indikator varchar(2) NOT NULL,
                            Inhalt varchar(255) NOT NULL,
                            INDEX(Inhalt));
```

Die Suche nach einem Eintrag in einer solchen Tabelle wird größtenteils in dem Datenfeld Inhalt erfolgen. Aus diesem Grund wird ein Index über dieses Feld erzeugt.

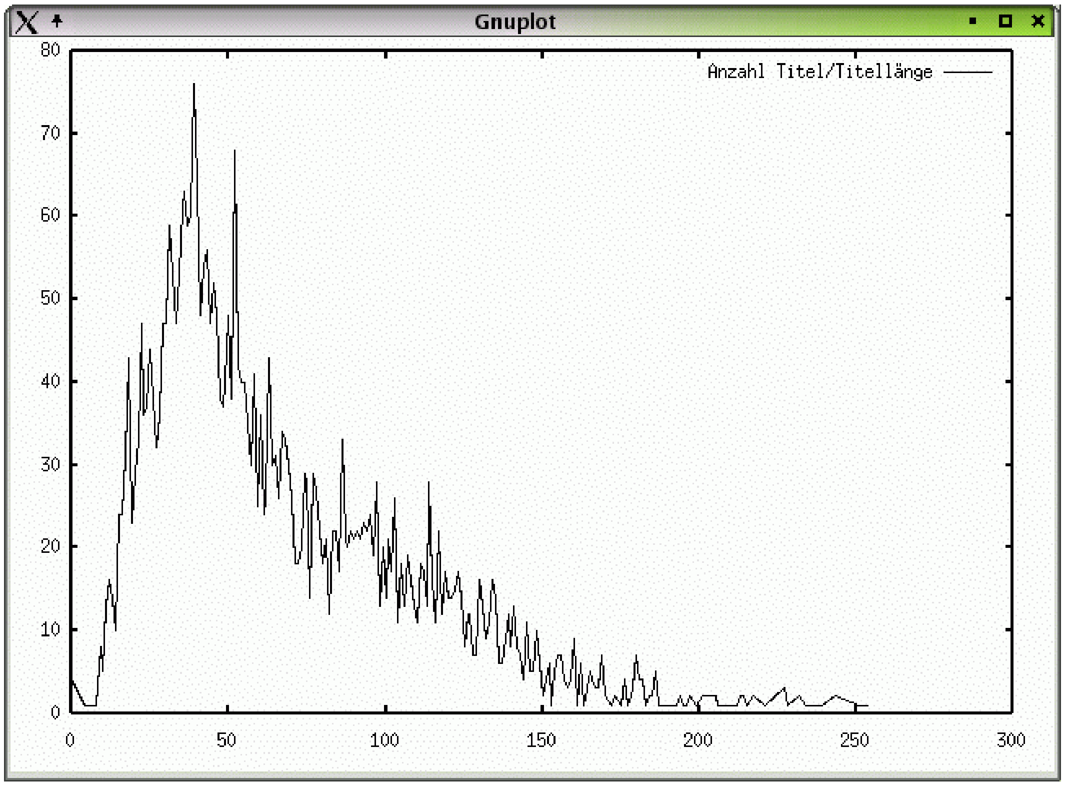

Abbildung 4.2: Verteilung der Titel(y-Achse) über die Titellänge(x-Achse)

Für die Anderen Tabellen haben ähnliche Überlegungen zu den verwendeten Datentypen geführt. Im Folgenden wird für jede Tabelle nur noch der SQL-Befehl dargestellt.

Die Tabelle `didtable` wird benötigt, um die Datensatz-IDs zu speichern und zu erzeugen.

```
CREATE TABLE didtable (DID INT AUTO_INCREMENT PRIMARY KEY);
```

Das Schlüsselwort `AUTO_INCREMENT` bedeutet, dass der Wert dieses Datenfeldes bei jedem `INSERT`-Befehl automatisch inkrementiert wird. `PRIMARY KEY` sagt aus, dass das Datenfeld die Primärschlüsselbedingung erfüllt (Eindeutigkeit).

Die folgenden vier Tabellen (`verlag_stamm`, `serien_stamm`, `koerpersch_stamm`, `autor_stamm`) sind die letzten Tabellen, die für die Literaturstammdaten benötigt werden. Die Tabelle für die MAB2-Beschreibungsdaten `stamm_def` wird im Abschnitt Konfiguration beschrieben. Mit dieser Tabelle ist die Stammdatenbank `stamm` vollständig.

```
CREATE TABLE verlag_stamm (VerlagID INT AUTO_INCREMENT PRIMARY KEY,
                           Name VARCHAR(255),
                           Ort VARCHAR(255));

CREATE TABLE serien_stamm (SerienID INT AUTO_INCREMENT PRIMARY KEY,
```

```
                            Name VARCHAR(255));

CREATE TABLE koerpersch_stamm (KoerperschID INT AUTO_INCREMENT PRIMARY KEY,
                               Name VARCHAR(255),
                               Ort VARCHAR(255));

CREATE TABLE autor_stamm (AutorID INT AUTO_INCREMENT PRIMARY KEY,
                          Name VARCHAR(255),
                          Koerpersch VARCHAR(255),
                          Signatur VARCHAR(255),
                          Typ VARCHAR(255));
```

Die Exemplardaten und die Bestelldaten werden in der Lokaldatenbank `bib` gespeichert. Wie schon im Kapitel 3 besprochen wird nicht das MAB-LOKAL-Format für die Exemplardaten verwand, sondern ein eigenes Format. Zur eindeutigen Identifizierung des Exemplardatensatzes wird der Primärschlüssel `ExemplarID` eingeführt. Dieser Primärschlüssel hat aus den selben Überlegungen zur Literaturstammdatensatz-ID den Datentyp `INT`.

Zum sortierten Einstellen und zur Identifizierung eines Exemplares ist dieses mit einer Signatur versehen. Die Signatur besteht aus alphanummerischen Zeichen. Als weitere, eindeutige Kennung des Exemplares innerhalb der Bibliothek wird eine Inventarnummer verwendet.

Es gibt drei Datenfelder des Typs `TINYINT` diese Datenfelder sind Flags, die zur weiteren Bearbeitung notwendig sind. Die Felder `Eitbar` und `Etisig` geben an, ob zu diesem Exemplar noch ein Barcode-Etikett bzw. ein Signatur-Etikett gedruckt werden muss. Ob ein Buch zur Ausleihe zugelassen ist, sagt das Feld `Sperre`.

Die Struktur der Tabelle `exemplare` wird durch den SQL Befehl

```
CREATE TABLE exemplare (ExemplarID INT AUTO_INCREMENT PRIMARY KEY,
                        Signatur VARCHAR(255),
                        Standort VARCHAR(255),
                        Invnummer INT,
                        BestellerID INT,
                        Kommentar TEXT,
                        LieferantID INT,
                        Status VARCHAR(50),
                        Statistik INT DEFAULT 0,
                        Sperre TINYINT DEFAULT 0,
                        Etibar TINYINT DEFAULT 1,
                        Etisig TINYINT DEFAULT 1,
                        Bearbeiter VARCHAR(255),
                        StammID INT,
                        Aufndatum DATE);
```

dargestellt.

Abschließend wird die Struktur der Tabelle `bestellung` betrachtet. Sie hat als Primärschlüssel das Feld `BestID`. Das Feld `ExID` ist ein Fremdschlüssel. Da aber MySQL keine referenzielle Integrität unterstützt und somit keine Fremdschlüssel, wird von diesem Feld nur verlangt, dass es nicht leer sein darf. Die Felder `Druck` und `Rechnung` sind wiederum Flags. `Druck` besagt, dass dieser Bestelldatensatz noch zu Drucken ist. Um automatisch Rechnungsdatensätze zu erzeugen, wird das Flag `Rechnung` benötigt.

```
CREATE TABLE bestellung (BestID INT AUTO_INCREMENT PRIMARY KEY,
                         ExID INT NOT NULL,
                         Druck TINYINT,
                         Rechnung TINYINT,
                         BDatum DATE NOT NULL,
                         LDatum DATE,
                         MDatum DATE);
```

Datenhaltungsschicht

Hier sind die Klassen implementiert, deren Struktur in Abschnitt 3.7.1 erläutert wurde. Das Lesen und Schreiben der Daten besteht aus dem Umsetzten der Attribute in SQL-Statements. Dies wird hier, stellvertretend für sämtliche Lese- und Schreiben-Funktionen der Klassen, am Beispiel der Klasse `Autor` gezeigt. Die Parameter `$lanme` und `$pw` werden zur Anmeldung an die Datenbank benötigt. Sie sind nach der Anmeldung an das System in sogenannten Session-Variablen gespeichert. Diese Session-Variablen sind allerding nur in der Darstellungsschicht sichtbar und müssen somit als Funktionsparameter zur Datenhaltungsschicht durchgereicht werden. Die Funktion `readDB($lname, $pw, $id)` liest die Daten des Autors mit der ID `$id` und speichert sie in den Attributen der Klasse ab.

```
function readDB($lname, $pw, $id){
  $connect = mysql_connect("localhost",$lname,$pw);
  if ($connect) {
    $sql = "SELECT * from autor_stamm WHERE AutorID = \"$id\" ";
    $result = mysql_db_query("stamm", $sql);
    $erg = mysql_fetch_array($result, MYSQL_ASSOC);
    $this->name = $erg[Name];
    $this->ID = $erg[AutorID];
    $this->koerperschaft = $erg[Koerpersch];
    $this->signatur = $erg[Signatur];
    $this->typ = $erg[Typ];
  } else {
    fehler(mysql_error(), mysql_errno());
  }
}
```

Um einen Datensatz abzuspeichern wird die Funktion `writeDB($lname, $pw)` verwendet. Diese Funktion entscheidet, ob der Datensatz neu angelegt oder geändert wird. Wenn das Attribut `$ID` leer ist wird ein neuer Datensatz erzeugt. Dabei wird eine neue ID erzeugt, die die Funktion an den Aufrufer zurückgibt.

```
function writeDB($lname, $pw){
  $id;
```

```
    $connect = mysql_connect("localhost",$lname,$pw);
    if ($connect) {
      if ($this->ID == "") {
        $sql = "INSERT INTO autor_stamm SET Name = \"$this->name\",
                            Koerpersch = \"$this->koerperschaft\",
                            Signatur = \"$this->signatur\",
                            Typ = \"$this->typ\" ";
      } else {
        $sql = "UPDATE autor_stamm SET Name = \"$this->name\",
                            Koerpersch = \"$this->koerperschaft\",
                            Signatur = \"$this->signatur\",
                            Typ = \"$this->typ\"
                            WHERE AutorID = \"$this->ID\"";
      }
      mysql_db_query ("stamm", $sql);
      if ($this->ID == "") {
        $this->ID = mysql_insert_id();
      }
      mysql_db_query ("stamm", "FLUSH TABLES");
    }
    return $this->ID;
  }
```

Eine Ausnahme beim Schreiben und Lesen von Daten bildet die Klasse `litwerk`, da sie ihre Daten aus mehreren Tabellen zusammentragen muss. Beim Lesen der Daten schaut die Methode `readDB()` erst in der Tabelle `stam_def` nach, welche Indizierungstabellen existieren. Danach werden die Daten aus den Indizierungstabellen gelesen und die Referenzen aufgelöst. Beim Schreiben der Daten schaut die Funktion `writeDB()` bei jedem Attribut nach in welcher Tabelle es gespeichert wird und erzeugt, falls notwendig, eine Referenz.

Datenbearbeitungsschicht

In der Datenbearbeitungsschicht ist die eigentliche Anwendung realisiert. Sie bearbeitet die von der Datenhaltungsschicht bereitgestellten Daten für die Darstellungsschicht auf und umgekehrt. In der Literaturverwaltung kommen zusätzlich noch Funktionen hinzu zum automatischen Generieren der Signaturen, zum Ausdrucken der Barcode- und Signaturetiketten und zum Ausdrucken der Bestellbriefe. Eine Signatur wird in der Fachrichtungsbibliothek Informatik nach bestimmten Regeln gebildet. Eine Signatur für Monographien besteht aus vier Teilen. Der erste Teil besteht aus den ersten drei Buchstaben des Autorennachnamens in Großbuchstaben, falls dieser nicht mit `Sch`, `St` oder `Sp` anfängt. Diese Buchstabengruppen werden als ein Buchstabe gewertet. Der zweite Teil der Signatur wird aus dem ersten Buchstaben des Autorenvornamens als Kleinbuchstabe erzeugt. Sollte es schon ein Autor mit einem solchen Kürzel geben, wird hinter dem Kleinbuchstaben eine Zahl (beginnend mit 2) hochgezählt bis dieses Kürzel noch nicht existiert. Der dritte Teil der Signatur besteht aus dem Erscheinungsjahr der Monographie, gefolgt von einem Doppelpunkt. Hinter dem Doppelpunkt wird wieder bei 1 anfangend hochgezählt bis das resultierende Kürzel nicht mehr existiert. Der vierte Teil der Signatur bezeichnet das Exemplar. Es besteht aus einer fortlaufenden Nummer gefolgt von einem Punkt und anschließend das Wort `Ex`. Auf Grund dieser Regel würde diese Diplomarbeit die Signatur `KRÖ d2 02:1 1.Ex` bekommen. In BiVIS wird zu jedem Autor das passende Signaturkürzel, die ersten zwei Teile der Signatur, automatisch vergeben. Zur Erzeugung des Signaturkürzel in den Stammdaten, erster bis dritter Teil der Signatur wird das Signaturkürzel des ersten genannten Autors des Werkes sowie die zweistellige Jahreszahl des Erscheinungsjahres genommen und dann solange hochgezählt bis das entstandene Signaturkürzel nicht existiert. Jetzt hat jede Monographie ihr eigenes Signaturkürzel. Die Funktion `create_ex_sig($lname, $pwd, $ID)` liest das Signaturkürzel des Stammdatensatzes mit der ID `$ID` und zählt einfach die Exemplarangabe hoch bis es eine Signatur findet, die noch nicht vergeben ist. Diese Signatur gibt sie and die aufrufende Funktion zurück.

```
function create_ex_sig($lname, $pwd, $ID){
  $data = select_stamm($lname, $pwd, $ID);
  $i = 1;
  $sig = $data[signatur] . " $i.Ex";
  $correct = false;
  $ex = new Exemplar;
  while(!$correct) {
    if ($ex->checkSig($lname, $pwd, $sig)) {
      $correct = true;
    } else {
      $i++;
      $sig = $data[signatur] . " $i.Ex";
    }
  }
  return $sig;
}
```

Zum Ausdrucken der Etiketten und Briefe wird LaTeXverwendet. Für die Barcode-Etiketten wurde in IBBIB ein eigenes Style-File geschrieben. Dieses Style-File wird in BiVIS wiederverwendet. Der Barcode ist ein Code128-Barcode. Das Programm, dass die Steuercodes zur Erzeugung des Barcodes berechnet, wurde von Perl nach PHP portiert. Es ist die Funktionen `codcxxviii($srcstr)` in der Datei `standard.php`. Die Signaturetiketten werden von der Funktion `print_sigetikett($etilist)` gedruckt. Für beide Etikettenarten wird eine Liste zu druckender Etiketten benötigt. Diese Listen werden durch die Funktion `etiToPrint($lname, $pwd, $type)` erzeugt. Der Parameter `$type` bezeichnet die Art der Etiketten. Hierbei steht `bc` für Barcodeetiketten und `sig` für Signaturetiketten. Der Anwender kann aus diesen Listen die zu druckende Etiketten auswählen.

Für die Bestellkorrespondenz wird ebenfalls das bestehende Briefformat von IBBIB wiederverwendet. Die Funktion `drucke_best($lname, $pwd, $vname, $nname)` durchsucht die Bestellungen nach noch zu druckenden Bestellungen. Die Funktion bekommt von der Datenhaltungsschicht diese Bestellungen gruppiert nach Lieferanten. Falls ein Buch mehrfach bestellt wurde, werden diese Bestellungen zusammengefasst und die Anzahl der Exemplare vorangestellt. Für jeden Lieferanten wird nun ein Brief in der festgelegten Korrespondenzsprache ausgedruckt. In BiVIS sind zur Zeit Englisch und Deutsch als Sprachen implementiert.

4.2.2 Kundenverwaltung

Tabellenstruktur

Die Kundenverwaltung erweitert die Lokaldatenbank `bib` um die Tabellen `kunden`, `kukart` und `knummern`. In die Tabelle `kukart` werden die ID's der Kunden gespeichert, deren Kundenkarte neu zu drucken ist. Aus dem SQL-Befehl

```
CREATE TABLE kukart (KundenID INT UNIQUE NOT NULL);
```

geht hervor, dass ein Kunde nur einmal in der Tabelle gespeichert werden kann. Es können also nicht mehrere Kundenkarten eines Kunden gleichzeitig gedruckt werden. Das wird durch den SQL-Befehl `UNIQUE` realisiert.

Um die bibliotheksinternen Kundennummern eindeutig zu erzeugen, wird die Tabelle `knummern`

```
CREATE TABLE knummern (Knummer INT NOT NULL);
```

benötigt. Wird ein Kunde aus dem System gelöscht wird dessen Kundennummer in diese Tabelle geschrieben. Beim Eintragen eines neuen Kunden wird zuerst diese Tabelle nach der kleinsten verfügbaren Kundennummer durchsucht. Sollte diese Tabelle leer sein, wird die größte Kundennummer aus den Kundendaten genommen und diese inkrementiert.

Die eigentlichen Kundendaten werden in der Tabelle `kunden`

```
CREATE TABLE kunden (Name VARCHAR(255),
             Vorname VARCHAR(255),
             HSuff VARCHAR(50),
             HPrae VARCHAR(50),
             Sex ENUM (\"w\", \"m\"),
             Strasse VARCHAR(255),
             PLZ VARCHAR(255),
             Ort VARCHAR(255),
             Telp VARCHAR(50),
             Telm VARCHAR(50),
             Teld VARCHAR(50),
             Dienstanschrift VARCHAR(255),
             Lehrstuhl VARCHAR(255),
             Email VARCHAR(255),
             Zugangsnr INT,
             KdNr INT,
             Typ VARCHAR(20),
             Abdauer INT,
             Eintritt DATE NOT NULL,
             Abenddat DATE,
             Ausleihdauer INT NOT NULL,
             KundenID INT AUTO_INCREMENT PRIMARY KEY);
```

gespeichert. Die Ausleihberechtigungsdauer `Abdauer` wird in Tagen angegeben. Wenn sie den Wert 0 hat, ist der Kunde unbeschränkt ausleihberechtigt.

Datenhaltungsschicht

In der Datenhaltungsschicht gibt es nur eine Klasse, die einen Kunde repräsentiert. Neben den Methoden zum Speichern, Löschen und Lesen der Daten enthält die Klasse `Kunde` eine Methode zum Berechnen einer neuen Kundennummer `getNewKdnr($lname, $pwd)`. Sie implementiert die im vorherigen Abschnitt beschriebene Berechnung einer neuen Kundennummer. Diese Berechnung wird von der Datenbank durchgeführt.

```
function getNewKdnr($lname, $pwd){
  if(mysql_connect("localhost", $lname, $pwd)){
    $result = mysql_db_query("bib","SELECT MIN(Knummer) AS mini FROM knummern");
    $erg = mysql_fetch_array($result, MYSQL_ASSOC);
    if ($erg[mini] == ""){
      $result = mysql_db_query("bib", "SELECT MAX(KdNr) AS maxi FROM kunden");
      $erg = mysql_fetch_array($result, MYSQL_ASSOC);
      if ($erg[maxi] == ""){
        $this->kd_nr = 0;
      } else {
        $this->kd_nr = $erg[maxi]+1;
      }
    } else {
      $this->kd_nr = $erg[mini];
    }
  }
  return $this->kd_nr;
}
```

Neben der Klasse `Kunde` enthält die Datenhaltungsschicht eine Funktion, die eine Zugangsliste bereitstellt. Diese Zugangsliste wird auch auf Datenbankebene erzeugt und sortiert nach Name und Vorname des Kunden.

```
function getZugangListe($lname, $pwd){
  if(mysql_connect("localhost", $lname, $pwd)){
    $result = mysql_db_query("bib","SELECT * FROM kunden
           WHERE Zugangsnr != \"\" ORDER BY Name, Vorname");
    $i = 0;
    while ($erg = mysql_fetch_array($result, MYSQL_ASSOC)){
      $list[$i] = array(KundenID=>$erg[KundenID],
                        Name=>$erg[Name],
                        Vorname=>$erg[Vorname],
                        Zugangsnr=>$erg[Zugangsnr],
                        KdNr=>$erg[KdNr],
                        Typ=>$erg[Typ],
                        Lehrstuhl=>$erg[Lehrstuhl]);
      $i++;
    }
    return $list;
  }
```

```
}
```

Datenbearbeitungsschicht

In der Datenbearbeitungsschicht werden, neben den Funktionen zur Datenaufbereitung, Funktionen bereitgestellt zum Drucken der Kundenkarten und zum Umwandeln der Kundennummer in das bibliotheksinterne Format.

Die Kundennummern in der Fachrichtungsbibliothek Informatik codieren neben der eindeutigen Nummer auch den Typ des Kunden. So hat ein Mitarbeiter mit der Nummer `0049` als Kundennummer `M-0049`. Somit steht vor dem Minuszeichen ein Großbuchstabe, der folgende Bedeutungen hat:

D steht für Diplomand

M steht für Mitarbeiter

G steht für Gast

S steht für Sonstige

Die Funktion `info_kdnr($typ, $kdnr)` erzeugt aus dem Typ `$typ` und der Kundennummer `$kdnr` das interne Format der Kundennummer.

```
function info_kdnr($typ, $kdnr){
  switch ($typ){
    case "Mitarbeiter": $knr = "M-". sprintf("%04d", $kdnr);
                        break;
    case "Diplomand":   $knr = "D-". sprintf("%04d", $kdnr);
                        break;
    case "Gast":        $knr = "G-". sprintf("%04d", $kdnr);
                        break;
    default:            $knr = "S-". sprintf("%04d", $kdnr);
  }
  return $knr;
}
```

Zum Drucken der Kundenkarteikarten wird auf die Formatvorlage aus IBBIB zurückgegriffen. Die Funktion `drucke_kukart($lname, $pwd)` erhält von der Datenhaltungsschicht die Liste zu druckender Kundenkarteikarten. Sie bereitet die Daten auf und druckt sie aus. Das Löschen aus der Tabelle `kukart` erfolgt erst, wenn der Anwender den korrekten Druck bestätigt.

4.2.3 Ausleihverbuchung

Tabellenstruktur

Wie in Kapitel 3 dargestellt wurde, werden für die Ausleihe zwei Tabellen benötigt. Die Tabelle `ausleihe` ist für das Speichern der Ausleihdaten verantwortlich. Jede Ausleihe wird durch den Primärschlüssel `AusleihID` eindeutig identifiziert. Die Schlüsselgenerierung erfolgt durch automatisches Hochzählen. Die Information zum Kunden wird als Referenz auf einen Kundendatensatz im Feld `KundenID` gespeichert. Dieses Feld darf nicht leer bleiben. Das Feld `BuchID` enthält die Referenz auf einen Exemplardatensatz, der das ausgeliehene Buch repräsentiert. Die Datumsfelder `Beginn` und `Ende` enthalten das Daten von Ausleihbeginn und Ausleihende. Das Ausleihende wird aus dem Ausleihbeginn und der im Kundendatensatz gespeicherten Ausleihdauer berechnet. Die Felder `Mahndatum` und `Mahnstufe` sind für das automatische Anmahnen von Bedeutung.

Die Tabellenstruktur wird durch den SQL-Befehl

```
CREATE TABLE ausleihe (AusleihID INT AUTO_INCREMENT PRIMARY KEY,
                       KundenID INT NOT NULL,
                       BuchID INT NOT NULL,
                       Beginn DATE NOT NULL,
                       Ende DATE NOT NULL,
                       Mahnstufe TINYINT,
                       Mahndatum DATE);
```

erzeugt.

Die Daten für Vorbestellungen werden in der Tabelle `vorbest` gespeichert. Die Felder `KundenID` und `BuchID` haben dieselbe Bedeutung wie in der Tabelle `ausleihe`. Im Datumsfeld `Datum` wird der Tag eingetragen, an dem Das Buch vorbestellt wurde. Um Bücher nicht ewig durch eine Vorbestellung von der Ausleihe zu sperren, wird das Feld `Bendatum` benötigt. Es enthält das Datum an dem der Kunde über die Rückgabe des Buches benachrichtigt wurde.

```
CREATE TABLE vorbest (VorbestID INT AUTO_INCREMENT PRIMARY KEY,
                      KundenID INT NOT NULL,
                      BuchID INT NOT NULL,
                      Datum DATE NOT NULL,
                      Bendatum DATE);
```

Datenhaltungsschicht

Die Datenhaltungsschicht wird hauptsächlich durch die zwei Klassen `ausleihe` und `vorbest` implementiert. Die Attribute der Klassen entsprechen den Felder der zugrundeliegenden Tabelle. Die Klasse `ausleihe` stellt Methoden zum Eintragen, Verlängern und Austragen zur Verfügung. Diese Methoden bestehen nur aus einem SQL-Befehl. Die Methode `init($lname, $pwd, $bid)` initialisiert eine Instanz der Klasse `ausleihe`. Sie setzt die Attribute der Klasse anhand des ausgeliehenen Buches. Um alle Ausleihen für einen Kunden zu bekommen wird die Methode `getAllAusleihen($lname, $pwd, $kid)` implementiert.

Die Klasse `vorbest` besteht aus den Methoden zum Initialisieren, Vorbestellen, Benachrichtigen und Löschen einer Vorbestellung. Die Methode `init()` ist gleich aufgebaut wie die gleichnamige Methode in der Klasse `ausleihe`.

In der Datenhaltungsschicht werden noch zwei Testfunktionen bereitgestellt. Die Funktion `ist_verliehen($lname, $pwd, $bid)` überprüft, ob ein Buch ausgeliehen ist. Dazu stellt sie eine Anfrage an die Datenbank, ob das

Buch mit der ID `$bid` in der Ausleihtabelle eingetragen ist. Wenn das Buch nicht verliehen ist liefert die Anfrage eine leere Ergebnismenge zurück. Die Funktion überprüft nun, ob die Ergebnismenge nicht leer ist und liefert dann entweder `TRUE` oder `FALSE` zurück.

```
function ist_verliehen($lname, $pwd, $bid){
  if (mysql_connect("localhost", $lname, $pwd)){
    $result = mysql_db_query("bib", "SELECT * FROM ausleihe
                                     WHERE BuchID = \"$bid\"");
    if(mysql_num_rows($result) != 0){
      return true;
    } else {
      return false;
    }
  }
  return false;
}
```

Die Funktion `ist_vorbestellt($lname, $pwd, $bid)` ist identisch aufgebaut. Sie schaut lediglich in der Tabelle `vorbest` nach, ob das Buch eingetragen ist.

Datenbearbeitungsschicht

Die interessanten Funktionen in der Datenbearbeitungsschicht sind die Funktionen zum Berechnen des Enddatums der Ausleihe und zum Drucken der Ausleihkarten. Die Berechnung des Enddatums übernimmt die Funktion

```
function calc_enddat($adauer, $lname, $pwd){
  $tag = date("d");
  $monat = date("m");
  $jahr = date("Y");
  $enddat = date("Y-m-d", mktime(0,0,0,$monat,$tag + $adauer,$jahr));
  $enddat = teste_datum($enddat, $lname, $pwd);
  return $enddat;
}
```

Die Funktion `teste_datum($enddat, $lname, $pwd)` testet, ob an dem errechneten Enddatum die Bibliothek geöffnet hat. Dazu benutzt sie die Funktion `feiertag($enddat)`. Die Funktion `feiertag($enddat)` berechnet die gesetzlichen Feiertage. Es gibt feste Feiertage, die immer auf demselben Datum liegen und es gibt Feiertage, die von Ostern abhängen. Ostern ist immer am ersten Sonntag nach dem ersten Frühjahrsvollmod. Um diesen Tag zu berechnen bietet PHP die Funktion `easter_days()`. Sie liefert die Differenz in Tagen zwischen Frühjahrsanfang (21. März) und Ostern. In dem folgenden Programmsegment wird diese Berechnung ausgeführt. `$dat[0]` steht hier für das Jahr.

```
//Ostern abhängige Feiertage:
//Fastnacht Ostermontag - 7 Wochen
//Christi Himmelfahrt: Ostersonntag + 39 Tage,
```

```
//Pfingsten: Ostersonntag + 49 Tage,
//Fronleichnahm: Ostersonntag + 60 Tage,
  $abst = easter_days($dat[0]);
  $fastnacht = date("md", mktime(0,0,0,3,21 + $abst - 47, $dat[0]));
//  echo $fastnacht, "<br>";
  $himmel = date("md", mktime(0,0,0,3,21 + $abst + 39, $dat[0]));
//  echo $himmel, "<br>";
  $pfmo = date("md", mktime(0,0,0,3,21 + $abst + 50, $dat[0]));
//  echo $pfmo, "<br>";
  $fron = date("md", mktime(0,0,0,3,21 + $abst + 60, $dat[0]));
//  echo $fron, "<br>";
  $karfr = date("md", mktime(0,0,0,3,21 + $abst - 2, $dat[0]));
//  echo $karfr, "<br>";
  $ostmo = date("md", mktime(0,0,0,3,21 + $abst + 1, $dat[0]));
//  echo $ostmo, "<br>";
```

Falls das an die Funktion `feiertag` übergebene Datum auf einen Samstag, Sonntag oder Feiertag fällt, liefert die Funktion `TRUE` zurück. `teste_datum()` erhöht nun das Datum um einen Tag und überprüft dieses Datum wieder. Diese Schleife durchläuft die Funktion solange, bis sie ein Datum gefunden hat, an dem die Bibliothek geöffnet ist. Dieses Datum gibt sie zurück.

Das Drucken der Ausleihzettel wird auf ein Postscript-Programm `zettelheader.ps` aufgebaut. Diese Programm wurde schon in IBBIB verwendet. Die Funktion

```
function drucke_karte($kd, $bd, $bis){
  $ses_temp_dir = "/tmp";
  $von = date("Y-m-d");
  $fp1 = fopen("zettelheader.ps", "rb");
  $content = fread ($fp1, filesize("zettelheader.ps"));
  fclose($fp1);
  $outfname = $ses_temp_dir . "/zettel.ps";
  $fp2 = fopen($outfname, "w");
  fwrite ($fp2, $content);
  $sig = uml2nqtex($bd[signatur]);
  $teil = strtok($sig, " ");
  $i = 1;
  while ($teil){
    $sigf[$i] = $teil;
    $i++;
    $teil = strtok(" ");
  }
  if ($i == 3) {
    $sigf[4] = "";
  }
  fwrite ($fp2, sprintf ("(%s) (%s) (%s) (%s) (%s) (%s)\n",
             uml2nqtex($bd[autoren]), uml2nqtex($bd[titel]),
             $sigf[1], $sigf[2], $sigf[3], $sigf[4]));
  fwrite ($fp2, sprintf ("(%s %s) (%s) (%s) (%s) (%s) \n",
             uml2nqtex($kd[Vorname]), uml2nqtex($kd[Name]),
```

```
              uml2nqtex($kd[Dienstanschrift]), $kd[Teld],
              $von, $bis));
  fwrite ($fp2, "() \n");
  fwrite ($fp2, "PrintZettel\n\n");
  passthru($printcmd . $outfname, $temp);
  passthru($printcmd . $outfname, $temp);
}
```

liest den Postscriptcode ein, ergänzt diesen um die fehlenden Angaben und schreibt das Ergebnis in eine neue Postscript-Datei. Diese Datei wird dann zweimal an den Drucker gesendet und vom Drucker interpretiert und ausgedruckt.

4.2.4 Buchhaltung

Tabellenstruktur

Durch das Modul `Buchhaltung` wird die Lokaldatenbank um zwei Tabellen ergänzt. Die Tabelle `konten` beinhaltet alle Informationen zu einem Konto. Die Struktur dieser Tabele wird durch folgenden SQL-Befehl erzeugt:

```
CREATE TABLE konten (KID INT AUTO_INCREMENT PRIMARY KEY,
                     Knr VARCHAR(50) NOT NULL,
                     KontoName VARCHAR(255),
                     Hauptkonto INT DEFAULT NULL,
                     Saldo DECIMAL(9,2) DEFAULT "0.00",
                     Budget DECIMAL(9,2) DEFAULT "0.00",
                     Buchungsstelle VARCHAR(255));
```

Das Feld `Hauptkonto` beinhaltet die Referenz auf das übergeordnete Konto. Die Datenfelder `Saldo` und `Budget` sind vom Datentyp `DECIMAL`. Dieser Datentyp ist ein Fix-Point-Datentyp und hat hier insgesamt 9 Stellen von den 2 Nachkommastellen sind. Es können somit Beträge bis zu 9.999.999,99 Euro gespeichert werden. Diese Höhe reicht auf jeden Fall für die Fachrichtungsbibliothek, die ein Jahresbudget von ca. 25.000 Euro hat.

Die Tabelle `buchungen` speichert die Angaben über eine Kontobewegung. Die Struktur wird durch den folgenden SQL-Befehl festgelegt.

```
CREATE TABLE buchungen (BID INT AUTO_INCREMENT PRIMARY KEY,
                        Kto INT NOT NULL,
                        Wert DECIMAL(9,2),
                        ExID INT,
                        BestID INT,
                        ReNr VARCHAR(255),
                        BuchNr VARCHAR(255),
                        Beschr VARCHAR(255),
                        Kommentar TEXT,
                        Typ VARCHAR(15),
                        Abgleich INT DEFAULT NULL,
```

```
                        ErstDat DATE,
                        RechDat DATE DEFAULT NULL,
                        AbglDat DATE DEFAULT NULL);
```

Datenhaltungsschicht

In der Datenhaltungsschicht werden die Klassen `Konto` und `Buchung` implementiert. Durch die in Kapitel 3 besprochene Baumstruktur der Konten werden die Methoden `deleteKonto($lname, $pwd)` und `calcSaldo($lname, $pwd, $konto)` der Klasse `Konto` rekursiv konstruiert.

```
function calcSaldo($lname, $pwd, $konto){
  $b = new Buchung;
  $b->kto = $konto;
  $summe = $b->sumBuchKto($lname, $pwd, $konto);
  $list = getUnterkonten($lname, $pwd, $konto);
  if (!empty($list)){
    foreach($list as $key=>$value){
      $summe += calcSaldo($lname, $pwd, $key);
    }
  }
  $k = new Konto;
  $k->ID = $konto;
  $k->readDB($lname, $pwd);
  $k->saldo = $summe;
  $k->writeDB($lname, $pwd);
  return $k->saldo;
}

function deleteKonto($lname, $pwd){
  //löscht ein Konto rekursiv (d.h. mit allen Unterkonten)
  $list=getUnterkonten($lname, $pwd, $this->ID);
  if (!empty($list)){
    foreach ($list as $key=>$value){
      $k = new Konto;
      $k->ID = $key;
      $k->deleteKonto;
    }
  } else {
    $b=new Buchung;
    $b->kto = $this->ID;
    $buchl = getBuchKto($lname, $pwd);
    if (!empty($buchl)){
      foreach ($buchl as $key=>$value){
        if ($this->hauptkonto = ""){
          $value->deleteBuch($lname, $pwd);
        } else {
```

```
            $value->moveBuch($lname, $pwd, $this->hauptkonto);
          }
        }
      }
      $sql = "DELETE FROM konten WHERE KID = \"$this->ID\"";
      if (mysql_connect("localhost", $lname, $pwd)){
        mysql_db_query("bib", $sql);
      }
    }
  }
```

Die Methode `deleteKonto($lanme, $pwd)` verschiebt die enthaltenen Buchungen zum übergeordneten Konto. Nur wenn das zu löschende Konto ein Hauptkonto ist, es also keine übergeordneten Konten hat, werden die enthaltenen Buchungen gelöscht.

Die Summe aller Buchungen eines Kontos berechnet die Methode `sumBuchKto($lname, $pwd)` der Klasse `Buchung`. Sie benutzt dafür die Möglichkeit der Datenbank Felder aufzusummieren:

```
  function sumBuchKto($lname, $pwd){
    $sql = "SELECT SUM(Wert) as Summe FROM buchungen WHERE Kto = \"$this->kto\"";
    if (mysql_connect("localhost", $lname, $pwd)){
      $result = mysql_db_query("bib", $sql);
      $erg = mysql_fetch_array($result, MYSQL_ASSOC);
      return $erg[Summe];
    }
    return 0;
  }
```

Datenbearbeitungsschicht

Die Datenbearbeitungsschicht der Buchhaltung ist nicht sehr interessant, da die Berechnungen für die Buchhaltung alle auf Datenbankebene ablaufen. Die einzige Aufgabe, die die Datenbearbeitungsschicht in der Buchhaltung hat, ist die Daten für die Darstellungsschicht aufzubereiten.

4.2.5 Basismodul

Die Aufgaben des Basismoduls bestehen aus der Benutzerverwaltung, der Zugriffskontrolle und der Konfiguration. Da die Implementierung der Benutzerverwaltung und der Konfiguration nur daraus besteht Daten zu speichern und bereitzustellen wird hier nicht näher auf die Implementierung eingegangen.

Die Zugriffskontrolle kann man in zwei Teile aufteilen. Der erste Teil ist der Schutz vor unerlaubten Zugriff Dritter. Der zweite Teil besteht darin, dass nicht jeder Benutzer die gesamte Funktionalität von BiVIS nutzen darf.

Um das System vor unerlaubten Zugriffen zu schützen, muss sich jeder Benutzer beim Start anmelden. Der Anmeldevorgang besteht darin sein Login-Name und Passwort dem System mitzuteilen. Das System überprüft nun die Anmeldetaten, indem es versucht den Benutzer an der Datenbank anzumelden. Die Authentifizierung über die Datenbank hat den Vorteil, dass wenn die Datenbank nicht verfügbar ist bekommt der Anwender dies schon direkt beim Anmeldeversuch mitgeteilt. Ein Arbeiten mit dem System ohne Datenbankzugriff ist somit nicht möglich. Wenn das System den Benutzer bei der Datenbank erfolgreich angemeldet hat, liest es die Anwenderdaten aus der Datenbank und startet eine Session. Eine solche Session bietet die Möglichkeit sogenannte Session-Variablen abzuspeichern. In diesen Variablen wird zum Beispiel der Login-Name und das Passwort des Benutzers abgespeichert. Beim Anmelden wird auch noch eine boolsche Variable `$val_ses`) mit dem Wert `TRUE` registriert. Beim Starten jedes Skriptes der Darstellungsschicht werden die Session-Daten eingelesen und überprüft. Sollte die Session nicht mehr gültig sein, wird der Benutzer angewiesen sich neu anzumelden. Diese Überprüfung erfolgt mit folgendem Codesegment:

```
session_start();
if(! $val_ses) {
  session_not_valid();
} else {
  //Auszuführender PHP-Code
}
```

Die Funktion `session_not_valid()` aus der Datei `standard.php` erstellt die HTML-Seite mit der Information, dass die Session nicht mehr gültig ist.

Die Zugriffsrechte auf bestimmte Programmteile ist auf die Module beschränkt. Die Datenbank enthält eine Tabelle mit den Information über die in BiVIS verfügbaren Module.

```
CREATE TABLE modules (ModuleID INT AUTO_INCREMENT PRIMARY KEY,
                      ProgName VARCHAR(255) NOT NULL,
                      URI VARCHAR(255) NOT NULL);
```

Die Tabelle `access` enthält eine Matrix von BenutzerIDs und ModulIDs. In dieser Matrix ist festgelegt, ob ein Benutzer das entsprechende Modul ausführen darf. Die Kontrolle hierüber erfolgt beim Systemstart. Jeder Benutzer bekommt sein individuelles Hauptmenü angezeigt. Gesperrte Module werden nicht angezeigt.

Beim Installieren des Basismoduls wird der Administrator-Account für die Datenbank und BiVIS erstellt. Im Folgenden wird das Installationsskript für die Datenbank gezeigt:

```
DROP DATABASE IF EXISTS bib;
CREATE DATABASE bib;
USE bib;
GRANT ALL ON *.* TO bivisadmin@localhost IDENTIFIED BY "bivis"
          WITH GRANT OPTION;

CREATE TABLE modules (ModuleID INT AUTO_INCREMENT PRIMARY KEY,
                      ProgName VARCHAR(255) NOT NULL,
                      URI VARCHAR(255) NOT NULL);

INSERT INTO modules SET ProgName = "Administration",
                        URI = "konfig.php";
```

```
INSERT INTO modules SET ProgName = "Einstellungen",
                        URI = "einstell.php";

CREATE TABLE user (UserID INT AUTO_INCREMENT PRIMARY KEY,
                   Login VARCHAR(255) NOT NULL,
                   Name VARCHAR(255) NOT NULL,
                   Vorname VARCHAR(255) NOT NULL,
                   Strasse VARCHAR(255),
                   PLZ VARCHAR(10),
                   Ort VARCHAR(255),
                   Email VARCHAR(255),
   Pwd VARCHAR(255) NOT NULL);

INSERT INTO user (Login, Name, Vorname, Strasse, PLZ, Ort, Email)
       VALUES ("bivisadmin", "Kröper", "Dag",
               "Im Maigen 8", "66693", "Mettlach",
               "kroeper@cs.uni-sb.de");

CREATE TABLE access (UserID INT NOT NULL,
                     ModuleID INT NOT NULL,
                     Allow INT NOT NULL);

INSERT INTO access SET UserID = 1, ModuleID = 1, Allow = 1;
INSERT INTO access SET UserID = 1, ModuleID = 2, Allow = 1;

CREATE TABLE lok_def (MAB_ID INT PRIMARY KEY,
                      Name varchar(255),
                      Descr TEXT);


DROP DATABASE IF EXISTS stamm;
CREATE DATABASE stamm;
USE stamm;
GRANT ALL ON *.* TO bivisadmin@localhost IDENTIFIED BY "bivis"
          WITH GRANT OPTION;

CREATE TABLE stamm_def (MAB_ID INT PRIMARY KEY,
                        Tabelle varchar(255) NOT NULL,
                        Name varchar(255),
                        Descr TEXT);

USE mysql;
GRANT ALL ON *.* TO bivisadmin@localhost IDENTIFIED BY "bivis"
          WITH GRANT OPTION;
```

4.3 Benutzungsschnittstelle von BiVIS

Da eine komplette Beschreibung der Benutzungsschnittstelle (GUI) sehr umfangreich wäre, wird sie hier anhand der Titelaufnahme erläutert.

Die Benutzungsschnittstelle wird in zwei Teilen implementiert. Der erste Teil beschäftigt sich nur mit der Visualisierung. Im zweiten Teil werden die Kontrollstrukturen implementiert.

Für die Visualisierung ist das PHP-Skript `formulare.php` verantwortlich. Dieses Skript stellt die verschiedenen Formulare als Funktionen zur Verfügung.

In Abbildung 4.3 wird die Ausgabe des Anmeldeformulares gezeigt.

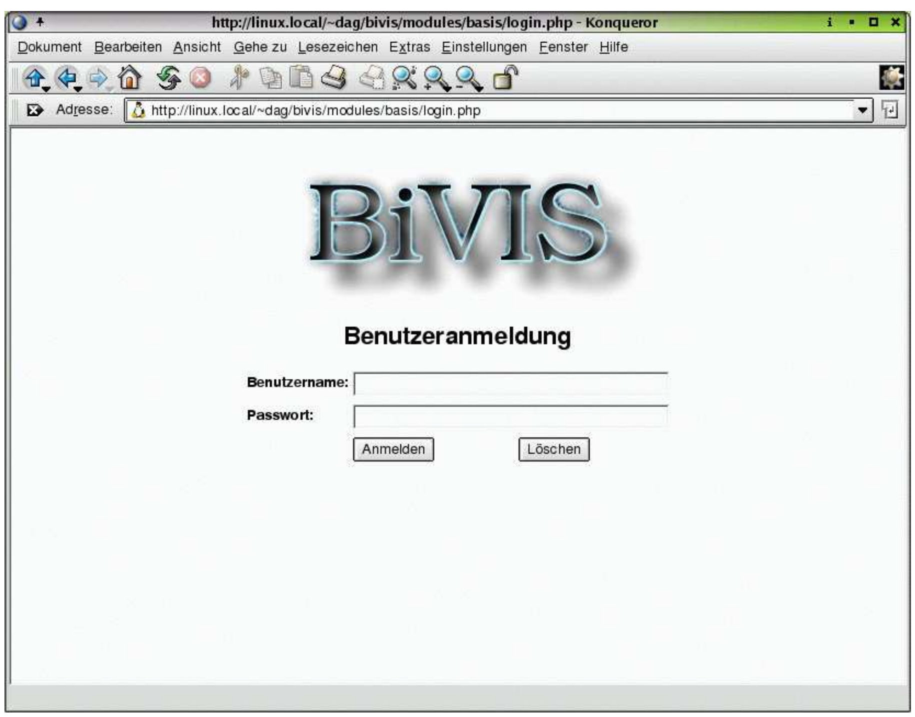

Abbildung 4.3: Anmeldeformular von BiVIS

Nach erfolgter Anmeldung durch das Kontrollskript `login.php` wird das Skript `start.php` aufgerufen. In `start.php` wird das Hauptmenü (Abbildung 4.4) generiert und dargestellt.

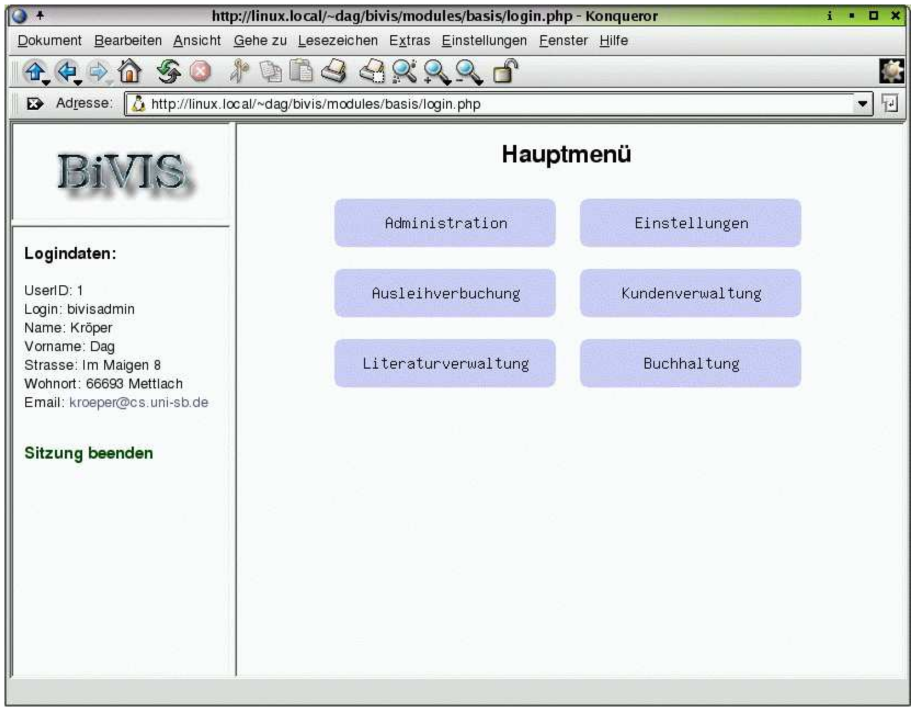

Abbildung 4.4: Hauptmenü von BiVIS

Das Hauptmenü bietet den Aufruf der einzelnen Module an. Dies ist durch Links realisiert. Durch das Anklicken einer Schaltfläche gelangt man zu dem Kontrollskript, das das Menü des entsprechenden Modules erzeugt. In Abbildung 4.5 wird die Ausgabe des Kontrollskriptes `literatur.php` gezeigt.

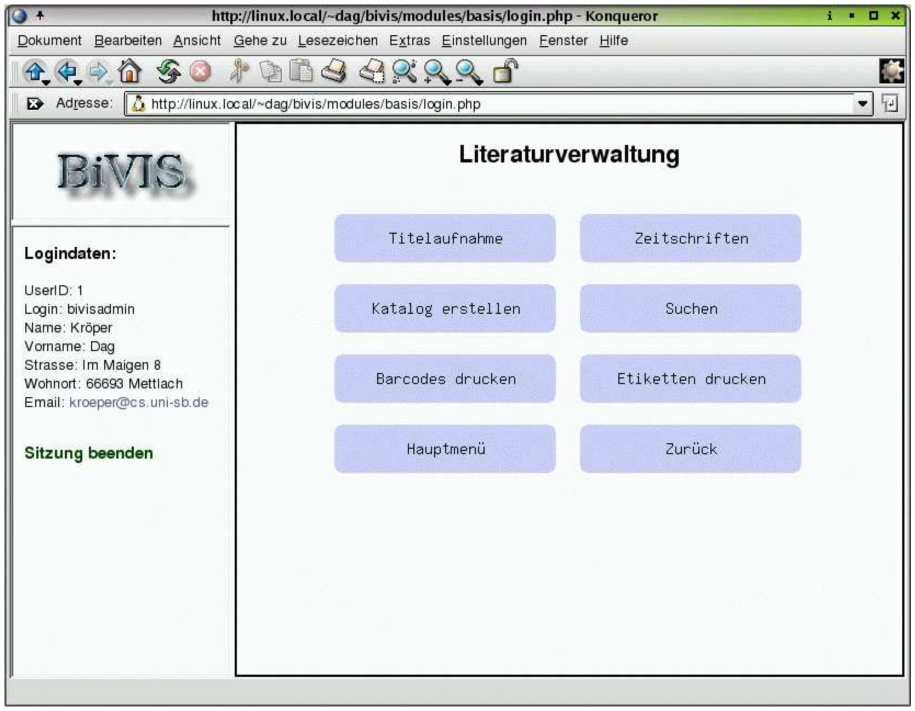

Abbildung 4.5: Literaturverwaltung

Dieses Menü ist wie das Hauptmenü mit Links realisiert. Durch Anklicken der Schaltfläche `Titelaufnahme` wird die Kontrolle an das Skript `titelaufn.php` weitergegeben. Es erscheint das Menü für die Titelaufnahme (Abbildung 4.6).

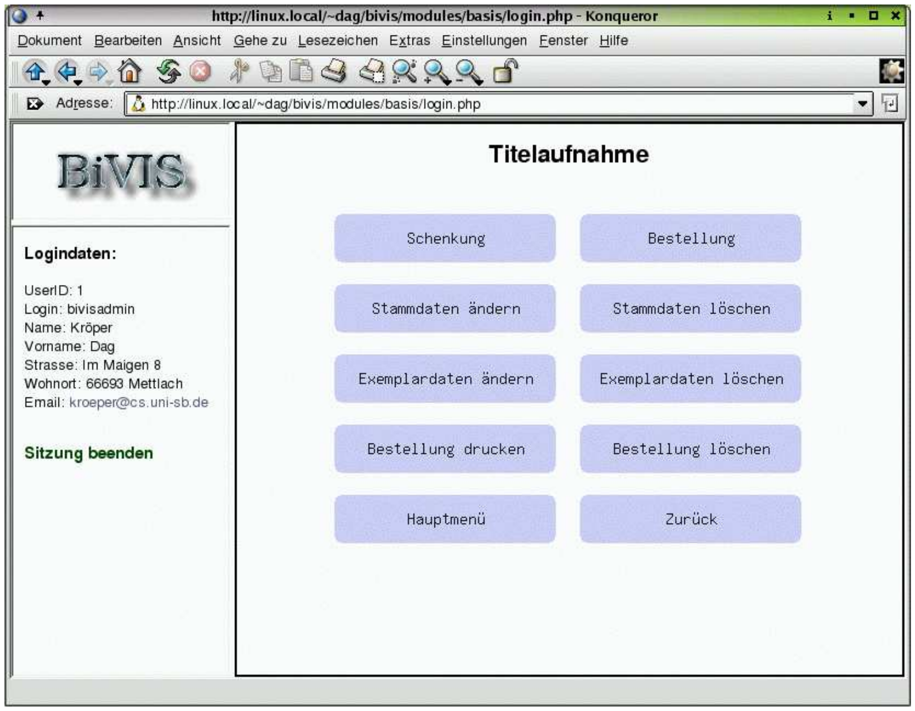

Abbildung 4.6: Menü der Titelaufnahme

Durch Anklicken der Schaltfläche `Schenkung` wird das Skript `suchestamm.php` aufgerufen. Dieses Skript bekommt als Parameter den Skriptname des Skriptes mit, das ausgeführt werden soll, wenn die Auswahl des Stammdatensatzes abgeschlossen ist. Dieses Skript ruft die Formularfunktion `suche_stamm_form("suchestamm.php")` auf. Diese Funktion erzeugt das Eingabeformular aus Abbildung 4.7.

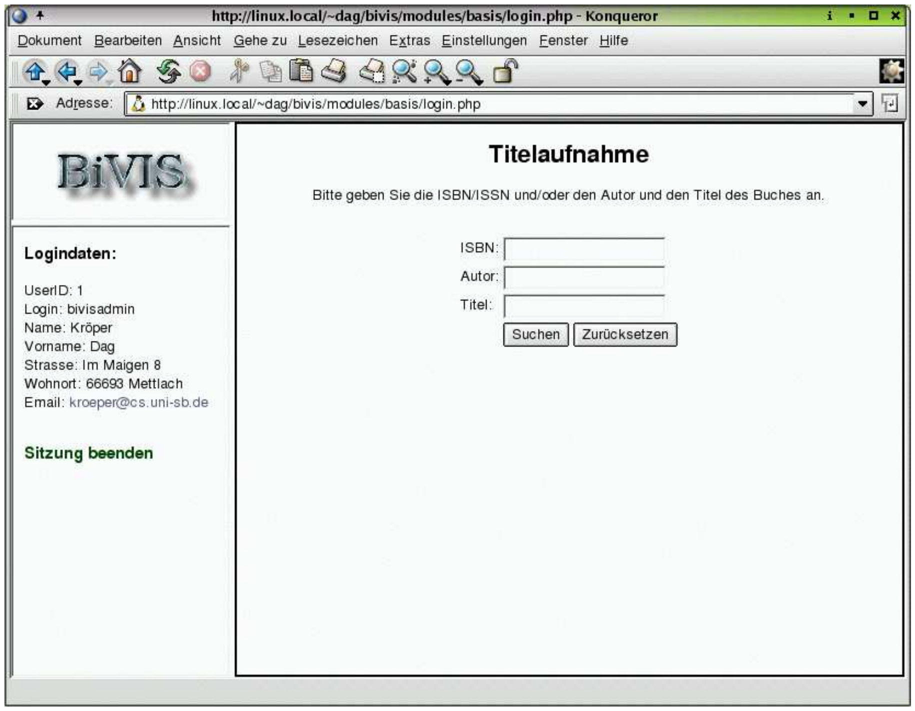

Abbildung 4.7: Formular zum Suchen eines Stammdatensatzes

Nach Eingabe der Suchdaten wird ein passender Stammdatensatz gesucht. Wenn ein oder mehrer Datensätze gefunden werden, wird durch das Skript `suche_stamm()` die Funktion `select_result_form()` aufgerufen. Diese Funktion erstellt eine Auswahlliste der gefundenen Stammdatensätze. Wird kein Stammdatensatz gefunden wird die Möglichkeit angeboten eine neue Suche einzugeben oder einen neuen Stammdatensatz anzulegen. Hier wird der Weg beschrieben einen neuen Stammdatensatz anzulegen. Dazu wird das Skript `createstamm.php` aufgerufen. Es bekommt als Parameter die eingegebenen Suchmuster. Das Skript `createstamm.php` ruft die Funktion `create_stamm_form()`, die die Eingabemaske (Abbildung 4.8) generiert.

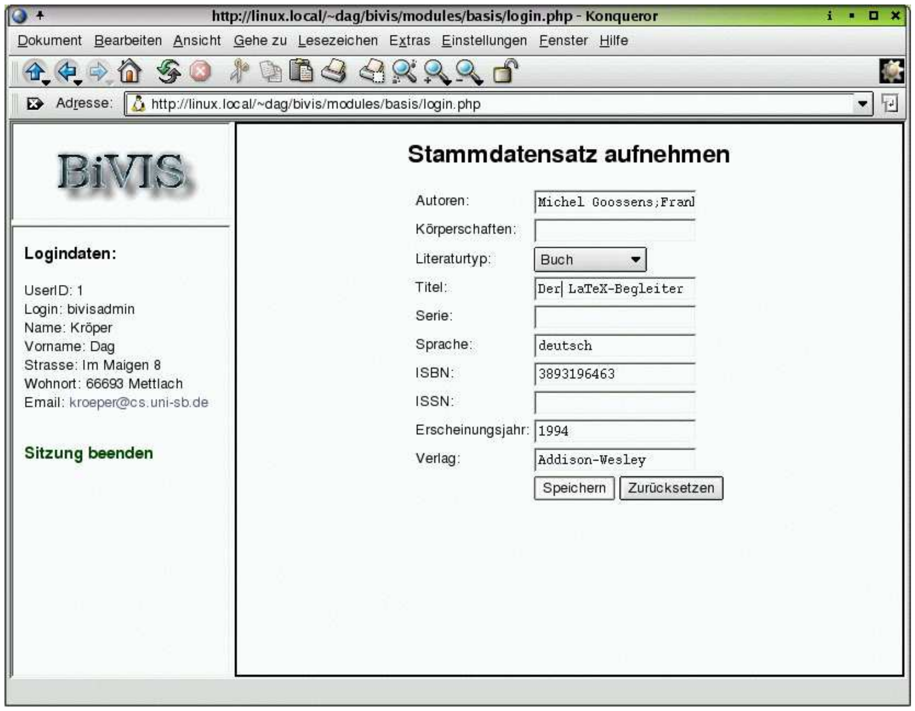

Abbildung 4.8: Eingabeformular der Stammdaten

Während dem Speichern der Stammdaten werden die Eingaben getestet (Korrektheit der ISBN, Überprüfen von Autor, Körperschaft...). Jetzt kommt der Parameter des Skriptes `suchestamm.php` zum tragen. Er wurde über sämtliche Skripte weitergereicht. Es wird nun dieses Skript aufgerufen. In diesem Fall ist es `buchexneu.php`. Dieses Skript steuert die Generierung eines Exemplardatensatzes. Dazu ruft es die Funktion `create_exemplar_form()` auf. Diese Funktion erzeugt die Eingabemaske für die Exemplardaten (Abbildung 4.9).

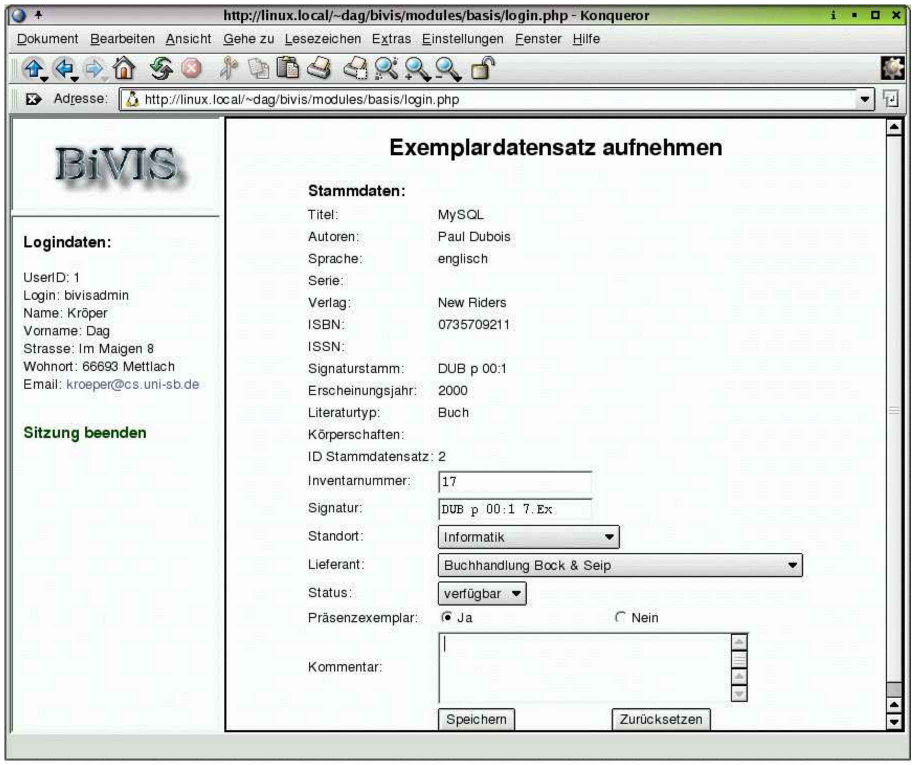

Abbildung 4.9: Eingabeformular der Exemplardaten

Zum Abschluss werden die Exemplardaten gespeichert und die Anzeige springt wieder zurück zur Literaturverwaltung.

Beim Betrachten dieses Beispieles wird ersichtlich, dass die Benutzungsschnittstellen nur aus modalen Dialogen besteht. Normalerweise sollte man bei Benutzungsschnittstellen modale Dialoge vermeiden. Das ist hier aber, durch die Verwendung von HTML-Seiten als Benutzungsoberfläche, nicht möglich. Die Benutzerführung durch BiVIS ist an die Arbeitsweisen in der Bibliothek angepasst. Das wird auch aus diesem Beispiel ersichtlich.

Kapitel 5

Zusammenfassung

In dieser Arbeit wurde die Planung und die Realisierung eines neuen Bibliotheksverwaltungssystem dargestellt. Die Problematik in der Planungsphase des Systems war es, das neue System zu modellieren. Als erster Schritt der Planungsphase steht die Beschreibung des Bibliothekalltags. Daraus werden einzelne Aufgaben, die das System später haben soll, ersichtlich. Als nächste wurde das bestehende System analysiert. Daraus ergaben sich zum großen Teil die Daten, die permanent gespeichert werden.

Der nächste Punkt war die Auswahl des Prozess-Modells. Hier wurden zwei Modelle verglichen und nach Abwägung der Vor- und Nachteile, das aus meiner Sicht, bessere Prozessmodell ausgewählt. Jetzt konnten die Anforderungen an das neues Bibliothekssystem festgehalten werden. Dies geschah in Zusammenarbeit mit der Bibliotheksleitung. Als Ergebnis entstand das Pflichtenheft (Anhang A).

Aufbauend auf dem Pflichtenheft wurde BiVIS entworfen. Bei dem Entwurf wurde die Grundstruktur von BiVIS, das Client-Server-Modell und der Modulare Aufbau, festgelegt. Desweiteren wurden die Schnittstellen definiert. Als weiterer Punkt steht die Entwicklung der Datenbank. Hier wurden die Zusammenhänge der persistent zu speichernden Daten anhand von ER-Modellen erläutert.

Die Implementierung bildete den Abschluss dieser Arbeit. Hier wurde zuerst entschieden, welche Programmiersprache und welches Datenbanksystem verwendet wird. Als nächstes wurde auf die Schwerpunkte der Implementierung eingegangen. Es wurde die Struktur der Tabellen und die Implementierung wichtiger Methoden und Funktionen gezeigt. Zum Abschluss wurde die Benutzungsschnittstelle vorgestellt. Es wurde dort der modale Aufbau der Dialoge begründet und an einem Beispiel der Ablauf der GUI erläutert.

Anhang A

Pflichtenheft

A.1 Funktionalität von BiVIS aus der Sicht der Anwender und Kunden

Wir betrachten hier die Funktionalität, die BiVIS aus der Sicht der Anwender und Kunden mitbringt. Dies kann man am anschaulichsten darstellen, indem man die Abläufe in der Fachrichtungsbibliothek beschreibt.

A.1.1 Bestellung und Schenkung von Büchern und Zeitschriften

Was wäre eine Bibliothek ohne Bücher oder wie kommen die Bücher in die Bibliothek? Es gibt hier grundsätzlich zwei Wege: die Bücher werden bestellt oder die Bibliothek bekommt die Bücher geschenkt. Der wichtigste Unterschied ist, dass es bei der Bestellung eine Rechnung gibt, die bezahlt werden muss. Also muss bei einer Bestellung ein Rechnungsdatensatz erzeugt werden.

Bestellung von Büchern

Betrachten wir nun den Vorgang der Bestellung. Bei einer Bestellung gibt der Anwender das zu bestellende Buch mit einigen Zusatzinformationen in das System ein. Diese Zusatzinformationen sind zum Beispiel der Lieferant, voraussichtlicher Preis oder Anzahl der Exemplare. Aus den eingegebenen Daten erstellt das System automatisch die entsprechende Einträge in der Lokaldatenbank. Das System druckt nun auf Anweisung des Benutzers einen Bestellbrief für einen Lieferanten aus, oder versendet die Bestellung per E-Mail.

Zu diesem Zeitpunkt wurde von dem System ein Literaturstammdatensatz für jeden bestellten Titel und ein Lokaldatensatz und ein Rechnungsdatensatz für jedes bestelle Exemplar dieses Titels angelegt. Der Status jedes bestellten Exemplares wird auf 'bestellt' gesetzt.

Der Biblitotheksmitarbeiter muss nun die Lieferung der Bücher überwachen. Das System unterstützt ihn dabei, indem es dem Anwender mitteilt, welche Bestellungen, nach einem voreingestellten Zeitraum, noch nicht eingetroffen sind.

Trifft ein Buch ein, so wird der Status dieses Exemplars gesetzt und die Signatur zum Druck freigegeben. Der Rechnungsdatensatz kann jetzt aktualisiert werden. Das bestellte Buch ist nun für die Kunden der Bibliothek verfügbar.

Schenkung

Wird ein Buch als einen Schenkung inventarisiert, dann wird nur ein Stammdatensatz (falls noch nicht vorhanden) und ein Lokaldatensatz erzeugt. Da das Buch im allgemeinen schon vorliegt, kann man direkt den Status auf einen entsprechenden Wert setzen und die Signatur zum drucken freigeben.

Zeitschriften

Die Bestellung ist bei Zeitschriften identisch zu den Büchern. Es unterscheidet sich nur die Überwachung der Lieferung. Da Zeitschriften in bestimmten Zeitabständen erscheinen, muss der Eingang jedes Heftes überprüft werden. Wenn ein Jahrgang komplett ist wird er zum Binden freigegeben und in das Zeitschriftenarchiv eingestellt.

A.1.2 Kunden und Ausleihe

Kunden

Für jeden Kunden der Bibliothek wird ein Kundendatensatz, der Angaben zur Person und deren Ausleihen speichert. Damit eine Person ausleihberechtigt wird, muss er einen schriftlichen Antrag an die Bibliothek stellen. Nach dem Erstellen des Kundendatensatzes werden die Kundenangaben auf eine Kundenkarteikarte gedruckt. Der Kunde erhält daraufhin eine Begrüssungs E-Mail und ist ab diesem Zeitpunkt ausleihberechtigt in der Bibliothek für die Dauer eines vorher festgelegten Zeitraums.

Ausleihe

Ein ausleihberechtigter Kunde kann Bücher, die nicht von der Ausleihe gesperrt sind, für einen festgelegten Zeitraum ausleihen. Die Bibliothek erlaubt keine Selbstverbuchung, d. h. der Kunde muss das Buch von der Aufsicht als Ausleihe verbuchen lassen. Zur Ausleihverbuchung gibt der Anwender die Kundennummer des Ausleiher und die Signaturen der Bücher ein. Das System speichert die Ausleihe, das Ausleihdatum und das Ausleihendedatum und druckt für jedes ausgeliehenes Buch einen Ausleihzettel. Der Ausleiher hat die Möglichkeit nach Ablauf der Leihfrist das Buch um die eingestellte Leihfrist verlängern zu lassen. Bei Rückgabe der Bücher wird das Rückgabedatum gespeichert, der Datensatz anonymisiert und für statistische Zwecke erhalten. Die Ausleihendedaten werden regelmäßig von dem Bibliothekssystem automatisch überprüft. Sollte eine Leihfrist überschritten werden mahnt das System automatisch die ausstehenden Bücher bei dem Ausleiher an. Dies erfolgt per E-Mail und als Brief.

Vorbestellung

Falls ein Buch verliehen sein sollte kann ein Kunde dieses Buch für sich vorbestellen lassen. Der Ausleiher dieses kann es nicht mehr verlängern lassen und muss es nach Ablauf der Leihfrist zurückbringen. Nach Rückgabe des Buches wird der Vorbesteller automatisch benachrichtigt. Nur der Vorbesteller kann jetzt dieses Buch ausleihen. Nach einem festgelegten Zeitraum wird der Vorbestellungsvermerk gelöscht und das Buch steht wieder für alle Ausleiher bereit.

A.2 Zielbestimmungen

Das vorhandene Bibliotheksverwaltungssystem *IBBIB* der Fachrichtungsbibliothek Informatik soll durch ein wartungsfreundlicheres System ersetzt werden. Dabei darf keine bestehende Funktionalität verloren gehen.

A.2.1 Musskriterien

- Titelaufnahme
- Abgleichen der Autoren, Serien, Körperschaften und Verlage während der Titelaufnahme mit Stammdaten.
- Bestellungen aufnehmen/drucken oder versenden (E-Mail).
- Lieferungen bearbeiten, verwalten und ausstehende Lieferungen anmahnen.
- Zeitschriften bestellen und verwalten.
- Kunden eintragen, bearbeiten und löschen.
- Ausleihe eintragen, verlängern, austragen und ausstehende Ausleihen automatisch anmahnen.
- Vorbestellungen zur Ausleihe aufnehmen und automatische Benachrichtigung wenn die Ausleihe verfügbar ist.
- Export nach MAB2 zum Datenaustausch mit der Universitätsbibliothek.
- Katalogerstellung zum Datenaustausch mit IBIS.
- Buchhaltung: Konten einrichten, Budget verwalten, Statistiken erstellen.
- Leichtes Portieren auf andere Plattformen.
- Zugangskontrolle: Nicht jeder Anwender hat die selben Aufgaben und damit unterschiedliche Zugangsrechte auf das Programm und die Datenbank. Ausserdem sollen nicht Dritte von ausserhalb (Internet) auf das System zugreifen können.

A.2.2 Wunschkriterien

- Inventurunterstützung
- Import externen Daten (MAB 2, Refer)
- Statistik für Ausleihverbuchung
- Web-OPAC
- freie Erweiterbarkeit der Kundendaten

A.2.3 Abgrenzungskriterien

- Kein erstellen eigener Korrespondenz

A.3 Produkteinsatz

Das Produkt dient zur Verwaltung der Fachrichtungsbibliothek Informatik.

A.3.1 Anwendungsbereiche

- Literaturbestandsverwaltung
  - Bestellen neuer Literatur
  - Eintragen neuer Literatur in die Datenbestände
  - Verfügbarkeitsstatus für jedes Exemplar festlegen
  - Rechnungsstatus für jedes Exemplar führen
    - Bestellung: Vorraussichtliche Kosten für dieses Exemplar (Bestellwert)
    - Rechnungseingang: auf der Rechnung ausgewiesener Betrag
    - Nach Bezahlung: tatsächlicher Betrag der überwiesen wurde (kann durch Umrechnungskurse von Rechnungseingang variieren)
- Kundenstammverwaltung
- Ausleihverbuchung

A.3.2 Zielgruppen

MitarbeiterInnen der Fachrichtungsbibliothek Informatik

A.3.3 Betriebsbedingungen

Büroumgebung

A.4 Produkt-Umgebung

Das Produkt hat eine Client-Server-Struktur.

A.4.1 Software

Server

- Unix Betriebssystem
- Web-Server
- Serverpage-scripting-Language
- Datenbank-Schnittstelle

Diese Software sollte standardisiert und leicht verfügbar sein. Deshalb wird als Web-Server Apache, als Serverpage Scripting Language PHP und als Datenbank-Schnitstelle MySQL verwendet.

Client

- Web-Browser der HTML 3.0 versteht.

A.4.2 Hardware

Als Hardware wird günstige Standardhardware verwendet (z.B. PC). BiVIS soll allerdings auch auf der bestehenden Hardware (SUN Workstations) laufen.

A.4.3 Orgware

Eine SSL-Netzwerkverbindung zwischen Client und Server.

A.5 Produkt-Funktionen

Im folgenden verwende ich den Begriff 'Buch' stellvertretend für die Gesamtheit der literarischen Werke.

A.5.1 Literaturbestandsverwaltung

Titelaufnahme

/F1100/ Literaturstammdaten aufnehmen
ermöglicht die Eingabe der Literaturstammdaten.

/F1110/ Literaturstammdaten löschen
ermöglicht einen Literaturstammdatensatz aus den Literaturstammdaten zu löschen. Es wird überprüft, ob der Datensatz noch mit einem anderen Datensatz verknüpft ist.

/F1120/ Literaturstammdaten bearbeiten
erlaubt das nachträgliche Bearbeiten der bereits aufgenommenen Stammdaten.

/F1121W/ Literaturstammdaten importieren
Mit dieser Funktion besteht die Möglichkeit Stammdaten von externen Quellen im MAB2-Format zu importieren.

/F1130/ Überprüfen auf doppelte Einträge
überprüft, ob der eingegebene Datensatz bereits in den Literaturstammdaten vorhanden ist.

/F1140/ Autoren überprüfen
überprüft, ob der eingegebene Autor/Editor in den Autorenstammdaten vorhanden ist.

/F1150/ Autorenstammdaten aufnehmen
ermöglicht es neue Autoren in die Autorenstammdaten aufzunehmen.

/F1160/ Autorenstammdaten löschen
löscht einen angegebenen Autor aus den Autorenstammdaten. Es wird überpfüft, ob der Autor noch mit einem anderen Datensatz verknüpft ist.

/F1170/ Autorenstammdaten bearbeiten
ermöglicht das Änderen eines bereits eingegebenen Autorendatensatzes.

/F1180/ Körperschaft überprüfen
Abgleich der Körperschaft mit der Körperschaftstammdatenbank.

/F1190/ Körperschaftstammdaten aufnehmen
ermöglicht es neue Körperschaften in die Körperschaftstammdaten aufzunehmen.

/F1200/ Körperschaftstammdaten löschen
überprüft, ob eine angegeben Körperschaft noch mit einem anderen Datensatz verknüpft ist. Falls nicht löscht sie den Datensatz.

/F1210/ Körperschaftstammdaten bearbeiten
ermöglicht einen bereits eingegebenen Körperschaftstammdatensatz zu ändern.

/F1220/ Verlag überprüfen
Abgleich der Verlagsdaten mit der Verlagstammdatenbank.

/F1230/ Verlagstammdaten aufnehmen
ermöglicht es neue Verlage in die Verlagstammdaten aufzunehmen.

/F1240/ Verlagstammdaten löschen
löscht einen angegebenen Verlag aus den Verlagstammdaten. Es wird überprüft, ob dieser Datensatz noch mit einem anderen Datensatz verknüpft ist.

/F1250/ Verlagstammdaten bearbeiten
ermöglicht einen bereits eingegebenen Verlag zu ändern.

/F1260/ Serie überprüfen
Abgleich der Seriendaten mit der Serienstammdatenbank

/F1270/ Serienstammdaten aufnehmen
ermöglicht es neue Serientitel in die Serienstammdaten aufzunehmen.

/F1280/ Serienstammdaten löschen
löscht einen angegebenen Serientitel aus den Serienstammdaten. Es wird überprüft, ob dieser Datensatz noch mit einem anderen Datensatz verknüpft ist.

/F1290/ Serienstammdaten bearbeiten
ermöglicht es einen bereits eingegebenen Serientitel in den Serienstammdaten zu ändern.

/F1300/ Lokaldaten aufnehmen
ermöglicht das Eingeben der Lokaldaten in die Lokaldatenbank.

/F1310/ Lokaldaten löschen
löscht einen Lokaldatensatz. Wenn dieser Datensatz noch mit einem anderen Datensatz verknüpft ist (ausser Stammdaten) ist das Löschen nicht möglich.

/F1320/ Lokaldaten bearbeiten
ermöglicht es einen angegebenen Lokaldatensatz zu löschen.

/F1330/ Signatur generieren
generiert nach vorgegebenen Regeln eine Signatur.

/F1331/ Signaturetiketten drucken.

/F1340/ Anmahnen ausstehender Lieferungen
mahnt ausstehende Lieferungen an, die die Lieferfrist überschritten haben.

/F1350/ Status setzen
setzt den Status eines Buches auf 'vermisst', 'ausgeliehen', 'vorhanden', 'bestellt', 'vergriffen', 'vorübergehend nicht verfügbar' oder 'gesperrt'.

/F1360/ Daten exportieren
exportiert die Literaturdaten in das MAB2-Format.

Zeitschriften

Hier werden die zusätzlichen Funktionen für die Zeitschriftenverwaltung definiert.

/F1500/ Zeitschriften Lieferung überprüfen
überprüft den Hefteingang der Zeitschriften anhand eines voreingestellen Zeitraums.

/F1510/ Anmahnen ausstehender Hefte
mahnt ausstehende Hefte an, die die Lieferfrist überschritten haben.

/F1520/ Übertragen kompletter Zeitschriftenbände (Volumes) in das Zeitschriftenarchiv

/F1530W/ Unterstützen beim Binden
soll beim Binden unterstützen. Sie soll alle nötigen Angaben (Einbandfarbe, Schriftfarbe, Buchrückenaufdruck) von einem zu bindenden Zeitschriftenband ausgeben.

Bestellungen

/F1700/ Bestellung aufnehmen
ermöglicht es Bestellungen aufzunehmen. Sie überprüft, ob ein bestelltes Buch schon in den Literaturstammdaten und in den Lokaldaten vorhanden ist. Ist das Buch nicht in den Literaturstammdaten vorhanden wird ein neuer Literaturstammdatensatz erzeugt.

/F1710/ Bestellung als Brief drucken

/F1720/ Bestellung als E-Mail senden

/F1730/ Bestellung überprüfen
überprüft anhand eines eingestellten Zeitraums ausstehende Bestellung.

/F1740/ Bestellung anmahnen
mahnt eine überfällige Bestellung bei dem entsprechenden Lieferanten an. Dies kann sowohl schriftlich als auch per E-Mail sein.

A.5.2 Kundenverwaltung

/F2100/ Kundendaten aufnehmen
ermöglicht neue Kunden in die Kundendaten aufzunehmen.

/F2110/ Kundendaten löschen
ermöglicht einen Kunden aus den Kundendaten zu löschen. Wenn dieser Kunde noch etwas ausgeliehen hat, kann der Datensatz nicht gelöscht werden.

/F2120/ Kundendaten bearbeiten
ermöglicht bereits eingetragene Kundendaten zu ändern.

/F2130/ Ausleihberechtigungsdauer eintragen
ermöglicht eine Ausleihberechtigungsdauer einzutragen, die von der Standardausleihdauer abweicht.

/F2140/ Ausleihberechtigungsdauer verlängern
ermöglicht es eine Ausleihberechtigungsdauer um einen angegebenen Zeitraum zu verlängern.

/F2150/ Ausleihberechtigungsdauer überprüfen
überprüft, ob die Ausleihberechtigungsdauer abgelaufen ist. Wenn die Ausleihberechtigungsdauer abgelaufen ist wird der Kunde automatisch per E-Mail darüber informiert.

/F2151/ Kundenkarteikarte drucken
druckt eine Kundenkarteikarte.

/F2152/ Liste der zugangsberechtigten Kunden ausgeben.

/F2160W/ Benachrichtigung über neue Bücher
Der Kunde bekommt die Möglichkeit sich über neu eingetroffene Bücher per E-Mail informieren zu lassen.

/F2170W/ Benachrichtigung über neue Zeitschriften
Der Kunde bekommt die Möglichkeit sich über neu eingetroffene Zeitschriftenhefte informieren zu lassen. Der Kunde kann sich die Zeitschriften über die er sich informieren lassen will, aus dem Zeitschriftenbestand aussuchen.

A.5.3 Ausleihverbuchung

/F3100/ Bücher eintragen
ermöglicht Bücher bei einem Kunden als Ausleihe einzutragen. Es können nur Bücher mit dem Status ´verfügbar´ eingetragen. Sollte ein vorbestelltes Buch eingetragen werden, wir der Vorbestellungsvermerk gelöscht. /F3170/

/F3101/ Ausleihkarte drucken
druckt eine Ausleihkarte.

/F3110/ Bücher verlängern
ermöglicht die ausgeliehenen Bücher eines Kunden um den festgelegten Ausleihzeitraum zu verlängern. Ein Buch kann nur verlängert werden, wenn es von diesem Kunden ausgeliehen wurde und nicht vorbestellt ist.

/F3120/ Bücher austragen
ermöglicht, dass ein zurückgegebenes Buch bei dem ausleihenden Kunden auszutragen. Sie ruft ausserdem /F3160/ auf.

/F3130/ Kurzausleihe eintragen (Nachtausleihe)
ermöglicht es Kurzausleihen einzutragen. Sollte der Ausleiher nicht im Kundenstamm sein gibt es die Möglichkeit die Ausleiherdaten einzugeben und für diese Ausleihe zu speichern.

/F3140/ Kurzausleihe austragen
ermöglicht zurückgegebene Kurzausleihen auszutragen. Sie überprüft auch, ob die Ausleihfrist eingehalten wurde.

/F3150/ Vorbestellungen eintragen
ermöglicht, dass ein Kunde ein ausgeliehenes Buch für sich zur Ausleihe vormerken lassen kann.

/F3160/ Vorbestellungen überprüfen
überprüft, ob vorbestellte Werke inzwischen den Status ´verfügbar´ haben und falls sie verfügbar sind, benachrichtigt die Funktion den Kunden per E-Mail.

/F3170/ Vorbestellung löschen
löscht einen Vorbestellungsvermerk.

/F3180/ Ausleihstatistiken erstellen
Erstellt vordefinierte Statistiken.

A.5.4 Buchhaltung

/F4100/ Hauptkonto anlegen
legt ein Hauptkonto an.

/F4110/ Unterkonto anlegen
legt ein Unterkonto zu einem bestehendem Hauptkonto an.

/F4120/ Unterkonto löschen
löscht ein Unterkonto. Die enthaltene Buchungen werden dem übergeordneten Konto zugeordnet.

/F4130/ Buchung eingeben
ermöglicht es Buchungen anhand eingegangener Rechnungen einzugeben.

/F4140/ Buchung stornieren
ermöglicht es falsch eingegebene Buchungen zu stornieren.

/F4141/ Konto abgleichen
ermöglicht es Buchungen in einem Hauptkonto anhand eines Kontoauszuges abzugleichen

/F4150/ Budget einrichten
ermöglicht es ein Jahresbudget zu einem Hauptkonto einzurichten.

/F4160/ Budget abgleichen
bildet die Differenz zwischen dem Jahresbudget und der Summe der eingetragenen Buchungen.

/F4170/ Statistik erstellen
Erstellt vordefinierte Statistiken.

A.5.5 Konfiguration

/F5100/ Anwenderdaten eintragen
trägt die Angaben über die Anwender ein.

/F5110/ Anwenderdaten ändern
ermöglicht Angaben über einen Anwender zu ändern.

/F5120/ Anwenderdaten löschen
löscht einen Anwender.

/F5130/ Berechtigungen setzen
setzt die Zugriffsberechtigungen für einen Anwender auf die Datenbank und Programmteile.

/F5140/ MAB2-Feld Tabelle zuweisen
weist einem MAB2-Feld eine Indizierungstabelle zu. Das MAB2-Feld wird anhand seiner Identifikationsnummer zugeordnet.

/F5150/ Neues Anwender-Feld definieren
ermöglicht es ein benutzerdefiniertes Feld einzufügen. Es können nur Identifikationsnummern benutzt werden, die nicht im MAB2-Format beschrieben sind.

/F5160/ Anwender-Feld löschen
Es können nur Anwender-Felder gelöscht werden. Der Inhalt dieser Felder geht dann verloren.

/F5170/ Lieferantendaten aufnehmen
ermöglicht die Eingabe von Lieferantendaten.

/F5180/ Lieferantendaten bearbeiten
ermöglicht eingegebene Lieferantendaten zu bearbeiten.

/F5190/ Lieferantendaten löschen
löscht Lieferantendaten.

A.5.6 Programmsteuerung

/F6100/ Anwender anmelden
meldet einen Anwender zu einer Sitzung am Bibliothekssystem an.

/F6110/ Anwender abmelden
meldet einen Anwender von einer laufenden Sitzung ab.

/F6120/ Neue Indezierungstabelle erstellen
erstellt eine neue Indizierungstabelle.

/F6130/ Indizierungstabelle löschen
löscht eine Indizierungstabelle. Die zugewiesenen Anwendungs- oder MAB2-Felder müssen einer anderen Indezierungstabelle zugewiesen werden. Es können nur vom Anwender hinzugefügte Indizierungstabellen gelöscht werden.

A.6 Produkt-Daten

A.6.1 Stammdaten

/D100/ Literaturstammdaten
Umfasst mindestens folgende Felder: Titel; Autoren- und Editorenliste; ISBN oder ISSN; Verlag; Serie; Körperschaft; Liste der vorhanden Exemplare; Erscheinungsjahr.

/D110/ Autorenstammdaten
Hier werden die Angaben zu den Autoren oder Editoren festgelegt. Autorenname; Kürzel für Signaturerzeugung.

/D120/ Serienstammdaten
speichert die Serientitel.

/D130/ Körperschaftsstammdaten
speichert die Angaben über Körperschaften.

/D140/ Verlagstammdaten
speichert die Angaben über Verlage.

/D150/ Lieferantendaten
speichert die Angaben über Lieferanten. Folgende Felder sollten vorhanden sein: Name;Anschrift;E-Mail-Adresse.

A.6.2 Lokaldaten oder Exemplardaten

/D200/ Literatur-Lokaldaten
Hier werden die Angaben über die in einer Bibliothek vorhanden Buchexemplare gespeichert.

/D210/ Kundendaten
speichert die Angaben der Kunden.

/D220/ Daten für die Ausleihverbuchung
Signatur;Ausleiher;Ausleihbeginn;Ausleihende; Mahnstufe.

/D230/ Rechnungsdaten
Die Rechnungsdaten umfassen alle Daten, die mit der Buchhaltung zu tun haben.

A.6.3 Konfigurationsdaten

/D300/ MAB2-Beschreibungsdaten
In diesen Daten werden die MAB2-Feldbezeichner und die Zuordnung zu den entsprechenden Tabellen gespeichert.

/D310/ Programmeinstellungen
Hier sind die Konfigurationsdaten des Programms gespeichert. Es wird die Adresse der Stammdatenbank festgelegt.

/D320/ Benutzerdaten

Diese Daten umfassen die Angaben zu den Benutzern des Produktes und die Zugriffsrechte auf die einzelnen Funktionen.

A.7 Produkt-Leistungen

/L100/ Die Anzahl der Datensätze ist nur durch die Kapazität des Servers beschränkt.

A.8 Benutzeroberfläche

/B100/ Für die Benutzung der Bibliotheksverwaltung ist ein Web-Browser vorgesehen.

A.9 Qualitäts-Zielbestimmung

	sehr wichtig	wichtig	weniger wichtig	unwichtig
Zuverlässigkeit	x			
Robustheit	x			
Effizienz			x	
Benutzerfreundlichkeit		x		
Portierbarkeit	x			
Wartbarkeit	x			
Migration	x			

A.10 Globale Testfälle

/T100/ Es wird der gesamte Funktionsumfang im Parallelbetrieb mit der alte Software getestet.

A.11 Glossar

Autorenstammdaten sind Angaben über Autoren, wie Namen, Vornamen, usw. Die Autorenstammdaten werden verwaltet um die Autoreneinträge der Literaturdaten eindeutig zu halten.

IBIS ist die Suchmaschine der Informatikbibliotheken in Saarbrücken. Folgende Institute und Fachrichtungen sind daran beteiligt: Computerlinguistik, Deutsches Forschungsinstitut für Künstliche Intelligenz, Fachrichtung Informatik und das Max-Planck-Institut für Informatik. Sie stellt den Bestand dieser Bibliotheken tagesaktuell im Internet bereit.

Körperschaftstammdaten sind Angaben über Körperschaften, wie Namen, Anschrift, usw. Die Körperschaftstammdaten werden verwaltet um die Körperschaftseinträge der Literaturdaten eindeutig zu halten.

Literaturstammdaten sind die Angaben über ein Buch die nicht an ein Exemplar gebunden sind, wie z.B. der Titel, der Autor, ISBN, usw.

Lokaldaten Können auch als Exemplardaten bezeichnet werden. Das sind die Daten, die einem Exemplar in einer Bibliothek zugeordnet werden.

Serienstammdaten sind Angaben über Serien, wie Serientitel, usw. Die Serienstammdaten werden verwaltet um die Serieneinträge der Literaturdaten eindeutig zu halten.

Signatur ist ein eindeutiges Kürzel für ein Buchexemplar. Die Signatur wir auch verwendet um ein Buch leichter im Regal zu finden.

Verlagstammdaten sind Angaben über Verlage, wie Namen, Anschrift, usw. Die Verlagstammdaten werden verwaltet um die Verlageinträge der Literaturdaten eindeutig zu halten.

Web-OPAC Suchmaske zum Recherchieren des Datenbestandes im Internet.

Literaturverzeichnis

[AS00] Heuer Andreas and Gunter Saake. *Datenbanken: Konzepte und Sprachen.* MITP-Verlag, zweite edition, 2000.

[AS01] Christoph Aigner and Thomas Strohmaier. *PHP – GE-PACKT.* die GE-PACKTE Referenz. MITP-Verlag, erste edition, 2001.

[Bal96a] Helmut Balzert. *Lehrbuch der Software-Technik – Software-Entwicklung.* Lehrbücher der Informatik. Spektrum, Akademischer Verlag, erste edition, 1996.

[Bal96b] Helmut Balzert. *Lehrbuch der Software-Technik – Software-Management Software-Qualitätssicherung Unternehmensmodellierung.* Lehrbücher der Informatik. Spektrum, Akademischer Verlag, erste edition, 1996.

[Die99] Die Deutsche Bibliothek. *MAB2:Maschinelles Austauschformat für Bibliotheken*, 1999.

[DuB00] Paul DuBois. *MySQL.* New Riders Publishing, erste edition, 2000.

[Inc85a] Adobe Systems Incorporated, editor. *PostScript Language - Reference Manual.* Addison-Wesley, 1985.

[Inc85b] Adobe Systems Incorporated, editor. *PostScript Language - Tutorial and Cookbook.* Addison-Wesley, 1985.

[Inc99] Adobe Systems Incorporated, editor. *PostScript Language Reference.* Adobe, 1999.

[Kop92] Helmut Kopka. *LaTeX– Eine Einführung.* Addison-Wesley, 1992.

[MyS] *MySQL Online Dokumentation.* http://www.mysql.com/documentation/mysql/.

[PHP] *PHP Online Dokumentation.* http://php3.de/manual/de/.

[Sel] *SelfHTML.* http://selfhtml.teamone.de/.

[SL00] Rolf D. Stoll and Gudrun Anna Leierer. *PHP 4 + MySQL.* Internet intern. Data Becker, erste edition, 2000.

[Wei99] Prof. Dr. Ing. Gerhard Weikum. *Skript zur Vorlesung: Datenbanksysteme WS 98/99.* Universität des Saarlandes, 1999.

[Zel02] Prof. Dr. Andreas Zeller. *Skript zur Vorlesung: Softwaretechnik I WS 01/02.* Universität des Saarlandes, 2002.