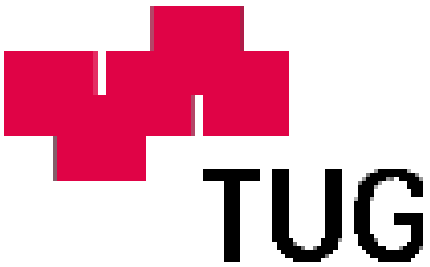
Jürgen Hartler

Entwicklung einer Datenbank für psychische Krankheiten

Diplomarbeit



Institut für Elektro- und Biomedizinische Technik
Technische Universität Graz
Inffeldgasse 18, A - 8010 Graz
Vorstand: Univ.-Prof.Dipl.-Ing.Dr.techn. Gert Pfurtscheller

Betreuer:
Ao. Univ.-Prof. Dipl-Ing. Dr. techn. Zlatko Trajanoski
Dipl.-Ing. Michael Maurer

Begutachter:
Ao. Univ.-Prof. Dipl-Ing. Dr. techn. Zlatko Trajanoski

Graz, Mai 2002

Entwicklung einer Datenbank für psychische Krankheiten

Psychische Krankheiten zählen zu den am häufigsten vorkommenden Krankheiten auf der Welt. Dennoch wurde bis heute noch kein informationstechnisches Hilfsmittel zum Erlangen von diagnostischen Parametern für psychische Krankheiten entwickelt, das auf einem standardisierten Fragebogen basiert und eine Datenbank verwendet.

Damit dies ermöglicht wird, wurde die bereits bestehende elektronische Version eines in den USA verwendeten standardisierten psychischen Fragebogens (SCID, Structured Clinical Interview for Diagnostic and Statistical Manual IV Axis I Disorders) adaptiert und erweitert. Mit Hilfe von Java, Java Enterprise Beans und Oracle wurde eine neue plattformunabhängige Software entwickelt, die die Konsistenz der über den standardisierten Fragebogen erhaltenen Daten gewährleistet. Durch die Verwendung der auf Java Server Pages und Servlets basierenden Struts-Technologie wurde ein Werkzeug geschaffen, das sowohl die Verwaltung der Daten als auch die Auswertung der Daten über einen Web-Browser erlaubt.

Die entwickelte Datenbank erfüllt die Anforderungen einer relationalen Datenbank und die auf dieser Datenbank basierenden Software stellt ein nützliches Hilfsmittel dar, welches die Konsistenz, die Verwaltung und die Auswertung der Daten ermöglicht.

Schlüsselworte: Structured Clinical Interview, Datenbank, psychische Krankheiten, Bionformatik, Java

Development of a Database for Mental Illnesses

Mental Illnesses are among the most debilitating and widespread disabilities worldwide. Despite this, there has never been an electronic tool available for diagnosing the full range of psychiatric illnesses.

Therefore a computerized version of a standardized psychiatric interview, that is widely used in the United States (SCID, Structured Clinical Interview for Diagnostic and Statistical Manual IV Axis I Disorders), has been developed. This novel, platform-independent software is based on Java, Java Enterprise Beans and Oracle to assure the consistency of the data derived from the questionnaire. The collected interviews can be administered by a web tool that has been developed based on the Struts-framework using Java Server Pages and Servlets.

The developed software suite fulfils all the requirements regarding usability and consistency of the data. Because of the streamline nature of this program, manual data entry for statistical analysis becomes obsolete, and researchers can be assured of the accuracy of their findings. So it is prone to become a useful tool in clinical research of mental disorders.

Key Words: Structured Clinical Interview, Database, Mental Disorders, Bioinformatics, Java

Inhaltsverzeichnis

Abbildungsverzeichnis

Glossar

API Application Interface

DALY Disability-Adjusted Life Year; entspricht dem Verlust eines Lebensjahrs einer Person durch vorzeitigen Tod oder das Leben mit dieser Behinderung

DBMS Datenbankmanagementsystem; Programm, dass die Datenbank verwaltet

DOM Document Object Modell

DSM Diagnostic and Statistical Manual

DTD Document Type Definition

EIS Enterprise Information System

EJB Enterprise Java Bean

eSCID electronic SCID

Expertensystem wissensbasiertes Programm zur automatischen Generierung einer Diagnose

Genexpressionsmuster Ausbildung der in einem Gen festgelegten Eigenschaften

HTML Hypertext Markup Language

HTTP Hypertext Transfer Protocol

IP Internet Protocol

JavaBeans serverseitige, plattformneutrale Komponenten

JAXP Java API for XML Processing

JDBC Java Database Connectivity

JDK Java Development Kit

JNDI Java Naming and Directory Interface

JSP Java Server Page

MVC Model View Controller

Microarray Methode für die parallele Analyse von Genexpressionen

ODBC Open Database Connectivity

Parser Programm, dass zum Einlesen von Dateien verwendet wird

PDF Portable Document Format

Query Datenbankanfrage

RMI Remote Method Invocation

SAX Simple API for XML

SCID Structured Clinical Interview for DSM-IV Axis I Disorder

Servlet serverseitige Komponente, die Anfragen vom Web-Client abarbeitet und beantwortet

SGML Standard Generalized Markup Language

SQL Structured Query Language

Struts ein verlässliches und getestetes MVC-System zum Konstruieren von Web-Applikationen mit JSPs und Servlets

TCP Transmission Control Protocol

UDP User Datagram Protocol

URL Uniform Resource Locator

VM Virtual Machine

Xalan XSLT-Prozessor zur Umwandlung von virtuellen XML-Dokumenten in reale XML-Dokumente

Xerces XML-Parser

XML Extensible Markup Language

XSLT XML Stylesheet Language for Transformation

Kapitel 1

Einleitung

Psychische Krankheiten gehören zu den am häufigsten vorkommenden Krankheiten. Allein in den USA leiden 9.5% (18.8 Millionen Menschen) [25] der Erwachsenen an Depressionen.

Ursachen für Behinderungen	Kosten (in DALYs)
Depressionen	42,972
Tuberkulose	19,673
Verkehrsunfälle	19,625
Alkoholmissbrauch	14,848
selbstzugeführte Verletzungen	14,645
Manisch-depressive Krankheiten	13,189
Krieg	13,134
Gewalt	12,955
Schizophrenie	12,542
Eisenmangelanämie	12,511

Abbildung 1.1: Die zehn führenden Ursachen für Behinderungen und Arbeitsunfähigkeit[22]. Die verwendete Einheit DALY (Disability-Adjusted Life Year) wurde von Harvard Forschern entwickelt. Hierbei entspricht ein DALY einem Verlust von einem Lebensjahr einer Person durch vorzeitigen Tod oder das Leben mit dieser Behinderung

Dies bringt nicht nur enorme Kosten im Gesundheitswesen sondern auch eine Reduktion der Arbeitskraft der Bevölkerung mit sich. Außerdem beziehen sich die 18.8 Millionen Menschen nur auf diejenigen, die an der Krankheit leiden. Zusätzlich werden noch die, die sich um die Kranken kümmern, in Mitleidenschaft

gezogen. Es wird nicht nur das Leben des Leidenden zerstört, sondern es gehen manchmal ganze Familie in die Brüche.

1.1 Ursachen für psychische Krankheiten

Depressionen treten in manchen Familien in gehäufter Form auf, wobei bei den Kranken ein verändertes genetisches Muster festgestellt werden kann. Dies würde zu der Schlussfolgerung führen, dass psychische Krankheiten vom genetischen Muster abhängen. Jedoch treten nicht bei allen Personen, die dieses krankhafte Muster aufweisen, psychische Störungen auf. Es trifft vor allem Personen mit einer eher pessimistischen Lebenseinstellung, einer geringen Selbstachtung oder die von Stress geplagt werden. Dies wiederum deutet darauf hin, dass psychische Krankheiten von der jeweiligen Lebensumgebung abhängen. Es wurde erst kürzlich gezeigt, dass physische Krankheiten, wie zum Beispiel Schlaganfall Herzinfarkt, Krebs, Parkinson und hormonelle Störungen, auch von psychischen Störungen begleitet werden können [25]. Das ist vor allem dadurch zu erklären, dass das Gehirn kein starres Gebilde ist, sondern sich während des Lebens ständig verändert.

Meistens stehen am Beginn jeder psychischen Krankheit genetische Faktoren, psychologische Faktoren und/oder Umgebungsfaktoren (z.B. ein schwerer Verlust, schwierige Beziehungen, finanzielle Probleme oder andere stressbehaftete Änderungen des Lebensmusters) als Auslösungsfaktoren, während wiederkehrende Ausbrüche der Krankheit oft schon durch etwas Stress hervorgerufen werden können oder manchmal gar keinen Auslöser brauchen.

Wie auch in Abbildung 1.2 dargestellt ist, werden psychische Krankheiten sowohl von genetischen Faktoren als auch von Umgebungsfaktoren hervorgerufen.

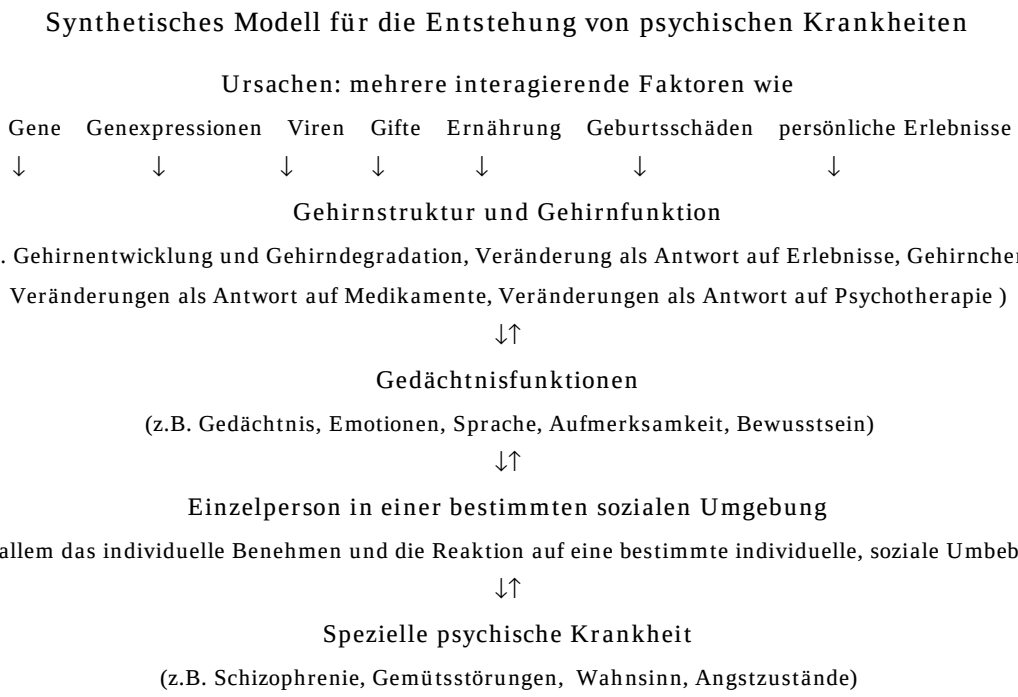


Abbildung 1.2: Zusammenspiel verschiedener Komponenten zur Entstehung einer psychischen Krankheit [2]; psychische Krankheiten sind nie die Folgeprodukte einer einzigen Ursache, sondern erst durch das Zusammenspiel verschiedener Komponenten wird eine psychische Krankheit ausgelöst

Da die meisten menschlichen Krankheiten "komplexe Krankheiten" sind, werden sie nicht nur von einem Gen, sondern von mehreren verursacht. Jedes defekte Gen hat meistens nur einen kleinen Nebeneffekt zur Folge, aber die Akkumulation dieser Effekte und die Interaktion dieser Gene bewirken dann meistens den Ausbruch der Krankheit. Dabei kommen noch folgende zwei Faktoren hinzu, die die Auswirkungen von psychischen Krankheiten auf genetischer Ebene beeinflussen:

- Penetranz: wenn ein Gen penetrant ist, kommt dessen Wirkung auch zur Geltung (entspricht einer Alles-Oder-Nichts Regel)
- Expression: beschreibt die Ausprägung der Wirkung (nicht Alles-Oder-Nichts); diese kann von sehr schwach bis sehr stark reichen

1.2 Programm zur Evaluierung und Verwaltung von psychologischen Daten

1.2.1 Der Fragebogen

Am Beginn einer Behandlung gegen psychische Krankheiten steht eine physische Untersuchung, da gewisse Medikamente oder virale Effekte die selben Symptome wie eine psychische Erkrankung aufweisen können. Dannach erfolgt eine psychologische Evaluierung, die eine komplette Anamnese der Symptome beinhalten muss [25]:

- Wann begannen die Symptome?
- Wie lange dauerten sie?
- Wie stark waren sie?
- Traten sie schon zuvor auf?
- Einnahme von Medikamenten, Drogen und Alkohol?
- Sind psychische Krankheiten schon öfters in der Familie vorgekommen?

Der Vorteil bei einem Fragebogen ist, dass durch die Standardisierung der Fragen ein Vergleich der Antworten für die Klassifikation der psychischen Krankheiten, wie sie für Studienzwecke benötigt wird, möglich ist. SCID ("Structured Clinical Interview for DSM-IV Disorders"; wobei DSM für Diagnostic and Statistical Manual steht) ist ein standardisierter klinischer Fragebogen, der aus ungefähr 200 Seiten und über 600 Fragen besteht und die zuvor genannten Kriterien erfüllt. Da der Diagnoseschlüssel des Fragebogens sehr komplex und langwierig ist, würde ein Programm, das die Erstellung dieser Diagnosen übernehmen würde, einen enormen Zeitgewinn bringen. Das Programm eSCID entspricht einer Implementierung des SCID Fragebogens, welches mit Hilfe eines Diagnoseschlüssels eine Diagnose generiert und die erhaltenen Daten speichert.

1.2.2 Die zentrale Verwaltung in einer Datenbank

Die zentrale Speicherung in einer Datenbank ermöglicht die vollständige Verfügbarkeit und die Verwaltung der Daten jedes einzelnen Patienten. Durch den Vergleich von verschiedenen Befragungen ergibt sich weiters ein sehr nützliches analytisches Werkzeug für die Forschung und Klinik, das den Ärzten ermöglicht diagnostische Parameter aus den Krankenakten zu errechnen.
In der Arbeitsgruppe Bioinformatik (am Institut für Elektro- und Biomedizinische Technik / Abteilung Biophysik) wird momentan versucht an Hand von Genexpressionsprofilen mit Hilfe der Microarray-Technologie Patienten mit psychischen Krankheiten zu klassifizieren. Damit man einen sinnvollen Vergleich von Genexpressionsmustern zwischen Gesunden und Kranken durchführen kann, muss man eine genügend große Anzahl von Patienten zur Verfügung haben. eSCID wäre eine ideale Basis um eine genügend große Anzahl von Versuchspersonen zu eruieren und klassifizieren. Mit eSCID als Basis und der Micro-Array-Technologie könnten eine Reihe von pathologischen Veränderungen in Genen erkannt werden. Dabei bleibt allerdings zu beachten, dass Krankheiten mit überlappenden Symptomen oft in eine große Gruppe geworfen werden (z.B. Depressionen), obwohl sie aus einer Gruppe von separaten Krankheiten bestehen können [11].

1.3 Aufgabenstellung

Den Ausgangspunkt bildete eine bereits vorhandene Version von eSCID, die den Fragebogen und dessen Logik enthielt und bei der die beantworteten Fragen lokal gespeichert wurden. Für dieses Programm sollte eine Datenbank und ein Web-Interface entwickelt werden. Die zu entwickelnde Software sollte folgende Eigenschaften haben:

- benutzerfreundlich
- plattformunabhängig
- netzwerkverbindungsunabhängig

- Datenbankanbindung möglich

Darauf aufbauend sollten folgende Veränderungen bzw. Neuerungen vorgenommen werden:

1. Änderungen am Programm eSCID:
   - Implementierung eines neuen Moduls zur Anmeldung und Synchronisierung mit dem Server
   - Implementierung eines Moduls, bei dem der Interviewer eine Diagnose über den Patienten abgibt, ohne vorher das automatisch generierte Resultat zu sehen
   - Die lokale Datei soll weiters folgende Daten speichern können:
     - Ø Eindeutige Identifizierungsnummer des einzelnen Fragebogens
     - Ø Versionsnummer des verwendeten Fragebogens
     - Ø Informationen über die Studie
     - Ø Speicherung und Auswertung des Interviews
     - Ø Bestätigung einer erfolgreichen Übertragung an die Datenbank

   Kommunikation mit dem Server
   - eine Kommunikation mit dem Server soll aufgebaut werden und alle noch nicht gesendeten Dateien sollen zum Server gesendet werden
   - die Patientendateien sollen vom Server ausgewertet werden (automatische Generierung des Resultats) und das Ergebnis soll an das Programm zurückgesendet werden; dort werden die vollständigen Patientendaten gespeichert und eine HTML- und eine PDF-Datei erzeugt
   - nach dem Ende der Transaktion werden die Daten an die Datenbank übertragen
   - die Verschlüsselung der Daten soll in allen Punkten berücksichtigt oder implementiert werden

2. Erstellung der Grundstruktur für ein Web-Interface, in das man sich einloggen und einfache Auswertungen und Abfragen vornehmen kann; dabei soll auch die Verschlüsselung der Daten berücksichtigt bzw. implementiert werden

Mögliche Abfragen:

- behandelte Patienten in einem gewissen Zeitraum
- nach Namen bzw. interner Spitalsnummer suchen
- Patienten mit bestimmten Krankheiten anzeigen
- alle Patienten einer Studie anzeigen

3. Zusätze:
    - Speichern der Versionsnummer des Fragebogens in der Datenbank
    - Verschlüsselung der übertragenen Daten
    - Usability-Test
    - Stresstest

Kapitel 2

Methoden

In diesem Kapitel wird näher auf die verwendete Programmiersprache und auf die Vorgehensweise eingegangen.

2.1. Programmierwerkzeuge

Als Programmiersprache wurde Java (Sun Microsystems, Inc., 901 San Antonio Road, Palo Alto CA 94303 USA, www.sun.com) verwendet. Weiters wurden außer dem Java Development Kit (JDK) das JAXP_Winter Package (The Apache Software Foundation, www.apache.org) mit einem XML-Parser (Xerces) und einem XML-Processor (Xalan), die Enterprise Java Beans (EJBs) und die Jakarta-Struts (The Apache Software Foundation, www.apache.org) Technologie verwendet.

2.1.1 Java

Geschichte von Java

Die Anfänge von Java liegen in einem portablen Interpreter namens Oak, der Mitte 1992 entwickelt wurde und zur Steuerung und Integration von Geräten bestimmt war. Da das World-Wide-Web zu dieser Zeit eine kritische Größe erreichte,

erkannte man die Bedeutung einer plattformunabhängigen Programmiersprache, mit der neben textuellen Inhalten auch Programme transportiert und ohne Installations- oder Portierungsaufwand auf einem beliebigen Zielrechner ausgeführt werden können. Deswegen wurde im Herbst 1994 mit Hilfe von Oak die erste Version von WebRunner fertiggestellt, einem Web-Browser, der in der Lage war kleine Programme aus dem World-Wide-Web zu laden und innerhalb des Browsers auszuführen. Dieses Programm wurde in den nächsten Monaten stabilisiert, in HotJava umbenannt und im Mai auf der SunWorld '95 der Öffentlichkeit vorgestellt, dabei wird der 23. Mai 1995 als die Geburtsstunde von Java angesehen [19]. Im Mai 2000 wurde schließlich die endgültige Version des hier verwendeten JDK 1.3 ausgeliefert.

Eigenschaften von Java

Java lässt sich mit folgenden Eigenschaften charakterisieren [12, 19]:

- Syntax ähnlich wie C oder C++
- einfach
- objekt-orientiert
- verteilt
- robust
- sicher
- architekturneutral
- Compiler- und Interpretersprache
- portabel
- dynamisch
- Parallelisierung einzelner Prozesse möglich

Java profitiert auch davon, dass viele Features von C++ nicht implementiert wurden und die Sprache dadurch schlank und übersichtlich wurde.
In den üblichen Programmiersprachen wurden die Quelldateien vom Compiler direkt in Maschinensprache übersetzt. Bei Java hingegen werden die Quelldateien in einen Code für eine virtuelle Maschine (VM) übersetzt. Diese Maschinensprache

für die virtuelle Maschine wird als Bytecode bezeichnet. Dieser Bytecode wird dann wiederum von der virtuellen Maschine in den prozessorspezifische Maschinencode übersetzt [21] (siehe Abbildung 2.1).

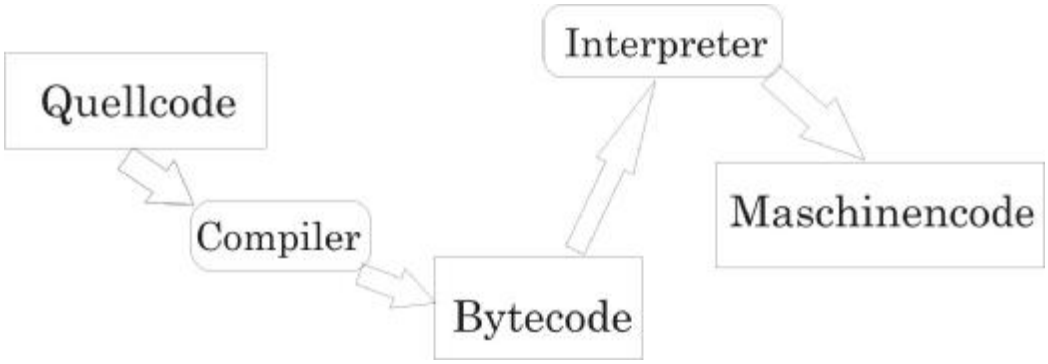


Abbildung 2.1: Vom Quellcode zur Maschinensprache bei Java; bei Java wird der Quellcode zuerst in einen Bytecode für die virtuelle Maschine übersetzt; danach wird der Bytecode von der virtuellen Maschine (Interpreter) in den Maschinencode übersetzt

Der große Nachteil von Java liegt allerdings darin, dass durch den Interpreter Performanceprobleme verursacht werden, die es bei plattformspezifischen Sprachen wie C oder C++ nicht gibt.

Standardkomponenten von Java

Von Java werden folgende Standardkomponenten zur Verfügung gestellt [4]:

- grundsätzliche Komponenten: Objekte, Strings, Nummern, Ein- und Ausgabe, Datenstrukturen, Systemeinstellungen, Datum und Uhrzeit, usw.
- Applets: ein Satz von Konventionen, der von Applets benutzt wird
- Netzwerkkomponenten: URLs (Uniform Resource Locator), TCP (Transmission Control Protocol), UDP (User Datagram Protocol) Sockets, und IP (Internet Protocol) Adressen
- Vielsprachigkeit: Hilfen zum erzeugen von Programmen, die in der entsprechenden Sprache des Landes erscheinen
- Sicherheit: auf hoher und niedriger Ebene, die elektronische Signaturen, öffentliche und private Schlüsselverwaltung, Zugriffskontrollen und Zertifikate beinhalten

- Software-Komponenten: bekannt unter dem Namen JavaBeans, die in bestehende Komponentenarchitekturen eingebaut werden können
- Java Datenbank Verbindung (JDBC): ermöglicht den Zugriff zu einer breiten Palette von relationalen Datenbanken

2.1.2 Enterprise Java Beans (EJBs)

Bei einem Bean handelt es sich um eine serverseitige, plattformneutrale, wiederverwendbare Komponente, (wobei es sich bei einer Komponente um eine wiederverwendbare Software handelt) die zur Erstellung von Anwendungen verwendet werden kann, und zwar mit weitaus größerer Entwicklungseffizienz [26]. Enterprise Java Beans sind serverseitige Komponenten, die die *Business Logic* einer Applikation umschließen, wobei die *Business Logic* der Code ist, der die Funktionalität einer Applikation beinhaltet. Der Vorteil von EJBs liegt darin, dass es die Entwicklung von großen verteilten Applikationen erleichtert. Die Gründe dafür sind [4]:

- der EJB-*Container* bietet Dienste auf der Systemebene; dadurch kann sich der Entwickler auf die Lösung von *Business* Problemen konzentrieren; der *Container* ist verantwortlich für Dienste wie das Managen von Transaktionen und die Sicherheits-Autorisierung
- nicht der Client, sondern die Beans enthalten die *Business Logic*, dadurch kann der Client sich auf die clientseitige Präsentation konzentrieren und muss keine *Business*-Regeln implementieren oder auf die Datenbank zugreifen; als Folge können clientseitigen Programme auch auf schwächeren Maschinen laufen
- dadurch, dass EJBs portable Komponenten sind, kann der Appliktations-Assembler neue Applikationen von existierenden Beans erzeugen; diese Applikationen können dann auf jedem J2EE kompatiblen Server laufen

Man unterscheidet zwischen folgenden Arten von Beans [4]:

- *Session-Beans*: erledigen Aufgaben für den Client
- *Entity-Beans*: repräsentieren Objekte, die dauerhaft gespeichert werden (entsprechen beispielsweise einem Zeileneintrag in einer Datenbank)
- *Message-Driven-Beans*: fungieren als Horcher für das Java Message Service API, durch das Nachrichten asynchron abgearbeitet werden

Dabei wird von der J2EE Plattform ein verteiltes Applikationsmodel, das aus mehrere Ebenen besteht, verwendet. Das bedeutet, dass die Applikations-Logik in Schichten (engl.: *tiers*) unterteilt wird mit den folgenden Funktionen (siehe Abbildung 2.2):

- Client-Tier, die auf der Client-Maschine läuft
- Web-Tier, die auf dem J2EE-Server läuft
- Business-Tier, läuft ebenfalls auf dem J2EE-Server
- *Enterprise Information System* (EIS)-Tier, die auf dem Datenbankserver läuft

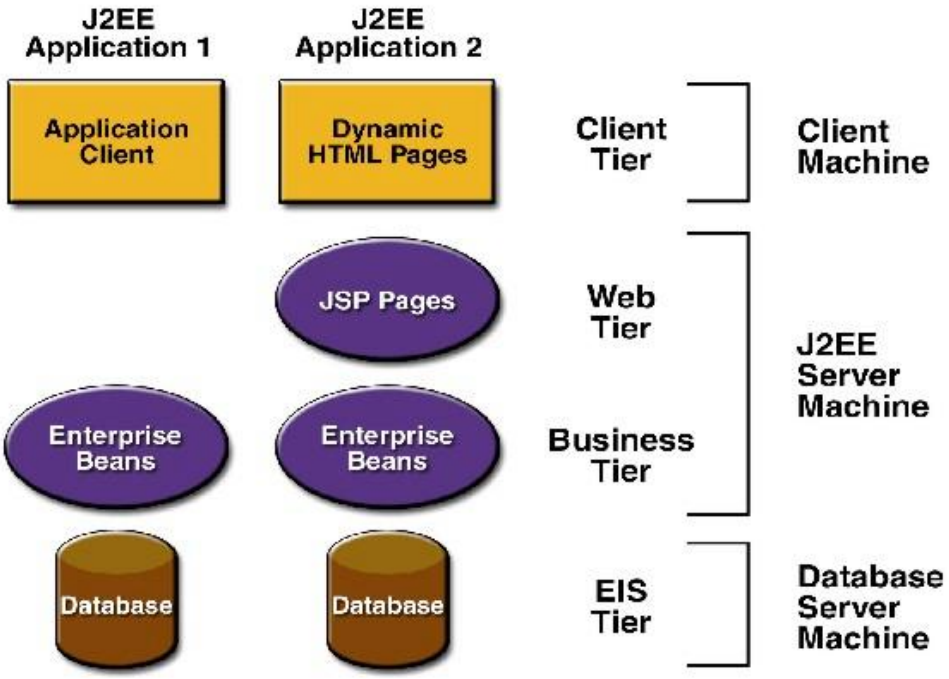


Abbildung 2.2: Schichten einer Applikation [4]; bei diesem Modell wird die Applikation in 3 Schichten unterteilt; der Vorteil besteht darin, dass jede der Schichten auf einer eigenen Maschine laufen kann

RMI

Zur Kommunikation zwischen Client und Server wird bei den EJBs die *Remote Method Invocation* (RMI) verwendet. RMI ermöglicht es, dass ein Objekt, das auf einer Java Virtual Machine läuft, Methoden auf ein Objekt anwendet, das auf einer anderen Java Virtual Machine läuft. Dabei werden von der Serverapplikation sogenannte *remote objects* zur Verfügung gestellt, die dann über Referenzen zugänglich sind. RMI stellt hierbei Mechanismen zur Verfügung, damit der Client mit dem Server kommunizieren und Informationen austauschen kann [28, 30].
Dabei müssen verteilte objektorientierte Applikationen folgende Operationen durchführen können:

- *remote objects* lokalisieren: Applikationen können *remote objects* entweder über eine einfache RMI-Benennungsroutine (*rmiregistry*) ansprechen oder *remote objects* Referenzen als Teil ihrer normalen Operationen übergeben

- mit *remote objects* kommunizieren: Details bezüglich der Kommunikation werden von RMI gehandhabt; für den Java Programmierer sieht es aus, als würde er eine Methode von einem Objekt verwenden
- Klassen-Bytecode für Objekte laden: RMI erlaubt es dem Aufrufenden Objekte an *remote objects* zu übergeben; RMI stellt Mechanismen zum Laden von Objektcode und zum Übermitteln von Daten zur Verfügung

Wie bereits erwähnt, sieht es für den Client-Programmierer so aus als würde er eine Methode aufrufen. In Wirklichkeit liefert RMI, nachdem es in der *rmiregistry* nachgesehen hat, ein Objekt zurück, das dem Client für Aufrufe zur Verfügung steht. Es handelt sich hierbei um ein Rumpfobjekt, das RMI-*stub*, welches als lokaler Proxy dient und den Aufruf zum RMI-*skeleton* auf der Serverseite weiterleitet. Hier wird dann die tatsächliche Implementierung der Methode aufgerufen, diese ausgeführt, und eventuelle Rückgabeparameter werden an den Client gesendet [14].

Das Session- Fassade Designmuster

Dieses Designmuster ist vor allem dann sinnvoll, wenn die Komplexität der Objekt-Interaktionen in der *Business Logic* für den Benutzer versteckt ablaufen soll. Es ist ein Garant für weniger Übergabeparameter zum Client, jedoch steigt damit die Komplexität der serverseitigen Komponenten.
Hierbei wird die komplexe *Business Logic* in *Session-Beans* ausgeführt, während die Speicherung von Daten, die von mehreren Objekten zugänglich sind, den *Entity-Beans* überlassen wird (siehe Abbildung 2.3).

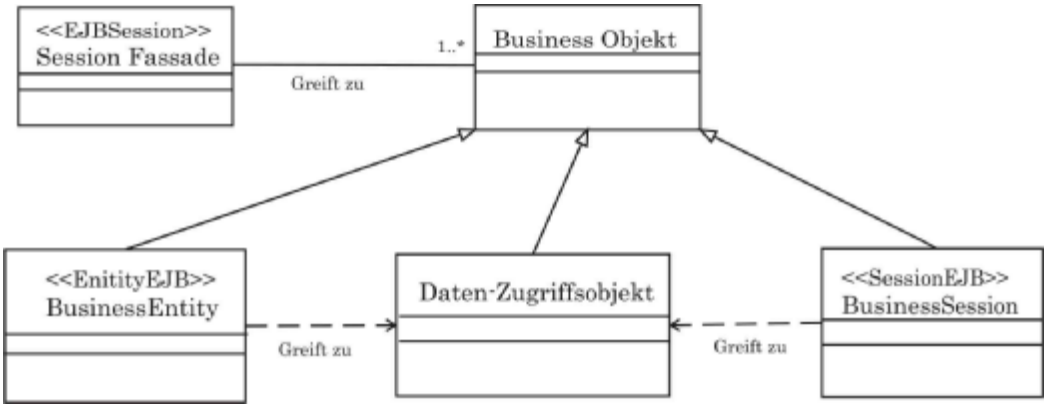


Abbildung 2.3: Klassendiagramm des Session Fassade Designmusters [27]; zeigt dass die Gesamtheit der Zugriffsoperationen über ein einziges Bean abgewickelt werden

Allgemein wird eine Session Fassade aus folgenden Gründen gewählt [27]:

- enge Kopplungen zwischen dem Client und den *Business*-Objekten werden vermieden
- es reduziert die Anzahl der vom Client verwendeten Methoden, da es sonst zu Netzwerksleistungseinbußen kommen könnte
- der Client greift über eine Methode auf den Server zu
- reduziert die Anzahl der *Business*-Objekte in der Dienstschicht (jene Schicht des Netzwerks, auf die der Client zugreifen kann) des Netzwerks
- es gibt einen zentralisierten Arbeitsfluss, da die unterliegenden Interaktionen und Abhängigkeiten zwischen den *Business*-Objekten versteckt werden

2.1.3 Struts

Die Implementierung einer Applikation mit Java Server Pages (JSPs) erfolgt durch eines der folgenden zwei Programmiermodelle [16]:

- Modell 1: hier besteht die Web-Applikation aus einer Sammlung von JSPs, die zum größten Teil unabhängig voneinander arbeiten

- Modell 2: stellt ein Programmierschema dar, das auf dem Model-View-Controller (MVC) System basiert (siehe Abbildung 2.3); Dieses System besteht aus einem *"Model"*, in dem sich die *Business Logic* befindet, einer *"View"*, die die generierte JSP ist und einem *"Controller"*, welcher aus einem Servlet oder einer Sammlung von Servlets besteht, um eine zentralisierte Prozessbehandlung zu gewährleisten

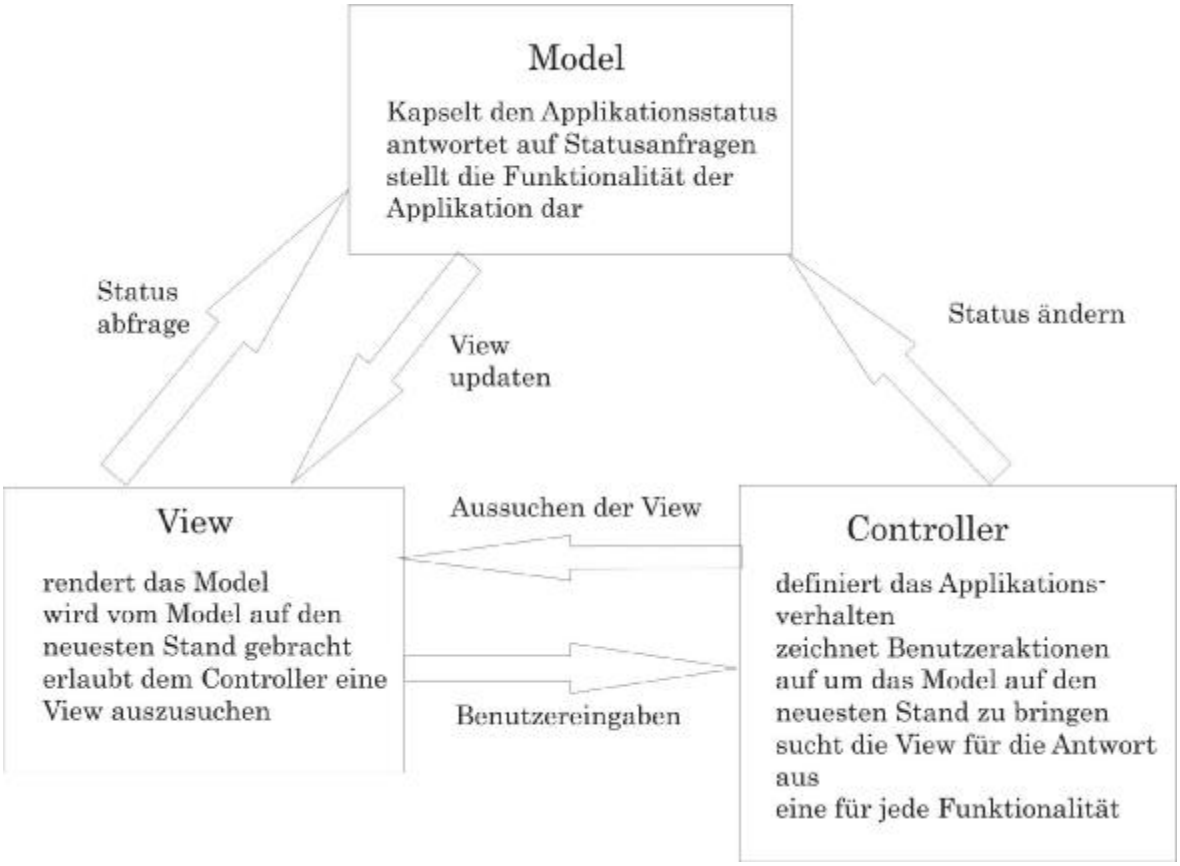


Abbildung 2.4: MVC-System; der *Controller* beinhaltet die Logik und bearbeitet anstehende Ereignisse; das *Model* enthält alle Daten oder weiß wo diese Daten stehen; die *View* ist für die Darstellung der Daten verantwortlich

Das MVC-Konzept besteht aus drei Teilen:

- der *Controller* beinhaltet die Logik und bearbeitet anstehende Ereignisse
- das *Model* enthält alle Daten oder weiß wo diese Daten stehen
- die *View* ist für die Darstellung der Daten verantwortlich

Alle Ereignisse (*Events*) die auftreten können, laufen über den *Controller*. Dieser entscheidet ob das Ereignis für die *View* oder für das *Model* gültig ist und informiert sie daraufhin über Änderungen. Die neuen Informationen werden vom *Model* ausgewertet und überarbeitet. Falls es dadurch zu Änderungen im *Model* kommt, informiert das *Model* die *View*. Kommt es zu Änderungen, holt sich die *View* ihrerseits die neuen Daten und stellt diese dar.
Durch die Verwendung des MVC-Konzepts ergeben sich folgende Vorteile:

- Unabhängigkeit des *Models* von der *View*; es ist nur eine definierte Schnittstelle zwischen beiden notwendig
- es können mehrere *Views* auf ein *Model* zugreifen; dabei muss nur sichergestellt sein, dass der *Controller* alle *Views* über Änderungen im *Model* benachrichtigt

Der große Vorteil in der Verwendung des Modell 2 besteht darin, dass die Web-Applikation leichter wartbar und erweiterbar wird, während ein Modell 1 schneller implementierbar ist. Folglich ist es um so sinnvoller ein Modell 2 zu verwenden, je größer die Web-Applikation wird. Das große Problem liegt allerdings darin, dass man erst ein eigenes MVC-Schema festlegen muss. Struts ist ein MVC-Schema (siehe Abbildung 2.4). Außer den bereits genannten Vorteilen eines MVC-Schemas ist es verlässlich und bereits getestet.

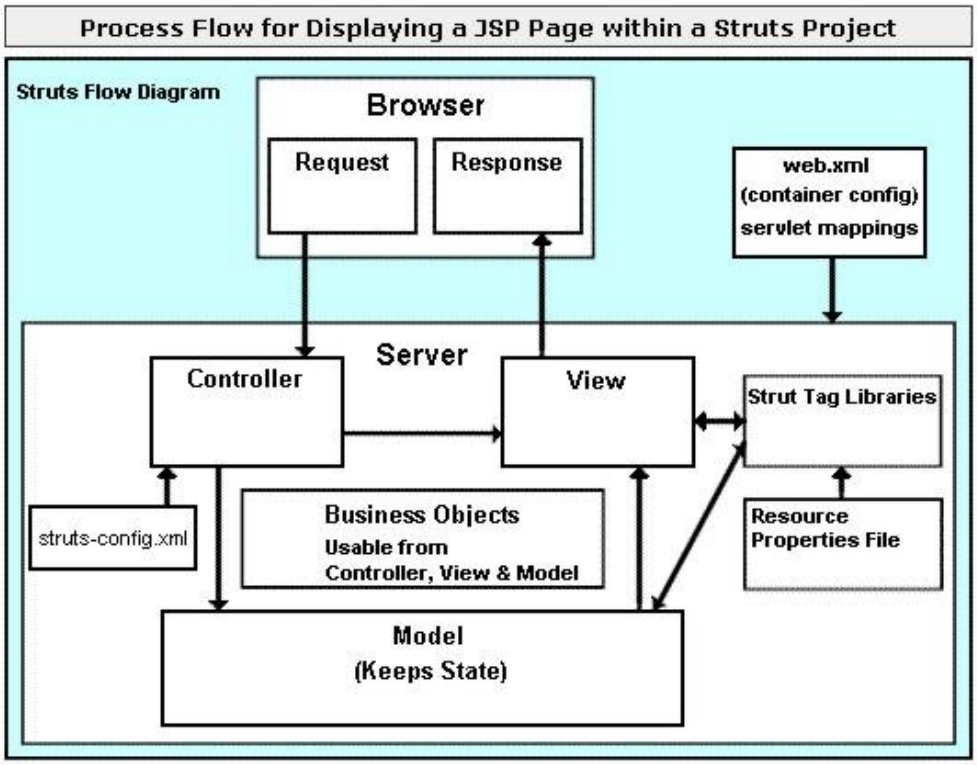


Abbildung 2.5: Prozessabläufe innerhalb des Struts-MVC-Modells [16]; Zusammenspiel zwischen *Model*, *View* und *Controller* mit zusätzlichen Komponenten, die von Struts verwendet werden

- web.xml: wenn der JSP-*Container* gestartet wird, wird zuerst über die web.xml bestimmt ob die angeführten Action-Servlets existieren, dann werden vom *Container* die Anfragen an die entsprechenden Servlets weitergeleitet, damit bei der Ausführung das richtige Action-Servlet zur Anwendung kommt
- request: entsteht bei Benutzeranfragen und wird vom *Controller* abgefangen
- response: Seite die dem Benutzer am Ende des Prozesses angezeigt wird
- controller: der *Controller* ist ein Servlet, das bestimmt welche Aktion notwendig ist und dann die Informationen sendet, die von einem Action-Bean verarbeitet werden
- model: Objekt, das die Anfragen des Benutzers übernimmt und die Resultate während der Dauer des Prozesses speichert

- view: meistens eine JSP, die zur Darstellung der Ausgabe für den Benutzer verwendet wird
- struts-config.xml: speichert die Action-Verweise (eine solche Datei fördert die Modularität)
- struts tag libraries: sind Struts-Komponenten, die eine einfache Integration des Struts-Programmierschemas in die Projekt-Logik erlauben
- property file: darin werden Nachrichten gespeichert, die von einem Objekt oder von einer JSP benutzt werden können
- business objects: sind Module, die verwendet werden um bestimmte Anfragen auszuführen (z.B.: Berechnungen, Speichern in die Datenbank, ...)

Der Nachteil von Struts ist allerdings, dass es keine dynamisch erzeugte Variablen, Eingabefelder, etc. gibt. Daher ist es sinnvoll für eine dynamisch erzeugte Ausgabe das Model 1 zu verwenden und somit Java-Code in die JSP zu schreiben, während die allgemeine Verwaltung weiterhin den Struts überlassen wird.

2.2 Der J2EE Server

Durch einen J2EE Server werden folgende Dienste zur Verfügung gestellt [27]:

- Benennung und Verzeichnisse: Programmen wird durch das *Java Naming and Directory Interface* (JNDI) die Lokalisierung von Diensten und Komponenten ermöglicht
- Authentifizierung: stärkt die Sicherheit, da von Benutzern das Eingeben eines Login-Namens und eines Kennworts erforderlich ist
- HTTP (Hypertext Transfer Protocol): ermöglicht Zugriff auf Servlets und Java Server Pages durch Web-Browser
- EJB: dem Client wird es ermöglicht Methoden von Enterprise Java Beans zu verwenden

2.3 JDBC (Java Database Connectivity)

Bei JDBC handelt es sich um ein *Application Interface* (API), das den virtuellen Zugriff auf tabellarische Daten über die Programmiersprache Java ermöglicht. Es bietet eine Datenbankmanagement übergreifende Verbindungsmöglichkeit auf eine breite Palette von SQL (Structured Query Language) Datenbanken [29]. Vor allem hat es den Vorteil, dass es wie alle anderen Java Programme auf andere Systeme portierbar ist. Hauptaugenmerk wurde darauf gelegt, dass dauernd verwendete Datenbankanfragen (z.B. SELECT) einfach gehalten werden [8].
JDBC führt folgende 3 Dinge aus:

- erstellt eine Verbindung mit einer Datenbank
- sendet SQL-Befehle
- führt die Resultate aus

JDBC ist ein *Application Interface* auf einer niederen Ebene (weil es die SQL-Befehle direkt verwendet). Es wurde als Basis für eine Schnittstelle auf höherer Ebene entwickelt, wobei eine Schnittstelle auf einer höheren Ebene ein verständlicheres und benutzerfreundliches API verwendet, das im Hintergrund in eine Schnittstelle auf niederer Ebene übersetzt wird, wie zum Beispiel JDBC. Weiters unterstützt JDBC sowohl das 2-Ebenen- als auch das 3-Ebenen Modell für den Zugriff auf die Datenbank.
Probleme bereitet allerdings, dass, obwohl SQL die Standardsprache für den Zugriff auf relationale Datenbanken ist, SQL nicht so standardisiert ist, wie man es gerne hätte. Das liegt vor allem daran, dass die Datentypen, die von verschiedenen Datenbankmanagementsystemen (DBMS) verwendet werden unterschiedlich sind. JDBC verwendet hierfür generische SQL-Datentypen. Ein weiteres Problem ist, dass obwohl die meisten Datenbankmanagementsysteme die Grundfunktionen von SQL benutzen, sie jedoch nicht die SQL-Syntax oder SQL-Semantik für erweiterte Funktionen unterstützen. Solange SQL noch nicht wirklich standardgemäß von allen Datenbankmanagementsystemen unterstützt wird, muss das JDBC-API SQL

so unterstützen wie es gerade ist. JDBC geht mit diesem Problem auf drei Arten um [13]:

- es erlaubt alle Such-Strings an das darunterliegende DBMS weiterzugeben, dadurch kann die Anwendung so viel Funktionalität benutzen wie sie will, wobei jedoch der Programmierer das Risiko eingeht, dass sein Programm nicht mit jedem DBMS kompatibel ist
- ein anderer Weg ist es, Ausstiegsklauseln in ODBC-Art (Open Database Connectivity) zu verwenden. Die Ausstiegssyntax enthält Standard-JDBC-Syntax für mehrere bekannte Bereiche, bei denen eine Divergenz bei der SQL-Syntax auftritt
- bei komplexeren Problemen wird eine beschreibende Information über das DBMS über das *Interface* DataBaseMetaData angeboten, sodass sich die Applikation an die Erfordernisse und Möglichkeiten der Datenbank anpasst

JDBC besteht aus folgenden Produktkomponenten:

- JDBC *driver manager*: die Primärfunktion besteht darin, die Java Anwendung mit dem richtigen JDBC Treiber zu verbinden
- JDBC *driver test suite*: testet ob der JDBC Treiber die Applikation ausführen wird
- JDB-ODBC-*bridge*: erlaubt es, ODBC Treiber als JDBC Treiber zu verwenden

2.4 Die Datenbank

2.4.1 Relationale Datenbanken

Der Grundgedanke hinter dem relationalen Modell ist, dass eine Datenbank aus einer Serie von ungeordneten Tabellen (oder Relationen besteht), die durch den Gebrauch von nicht-prozeduralen Operationen, welche Tabellen zurückliefern,

manipuliert werden können. Beim Design einer Datenbank sollte man am besten die wirkliche Welt modellieren. Dabei repräsentieren die Tabellen Gegenstände (Entities) in der realen Welt. Diese Tabellen bestehen wiederum aus Zeilen und Spalten. Hierbei darf jede Zeile einer Tabelle in einer relationalen Datenbank nur einmal vorkommen. Diese Einmaligkeit einer Zeile in einer Tabelle wird durch einen sogenannten Primärschlüssel erreicht, bei dem niemals eine andere Zeile vorhanden sein darf, die den selben Wert für den Primärschlüssel beinhaltet. Ein einfacher Primärschlüssel besteht aus einer Spalte, während ein zusammengesetzter Primärschlüssel aus mehreren Spalten besteht. Weiters gibt es in einer relationalen Datenbank noch sogenannte Fremdschlüssel. Diese sind eine Spalte in einer Tabelle, deren Sinn es ist einen Primärschlüssel in einer anderen Tabelle zu referenzieren, das bedeutet auf eine Zeile in einer anderen Tabelle hinzudeuten. Mit diesem Prinzip sind sowohl 1:n, als auch n:1 Relationen realisierbar. Für n:m Relationen benötigt man eine zusätzlichen Hilfstabelle.

2.4.2 Normalisierung

Damit eine Datenbank den Anforderungen einer relationalen Datenbank entspricht, muss sie normalisiert werden. Dabei versteht man unter Normalisierung den Prozess der Vereinfachung des Designs der Datenbank, damit sie die optimale Struktur erreicht. Um den minimalen Anforderungen einer relationalen Datenbank gerecht zu werden, müssen die ersten drei Normalformen eingehalten werden [10]:

- Erste Normalform: Spalteneinträge müssen atomar sein; das bedeutet, dass in jeder Spalte nur ein Wert existiert und nicht ein Array oder eine Liste von Werten
- Zweite Normalform: die Tabelle muss in erster Normalform sein und jede Nichtschlüsselspalte ist vollständig vom Primärschlüssel abhängig; das bedeutet, dass Tabellen nur Daten speichern sollten, die zu einem

Gegenstand (*Entity*) gehören und dieser Gegenstand sollte durch seinen Primärschlüssel beschrieben werden

- Dritte Normalform: die Tabelle muss in zweiter Normalform sein und keine Nichtschlüsselspalte darf transitiv von einer Nichtschlüsselspalte abhängig sein

2.4.3 Integritätsregeln

Jede Datenbank muss zwei grundlegende Integritätsregeln erfüllen und zwar die *Entity*-Integrität und die referenzielle Integrität.
Die *Entity*-Integrität besagt nur, dass der Primärschlüssel Daten enthalten muss [10] (er muss vorhanden sein). Diese Regel gilt sowohl für einfache als auch für zusammengesetzte Schlüssel. Bei zusammengesetzten Schlüsseln darf folglich keine der Spalten, die Teil des Schlüssels ist, keine Daten enthalten.
Die referenzielle Integrität besagt, dass die Datenbank keine Fremdschlüssel enthalten darf, die nicht in einer anderen Tabelle als Primärschlüssel existieren [10]. Das führt zu:

- eine Reihe darf nicht zu einer Tabelle mit einem Fremdschlüssel hinzugefügt werden, wenn der referenzierte Schlüssel nicht in der referenzierten Tabelle existiert
- wenn der Wert in jener Tabelle, die durch einen Fremdschlüssel referenziert wird, geändert wird (oder die gesamte Zeile gelöscht wird), dürfen die Zeilen in der Tabelle mit dem Fremdschlüsseln nicht ungeändert verbleiben

Es gibt drei Möglichkeiten einem Verstoß gegen die referenzielle Integrität vorzubeugen:

- nicht erlauben: das Ändern des Primärschlüssels ist gänzlich verboten
- die ganze Kaskade ändern: die Änderung des Primärschlüssels ändert automatisch alle abhängigen Tabellen mit, beim Löschen des Eintrags werden auch alle abhängigen Reihen in den anderen Tabellen gelöscht

- auf Null setzen: die abhängigen Fremdschlüssel werden automatisch auf Null gesetzt

2.4.4 Transaktionen und operationale Integrität

Als operationale Integrität wird die Bewältigung von zwei speziellen Problemen bezeichnet:

- gleichzeitiger bzw. quasiparalleler Zugriff mehrerer Benutzer auf gemeinsame Daten muss synchronisiert werden
- Dateninkonsistenz verursacht durch physikalische Speicherfehler

Die Maßnahme zur Erfüllung der operationalen Integrität wird als Transaktionskonzept bezeichnet. Eine Transaktion ist entweder das Ändern von Daten oder die Erstellung einer neuen Reihe in der Datenbank. Diese Operationen können zu einem Datenverlust oder zu inkonsistenten Datenbeständen führen. Das kann allerdings durch folgende Maßnahmen verhindert werden:

- Atomarität: entweder die gesamte Sequenz von Operationen wird ausgeführt (Abschlussbefehl einer Transaktion *"Commit"*) oder keine (der Abbruchbefehl einer Transaktion *"Rollback"* verwirft alle bis zum letzten *"Commit"* eingelangten Befehle und führt dadurch keine der Operationen dieser Transaktion durch)
- Konsistenz: durch Transaktionen geänderte Daten oder Datenobjekte sind erst nach deren Abschluss für andere Benutzer sichtbar und können erst dann durch neue Transaktionen manipuliert werden
- Dauerhaftigkeit: Abgeschlossene Transaktionen sind persistent (z.B. ein Systemfehler darf keine Dateninkonsistenz hervorrufen)

Die Synchronisation von Transaktionen wird durch das relationale Datenbankmanagementsystem erfüllt.

2.4.5 Datenbankentwurf mit dem Entity-Relationship Modell

Die Abbildung von Informationsstrukturen der Wirklichkeit auf das Schema einer Datenbank erfolgt indirekt über ein semantisches Datenmodell, welches dann auf das konzeptionelle Schema (Datenbankobjekte) transformiert wird [17]. Für diesen Zweck wird als semantisches Datenmodell am häufigsten das Entity-Relationship Modell verwendet. Der Vorteil liegt darin, dass es unabhängig vom hierarchischen, netzartigen, relationalem Datenmodell ist und eine anschauliche grafische Darstellungsmethodik bietet. Das Entity-Relationship Modell beschreibt anhand von Gegenständen (*Entities*) reale Gegebenheiten zwischen denen eine Beziehung (*Relationship*) besteht, wobei die *Entities* realen Gegenständen möglichst nahe kommen sollten. Zur Beschreibung der *Entities* werden Attribute verwendet und für die Abhängigkeit der *Entities* untereinander wird das Primärschlüsselkonzept verwendet.

2.5 XML

Bei XML (Extensible Markup Language) handelt es sich um ein universelles Format für strukturierte Dokumente. XML hat folgende Eigenschaften [5]:

- XML ist eine Methode, um strukturierte Daten in einer Textdatei darzustellen; das wird vor allem durch die Baumstruktur, die XML eigen ist, erreicht
- XML sieht fast aus wie HTML (Hypertext Markup Language), ist aber kein HTML; dies kommt vor allem daher, dass XML die selbe Tag-Struktur verwendet wie HTML, jedoch legt HTML die Bedeutung des Tags fest, während XML die Tags nur zur Abgrenzung benutzt und die Interpretation der Daten völlig der Applikation überlässt, die sie verarbeitet

- XML ist Text, aber nicht zum Lesen; XML-Dateien sind nicht wie HTML-Dateien zur Darstellung am Bildschirm bestimmt, sondern um Informationen zu übertragen oder als Konfigurationsdateien zu dienen
- XML besteht aus mehreren Teilen; außer der XML 1.0 Spezifikation, die Tags und Attribute definiert, gibt es noch:
  - Xlink: zum Hinzufügen von Hyperlinks zu einer XML-Datei
  - XSLT (XML Stylesheet Language for Transformation): eine Transformationssprache, die für das Umstellen, Hinzufügen oder Löschen von Tags und Attributen verwendet werden kann und auch zum Transformieren von Daten in einer XML-Datei in HTML Code dienen kann
  - DOM (Document Object Model): ist eine Standardmenge von Funktionsaufrufen um ein XML-Dokument zu manipulieren
  - XML Namespaces: ist eine Spezifikation für die Verknüpfung von einer URL (Uniform Resource Locator) mit einem einzelnen Tag oder einem Attribut
- XML ist ausführlich; das bedeutet, dass XML als Textformat redundante Teile enthält und folglich immer größer ist als binäre Formate
- XML entwickelte sich aus SGML (Standard Generalized Markup Language) heraus
- XML ist lizenzfrei, plattformunabhängig und gut unterstützt

2.5.1 DOM (Document Object Model)

Bei dem Document Object Model handelt es sich um eine plattformunabhängige und programmiersprachenneutrale Schnittstelle, die Programmen einen dynamischen Zugriff auf den Inhalt und die Struktur von XML-Dokumenten erlaubt [31]. Es ist eine Programmschnittstelle für XML und HTML.
DOM glieder sich in folgende 3 Teile [7]:

- Core: bietet ein Satz von Objekten auf niederer Ebene, die jedes strukturierte Dokument darstellen können; die Core-Schnittstelle stellt ein kompaktes und minimales Design für die Manipulation des Dokumentinhalts dar;
- XML: hat weitere, sich auf höherer Ebene befindende Schnittstellen für XML, die mit der Core-Spezifikation verwendet werden und einen näheren Einblick in das XML-Dokument erlauben
- HTML: hat weitere, sich auf höherer Ebene befindende Schnittstellen für HTML, die mit der Core-Spezifikation verwendet werden und die einen näheren Einblick in das HTML-Dokument erlauben

Sowohl die XML- als auch die HTML-Schnittstellen bestehen aus Objekten und Methoden, die einen leichteren und direkten Zugriff auf die Elemente des jeweiligen Dokuments erlauben.
DOM wurde in folgenden Ebenen entwickelt [7]:

- Ebene 1: konzentriert sich auf das Core-, HTML- und XML-Dokumentmodell; beinhaltet die Grundfunktionen für das Navigieren im Dokument und das Verändern des Dokuments
- Ebene 2: legt Funktionen zum Einteilen und Manipulieren von Informationen fest, die das Aussehen des Dokuments betreffen; weiters definiert es ein Ereignisbehandlungsmodell und unterstützt das Verwenden von Leerzeichen in XML
- Ebene 3: adressiert das Laden und Speichern des Dokuments und unterstützt Inhaltmodelle (wie DTDs (Dokument Type Definitions) und Schemas), mit denen eine Validierung des Dokuments möglich ist

Dabei handelt es sich bei DOM um ein Modell, das auf einer Baumstruktur basiert. Hier wird die Information in sogenannte Knoten, die auch Attribute besitzen können, gespeichert. Diese Struktur lässt eine hierarchische Ordnung zu (Abbildung 2.6).

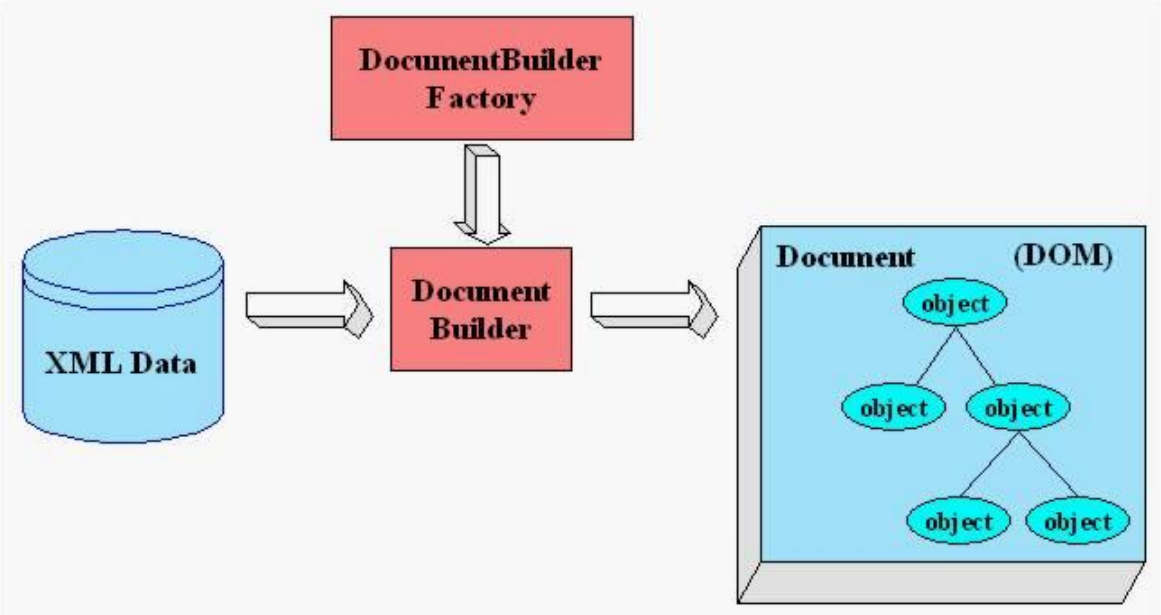


Abbildung 2.6 [3]: Das JAXP API; die XML-Daten werden mit Hilfe eines DocumentBuilders, der von der DocumentBuilderFactory zur Verfügung gestellt wird in eine DOM-Struktur transformiert; die DOM Struktur ist hierarchisch geordnet

2.5.2 Xerces

Xerces ist ein frei verfügbarer XML-Parser, der von Apache Software Foundation freigegeben ist. Hier stehen folgende zwei Schnittstellen zur Verfügung [6]:

- eine auf einer Baumstruktur basierende Schnittstelle (DOM)
- eine auf Ereignissen basierende Schnittstelle (SAX: Simple API for XML)

2.5.3 Xalan

Xalan ist ein XSLT- (*the XML Stylesheet Language for Transformation*) Prozessor, der in Java geschrieben ist. Xalan implementiert die gesamten Anforderungen an XSLT und unterstützt Java- und JavaScripterweiterungen. Dabei kann sowohl SAX als auch DOM zum Generieren der XML-Dateien verwendet werden. Xalan benutzt

Xerces als Parser, hat ein JAXP-API und kann XLST Stylesheets in Java-Bytecode kompilieren. Außerdem können mit Xalan aus bestehenden XML-Dokumenten HTML-Dateien erzeugt werden.

2.6 Usability

Die wichtigsten Kriterien für Benutzerfreundlichkeit sind folgende [23]:

- Aufgabenangemessenheit
- Selbstbeschreibungsfähigkeit
- Steuerbarkeit
- Erwartungskonformität
- Fehlertoleranz
- Individualisierbarkeit
- Lernförderlichkeit

Diese Ansprüche führen aber bei der Implementierung von Programmen oft zu Widersprüchen, sodass der Entwickler sich oft entscheiden muss welchem Kriterium er den Vorrang gibt. Zur Erleichterung dieser Entscheidungen, und zum Testen der Benutzerfreundlichkeit einer Web-Seite dienen Usability-Tests. Für die Entscheidungen eines Entwicklers sollte folgendes gelten [18]: "Die Web-Seite sollte so gestaltet sein, dass ein Benutzer intuitiv das richtige tut!"

2.6.1 Anzahl der Tests und der Benutzer

15 Testpersonen finden zwar alle Fehler einer Webseite, es identifiziert aber bereits einer 30% der Fehler. Beim Testen durch jeden weiteren Benutzer kommt man zwar zu weiteren Erkenntnissen, doch diese Benutzer stolpern auch immer wieder über dieselben Fehler wie die vorhergehenden, sodass kleinere Fehler nicht so auffallen. Daher ist es sinnvoller eine Web-Seite mit 3-5 Benutzern zu testen,

danach die entdeckten Probleme zu beseitigen und dann wieder zu testen, usw. [18].
Außerdem gelten bezüglich Usability folgende Regeln[18]:

- einen Benutzer zu testen ist um 100% besser als keinen zu testen
- einen Benutzer am Anfang eines Projektes testen zu lassen ist effektiver als 50 Benutzer, wenn das Projekt bereits fertig ist

2.7 Kriterien für die Wahl der verwendeten Methoden

Ausschlaggebend für die Verwendung von Java war vor allem, dass eSCID eigentlich für jeden Arzt im Krankenhaus verwendbar sein sollte, egal welches System er verwendet. Außerdem war der Geschwindigkeitsgewinn, den es eventuell bei einer anderen Sprache gegeben hätte, nicht so wichtig, da keine besonders komplizierten Berechnungen vorgenommen werden und keine all zu großen Datenmengen verschoben werden müssen.
Da ein Internetanschluss auf den verwendeten Rechnern (Laptops) wahrscheinlich nicht immer vorhanden ist, war für das Projekt ein Zwischenspeicher vonnöten.
Das XML-Format ermöglicht es Daten strukturiert darzustellen, egal welches System verwendet wird. Deswegen ist es als lokaler Zwischenspeicher ideal geeignet. Das Senden an den Datenbankserver kann zu einem späteren Zeitpunkt erfolgen.
Ein besonderes Augenmerk wurde auf die Skalierbarkeit der Anwendung gelegt, da es vor allem bei einem System wie diesem möglich sein muss, dass mehrere Benutzer gleichzeitig Transaktionen mit der Datenbank abwickeln. Aus Gründen der Skalierbarkeit wurde Oracle 9i und die EJBs verwendet.
Die Hauptgründe für die Wahl eines J2EE-Servers lagen darin, dass dieser einen EJB-*Container* besitzt, der *container-managed-persistence* unterstützt, das heißt, dass sich der Entwickler nur um die *Business Logic* kümmern muss, während die Befehle für Zugriffe auf die Datenbank vom *Container* übernommen werden (z.B. keine hartcodierten Suchmethoden). Das wiederum macht die Serversoftware für

verschieden Datenbanken verwendbar, da der *Container* das entsprechende datenbankspezifische Management übernimmt. Da aber ein direkter Zugriff auf *Entity-Beans* als nicht wünschenswert erachtet wird, wurde die Session Fassade verwendet. Die Session Fassade bringt den Vorteil mit sich, dass sich die Übergabeparameter zwischen Client und Server in Grenzen halten. Außerdem ist durch die alleinige Übergabe der XML-Dateien auch eine leichtere Verschlüsselung möglich, als wenn alle Parameter einzeln verschlüsselt werden müssten.

Als Server wurde ein Server der Marke Orion (Evermind - informatica, 9718 AS Groningen Sweden) gewählt. Der Hauptgrund für die Wahl des Orionservers lag vor allem darin, dass es sich bei Orion um einen J2EE-kompatiblen Server handelt. Außerdem ist er ein kostengünstiger, robuster, skalierbarer und von Oracle lizenzierter Server.

Struts wurden vor allem aus Gründen der möglichen Erweiterbarkeit den gewöhnlichen JSPs vorgezogen, wobei zu bemerken ist, dass vor allem bei dynamisch kreierten Tabellen trotzdem, obwohl durch Struts genau dies vermieden werden soll, Java-Code in der JSP vorhanden ist. Dies stört jedoch nicht die Wart- und Erweiterbarkeit der Applikation, da diese Inkonsistenz eigentlich nur bei Endausgaben verwendet wurde und keinen Einfluss auf die Programmstruktur an sich hat.

Kapitel 3

Ergebnisse

In diesem Kapitel wird näher auf die erzielten Ergebnisse eingegangen. Es wurden Veränderungen am Programm eSCID vorgenommen, eine Datenbank erstellt und eine serverseitige Software entwickelt, die sowohl eine Kommunikation zwischen der Applikation und dem Server ermöglicht als auch ein Web-Interface, das Zugriffe auf die Datenbank über einen Browser erlaubt.

3.1 Veränderungen am Programm eSCID

Die Applikation wurde um 3 Funktionen erweitert:

- Anmeldung und Synchronisierung mit dem Server: beim ersten Starten von eSCID und beim Verwenden des "Update"-*Buttons* ("Update"-*Button* siehe Abbildung 3.1)
- das Senden der XML-Datei: über den "Web"-*Button* ("Web"-*Button* siehe Abbildung 3.1)
- ein Modul, indem der Arzt seine eigene Diagnose abgeben kann ohne vorher das automatisch generierte Resultat zu sehen

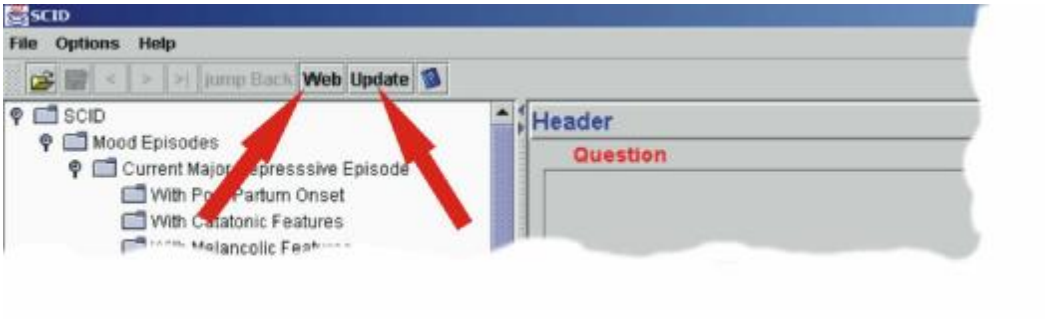


Abbildung 3.1: Ausschnitt aus dem Haupteingabefenster; Darstellung des "Web"- und des "Update"-*Buttons*; mit dem "Web"-*Button* wird das Senden des Fragebogens an den Server ermöglicht; durch den "Update"-*Button* wird die Registrierung aktualisiert und aktuelle Informationen vom Server geladen

3.1.1 Anmeldung und Synchronisierung der krankenhausspezifischen Daten

Nach dem Erststart von eSCID erscheint die Registrierungseingabemaske (siehe Abbildung 3.2). Nachdem die vom Administrator vergebene Krankenhausnummer eingegeben wurde, werden die aktuellen Registrierungsdaten, sowie eine Liste von zugangsberechtigte Personen und aktuell in diesem Krankenhaus laufende Studien vom Server heruntergeladen und lokal gespeichert. Die selbe Registrierungseingabemaske erscheint, nachdem der "Update"-*Button* betätigt wurde. Dadurch werden die lokal gespeicherten Informationen bezüglich der zugangsberechtigten Personen und der in diesem Krankenhaus laufenden Studien aktualisiert. Es ergeben sich folgende Vorteile:

- jeder behandelnde Arzt kann einem Krankenhaus zugeordnet werden
- durch das Herunterladen der Liste mit den zugangsberechtigten Personen kann keine Inkonsistenz zwischen der eSCID-Applikation und der Datenbank entstehen, weil dadurch jeder ausgefüllte Fragebogen genau einem, in der Datenbank existierenden, Arzt zugeordnet werden kann

- durch das Laden einer Liste mit den vorhandenen Studien ist immer eine eindeutige Zuordnung zwischen den ausgefüllten Fragebögen und den Einträgen in der Datenbank gegeben

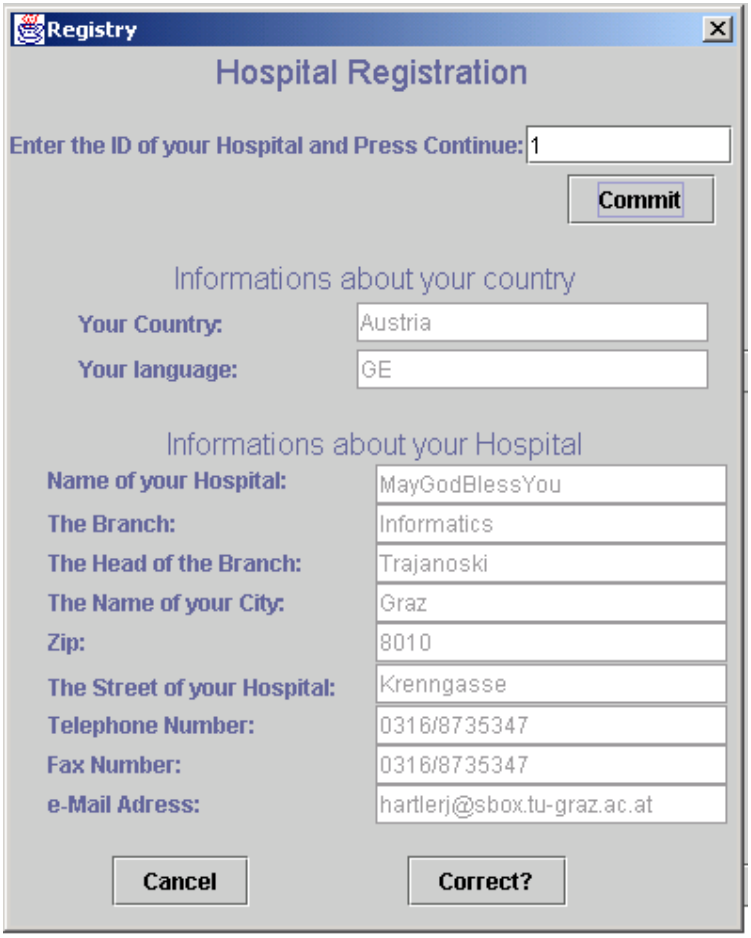


Abbildung 3.2: Registrierungseingabemaske; dient zum Anmelden und Synchronisieren mit dem Server; nachdem eine gültige Hospital-ID (ID: Identity) eingegeben wurde, werden die aktuellen Registrierungsdaten, sowie eine Liste mit zugangsberechtigten Personen und aktuell in diesem Krankenhaus laufende Studien vom Server heruntergeladen und lokal gespeichert; als Rückmeldung erhält der Benutzer die vollständig ausgefüllte Registrierungseingabemaske

3.1.2 Das Senden des ausgefüllten Fragebogens

Der "Web"-*Button* dient zum Senden von ausgefüllten Fragebögen (XML-Dateien). Hier kann vom Benutzer zwischen den noch nicht gesendeten Patientendateien

ausgewählt werden und nach einer Kennwortabfrage werden die ausgewählten Dateien an den Server gesendet. Dieser liefert nach dem Eintrag der Daten in die Datenbank eine automatisch generierte Diagnose zurück. Dann wird eine neue XML-Datei erstellt, welche auch die generierte Information beinhaltet, und aus dieser werden eine für den Benutzer einfach lesbare HTML-Datei und eine PDF-Datei erstellt (Abbildung 3.3).

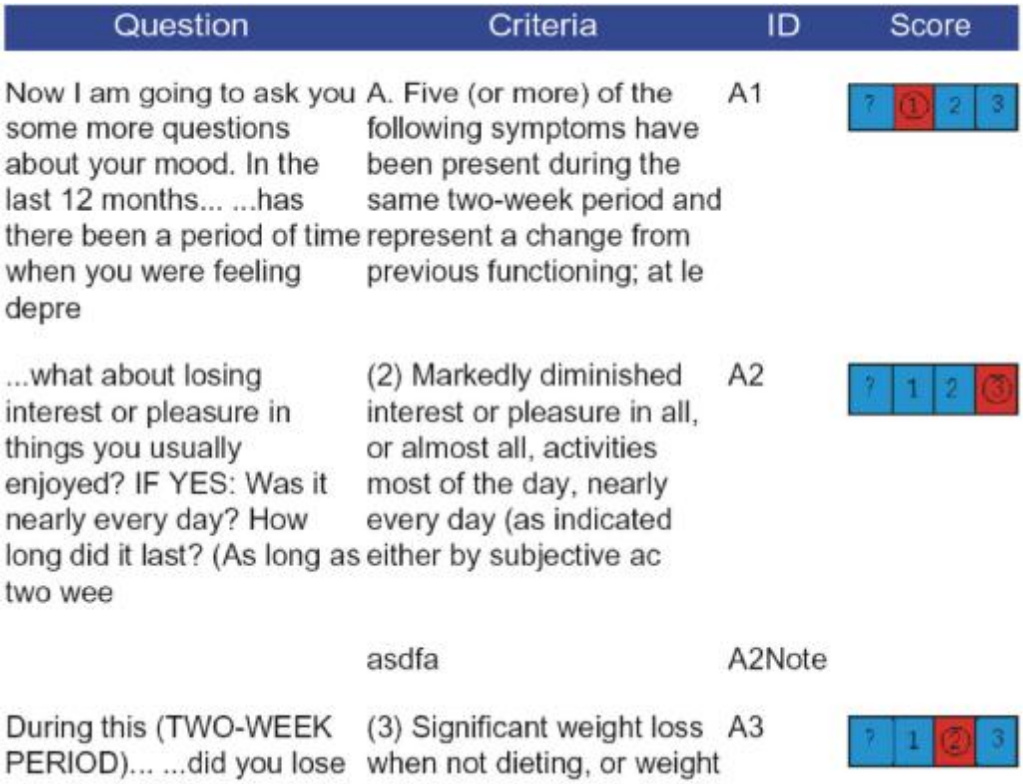


Abbildung 3.3: Ausschnitt aus der generierten PDF-Datei; nach dem der ausgefüllte Fragebogen an den Server gesendet wurde und eine automatisch erstellte Diagnose zurückgeliefert wurde, wird diese PDF-Datei kreiert, die sowohl die Diagnose als auch die beantworteten Fragen beinhaltet

3.1.3 Modul für die Diagnose

Nach Ausfüllen des Fragebogens bekommt der Arzt eine Reihe von Fragen gestellt, die seine eigene Diagnose betreffen (Ausschnitt: siehe Abbildung 3.4). Diese Fragen werden aus der selben XML-Datei gelesen, die die Fragen des Fragebogens

beinhaltet. Der Vorteil liegt darin, dass die Fragen einfach mit einem Texteditor geändert werden können. Die Antworten werden dann in die XML-Datei gespeichert, die alle anderen vom Patienten beantworteten Fragen beinhaltet.

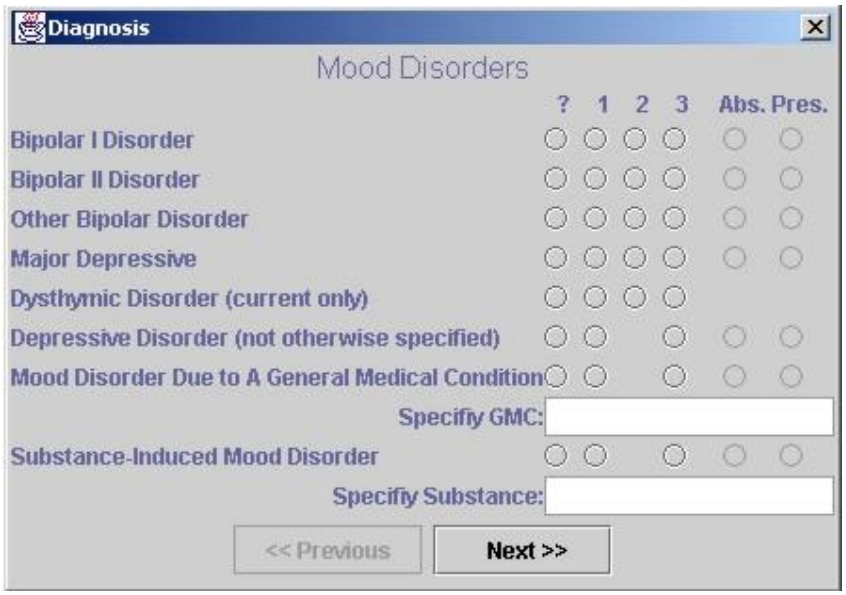


Abbildung 3.4: Ein Dialogfenster aus einer Reihe von Dialogfenstern, die es dem Arzt ermöglichen seine eigene Diagnose abzugeben, ohne vorher die automatisch generierte Diagnose zu sehen

3.2 Die serverseitige Architektur

Für die Session Fassade wurden 3 *Session-Beans* verwendet, die die wesentlichen Aufgaben der serverseitigen Anwendung widerspiegeln:

- PswdManagementBean: verwaltet die wesentlichen Sicherheitsüberprüfungen, wie die Kontrolle des Login-Namens und des Kennworts sowie das Zurückliefern der Registrierungsdatei, einer Datei mit den zugangsberechtigten Personen im Krankenhaus und eine Liste der aktuell laufenden Studien
- XMLManagementBean: übernimmt die Speicherung der über die XML-Datei übermittelten Daten, sowie die Generierung einer weiteren XML-Datei, die

eine automatisch generierte Diagnose beinhaltet, wobei diese Datei wieder an den Client zurückgeliefert wird

- RequestHandlerBean: ist für Anfragen, die vom Web-Interface stammen, verantwortlich

3.3 Der Aufbau der Datenbank

Die Struktur der Datenbank ist in Abbildung 3.5 dargestellt. Beschreibung der verwendeten Tabellen:

- T_Country: dient der Zuordnung der Spitäler zu einem Land
- T_Hospital: beinhaltet die Spitäler, die eSCID verwenden
- T_Person: hier sind alle Ärzte, die eSCID verwenden, Sekretärinnen, die für den Arzt die Dateien zum Server senden können, sowie Administratoren eingetragen
- T_Patient: enthält Name und Anschrift des Patienten
- T_Hosp_Nr: Hilfstabelle, die es dem Arzt ermöglicht den Patienten über eine krankenhausinterne Nummer zu suchen; diese Nummer wurde nicht in der Patiententabelle gespeichert, da der selbe Patient in verschiedenen Krankenhäusern behandelt werden und somit unterschiedliche krankenhausinterne Nummern haben kann
- T_Trial: speichert im Krankenhaus laufende Studien
- T_Session: für jede Untersuchung mit dem eSCID (jede gesendete XML-Datei) erfolgt ein Eintrag in die T_Session Tabelle
- T_Version: zu jeder Sitzung wird eine Versionsnummer des verwendeten eSCID Fragebogens mitgespeichert
- T_EScid: speichert die Antworten des Patienten auf den SCID-Fragebogen
- T_OView: speichert allgemeine Informationen über den Patienten (Ehestand, Schulbildung, ...)
- T_Diagnose: speichert die vom Arzt ausgefüllte Diagnose

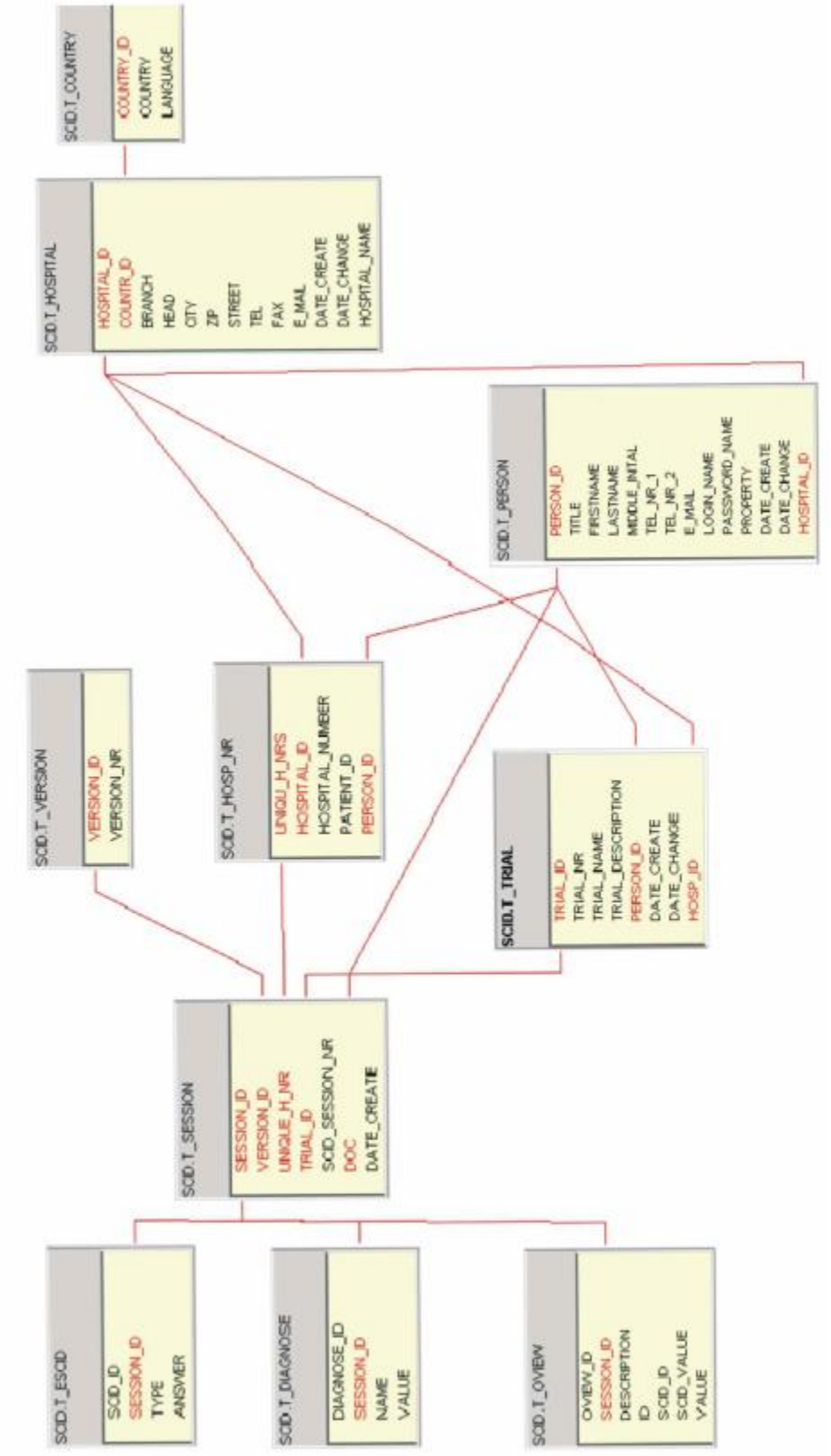


Abbildung 3.5: Entity-Relationshipmodell der Datenbank; diese Struktur entspricht den Anforderungen einer relationalen Datenbank; die vier Tabellen auf der linken Seite sind für die Speicherung des Fragebogens verantwortlich, während die anderen Tabellen die eindeutige Zuordenbarkeit der Daten gewährleisten;

3.4 Das Web-Interface

Beim Aufrufen der Web-Applikation erscheint zuerst eine Eingabemaske, bei der ein Login-Name und ein Kennwort einzugeben sind. Daraufhin wird der Benutzer entsprechend seiner Sicherheitsstufe eingeloggt. Bei dieser Programmversion wurde zwischen vier Sicherheitsstufen unterschieden:

- Secretary: es stehen alle Suchfunktionen, bis auf das Suchen nach einer bestimmten Diagnose zur Verfügung, dabei werden zwar die allgemeinen Daten des Patienten angezeigt, aber es besteht weder Zugriff auf die beantworteten Fragen noch auf die Diagnose, weiters können Angaben zur eigenen Person (wie z.B. Login-Name und Kennwort) und allgemeine Daten des Patienten geändert werden (z.B. Anschrift)
- User: hat alle Suchfunktionen zur Verfügung und kann auf alle Patientendaten zugreifen (entspricht einem gewöhnlichen Arzt in der Krankenhaushierarchie); weiters kann er neue *Secretaries* eintragen, sowie Angaben zu seiner eigenen Person ändern und auch neue Studien eintragen bzw. bestehende ändern
- Administrator: hat den selben Zugriff wie der *User*, der einzige Unterschied besteht darin, dass er auch neue *User* eintragen kann (entspricht einem Oberarzt in der Krankenhaushierarchie)
- Super User: ist eine einzige Person, die beim Erstellen der Datenbank automatisch eingetragen wird (entspricht dem Datenbankadministrator); ihm allein ist das Recht vorbehalten neue Krankenhäuser hinzuzufügen und Einträge von bestehenden Krankenhäusern zu ändern, sowie Administratorrechte zu vergeben

3.4.1 Die Suchfunktionen

Suchen nach Namen

Dabei hat der Benutzer entweder die Möglichkeit eine krankenhausinterne Nummer einzugeben und/oder einen Teil des Namens einzugeben. Dann wird entweder ein Patient mit dieser Spitalsnummer angezeigt, bzw. mehrere bei denen der eingegebene Teil mit dem Anfang des Namens übereinstimmt. Dabei ist anzumerken, dass dem behandelnden Arzt nur diejenigen Personen angezeigt werden, die von ihm behandelt wurden. Die selbe Suchfunktion steht auch zur Verfügung, wenn die allgemeinen Daten eines Patienten oder einer angestellten Person (hier keine krankenhausinterne Spitalsnummer vorhanden) geändert werden sollen.

Suchen nach Patienten, die in einem gewissen Zeitraum behandelt wurden

Hier kann der Arzt entweder das Anfangs- und das Enddatum des entsprechenden Zeitraums eingeben, und es werden daraufhin alle Patientenakten angezeigt, die in diesem Zeitraum vom eingeloggten Arzt erstellt wurden. Zusätzlich steht ein "All"-*Button* zur Verfügung, der alle Patienten anzeigt, die von diesem Arzt behandelt wurden.

Suchen nach Patienten, die einer Studie zugeordnet sind

Es wird zunächst eine Liste mit allen Studien, die es in diesem Krankenhaus gibt angezeigt. Daraufhin werden, nachdem eine Studie ausgewählt wurde, alle Patienten (auch diejenigen, die nicht von diesem Arzt behandelt wurden) dieser Studie angezeigt.

Suchen nach Krankheiten

Zunächst sieht der Arzt eine Liste von Krankheiten, und zwei Spalten mit *checkboxen*, wobei eine für die automatisch generierte Diagnose, und eine für die Diagnose des Arztes steht. Als Resultat werden die Patienten, auf die diese Diagnose zutrifft (und von diesem Arzt behandelt wurden), angezeigt.

3.4.2 Krankheitsspezifische Daten

Nachdem der Arzt einen Patienten ausgewählt hat (Auswahl siehe Abb.3.6), bekommt er die automatisch generierte Diagnose des Patienten angezeigt, und kann dann zwischen folgenden Möglichkeiten wählen:

- Compare: vergleicht die automatisch generierte Diagnose mit der vom Arzt abgegebenen Diagnose
- Compare Sessions: vergleicht alle Sitzungen eines Patienten, dabei wird sowohl die automatisch generierte, als auch die Diagnose des Arztes angezeigt
- Details: zeigt allgemeine Informationen des Patienten , wie z.B Ehestand, schulische Bildung, usw. und die Antworten auf den eSCID-Fragebogen an (siehe Abbildung 3.7)

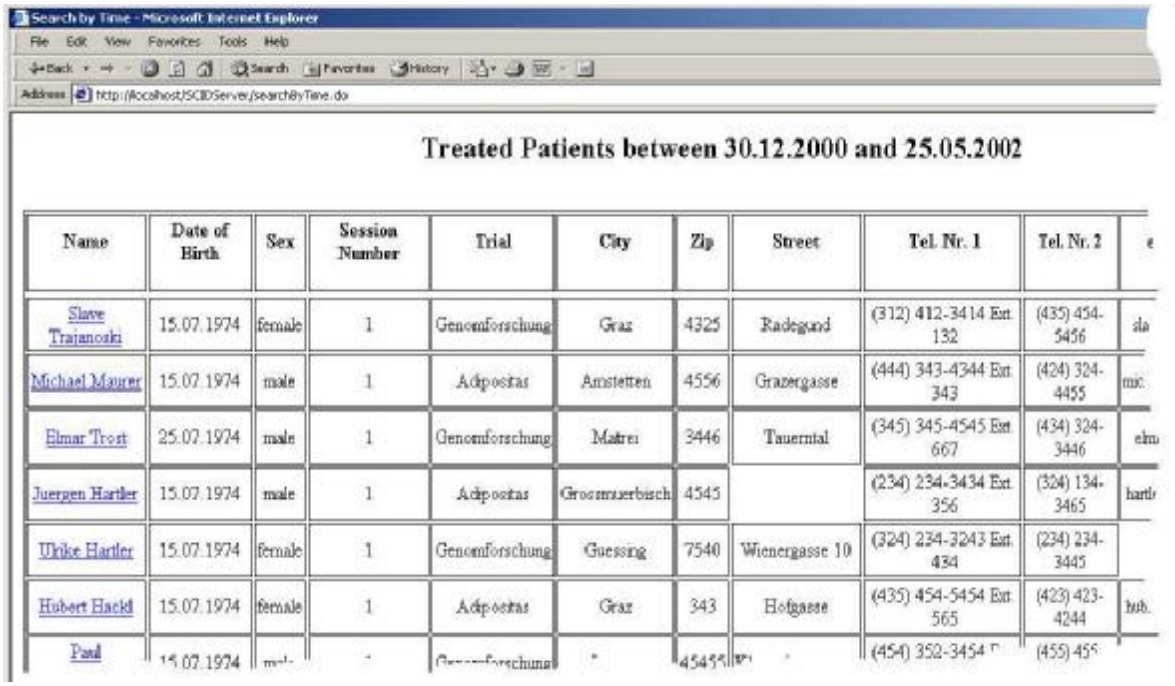


Abbildung 3.6: Auswahl eines Patienten über einen Link; eine Liste mit den Daten des Patienten wird angezeigt; ein Link, der sich auf dem Namen des Patienten befindet führt zur Diagnose des Patienten

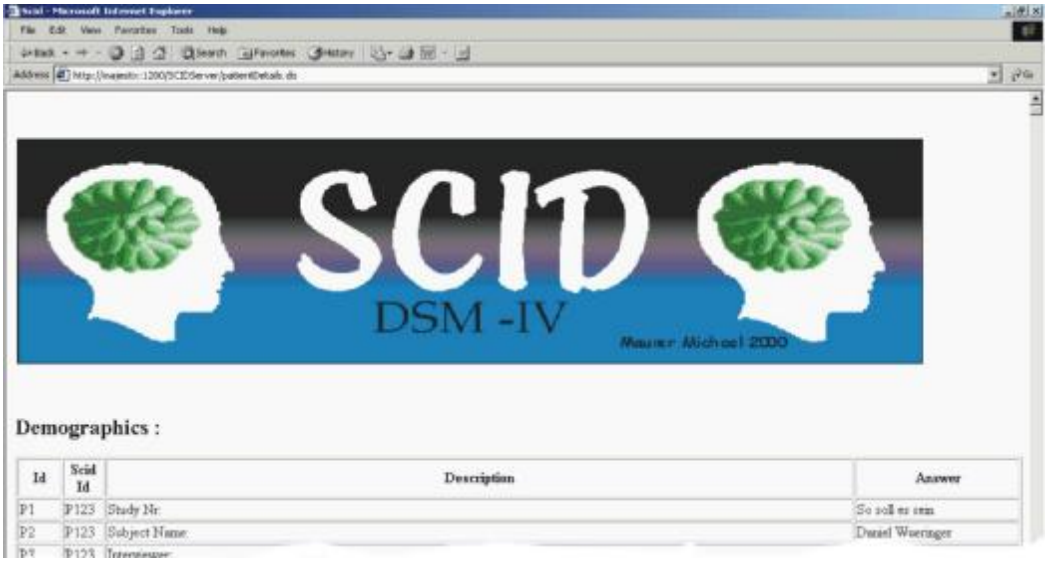


Abbildung 3.7: Anzeige der Details eine Patienten; hier werden allgemeine Informationen (Ehestand, Beruf), die Diagnose und die Antworten auf den eSCID-Fragebogen angezeigt

3.5 Der Usability-Test

Als Testpersonen standen 3 Medizinstudenten/Innen zur Verfügung. Es wurde jede der 6 gestellten Aufgaben (z.B. einen Patienten über seinen Namen suchen und die Antworten auf den eSCID-Fragebogen anzeigen) von jeder der Versuchspersonen gelöst. Dabei traten folgende Probleme auf:

- 2 von 3 erwarteten beim Suchen nach einer Studie eher ein Eingabefeld und nicht eine Liste, in der bereits alle Studien angeführt sind, während die dritte Person sich zwar eine Liste erwartete, aber nachdem sie auf die Eventualität eines Eingabefeldes angesprochen wurde, meinte, dass dieses nur sinnvoll wäre, wenn sehr viele Studien vorhanden wären
- allgemeine Verwirrung gab es zwischen den verwendeten Begriffen Person und Patient, obwohl jeder nach kurzem Nachdenken das richtige verwendete; die Meinung ging eher dahin, dass der Begriff User für Person passender wäre
- bei der Verlaufsdiagnose wurde von allen eher erwartet, dass die Krankheiten horizontal aufgelistet sind, und der Verlauf untereinander,

damit ein Verlauf der Krankheit leichter ersichtlich sei (momentan Krankheiten vertikal und Verlauf horizontal)

- 2 von 3 wollten allgemeinen Einträge des Patienten gleich in der Liste ändern, in der die zu ändernden Patienten angezeigt werden, anstatt über den Link (der sich auf dem Namen des Patienten befindet) auf das Eingabefeld zu gehen
- die Möglichkeit mit dem "Menu"-*Button* wieder in das Hauptauswahlmenü zu gehen wurde von einer Person überhaupt nicht erkannt, während die beiden anderen den *Button* erst später verwendeten; die Tendenz lag eher darin den "<<Back"-*Button* zu verwenden
- mit der Bezeichnung "Compare", um die automatisch generierte mit der eigenen Diagnose zu vergleichen, konnte keiner etwas anfangen

Weiters kam es aber nur zu sogenannten "Kajak-Problemen" [18], bei denen die Benutzer zuerst unsicher sind, dann eine oder mehrere Seiten zurück gehen, dann wieder die selbe Seite wie zuvor annavigieren und das Richtige tun. Allgemein hatte aber jede der drei Versuchspersonen den Eindruck, dass sie keine Probleme hätten auf Anhieb mit dem Programm umzugehen

3.6 Der Stresstest

Es wurde ein Stresstest mit 3 Benutzern durchgeführt, die gleichzeitig auf die selben und auf verschiedene Ressourcen zugriffen. Dieser Test wurde ohne merkbare Performanceeinbußen bestanden. Um die Strapazierbarkeit des Programms besser auszutesten, sollte noch ein Clientstresstest mit speziellen Clientoptionen durchgeführt werden.

Kapitel 4

Diskussion

In dieser Arbeit wurde eine Datenbank für psychische Krankheiten und die dazugehörige Software entwickelt. Diese Software stellt ein nützliches Werkzeug zur Verwaltung der Daten dar, das es ermöglicht diagnostische Parameter über ein Web-Interface anzuzeigen. In den nun folgenden Punkten werden die in Kapitel 3 beschriebenen Ergebnisse näher diskutiert.

4.1 Die Anmeldung

Die in Kapitel 3.1.1 besprochene Art der Anmeldung wurde vor allem aus Konsistenzgründen so gewählt. Dadurch ist eine Synchronisierung der gesendeten Daten mit den in der Datenbank gespeicherten Daten gewährleistet.
Der Nachteil besteht allerdings darin, dass jeder Rechner, der dieses Programm verwendet einen Internetanschluss haben muss (wenn auch nur zeitweilig). Dies wurde allerdings in Kauf genommen, da die so erhaltenen Daten für Studienzwecke nur dann sinnvoll sind, wenn sie auch zur Datenbank gesendet werden.

4.2 Verwendung von Enterprise-Java-Beans

Die großen Vorteile von EJBs sind vor allem in der Skalier- und Wiederverwendbarkeit der Komponenten zu suchen. Trotz der in Kapitel 2.1.2 besprochenen Vorteile gibt es auch ein paar Nachteile. So kann es beispielsweise bei komplizierten *Queries* und großen Resultsets zu Performanceeinbußen kommen. Deshalb wurde bei komplizierten Abfragen SQL der Vorzug gegeben.

4.3 Das automatisch generierte Resultat

Wie es der Aufgabenstellung (Kapitel 1.3) zu entnehmen ist, sollten die Fragebögen auf dem Server durch ein bereits existierendes Expertensystem ausgewertet werden. Da dieses Expertensystem bis zum Verfassen dieses Textes noch nicht zur Verfügung stand, wurde es nicht im Programm implementiert. Stattdessen wird momentan eine Dummy-Diagnose generiert.

4.4 Der Aufbau der Datenbank

Die in Kapitel 3.3 besprochene Struktur (siehe Abbildung 3.5) der Datenbank ermöglicht es, dass jede Sitzung eines Patienten sowohl einem Arzt als auch einem Krankenhaus zugeordnet wird und dass der Patient nur einmal in der Datenbank vorhanden ist, obwohl er in verschiedenen Krankenhäusern befragt wurde. Außerdem wurde durch eine Zwischentabelle, mit der krankenhausinternen Nummer auf die Gegebenheiten in einem Krankenhaus eingegangen. Ein Problem besteht allerdings darin, dass die behandelnden Ärzte im Falle eines Krankenhauswechsels noch einmal in die Datenbank eingetragen werden würden, da die Spitalsnummer ein Fremdschlüssel in der Personentabelle ist. Dieser Nachteil wurde aber in Kauf genommen, da bei einer Änderung der Personenakte auch der Patient nicht mehr zum entsprechenden Krankenhaus zuordenbar wäre.

Durch das Speichern der Versionsnummer des Fragebogens sind alte Einträge in der Datenbank genau der jeweiligen Version (und damit genau die Antworten den Fragen) zuordenbar. Dies sichert eine problemlose Erweiterbarkeit und Wiederverwendbarkeit von existierenden Daten.

4.5 Die Sicherheit

Die XML-Datei mit den sicherheitsrelevanten Informationen wird als Bytestrom über das Netz geschickt. Eine Verschlüsselung dieses Stroms sollte sich nicht als schwierig erweisen. Das Problem liegt in der Wahl des kryptographischen Verfahrens und der Schlüsselvergabe.
Das größere Problem liegt eher in der Verschlüsselung des Web-Interfaces. Es gibt zwar Literatur über die Verwendung von diversen Sicherheitskonzepten für die Verschlüsselung von Webseiten aber es gilt noch folgenden Punkte abzuklären:

- sind diese Verfahren für klinische Daten sicher genug
- sind diese Verfahren für jedes System anwendbar (Portierbarkeit)
- muss jeder Anwender einen eigenen Schlüssel erhalten und falls notwendig, wie erfolgt die Schlüsselübergabe
- soll das Web-Interface von jedem Rechner aus uneingeschränkt zugänglich sein oder nur von solchen, auf denen eSCID verwendet wird
- entstehenden Kosten für das verwendete Verfahren

4.6 Design und Usability

Das clientseitige Design sollte im wesentlichen den Erwartungen der Endbenutzer entsprechen und keine Problem mehr verursachen. Beim Web-Interface wurde nur die Funktion der einzelnen Komponenten implementiert, während das Design größtenteils nur auf das Notwendigste beschränkt wurde. Das Hauptaugenmerk lag

auf einer einwandfreien Funktion der Komponenten und einer leichten Erweiterbarkeit. Durch die Verwendung von Struts und der daraus folgenden Trennung zwischen Java-Code und HTML-Code, sollte sich die Verbesserung des Designs der JSPs als nicht besonders schwierig gestalten. Vor allem reicht für diese Tätigkeit die Kenntnis von HTML aus (keine Java-Kenntnisse notwendig).
Eine mögliche Homepagevorlage wurde bereits entwickelt (siehe Abbildung 4.1). Die Menüführung wurde in eine obere Leiste und in eine seitliche Leiste unterteilt. Dabei ist die obere Leiste das Hauptmenü und erst nachdem der Benutzer eine Auswahl getätigt hat erscheint das Untermenü in der Seitenleiste mit den zur Verfügung stehenden Funktionen.

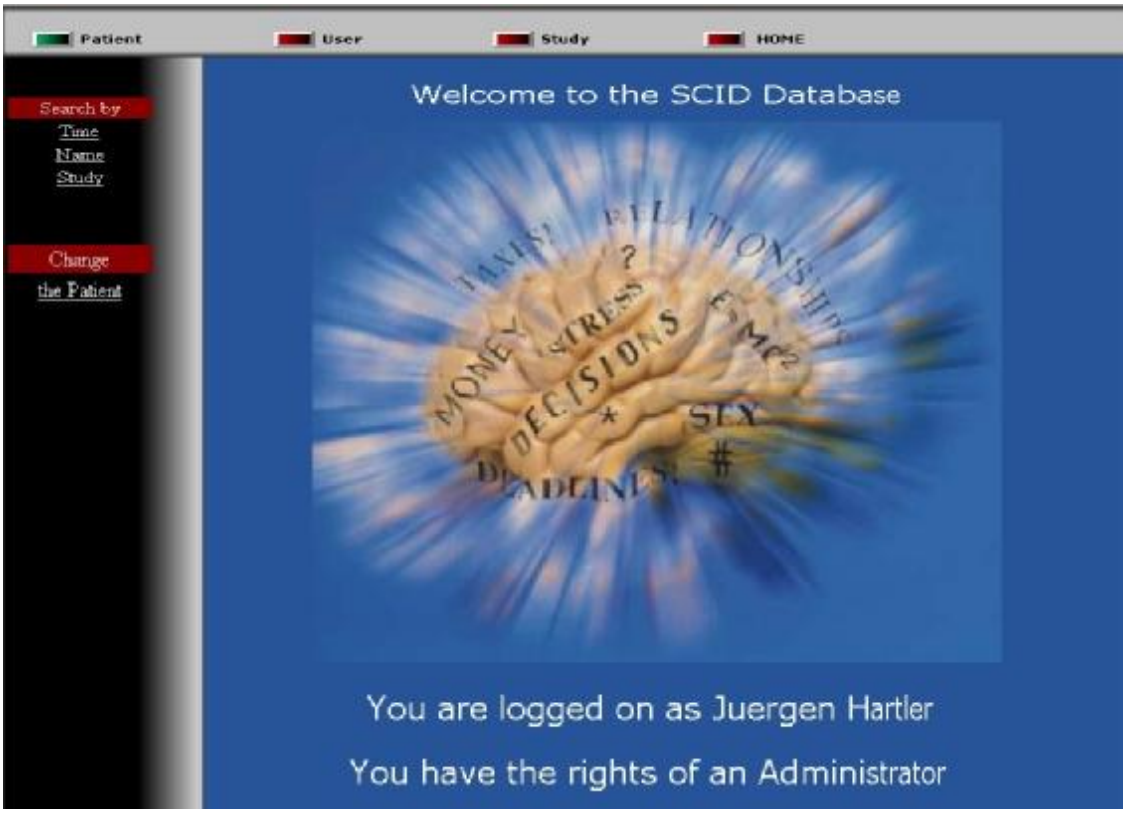


Abbildung 4.1: Mögliches Erscheinungsbild des Web-Interfaces von eSCID; nach einer Auswahl im oberen Menü stehen Funktionen auf der linken Seite zur Verfügung

Die Usability-Tests wurden mit Augenmerk auf die Navigierbarkeit durchgeführt, da ein entsprechendes Design noch nicht vorhanden ist. Die dabei erzielten Testergebnisse (siehe Kapitel 3.5) lassen bezüglich der Navigierbarkeit positive Rückschlüsse zu, sodass das momentane Konzept beibehalten werden sollte, und

nur kleinere Änderungen vorgenommen werden sollten. Der nächste Usability-Test sollte erst nach diesen Änderungen, und nachdem die einzelnen Seiten neu gestaltet wurden, durchgeführt werden.

4.7 Krankenhaushierarchie

Von der Entwicklerseite her erschien diese Lösung (*Super User, Administrator, User, Secretary*) als die sinnvollste, da sie es dem Krankenhaus ermöglicht die Verwaltung der Personen selbst vorzunehmen, der Datenbankbetreiber aber trotzdem die Oberaufsicht über die vorhandenen Personen hat. Außerdem können durch das Eintragen von *Secretaries* dem Arzt verwaltungstechnische Aufgaben abgenommen werden. Ob dieser Ansatz und ob die Zugriffsrechte auch wirklich so verteilt werden sollen, lässt sich erst nach Rücksprache mit den Endbenutzern beurteilen. Ein möglicher weiterer Ansatz wäre, beim Eintragen eines neuen Benutzers einen Security-Dialog (z.B. mit *checkboxen*) zu verwenden, in dem man individuell für jeden Benutzer ein Sicherheitsprofil festlegen kann.

4.8 Zusammenfassung

Abschließend bleibt noch anzumerken, dass die im Kapitel 1 gestellte Aufgabenstellung erfüllt wurde, und dass nach Klärung und/oder Implementierung der in diesem Kapitel angesprochenen Punkte ein nützliches Werkzeug zur Verwaltung und Auswertung von Daten, die psychische Krankheiten betreffen, zur Verfügung steht.
Die zentrale Speicherung der Daten bildet eine ideale Ausgangsbasis für den Vergleich von Diagnosetechniken und Krankheitsverläufen. Das Programm könnte an bestehende Krankenhausinformationssysteme angebunden werden und als Ausgangsbasis für krankenhausübergreifenden Studien zur Erforschung von psychischen Krankheiten dienen.

Die in nächster Zeit erwarteten ersten klinischen Erprobungen am National Institute of Mental Health (NIMH) sollten Aufschluss über einige der in diesem Kapitel angesprochenen Probleme bringen. Danach sollte diese Software in der klinischen Routine eingesetzt werden und somit eine wichtige Basis für Forschungen im Bereich der psychischen Krankheiten am NIMH und an der TU-Graz bilden.

Literaturverzeichnis

[1] Adatia R, Faiz A, et al.: *Professional EJB*. Wrox Press Ltd (2001)

[2] Andreasen NC: *Brave New Brain – Conquering Mental Illness in the Era of the Genome*. Oxford University Press (2001)

[3] Armstrong E: *The Java API for XML Processing (JAXP) Tutorial*. WWW, http://java.sun.com/xml/jaxp/dist/1.1/docs/tutorial/ (Version 1.1 Update 35: 10.12.2001)

[4] Bodoff S, Green D, et al.: *The J2EE™ Tutorial*. WWW, http://java.sun.com/j2ee/tutorial/1_3-fcs/index.html (24.04.2002)

[5] Bos B: *XML in 10 points*. WWW, http://w3.org/XML/1999/XML-in-10-points.html (27.03.1999)

[6] Cerami E: *Xerces Java: Quick Start*. WWW, http://ecerami.com/xerces/ (November 2000)

[7] Cover R: *The XML Cover Pages*. WWW, http://xml.coverpages.org/dom.html (13.12.2001)

[8] Fisher M: *Trail: JDBC (TM) Database Access*. WWW, http://java.sun.com/docs/books/tutorial/jdbc/index.html

[9] Garshol LM: *Xalan-Java*. WWW, http://www.garshol.priv.no/download/xmltools/prod/Xalan-J.html (07.05.2002)

[10] Getz K, Litwin P, et al.: *Microsoft Access 2 Developer's Handbook*. Sybex (1994)

[11] Gold PW, Goodwin FK, et al.: *Clinical and biochemical manifestations of depression. Relation to the neurobiology of stress(1).* N Engl J Med (1988); 319(6):348-53

[12] Gosling J, McGilton H, *The Java™ Language Environment: A White Paper.* WWW, ftp://ftp.javasoft.com/docs/papers/langenviron-pdf.zip (Mai 1996)

[13] Hilton G, Cattell R, et al.: *JDBC Database Access with Java: A Tutorial and Annotaded Reference*. WWW, http://java.sun.com/docs/books/tutorial/jdbc/index.html

[14] Jaworski J: *Java 2 Platform Unleashed*. Indianapolis, Sams (1999)

[15] Kendler KS, Walters EE, et al.: *The structure of the genetic and environmental risk factors for six psychiatric disorders in women. Phobia, generalized anxiety disorder, panic disorder, bulimia, major depression, and alcoholism*. Arch Gen Psychiatry (1995); 52(5):374-83

[16] Kochmer C: *An Introduction to Struts*. WWW, http://jspinsider.com/tutorials/jsp/struts/strutsintro.view (04.12.2001)

[17] Kreins DC, Brian Laskey: *Oracle Database Administration: The Essential Reference*. O'Reilly (April 1999)

[18] Krug S: *Don't Make Me Think - A Common Sense Approach to Web Usability*. Indianapolis, Circle.com.Library – New Riders Publishing (Oktober 2000)

[19] Krüger G: *Goto Java 2, 2.Auflage*. München - Boston - San Francisco - Harlow, England Don Mills, Ontario - Sydney - Mexico City - Madrid - Amsterdam, Addison-Wesley (2000)

[20] Kyte T: *Expert One-to-One: Oracle*. Wrox Press (excerpt posted November 2001), WWW,http://www.oracle.com/oramag/webcolumns/2001/4826_Chap01.pdf

[21] Lemay L, Cadenhead R: *Java 2 in 21 Tagen*. Markt und Technik Verlag (2002)

[22] Murray CJL, Lopez AD, et al.: *The Global Burden of Disease*. Geneva – Boston, World Health Organisation and Harvard University Press (1996)

[23] ÖNORM EN ISO 9241-10: *Ergonomische Anforderungen für Bürotätigkeiten mit Bildschirmgeräten; Teil 10 Grundsätze der Dialoggestaltung.* (1996)

[24] Steinberg D: *J2EE™ Design Patterns.* Java World (07.06.2001), WWW, http://java.sun.com/blueprints/patterns/index.html

[25] Strock M: *Depression*. NIH Publication No.00-3561

[26] Sun Microsystems: *Java Beans*. WWW, http://java.sun.com/products/javabeans

[27] Sun Microsystems: *Java 2 Enterprise Edition Developers Guide (Version 1.2.1).* WWW, http://java.sun.com/j2ee/sdk_1.2.1/techdocs/guides/ejb/html/Overview3.html

[28] Sun Microsystems: *Java Remote Method Invocation – Distributed Computing for Java: White Paper*. WWW, http://java.sun.com/marketing/collateral/javarmi.html

[29] Sun Microsystems: *JDBC Data Access API.* WWW, http://java.sun.com/products/jdbc

[30] Wollrath A, Waldo J: *The Java™ Tutorial: Trail RMI – An Overview of RMI Applications*. WWW, http://java.sun.com/docs/books/tutorial/rmi/overview.html

[31] W3C DOM Working Group: *Document Object Model (DOM)*. WWW, http://www.w3.org/DOM (29.03.2002)