Entwicklung eines webbasierten Auskunftssystems für Schienenfahrzeuge unter Verwendung von J2EE Technologien

Diplomarbeit

zur Erlangung des akademischen Grades

Diplom-Informatiker

eingereicht an der

Fachhochschule für Technik und Wirtschaft Berlin

Fachbereich Wirtschaftswissenschaften II

Studiengang Angewandte Informatik

1. Betreuer: Prof. Dr.-Ing. habil. Rüger Oßwald

2. Betreuer: Dipl.-Inf. Mathias Müller, ETC Transport Consultants GmbH

Eingereicht von Daniel Prusseit im November 2004

ETC TRANSPORT CONSULTANTS

FHTW
Fachhochschule für Technik und Wirtschaft Berlin

Inhalt:

1 Einleitung

1.1 Danksagung

An dieser Stelle danke ich Herrn Prof. Oßwald für das Zustandekommen dieser Arbeit durch seine guten Kontakte zur ETC Transport Consultants GmbH. Den Mitarbeitern der ETC, insbesondere Herrn Müller, danke ich für die großzügige Bereitstellung der technischen und fachlichen Voraussetzungen für die Umsetzung der Arbeit.
Ein großes Danke auch an meine Eltern, an meine Freundin Grete und an Euch andere geduldige Korrekturleser.

1.2 Motivation

Seit es Computer gibt, besteht die Notwendigkeit, für sie Software zu erstellen. Heutzutage sind Softwaresysteme aus unserem täglichen Leben nicht mehr wegzudenken, und viele praktische Dinge wären ohne funktionierende Software nicht denkbar. Das Erstellen guter Software ist jedoch mit der Entwicklung leistungsfähigerer Hardware nicht einfacher geworden. Beinahe über Nacht werden bisher unrealisierbar scheinende Anwendungen möglich. Ein Aspekt, der bereits Mitte des letzten Jahrhunderts im Wort „Softwarekrise“ zum Ausdruck kam und bis heute immer wieder zur Einführung neuer Softwaretechniken und zu Paradigmenwechseln führte.
An einem solchen Wendepunkt befinden wir uns nach Ansicht des Autors gerade wieder. Leistungsfähige Hardware sowie eine hohe Verbreitung und Akzeptanz des Internet werden sehr schnell dafür sorgen, dass sich die modernen komponenten- und objektorientierten Entwicklungstechniken endgültig durchsetzen. Nach dem „Internet Hype“ (siehe [Klinger04]) der letzten Jahre besteht jetzt die Herausforderung, leistungsfähige, wiederverwendbare und performante Webanwendungen zu entwickeln.

1.3 Zielstellung

Die vordergründige Zielstellung dieser Diplomarbeit ist die Erstellung eines Fahrzeuginformationssystems in Zusammenarbeit mit der ETC Transport Consultants GmbH in Berlin. Die durch den Verfasser zu erstellende Software ist ein Auskunftssystem zur Abfrage der in der ETC vorhandenen Schienenfahrzeugdatenbank. Sie ermöglicht eine Suche über Fahrzeuge und deren technische und wirtschaftliche Details und bietet Abfragemöglichkeiten über Unternehmen mit Bezug zum Schienenverkehr wie z.B. Verkehrsunternehmen und Fahrzeughersteller.
Das Auskunftssystem soll als eine Webanwendung entwickelt werden und zukünftig in der ETC eingesetzt werden. Eine weitere Vorgabe besteht in der Verwendung von modernen J2EE-Technologien, unter anderem durch Einsatz von EJB-Komponenten.
Vor der eigentlichen Entwicklung der Anwendung wird sich der Autor daher kurz mit dem Softwareentwicklungsprozess auseinandersetzen und die Idee und den Einsatz von Softwarekomponenten herausarbeiten. Hiernach wird sich der Verfasser mit der J2EE-Umgebung von Sun Microsystems befassen und die für die Entwicklung des Auskunftssystems geeigneten Technologien bestimmen. Dabei werden sich viele Möglichkeiten bieten, im universitären Rahmen erworbene Kenntnisse über verschiedene Techniken zu vertiefen und in der Praxis zu erproben.
Letzten Endes wird der Autor die Struktur des Auskunftssystems entwerfen und die Anwendung implementieren. Hierbei steht die effiziente Arbeit mit modernen Entwicklungswerkzeugen und eine enge Zusammenarbeit mit den späteren Nutzern in der ETC im Vordergrund.

1.4 Aufbau und Form der Arbeit

Die Diplomarbeit besteht aus insgesamt sechs Kapiteln. Die theoretische Vorbetrachtung, Analyse und kritische Auseinandersetzung mit Entwicklungstechnologien findet sich in den Kapiteln 2 und 3. Den größten Teil der Arbeit nimmt die Entwicklung der Software von der Anforderungsdefinition über den Entwurf bis zur Implementierung in Kapitel 4 ein. Kapitel 5 enthält schließlich eine Zusammenfassung und Wertung der Entwicklungsarbeit und gibt Ausblicke auf den Einsatz und Erweiterungsmöglichkeiten der Software.

Im Anhang ist weiterhin das im Verlauf der Entwicklung angefertigte Pflichtenheft für das Auskunftssystem enthalten.

Die Arbeit wurde mit Hilfe von Microsoft Word 2000 erstellt. Die Diagramme und Schemata entstanden überwiegend unter Verwendung von Visio 2000. Als zusätzliche Bildbearbeitungssoftware wurde Paint Shop Pro eingesetzt.

Um eine optimale Lesbarkeit des Dokumentes in Papierform zu garantieren, wurde der Fließtext in der Serifenschriftart Times New Roman in der Größe 11 Punkt gesetzt. Überschriften und Beschriftungen von Abbildungen, Tabellen und Quelltextauszügen liegt die serifenlose Schriftart Arial zu Grunde. Quelltextauszüge und einzelne Anweisungen wurden mit der Schriftart Lucida Console 10 Punkt umgesetzt.

Tabelle 1: Beispielquelltext mit Beschriftung

```
SELECT typ
FROM fahrzeuge
WHERE typ = 'ICE';
```

Beigelegte CD

Dieser Diplomarbeit ist eine CD-ROM beigelegt, die diesen Text im Portable Document Format (PDF) sowie im Format Microsoft Word (DOC) enthält.

Weiterhin beinhaltet die CD zwei Präsentationen im Format Microsoft Powerpoint (PPT). Während eine dieser Präsentationen die Grundlage der Anwendungsdemonstration in der ETC bildete, bietet die andere anhand animierter Bildschirmaufnahmen einen anschaulichen Eindruck der erstellten Anwendung.

2 Komponenten in der Softwareentwicklung

2.1 Traditionelle Vorgehensmodelle

Die Entwicklung einer korrekten und funktionsgerechten Software ist ein komplexer Vorgang. In [Dumke03, S.6] wird dieser Software-Entwicklungsprozess beschrieben als der „gesamte Prozess der Aufgabenstellung, Planung, Realisierung und Bewertung einer Software-/Hardware-Anwendung“. Dieser Entwicklungsprozess lässt sich durch verschiedene Vorgehensmodelle wie beispielsweise das Wasserfall-Modell oder die so genannte evolutionäre Entwicklung beschreiben. Diese Modelle unterteilen die Entwicklung in verschiedene Phasen, die von der groben Beschreibung bzw. einer Anforderungsdefinition zum Einsatz des fertigen Softwaresystems führen.

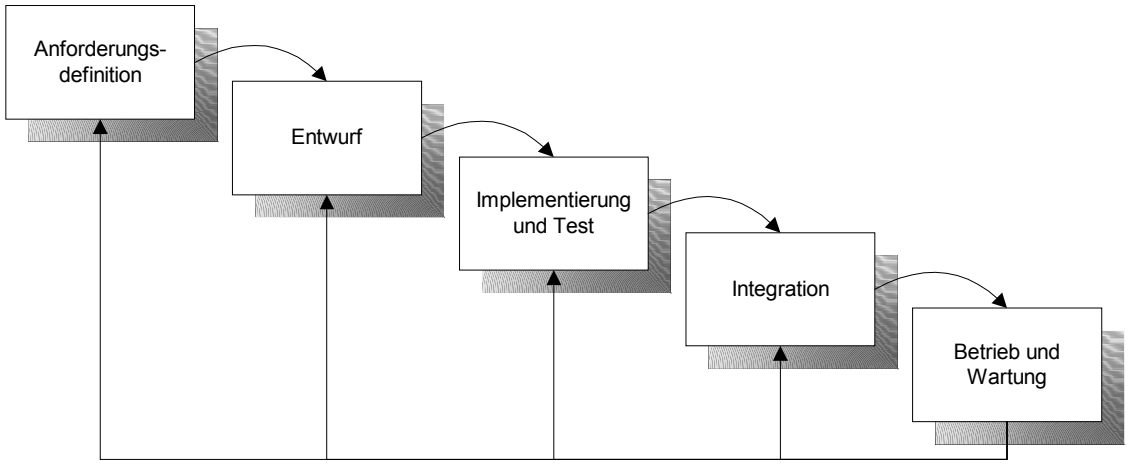

Abbildung 1: Das Wasserfall-Modell

In vielen Punkten sind die traditionellen Vorgehensmodelle zu starr, um den Anforderungen der Entwicklung zeitgemäßer Software gerecht zu werden. Nur wenige der Modelle berücksichtigen in Ansätzen Aspekte der Wiederverwendung bereits bestehender Software. Trotzdem stellt [Sommerv01, S.58] noch 2001 fest, dass das nicht mehr zeitgemäße Wasserfallmodell noch immer die „momentane Entwicklungspraxis widerspiegelt“.

2.2 Komponenten-Idee

Komplexe Software besteht aus einer Vielzahl von Teilen, die sich durch ihre Funktion im Gesamtsystem charakterisieren lassen. Beispielsweise könnte zu einem Bestellsystem eine Kundenverwaltung, eine Rechnungserstellung oder auch eine Funktion zum Sichern der Daten gehören. Während traditionelle Vorgehensmodelle die Entwicklung der Software „als Ganzes“ verfolgen, ist es doch vorstellbar, die genannten Teile einer Software als unabhängige, ersetzbare Bausteine zu erstellen und die eigentliche Anwendung somit nach dem Baukastenprinzip zusammenzusetzen:

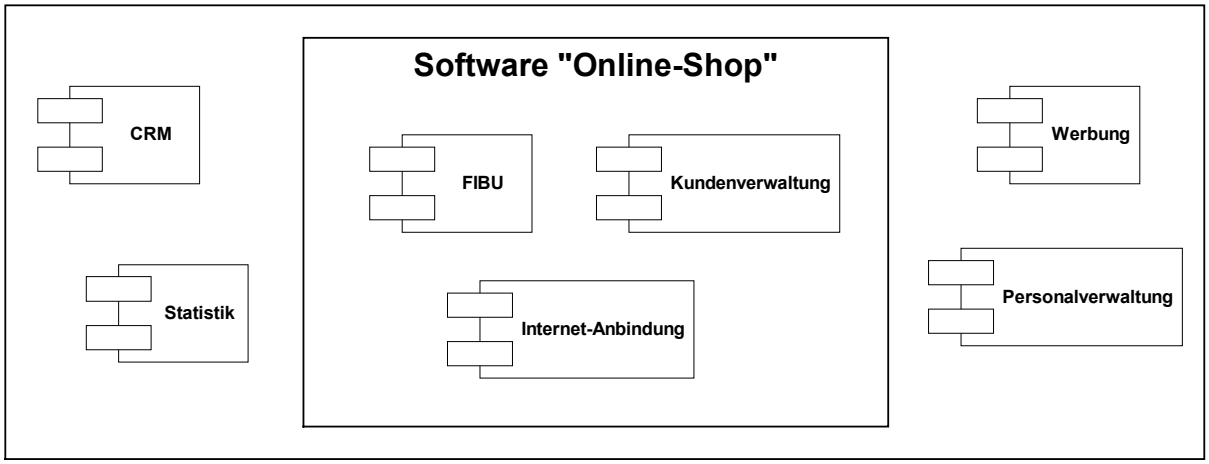

Abbildung 2: Komponenten als Teil einer Anwendung

Ansätze für wiederverwendbare Softwarebausteine finden sich bereits in verschiedenen Technologien, [Dumke03, S.350] führt beispielsweise die aus der Windows-Welt als DLL bekannten Modulbibliotheken oder den Einsatz von Standardsoftware an, beinhalten jedoch bei weitem nicht alle Vorteile moderner komponentenbasierter Techniken.

2.3 Der Komponentenbegriff

Zum Begriff der Software-Komponente existieren in der Literatur die verschiedensten Definitionen, die sich darin unterscheiden, welche Softwarestrukturen bereits als Komponenten bezeichnet werden oder welche konkreten Anforderungen an eine Komponente bestehen. Folgende Merkmale finden sich jedoch in beinahe allen Definitionen und charakterisieren eine Komponente sehr gut:

Tabelle 2: Eigenschaften einer Komponente

- Eine Komponente ist eine unabhängige, wiederverwendbare, ausführbare Softwareeinheit.
- Sie besitzt einen nichttrivialen Funktionsumfang.
- Interaktionen geschehen über nach außen definierte Schnittstellen.
- Der Quellcode der Komponente ist in der Regel nicht zugänglich.

[Dumke03, S.350] und [Zuser01, S.119] beschreiben eine Komponente zudem zwingend als Teile eines Systems. Dies schließt jedoch der Ansicht des Verfassers nach fälschlicherweise aus, dass eine einzelne Komponente isoliert existiert und eingesetzt wird.

Bis vor kurzem wurden Komponenten zusätzlich implizit durch technische Details wie Plattformunabhängigkeit oder Zustandsbehaftung definiert [Piemont99, S.294]. Diese eher für die Programmierung relevanten Eigenschaften finden sich in aktuellen Definitionen nicht mehr wieder. Für den Autor sehr treffend beschreibt jedoch [Piemont, S.293] Komponenten als „Black Boxen“, die zur Lösung einer komplexen Gesamtaufgabe verknüpft werden können.

2.4 Arten und Einsatz von Komponenten

Arten von Komponenten

Komponenten lassen sich unter anderem hinsichtlich ihrer angebotenen Funktionen, Größe und Wiederverwendbarkeit als auch über ihren Grad der Verteilung unterscheiden.
[Brown98] unterscheidet beispielsweise so genannte „COTS-Komponenten“ (commercial off-the-shelf components), d.h. große kommerzielle, voll integrierbare Komponenten von anderen Komponentenarten, die in ihren Schnittstellen mehr oder weniger variabel an vorhandene Umgebungen und andere Komponenten angepasst werden können.
Bei der Zusammenstellung einer Anwendung bietet sich die Möglichkeit, verschiedene Arten von Komponenten zu kombinieren, einige einzukaufen und andere selbst zu entwickeln.

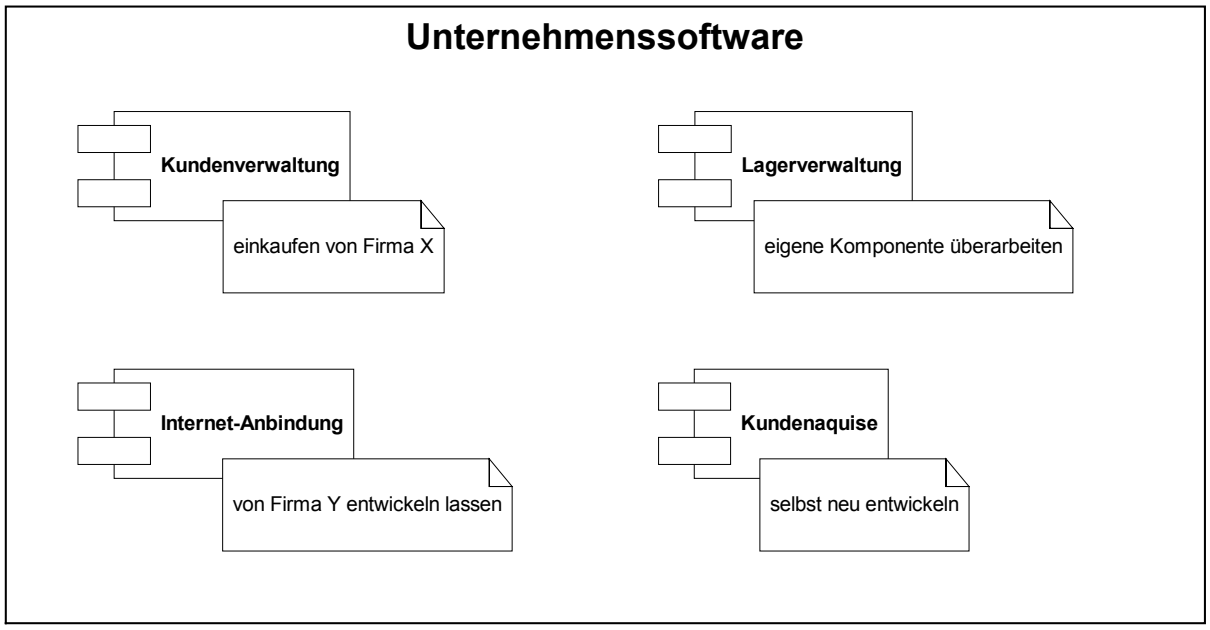

Abbildung 3: Komponenten kaufen, entwickeln oder entwickeln lassen?

Beispielsweise ist es denkbar, dass im Rahmen einer Entwicklung einer Unternehmenssoftware eine Kundenverwaltung als COTS-Komponente gekauft und der sehr spezifische Funktionsumfang der Neukundenaquise als so genannte „Qualifizierte Komponente“ selbst entworfen und umgesetzt wird.
Sind Komponenten über die Hardware verteilt, spricht man hinsichtlich der Laufzeitumgebung der Komponente auch von Client- und Serverkomponenten. Beispiel für ein clientseitiges Komponentenmodell sind die Java Beans, nicht zu verwechseln mit Suns serverseitigem Komponentenmodell der Enterprise Java Beans.

Wiederverwendung

Eine der zentralen Eigenschaften einer guten Softwarekomponente ist deren Wiederverwendbarkeit. [Sommerv01, S.316] und [Stevens00, S.254] nennen die entscheidenden Vorteile von Wiederverwendung:

Tabelle 3: Vorteile von Wiederverwendung

- höhere Zuverlässigkeit bereits praxiserprobter Komponenten
- Entwicklung solcher Komponenten durch Anwendungsspezialisten
- Bilden gewisser Standards hinsichtlich Schnittstellen, Oberflächen etc.
- Beschleunigung der Entwicklung neuer Software

Gleichzeitig wird jedoch angeführt, welche Probleme bei Wiederverwendung auftreten können und warum sie allgemein schwierig zu realisieren ist. Genannte Quellen und [Zuser01, S.180] nennen unter anderem folgende Gründe:

Tabelle 4: Nachteile von Wiederverwendung

- Finden und Auswahl der „richtigen" Komponente schwierig
- höhere Wartungskosten, da eine wiederverwendete Komponente zunehmend altert und inkompatibel wird
- „Aufblähung" von Software durch Einbinden vieler ungenutzter Funktionen
- unzureichende Dokumentation oder schlechter Programmierstil
- fehlende domänenunabhängige Datenmodelle
- psychologische Aspekte, die beispielsweise Neuentwicklungen als Herausforderung und „Fremdprodukte" nicht vertrauenswürdig erscheinen lassen

Komponentenorientierte und –basierte Entwicklung

Zusammenfassend unterscheidet man also in der Softwareentwicklung mit Komponenten zwischen dem Schreiben von neuen Komponenten (in der Literatur auch als „komponentenorientiertes Software Engineering" bezeichnet) und der Entwicklung basierend auf bestehenden Komponenten, d.h. Wiederverwenden oder Integrieren vorhandener Komponenten. [Stevens00, S.257f.] stellt in diesem Zusammenhang beispielsweise danach Fragen, wer im Unternehmen mit wessen Finanzen und Resourcen die wiederverwendbaren Komponenten entwickeln soll. Ist jedoch eine geeignete Komponente vorhanden, so ist deren Einsatz allemal „wichtiger, nutzbringender und einfacher" als eine Neuentwicklung. Eine Feststellung, die sich im Programmieralltag erst noch durchsetzen muss...

3 Die J2EE Technologien

3.1 Einführung

Mit zunehmender Verbreitung des Internet entstand binnen kurzer Zeit ein hoher Bedarf an verteilten Anwendungen. Der Begriff „Enterprise Computing“, so ist es bereits in der Einleitung bei [Farley03, S.3] zu lesen, bezeichnet nichts anderes als diese verteilten Anwendungen, unabhängig davon ob die „Enterprise“ ein großes Unternehmen oder eine kleine Organisation darstellt. In der J2EE-Architektur hat Sun Microsystems eine Reihe von Technologien zum Entwickeln verteilter Anwendungen zusammengefasst. Ein J2EE Applikationsserver stellt zur Laufzeit der Anwendung eine „wohl definierte Ablaufumgebung“ ([Kompf04, S.22]) zur Verfügung.

Die Elemente der J2EE

Die Spezifikation der J2EE besteht aus verschiedenen Elementen. Die für das Entwickeln einer Anwendung wichtigsten Elemente und ihr Zusammenspiel in einer Anwendung sind in der folgenden Abbildung aus [Sun04] dargestellt.

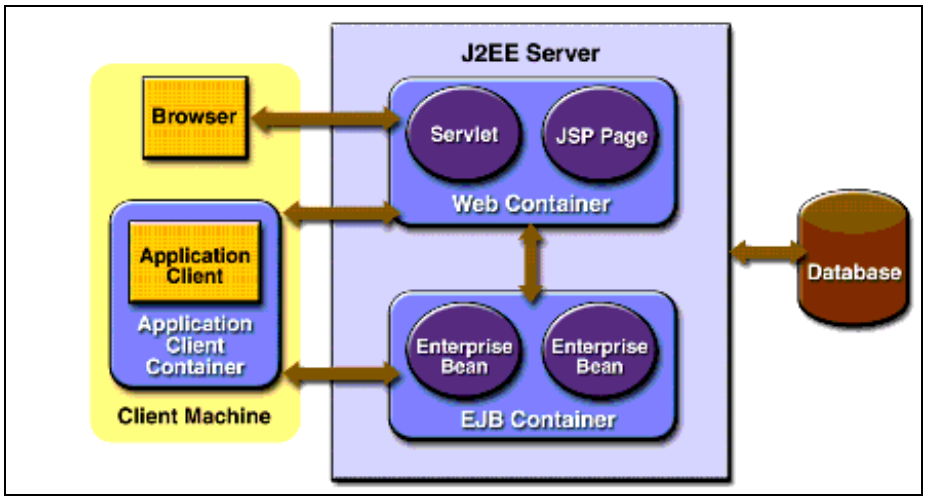

Abbildung 4: Wichtige Elemente der J2EE

Daneben existieren weitere APIs und Technologien, unter anderem zur Integration von Java und Web Services oder XML-basierter Dienste, die beispielsweise der Übersicht in [Farley03, S.16] entnommen werden können.

3.2 Enterprise Java Beans

Enterprise Java Beans (EJB) sind die standardisierte Architektur für verteilte Komponenten in Java. Sie sind die logische Erweiterung des in Java vorhandenen Frameworks für verteilte Objekte (RMI) und dem existierenden Komponentenmodell der Java Beans. Ihr Einsatz ermöglicht jedoch weitaus mehr als das entfernte Aufrufen von Methoden oder das Kapseln von Funktionalität und Anwendungsdaten. Durch die Ablaufumgebung einer EJB, den so genannten EJB Container, werden eine Reihe nützlicher Dienste wie Transaktionsmanagement und Persistenz zur Verfügung gestellt, um deren Implementierung sich der Anwendungsentwickler nicht kümmern muss.
Die aktuelle Spezifikation von EJB in der Version 2.1 enthält drei Typen von EJB: (siehe auch [Kompf04, S.25f.], [Farley03, S.242], [Zimmer00, S.288])

Tabelle 5: Arten von Enterprise Java Beans

Session Beans
Session Beans dienen zur Implementierung von Prozessen, Anwendungsfällen und Algorithmen. Hierbei unterscheidet man Session Beans, die nicht zustandsbehaftet und nur für einen einzelnen Methodenaufruf an den Client gebunden sind („stateless“) und solchen, die den Status genau eines Clients während ihrer gesamten Lebensdauer verwalten („stateful“).

Entity Beans
Entity Beans repräsentieren persistente Daten, die in der Regel in einer relationalen Datenbank abgelegt sind. Je nachdem ob der Entwickler die Zugriffsmechanismen über JDBC und SQL in der Bean selbst erstellt oder deren Erstellung dem EJB-Container überlässt, spricht man von „Bean Managed Persistence“ (BMP) und „Container Managed Persistence“ (CMP).

Message Driven Beans
Seit Einführung von EJB 2.0 ermöglicht dieser EJB-Typ eine asynchrone Interaktion zwischen Client und Server, indem nicht explizit Methoden der Bean aufgerufen werden, sondern die Bean auf Nachrichten (Messages) des Client reagiert.

In einem großen Software-Projekt, welches mit Hilfe von EJB umgesetzt wird, sind in der Regel alle drei Bean-Typen beteiligt.

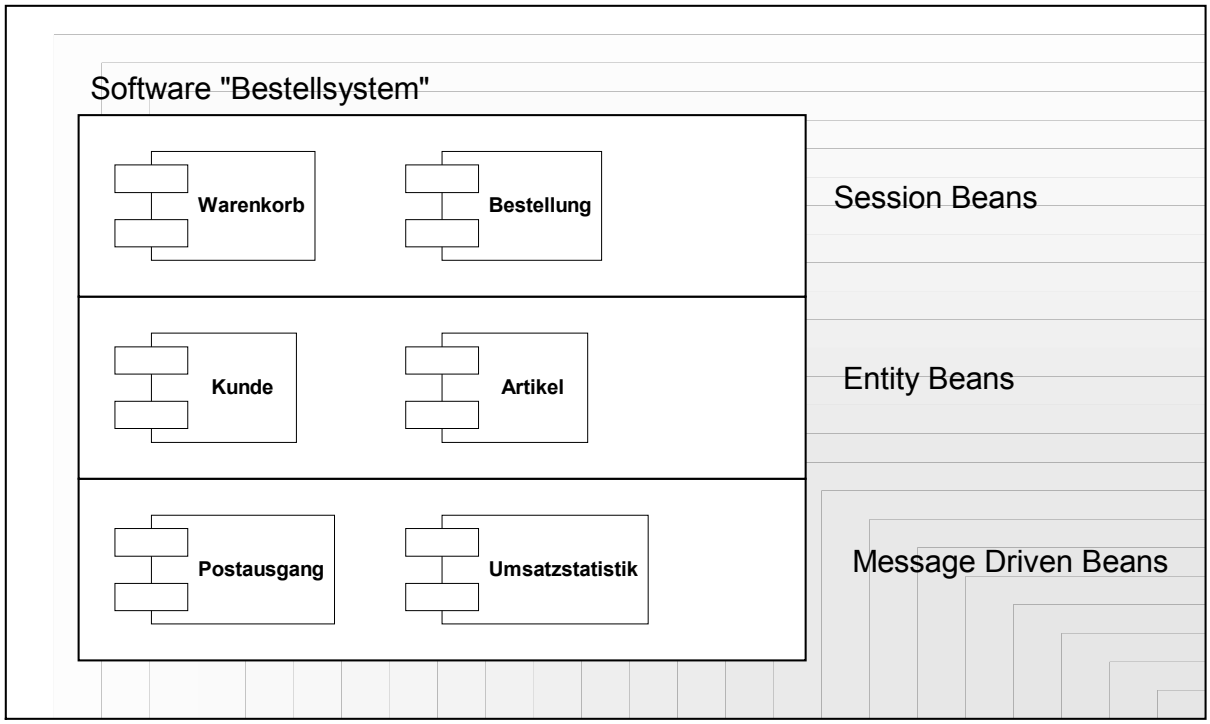

Abbildung 5: Bean-Typen in einem Softwareprojekt

In dem in der Abbildung dargestellten Beispiel sind mehrere Session-Beans für konkrete Nutzeraufträge und –transaktionen zuständig. Dabei wird beispielsweise die Logik und Transaktionalität eines Bestellvorgangs umgesetzt. Entity Beans realisieren die Sicht auf die Datenbank und bieten Zugriff auf Kundendaten und angebotene Artikel. Die in der derzeitigen Entwicklungspraxis noch relativ selten anzutreffenden Message Driven Beans werden hier zur Umsetzung des Postausgangs eingesetzt und registrieren jeden Bestellvorgang in einer Statistik.

Bestandteile einer EJB

Eine EJB besteht im Allgemeinen aus zwei Interfaces und einer Implementierung der Bean. Dabei stellt das so genannte Home-Interface Methoden zum Erzeugen, Finden und Zerstören einer Bean zur Verfügung. Die eigentlichen Geschäftsmethoden werden im Remote-Interface deklariert, welches in neuerer Literatur wie [Farley03, S.245] auch als Client-Interface bezeichnet wird.

Die folgende Abbildung aus [J2EE-Tut04] zeigt beispielhaft den Zugriff auf eine EJB, die ein Bankkonto darstellt. Das Home-Interface bietet Methoden zum Erstellen, Löschen und Auffinden eines Kontos, das Remote-Interface enthält die Geschäftsmethoden zum Einzahlen und Abheben von Geld.

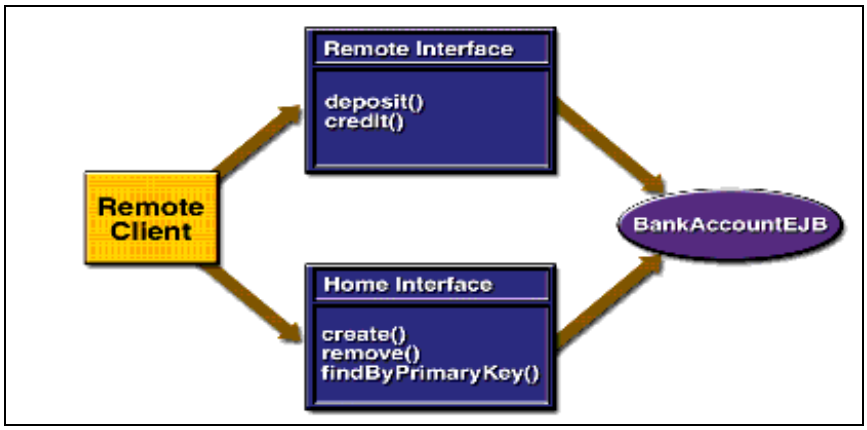

Abbildung 6: Zugriff auf eine EJB über Interfaces

Die Realisierung der Funktionalität erfolgt schließlich in der Implementierung der Bean. Die Implementierung von Methoden wie beispielsweise die Persistenzsteuerung oder das Auffinden einer Bean durch einen Primärschlüssel erfolgt beim entsprechenden Bean-Typ automatisch durch den EJB-Container.

3.2.1 Rollen in der Softwareentwicklung

Beim klassischen Software Engineering werden die direkt an der Erstellung der Software beteiligten Personen in der Regel in Entwicklungsteams oder Arbeitsgruppen zusammengefasst. Die Mitglieder eines Teams nennt [Zuser01, S.36]:

Tabelle 6: Zusammensetzung eines klassischen Entwicklungsteams

Projektleiter	trägt die wirtschaftliche und technische Verantwortung
Gruppenleiter	steht an der Spitze einer Arbeitsgruppe
Analytiker	erstellt Anforderungsanalyse, idealerweise kein Programmierer
Integrator	hauptverantwortlich für Entwurf, Schnittstellen, Inbetriebnahme
Programmierer	implementieren das Projekt
Tester	erstellt Testszenarien, testet fertige Anwendung, erstellt Fehlerberichte
Qualitätssicherer	überprüft Einhaltung der Entwicklungsvorgaben

Die Zusammenstellung eines solchen Teams ist ohne Zweifel sehr stark an den in 2.1 vorgestellten Vorgehensmodellen orientiert. Jedes Mitglied hat im Idealfall sehr gute Kenntnisse in genau dem Bereich, in dem es arbeitet. Dennoch sind Kenntnisse aus anderen Fachgebieten unbedingt für eine Zusammenarbeit nötig. Beispielsweise benötigt ein Tester Kenntnisse darüber, wo eine Software implementierungsabhängige Schwachstellen haben könnte. Weiterhin ist erkennbar, dass die Anwendung bei einer klassischen Rollenverteilung im Prinzip „in einem Stück“ durch eines oder bei größeren Projekten durch einige wenige Teams entworfen, entwickelt, getestet und installiert wird.

3.2.2 Rollen in der EJB-Entwicklung

Der Einsatz von J2EE-Technologien und insbesondere EJB in der Entwicklung verändert nicht nur das Design der Anwendung und die Techniken der Implementierung sondern auch nachhaltig die Rollenverteilung im Softwareentwicklungsprozess. Leider werden die Rollen im EJB-Entwicklungsprozess auch in aktueller Literatur oftmals entweder komplett ignoriert oder nur sehr kurz aufgezählt, obwohl sie nach Ansicht des Autors zu einem großen Teil zur Effizienz der Technologie beitragen!

[J2EE-Tut04] beschreibt die in der Abbildung gezeigten Rollen:

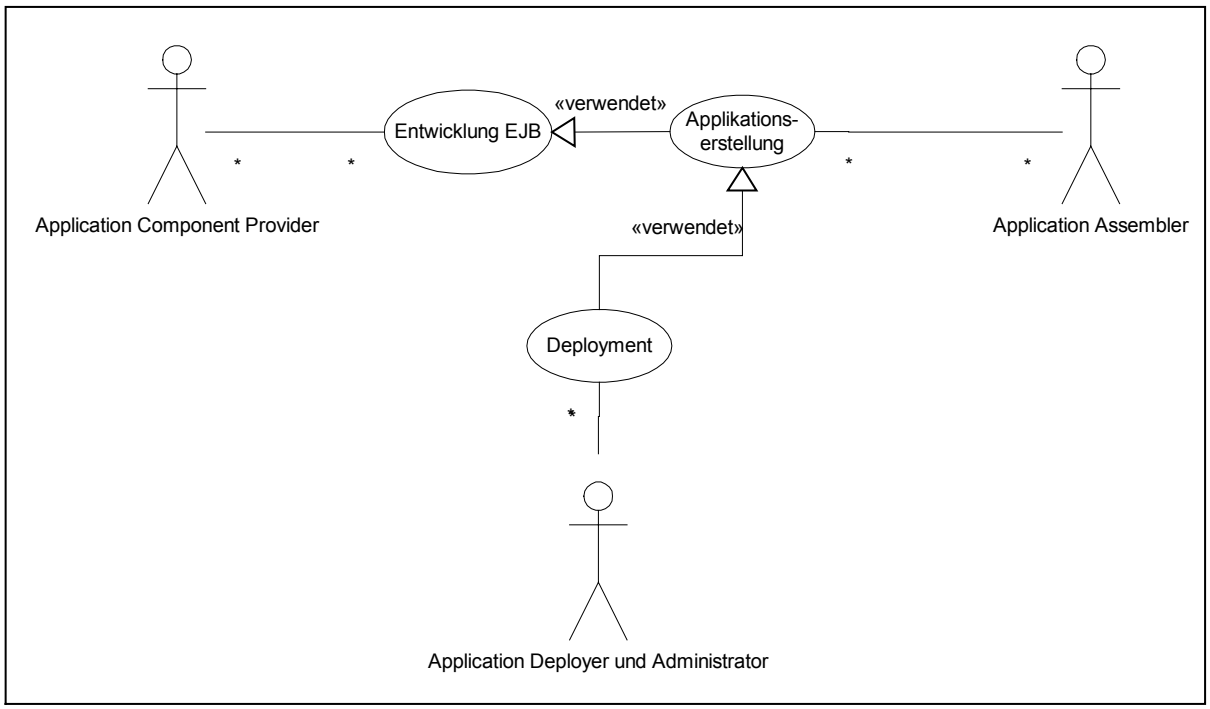

Abbildung 7: Rollen in der EJB-Entwicklung

Nachfolgend werden die dargestellten Rollen und ihre Aufgaben im Entwicklungsprozess beschrieben.

Tabelle 7: Rollen in der EJB-Entwicklung

Application Component Provider
Hiermit wird das Unternehmen oder die Person bezeichnet, die Webkomponenten, EJB, Applets oder Clients für J2EE-Umgebungen entwickelt. Dabei kann weiterhin zwischen dem Enterprise Bean Developer (entwickelt EJB), Web Component Developer (entwickelt Servlets und JSP, entwirft HTML) und dem J2EE Application Client Developer (schreibt Clients) unterschieden werden. Der Application Component Provider, teilweise auch als „Bean Provider“ bezeichnet, ist also für die einzelnen Komponenten, deren Design und Funktionalität und deren Testung verantwortlich.

Application Assembler
Der Application Assembler stellt die vom Component Provider entwickelten Komponenten zu einer Anwendung zusammen. Er ist für das Zusammenspiel zwischen Serverkomponenten und Clients verantwortlich und erstellt die für die Installation der Anwendung im Server (Deployment) nötigen Beschreibungen (Deployment Descriptor).

Application Deployer und Administrator
Sind die Komponenten einer Anwendung entwickelt und zu einer Applikation verbunden, installiert sie der Application Deployer im Anwendungsserver. Er administriert Dienste wie die Transaktionskontrolle oder Sicherheitsmechanismen und ist für die notwendige Infrastruktur (Rechner, Netzwerk, Datenbankverbindung) zuständig.

Neben diesen Rollen, die den konkreten Entwicklungsprozess einer Anwendung abdecken, bezieht beispielweise [Geese00, S.118] auch das Umfeld der Entwicklung in das Rollenmodell mit ein und benennt zusätzlich die Bereitsteller der Server und der EJB Container. („EJB Server Provider“, „EJB Container Provider“)

Vor- und Nachteile des EJB-Rollenmodells

Die in der J2EE-Spezifikation vorgeschlagene Rollenverteilung in der EJB-Entwicklung bringt eine Reihe von Vor- und Nachteilen mit sich, wobei sich die Vorteile vor allem in der Effizienz der Entwicklung niederschlagen und erkennen lassen, dass die Idee der Entwicklung mit Komponenten sich auch in der Personalstruktur fortsetzt:

Tabelle 8: Vorteile des EJB-Rollenmodells

Kompetenzbündelung	Thematisch gleiche bzw. zusammenhängende Vorgänge werden von der selben Person oder dem selben Personenkreis ausgeführt. Während in klassischen Entwicklungsteams beispielsweise für jede Entwicklung Fragen der Integration und Inbetriebnahme geklärt werden müssen, wird diese Aufgabe der zentrale Application Deployer bzw. Administrator übernehmen
Klar definierte Ziele	Klassische Rollenstrukturen führen im Zusammenhang mit den in 2.1 erwähnten Vorgehensmodellen oft dazu, dass die Arbeit im Team oftmals an den Nächsten weitergereicht wird. [Stevens00, S.71] spricht in diesem Zusammenhang sehr treffend vom „wirf es über die Mauer“-Prozess, auch wenn iterative Vorgehensmodelle diesen Effekt etwas abschwächen. In der EJB-Entwicklung dagegen sind als feste Ziele die Entwicklung funktionierender Komponenten, deren Installation und deren Einsatz und Wartung auf Rollen abgebildet.
Komponentenorientierung	Die Entwicklung mit modernen Komponententechnologien ist in traditionell strukturierten Entwicklungsteams nur unzureichend abbildbar. Das EJB-Rollenmodell definiert von Anfang an die Kompetenzbereiche Entwicklung, Zusammenstellung und Einsatz von Komponenten.

Die Nachteile der Rollenverteilung im EJB-Entwicklungsprozess sind überwiegend durch die Neuartigkeit der Technologie begründet. Die folgende Tabelle zählt einige der Nachteile auf, die sich für den Autor aus der neuen Rollenverteilung im Entwicklungsprozess ergeben.

Tabelle 9: Nachteile des EJB-Rollenmodells

noch keine definierte Feinstruktur	Wie genau der Application Content Provider die Entwicklung der Komponente aufteilt, welche Personen wann im Entwicklungsprozess beteiligt sind, ist nicht im Rollenmodell der EJB-Entwicklung enthalten. Hier sind in Zukunft praxisgerechte Personalaufteilungen analog einer „klassischen“ Teamzusammensetzung und geeignete Vorgehensmodelle zu entwickeln.
Rollen als Schwachstellen im Entwicklungsprozess	Einzelne Personen oder kleine Gruppen können unter Umständen sehr kritische Schwachstellen im Entwicklungsprozess (so genannte SPOF[1]) darstellen. Ein unzureichend vorbereitetes Deployment ist in der Regel folgenreicher als in ein der klassischen Rollenverteilung auftretender schlechter Tester oder Programmierer.
Umstellung nötig	Bis sich die neuartigen Personalstrukturen und Rollen bewährt und bis in die Personalabteilungen durchgesprochen haben, wird noch einige Zeit vergehen. Die Unterschiede in den Ansätzen sind zu groß, um „nebenbei“ das Personal umzugruppieren oder neue Strukturen aufzubauen.

3.2.3 Performance von EJB

Ein Hauptargument, das bislang gern noch gegen den Einsatz von EJB genannt wird, ist die angeblich schlechte Performance der resultierenden Anwendungen. Dies mag auf den ersten Blick stimmen, bringen doch EJB eine Häufung von Interfaces, Methodenaufrufen, Diensten und Netzverkehr mit sich. Doch gibt es eine Reihe von Optimierungsstrategien, um die Performance von auf EJB basierenden Anwendungen entscheidend zu verbessern. Leider werden diese Strategien bisher sehr wenig beziehungsweise häufig überhaupt nicht in der Literatur erwähnt. Der Entwickler ist somit bisher größtenteils auf Newsgroups und Diskussionsforen im Internet angewiesen oder muss selbst auf Kosten der eigenen Entwicklungszeit versuchen, die EJB-Anwendung zu optimieren.
Im Folgenden sind die wichtigsten Optimierungsansätze dargestellt.

[1] Single Point Of Failure, sinngemäß einzelne Fehlerquelle, kritische Stelle

Design Patterns

Ein Aufruf einer Geschäftsmethode einer EJB läuft immer über das Netzwerk ist somit grundsätzlich teuer. Gleichzeitig müssen auch die an die Methode zu übergebenden Daten und die Rückgabewerte verpackt und übertragen werden (Marshalling). Es wäre also von Vorteil, die Anzahl der Methodenaufrufe möglichst gering zu halten.
Hierfür wurden so genannte Design Patterns entworfen, Entwurfsstrukturen für Anwendungen mit EJB. Das derzeit populärste Pattern ist die so genannte Session Facade.

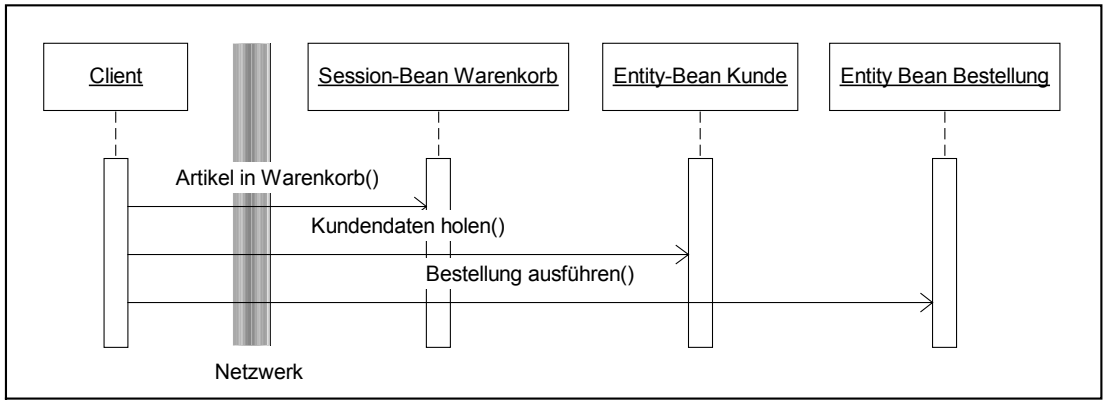

Abbildung 8: Anwendungsfall ohne Session Facade

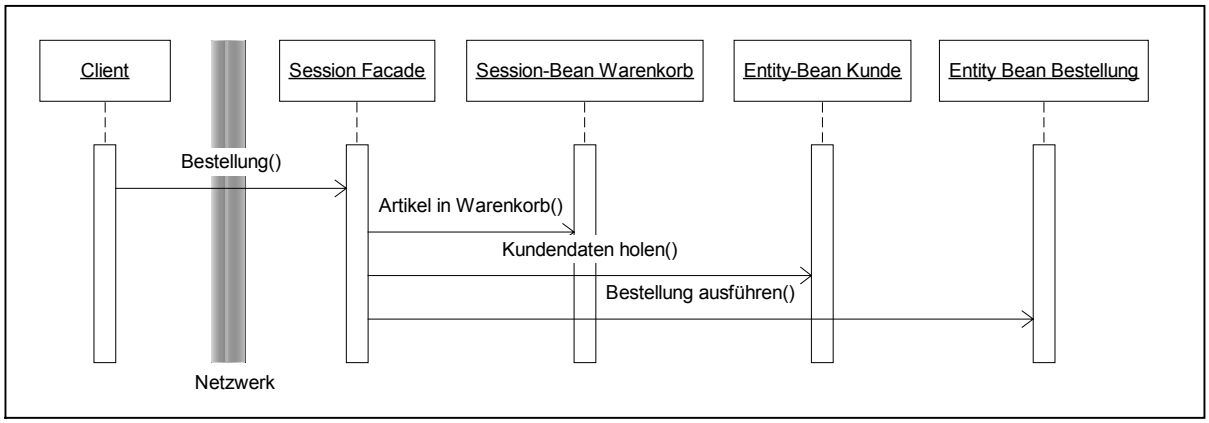

Abbildung 9: Einsatz einer Session Facade

Die erste Abbildung zeigt einen Anwendungsfall ohne Einsatz eines Design Pattern, in der folgenden Darstellung wurde der selbe Fall mit Hilfe einer Session Facade gekapselt. Wie deutlich zu sehen ist, enthält die Variante mit Session Facade nur noch einen Aufruf über das Netzwerk. Angenehmer Nebeneffekt: Gleichzeitig werden weitere Teile der Anwendungslogik vor dem Client verborgen. Einen guten Einstieg in Design Patterns für die J2EE-Umgebung findet sich in [J2EE-Blue04].

Entity Beans oder Session Beans

In einigen Fällen ist es sinnvoller, für den Datenzugriff statt einer Entity Bean eine Session Bean zu verwenden, woanders bringen jedoch Entity Beans mit ihren Eigenschaften Vorteile. [Kompf04, S.293f.] führt einige Entscheidungshilfen an:

Tabelle 10: Entity Beans oder Session Beans

Entwicklungszeit gegen Laufzeit
Mit den mittlerweile verfügbaren neuen Werkzeugen zur Modellierung von EJB kann erheblich Zeit bei der Anwendungsentwicklung gespart werden. Unterstützt die eingesetzte Entwicklungsumgebung beispielsweise automatisch Beziehungen zwischen Entity Beans und erkennt Fremdschlüsselbeziehungen, kann der Datenzugriff auf Kosten leichter späterer Performance-Einschränkungen wesentlich schneller entwickelt werden als durch von Hand Kodieren hunderter SQL-Statements, die zudem schlecht zu debuggen und zu warten sind.

SELECT oder UPDATE
Ein wichtiges Kriterium ist auch das Verhältnis von lesenden zu schreibenden Anfragen. Führt die Anwendung hauptsächlich viele parallele Updates aus, die vielleicht zudem immer auf die selben Tabellen zugreifen, sind Entity Beans effizienter und flexibler. Die Performance kann hier unter Umständen durch Kauf eines leistungsfähigeren Anwendungsservers gesteigert werden.

geschicktes SQL statt Entity Bean
Werden für einen Dialog viele Informationen aus vielen kleinen Tabellen benötigt, ist es in der Regel günstiger, mittels einer geschickt formulierten SELECT-Anweisung mit einem Join über alle relevanten Tabellen die Daten zu sammeln als mit einer Hand voll Entity Beans. Hierbei wird jedoch letztendlich auch die Erfahrung des Entwicklers den Ausschlag geben. Nicht jeder Programmierer ist auch SQL-Spezialist beziehungsweise Datenbankspezialisten mit großer SQL-Erfahrung sind sicher nicht immer erfahrene Entwickler auf dem Gebiet der objekt- und komponentenorientierten Programmierung.

Remote und Local Interfaces

Oftmals kommt es vor, dass sich beim Aufrufen von EJB-Methoden der Aufrufende und die EJB im selben Anwendungsserver und folglich in der selben virtuellen Maschine befinden. In diesem Fall macht es logischerweise keinen Sinn, Methodenaufrufe über das Netzwerk mittels RMI/JRMP/IIOP abzuwickeln. Für diesen Fall wurden in der EJB 2.0 Spezifikation die so genannten Local Interfaces eingeführt.
Beide Arten von Interfaces, Remote und Local, können nebeneinander existieren, sie müssen auch nicht zwingend die gleichen Methoden enthalten. Der Entwickler erhält also hier eine Designentscheidung, welche Dienste er „lokal“ mit gesteigerter Performance und unter Umgehung des Netzwerkes anbietet und welche er nach außen hin veröffentlicht.

Datenbankzugriffe minimieren

Auch in der Entwicklung von EJB gilt der Grundsatz der verteilten Anwendungen: Datenbankverbindungen wirken sich in der Regel am gravierendsten auf die Performance der Gesamtanwendung aus und sind in ihrer Anzahl zu minimieren.
Dies kann durch geschickte SQL-Statements, einen ausreichend großen Verbindungspool im Anwendungsserver, den Einsatz von PreparedStatements oder eine grob granulare Struktur der Entity Beans erreicht werden.

Java optimieren

Des Weiteren gelten für die Entwicklung von Anwendungen mit EJB selbstverständlich alle Regeln zur Geschwindigkeitssteigerung, die für Java allgemein gelten. Eine nach Ansicht des Autors sehr hilfreiche Übersicht, die auch weitere Tipps für die J2EE-Entwicklung enthält, findet sich in [Shirazi04].

4 Entwicklung des Auskunftssystems

4.1 Auswahl der Techniken

Bei der Entwicklung einer Anwendung sprechen heute eine Vielzahl von Aspekten dafür, eine Webanwendung zu erstellen:

Tabelle 11: Vorteile einer Webanwendung

auf Serverseite	auf Clientseite
zentraler Anwendungsserver	keine Installation nötig
daher leicht zentral wartbar	überall im Netz verfügbar
Verteilung auf Rechner/Ressourcen möglich	Browser als Clients frei verfügbar
standardisierte Server	geringe Kosten

Es lassen sich jedoch auch sehr schnell eine Reihe von Nachteilen finden, die gegen die Entwicklung einer Webanwendung sprechen:

Tabelle 12: Nachteile von Webanwendungen

- Abhängigkeit von Netzinfrastruktur und deren Eigenschaften
- wenig Einfluss auf Funktionalität und Stabilität des Browser-Clients
- komplexe Sicherheitsaspekte
- „Vertrauen“ in Anwendungsserver teilweise gering, jedoch nötig
- noch nicht im vollen Durchbruch und am Markt etabliert

Trotz der hier aufgeführten Nachteile geht der Trend jedoch eindeutig in Richtung Webanwendungen. Allgegenwärtige Floskeln wie das „Informationszeitalter“ verdeutlichen den Wunsch, überall auf Daten zugreifen zu können, und das Internet hat mittlerweile in beinahe allen Bereichen des täglichen Lebens Einzug gehalten.

Neue Technologien für neue Anwendungen

Für die Entwicklung einer Webanwendung steht mittlerweile eine unüberschaubare Vielzahl von Technologien zur Verfügung. Das Spektrum reicht dabei von extern generierten statischen HTML-Seiten über frei verfügbare Open Source[2] Technologien wie PHP bis zu großen kommerziellen, serverseitigen Ansätzen wie Microsofts .NET-Framework oder Suns J2EE-Umgebung.
Trotzdem bleibt nach wie vor die größte Schwierigkeit, als Softwareentwickler die Wünsche des Kunden nicht nur zu verstehen, sondern auch mit ihm zu kommunizieren und zu verhandeln und das Ergebnis schließlich auf eine Software abzubilden. Hier bewegen sich Entwickler und Kunde oft in verschiedenen

[2] frei verfügbare Software, siehe auch http://www.opensource.org/

Gedankenwelten. (vgl. auch [Kompf04, S.29]) Ein Beispiel verdeutlicht dieses Problem: Dem Kunden, der ein neues Auskunftssystem über Schienenfahrzeuge kaufen möchte, interessiert beispielsweise:

- Was kann das neue Auskunftssystem alles?
- Wie zeigt es technische Daten zu Fahrzeugen an?
- Kann ich zu einem Fahrzeug auch die Herstellerangaben abrufen?
- Kann es Fahrzeuglisten zu einem Verkehrsunternehmen ausgeben?

Der Entwickler würde dem Kunden jedoch folgende Fragen stellen wollen:

- Wie werden die Daten persistent gehalten?
- Wie erfolgt die Lastverteilung auf den Servern?
- Wie werden die Beziehungen zwischen den Entitäten abgebildet?

Diese Kluft zwischen der Welt des Kunden, der mit Schienenfahrzeugen und technischen Details umgeht und der des Entwicklers, der bereits die programmtechnische Umsetzung im Hinterkopf hat, wurde bisher hauptsächlich in der Phase der Anforderungsdefinition überbrückt, beispielsweise durch sehr umfangreiche Pflichtenhefte, Glossare oder ausgeklügelte Testszenarien.
Doch es gibt eine Möglichkeit, auch in späteren Phasen, im Anwendungsentwurf und selbst in der Implementierung, nicht die Gedankenwelt des Kunden zu verlassen.

Komponenten und EJB – Konzentration auf das Wesentliche

Bereits eine objektorientierte Herangehensweise erleichtert die Entwicklung in der Hinsicht, dass nunmehr Klassen wie „Fahrzeug“ und „Verkehrsunternehmen“ modelliert werden können. Allerdings löst allein der Einsatz von Objektorientierung nicht alle Probleme. Ein Objekt besitzt zwar einen „definierten Zustand“ und ein „definiertes Verhalten auf seine Umgebung“ ([Balzert00, S.35]), jedoch ist der Objektbegriff und die Definition der Umgebung eines Objektes beispielsweise im Java-Klassenmodell zu weit gefasst.
Die logische Folgerung ist das Zusammenfassen einer Menge von Objekten zu einem funktionalen Ganzen, einer Komponente (vgl. 2.2). Beispiele sind Java Beans, CORBA oder auch ActiveX.
Aus der Idee, der Komponente zudem eine Reihe von Diensten wie Persistenz oder die Steuerung von Transaktionen zuzusichern, entstanden die Enterprise Java Beans. (vgl. auch 3.2)
Java als objektorientierte Programmiersprache, JavaBeans als Komponenten und schließlich EJB ordnet [Kompf04, S.31] sehr übersichtlich hinsichtlich der aus der Idee entstehenden Anforderungen und Zusicherungen des jeweiligen Modells:

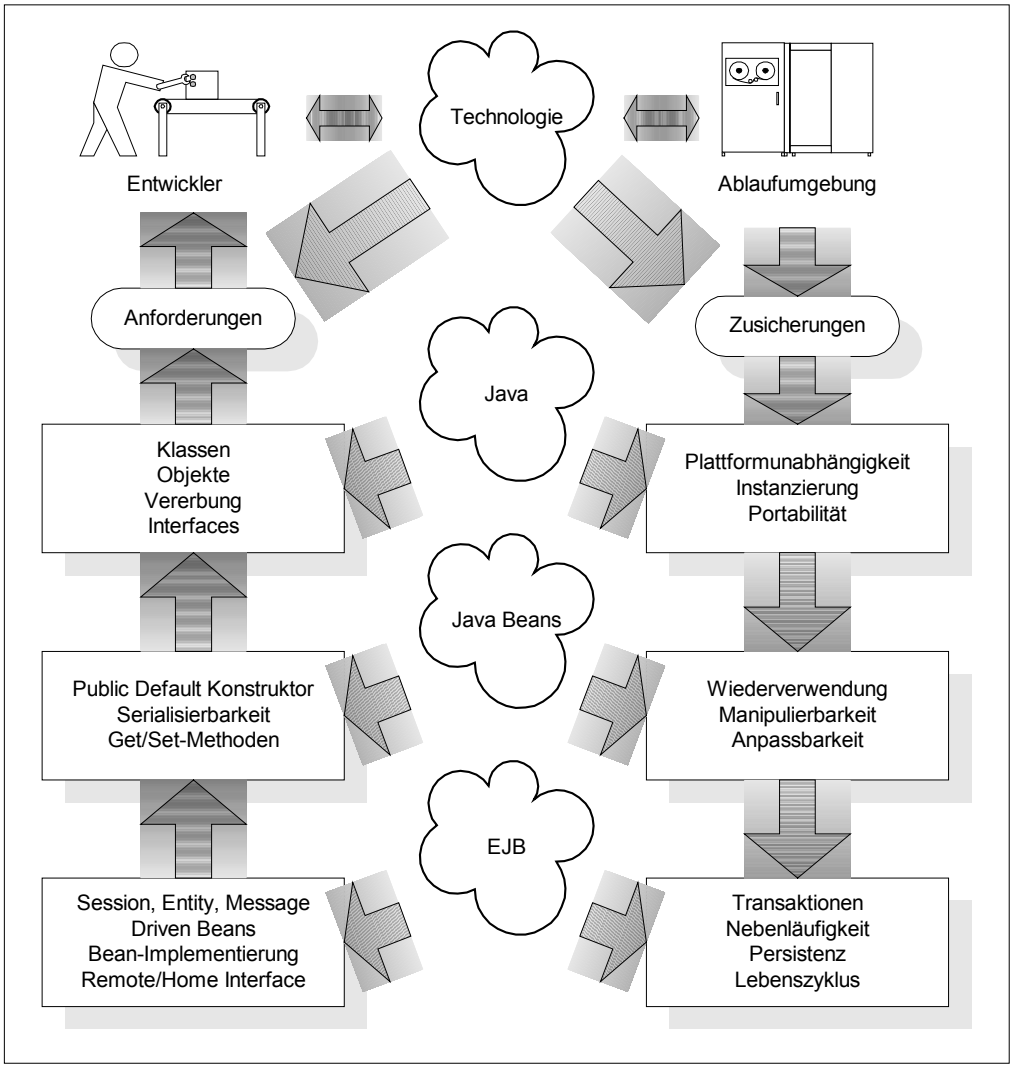

Abbildung 10: Anforderungen und Zusicherungen von Technologien

Mit dem Einsatz von Technologien aus der J2EE-Umgebung, insbesondere EJB ist es also möglich, dass der Entwickler gezielt auf die Wünsche des Kunden eingehen und sich gleichzeitig auf die vom Framework bereitgestellten Dienste verlassen kann. Eigenschaften wie Portabilität, Wiederverwendbarkeit oder Verwaltung des Lebenszyklus eines Software-Bestandteiles müssen weder mit dem Kunden verhandelt noch in späteren Phasen selbst umgesetzt werden, sondern werden von der jeweiligen technologischen Abstraktionsebene zugesichert.

4.2 Anforderungsdefinition

Nach der Auswahl der bei der Entwicklung einzusetzenden Technologien erfolgt die Definition der Anforderungen an die zu erstellende Software.

4.2.1 Zielstellung

Ziel der Entwicklung ist ein Auskunftssystems über Schienenfahrzeuge für die ETC Transport Consultants GmbH. Die zu erstellende Webanwendung soll es ermöglichen, die in der ETC vorhandene Datenbank über Schienenfahrzeuge und Unternehmen mit Bezug zum Schienenverkehr für Recherchen zu nutzen und technische und wirtschaftliche Details und Zusammenhänge zwischen den Fahrzeugen und Unternehmen abzufragen.
Durch entsprechende Vorgaben und Voraussetzungen in der ETC erfolgt die Entwicklung des Auskunftssystems als Webanwendung und unter Verwendung von modernen J2EE-Technologien.

4.2.2 Pflichtenheft

Das im Rahmen der Entwicklung entstandene und in Zusammenarbeit mit fachlich zuständigen Mitarbeitern der ETC überarbeitete Pflichtenheft des zu erstellenden Auskunftssystems ist im Anhang abgelegt.
Da die Software nicht nur eine reine Technologiestudie oder Musterentwicklung darstellen sondern im Praxisbetrieb in der ETC eingesetzt werden soll, war die Erstellung eines Pflichtenheftes nötig. So konnten die fachlichen und technischen Anforderungen und die spätere Umgebung des Produktes präzise definiert werden.

4.3 Design

Nach Auswahl der Techniken erfolgt der Entwurf des Systems. Hierbei werden vor allem die Anforderungen aus 4.2 umgesetzt und die in der Auseinandersetzung mit den Entwicklungstechnologien in den Kapiteln 2 und 3 gewonnenen Erkenntnisse eingebracht.

4.3.1 Datenmodell

Das Datenmodell, auf dem die eingesetzte Oracle-Datenbank und die Anwendung basieren, lag bei Projektbeginn als Entwurf vor. In mehreren Treffen mit fachlich zuständigen Mitarbeitern der ETC wurde das Modell mehrmals überarbeitet. Dabei wurden sowohl fachliche Details wie ergänzende Attribute ergänzt als auch datenbanktechnisch relevante Aspekte eingefügt. Unter anderem wurde dem Entwurf die Möglichkeit einer Versionierung der Datenbestände hinzugefügt.
Im Folgenden sind die wichtigsten Entitäten des Datenmodells und ihre Beziehungen zueinander beschrieben und in einem vereinfachten ERM-Diagramm dargestellt.

Tabelle 13: Wichtige Entitäten des Datenmodells

Fahrzeug
Diese Entität enthält allgemeine und technische Angaben zu einem Fahrzeug eines bestimmten Typs[3]. Dies sind beispielsweise Maße und Gewicht des Fahrzeugs oder Angaben zur Maximalgeschwindigkeit. Die Tupel dieser Relation enthalten noch keinerlei Informationen über Unternehmen, Einsatz und Bestände. Je nach Art des Fahrzeugs (Personen-, Güterwagen, Triebfahrzeug) existiert weiterhin ein Tupel in der jeweiligen schwachen Entität und ergänzt die Angaben um von der Fahrzeugart abhängige Details. Dies kann bei Triebfahrzeugen etwa die Zugkraft, bei Personenwagen die Anzahl der Sitzplätze sein.

Unternehmen
Diese Relation enthält allgemeine Angaben zu einem Unternehmen. Neben Grunddaten wie Name und Kontaktinformationen werden hier wirtschaftliche Details wie die Unternehmensgröße oder der Jahresumsatz erfasst. Je nach Art des Unternehmens existiert ein weiteres Tupel in einer oder mehrerer der schwachen Entitäten. Beispielsweise werden zu einem Verkehrsunternehmen die gefahrenen Personenkilometer oder zu einem Eigentümer die Möglichkeit des Ersatzteilservice erfasst.

[3] gemeint ist kein konkretes Fahrzeug mit Baujahr und Seriennummer, sondern ein allgemeiner Typ

Beziehungsrelationen

In welcher Beziehung Fahrzeuge und Unternehmen stehen, wird in mehreren Relationen erfasst. In einem Fahrzeugpool sind alle Fahrzeuge enthalten, die ein Verkehrsunternehmen im Bestand hat. Ein Hersteller produziert eine bestimmte Menge von Fahrzeugtypen. Eigentümer besitzen einen Fuhrpark, der als Fahrzeugpool von einem Verkehrsunternehmen eingesetzt wird. Ein Verkehrsunternehmen lässt seine Fahrzeuge bei einem bestimmten Instandhaltungsbetrieb warten.

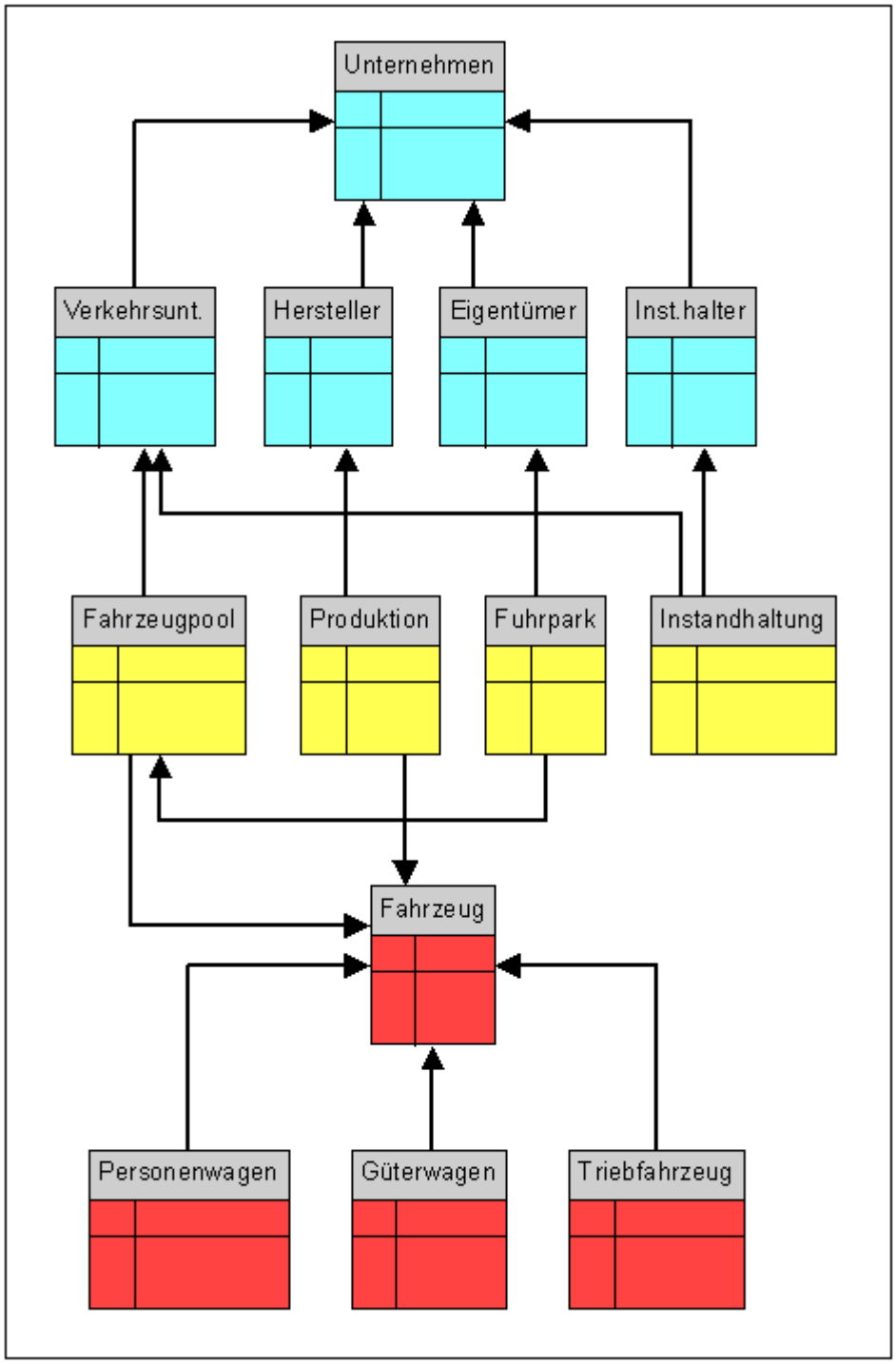

Abbildung 11: Datenmodell (Auszug)

Daneben existieren eine Reihe von Relationen, die für einzelne Attribute die Domäne darstellen. Dies sind unter anderem Länder, Strom- und Antriebssysteme und Fahrzeugarten.

4.3.2 Datenzugriff über EJB

Der Zugriff auf die Daten der Fahrzeugdatenbank erfolgt über Enterprise Java Beans (EJB), deren Struktur und Eigenschaften im Folgenden dargestellt sind.

EJB-Struktur

Für jeden inhaltlichen Teilbereich existiert eine EJB, über deren Geschäftsmethoden Listen und Details abgefragt werden können.

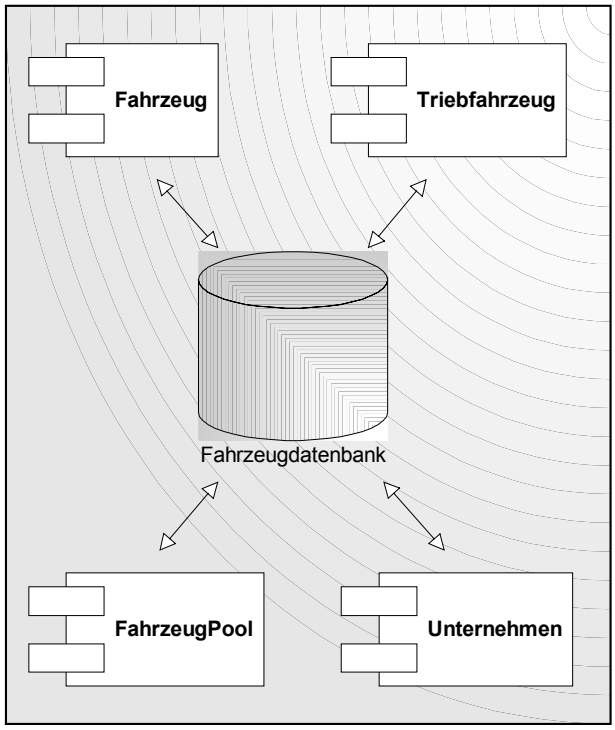

Abbildung 12: EJB-Zugriff auf die Fahrzeugdatenbank

Die Abbildung zeigt einige Beispiele für EJB in der Anwendung. Beispielsweise stellt die EJB Fahrzeug Listen aller im Datenbestand vorhandenen Fahrzeuge bereit oder ermöglicht die Abfrage von Details zu einem Fahrzeug. Die EJB Triebfahrzeug ermöglicht dagegen das Abrufen von Angaben zu Antrieben, Stromsystemen u.a. eines Triebfahrzeugs.

Bei der Zusammenstellung der Daten greifen die EJB jeweils auf eine Vielzahl von Relationen der Fahrzeugdatenbank zu, vor allem, um zu Fremdschlüsselbeziehungen Werte abzufragen und Werte aus Domänen zu beziehen. In der folgenden Abbildung ist dargestellt, wie die EJB Triebfahrzeug aus mehreren Tabellen Informationen zusammenträgt, um ein Triebfahrzeug hinsichtlich seiner technischen Details zu charakterisieren.

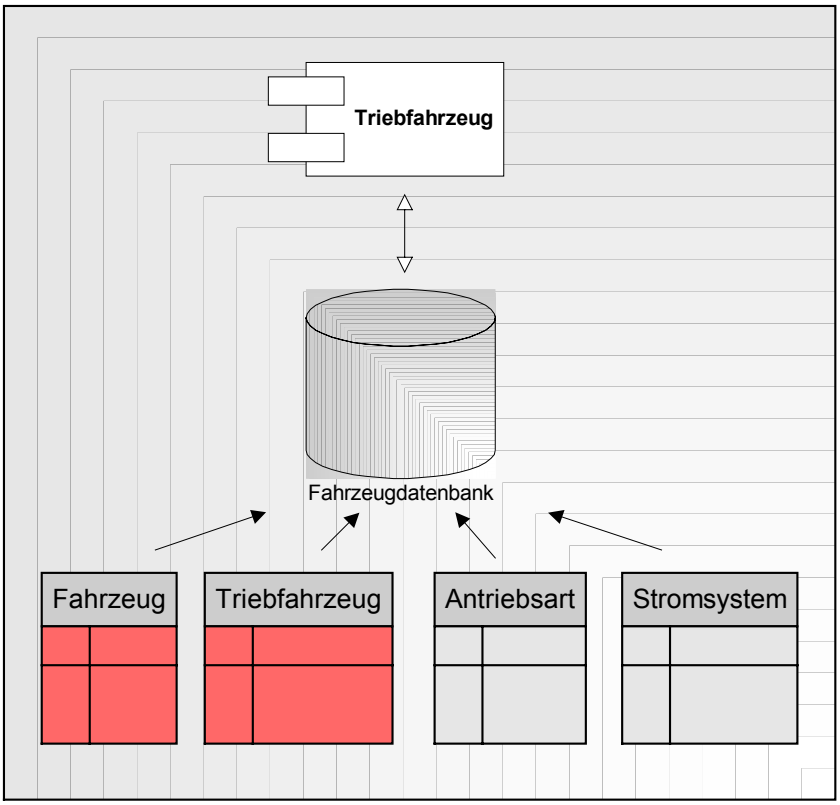

Abbildung 13: Eine EJB trägt Daten aus mehreren Tabellen zusammen

Die einzelnen EJB werden als zustandslose Session Beans modelliert. Wie in 3.2.4 erläutert, bietet sich der Einsatz von Session Beans anstelle von Entity Beans vor allem dann an, wenn für die resultierenden Dialoge Informationen aus einer Vielzahl von kleinen Tabellen zusammengetragen werden muss. Eine Session Bean mit einem geschickt formulierten SQL-Join über alle benötigten Tabellen ist in diesem Fall wesentlich performanter als die eigentlich erwartete Lösung mit mehreren Entity Beans
Der Datenzugriff wird im folgenden Abschnitt anhand einer konkreten EJB verdeutlicht.

Beispiel: Fahrzeug-EJB

Die EJB für Fahrzeuge stellt sowohl Listen aller in der Datenbasis enthaltenen Fahrzeuge als auch detaillierte Angaben zu konkreten Fahrzeugen zur Verfügung. Sie enthält weiterhin Methoden zur Ermittlung der Gesamtanzahl der Fahrzeuge und zur Abfrage ausgewählter Attribute. Beispielsweise lassen sich so für die Oberfläche aussagekräftige Fahrzeugtitel generieren oder für Detailansichten genaue Typenbezeichnungen abfragen.

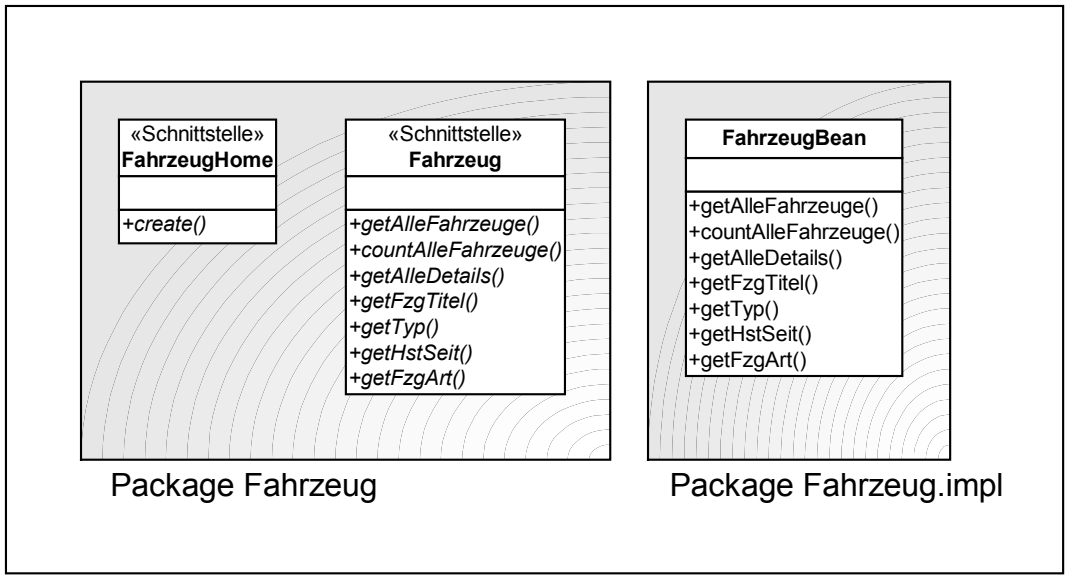

Abbildung 14: EJB Fahrzeug

In dieser Abbildung der Package-Struktur der EJB Fahrzeug, in der aus Gründen der Übersichtlichkeit Parameter und Rückgabewerte weggelassen wurden, sind alle Bestandteile und Methoden dargestellt. Die Home- und Remote-Schnittstelle befinden sich im Package „Fahrzeug“, die eigentliche Implementierung der Funktionalität in „Fahrzeug.impl“[4]. Um mittels der EJB beispielsweise die Liste aller Fahrzeuge abzurufen, sind folgende Schritte nötig:

Tabelle 14: Datenzugriff über Fahrzeug-EJB

1. Die EJB „Fahrzeug“ wird über den Java Namens- und Verzeichnisdienst JNDI ermittelt.
2. Man erhält die Schnittstelle „FahrzeugHome“.
3. Durch Aufruf der Methode „create()“ erzeugt man eine Instanz der Bean und erhält die „Fahrzeug“-Schnittstelle.
4. Der Aufruf von „getAlleFahrzeuge()“ stößt die in der Bean-Implementierung „FahrzeugBean“ enthaltene Funktionalität an.
5. In der „FahrzeugBean“ wird eine Datenbankverbindung zur Fahrzeugdatenbank aufgebaut, eine Abfrage an die Datenbank gesendet und die resultierende Fahrzeugliste in einer Datenstruktur abgelegt.
6. Die Datenstruktur mit der Fahrzeugliste wird zurückgegeben.

[4] siehe auch die Namenskonventionen für EJB in [Sun04]

4.3.3 Struktur der Applikation

3-Schichten-Architektur

Das Auskunftssystem stellt eine verteilte Anwendung dar. Ihm liegt eine 3-Schichten-Architektur zugrunde, welche die Trennung von Darstellung, Anwendungslogik und Datenhaltung beschreibt.

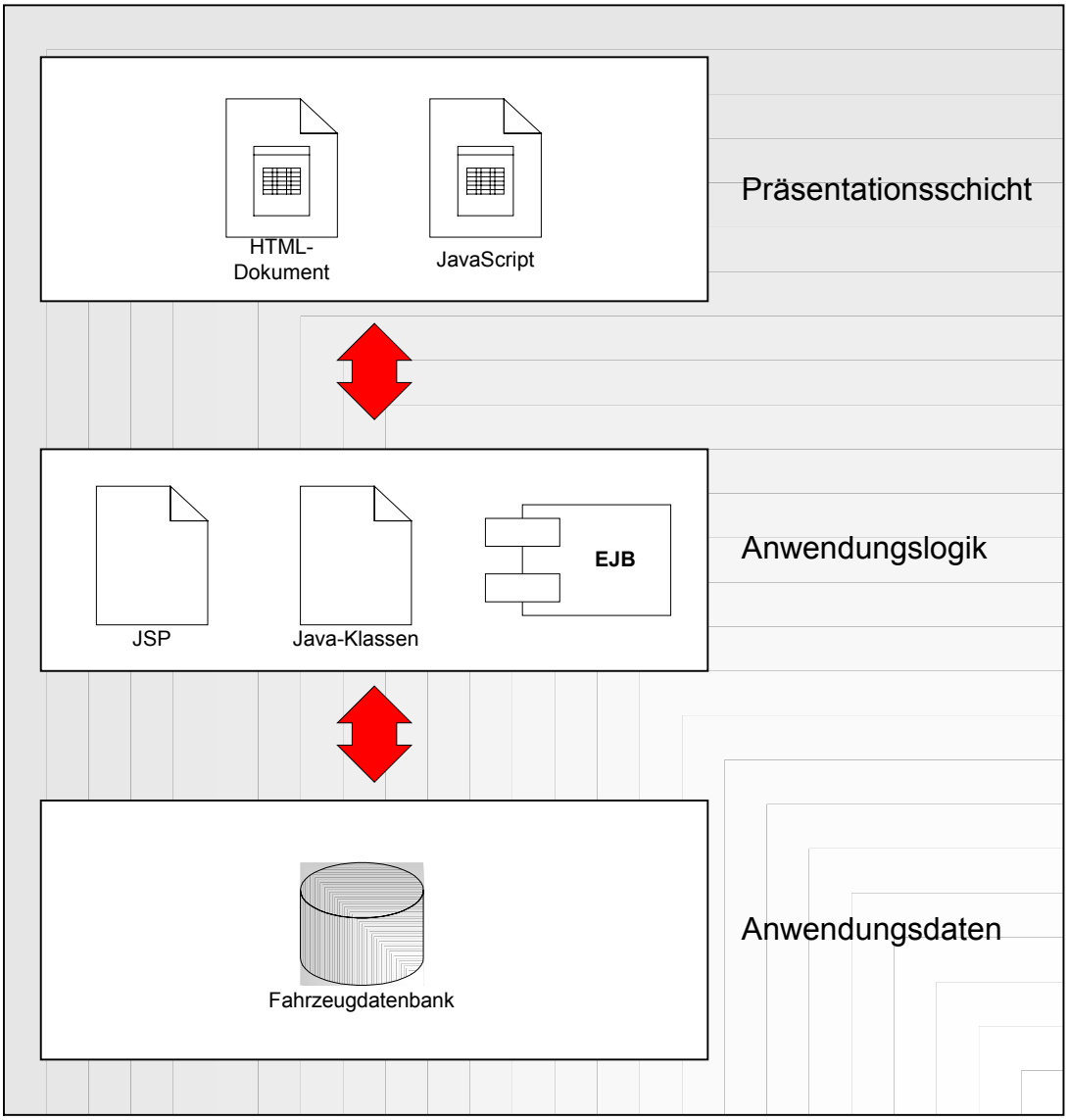

Abbildung 15: 3-Schichten-Architektur der Anwendung

Weiterhin ist die Anwendung physisch verteilt. Die Daten werden auf einem Datenbankserver persistent gehalten. In einem Anwendungsserver laufen die Komponenten und Klassen der Anwendungslogik. Der Zugriff auf die Anwendung erfolgt von Clients, an die architekturbedingt nur minimale Anforderungen bestehen (thin clients[5]).

Die genauen Anforderungen an die einzusetzende Hardware sind in Anhang, Kapitel A3.2 nachzulesen.

[5] zur Thematik von “thin clients” s.a. ausführlich in [Brown01, S.80ff.]

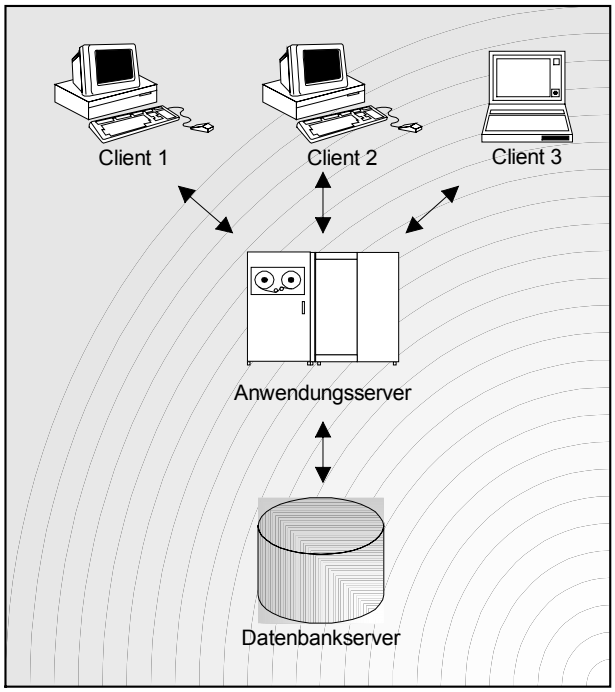

Abbildung 16: Physische Verteilung der Anwendung

Package-Struktur

Packages, auch Pakete genannt, fassen verwandte Klassen zusammen und stellen den enthaltenen Klassen einen Namensraum bereit. Die Namen der Packages sollen zur Unterscheidung der Klassen global eindeutig sein. Nach [Flanagan01, S.80f], welcher sich auf [Sun04] bezieht, sollten die Package-Namen in einem Projekt immer mit dem eigenen Internet-Domänennamen beginnen.
Die vom Ersteller dieser Arbeit wiederverwendeten bzw. überarbeiteten Komponenten in der ETC enthielten jedoch keine solche Sun-konforme Package-Struktur.
Bei der Wahl neuer Package-Namen orientierte sich der Autor an diesen vorhandenen Namen. Die wichtigsten Pakete und ihr funktionaler Kontext sind in der folgenden Abbildung dargestellt.

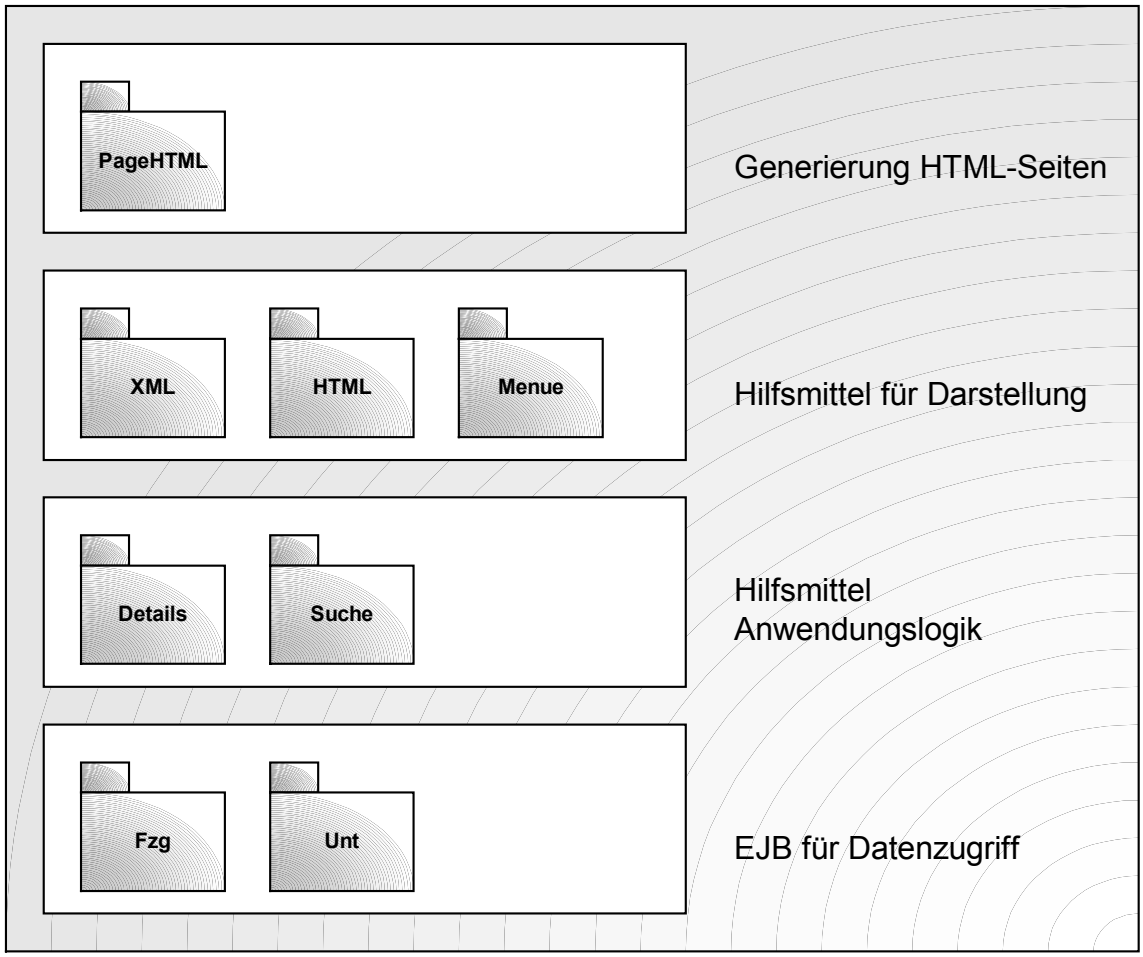

Abbildung 17: Packagestruktur der Anwendung

Die enthaltenen Klassen und die in den Packages umgesetzte Funktionalität ist folgender Übersicht zu entnehmen:

Tabelle 15: Packages und deren Inhalt

Package	enthaltene Klassen/Funktionalität
PageHTML	Generierung der HTML-Seiten für die Darstellung
XML	XML-Dokumente zur Konfiguration des Hauptmenüs, der List Views und der Suchfunktionalität, Klassen zur Interpretation der XML-Dokumente
HTML	Generierung des HTML-Codes verschiedener Steuerelemente wie z.B. List Views, Hauptmenü, Nachrichtenfenster,...
Menue	Generierung des HTML-Codes des Anwendungsmenüs.
Details	Klassen, die einzelne Datensätze darstellen (ein Fahrzeug, Unternehmen, technisches Detail,...)
Suche	Klassen zur Umsetzung der Suchfunktionalität
Fzg, Unt	EJB für den Zugriff auf die Datenbank

Wiederverwendung von Oberflächen-Steuerelementen

Bei der Gestaltung der Oberfläche konnten in der ETC vorhandene Steuerelemente wiederverwendet werden. Teilweise war jedoch eine Überarbeitung, Anpassung und Verallgemeinerung dieser Elemente nötig.

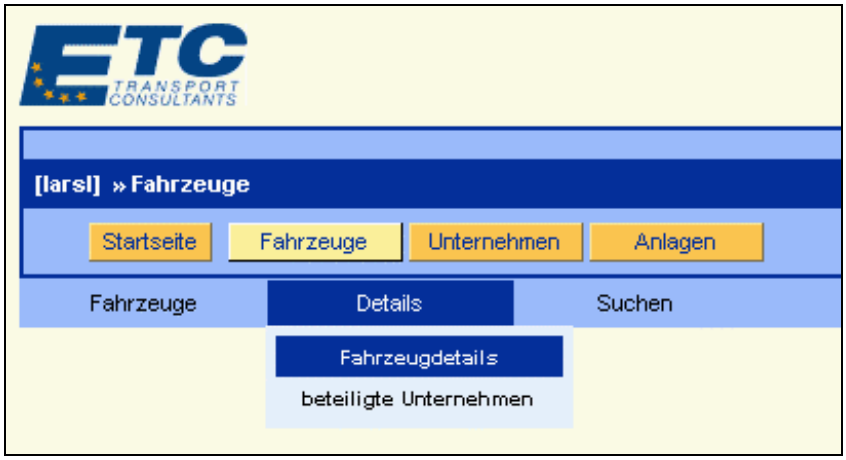

Abbildung 18: Menü-Steuerelemente

Die Verwendung von Menü-Steuerelementen hat für die Entwicklung und den Einsatz in der Webanwendung eine Reihe von Vorteilen. Beispielsweise führt ein allgemeiner, anpassbarer Entwurf eines frei anpassbaren Menüs dazu, dass sich der Entwickler bei der späteren Umsetzung der Dialoge nicht mehr um Navigationsstrukturen und Verweise kümmern muss. Aus Sicht des Anwenders hat der Einsatz solcher Menüs zudem den Vorteil, dass er sich in der Bedienung der Webanwendung schneller zurecht findet, da er Strukturen wie Drop-down-Menüs aus herkömmlichen Anwendungen gewohnt ist.

Tabelle 16: Wiederverwendete Menü-Steuerelemente der ETC

Steuerelement	Funktion, Einsatz
Hauptmenü (MM)	Umschalten zwischen den Einstiegspunkten der Recherche (Fahrzeuge, Unternehmen, Testanlagen), Rücksetzen aller getätigten Eingaben
Anwendungsmenü (SubMenue)	Verweise auf zugehörige Daten, Aufruf von Such- und Filterfunktionen, Rücksprung auf vorige Ansicht

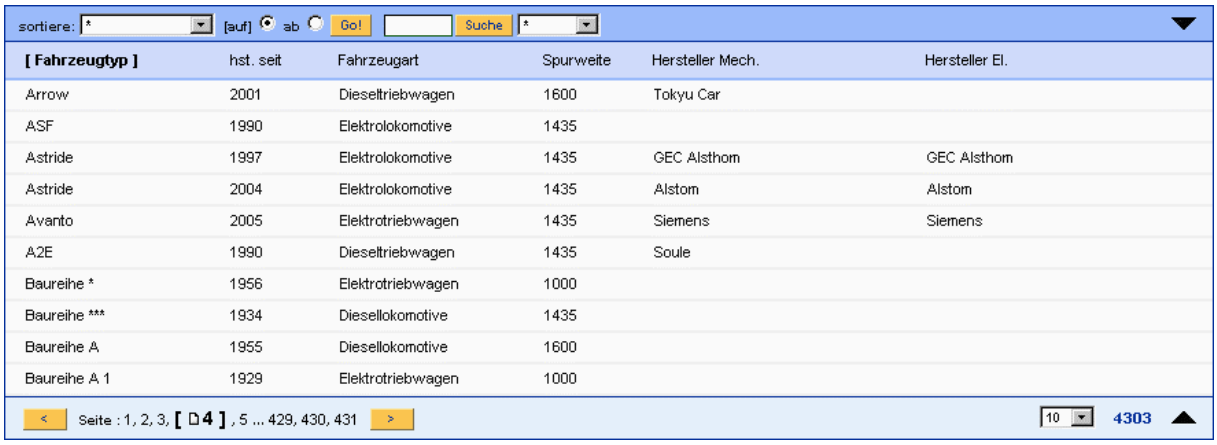

Abbildung 19: Listen-Steuerelement

Weiterhin wurde das in der ETC vorhandene Listen-Steuerelement in der Entwicklung des Auskunftssystems wiederverwendet. Es bietet eine flexible Darstellung von Listen und ist über XML an die Anforderungen des jeweiligen Einsatzes anpassbar. Die Liste enthält bei Bedarf die Möglichkeit, die angezeigten Datensätze zu sortieren sowie schnell in ihnen nach Schlagworten zu suchen.

Dialogaufbau

Die in der Anwendung auftretenden Dialoge unterliegen einer festen Struktur. Dies hilft nicht nur dem Anwender beim Navigieren und Auffinden der gesuchten Informationen sondern ermöglicht auch eine schnelle Realisierung von neuen Dialogen.

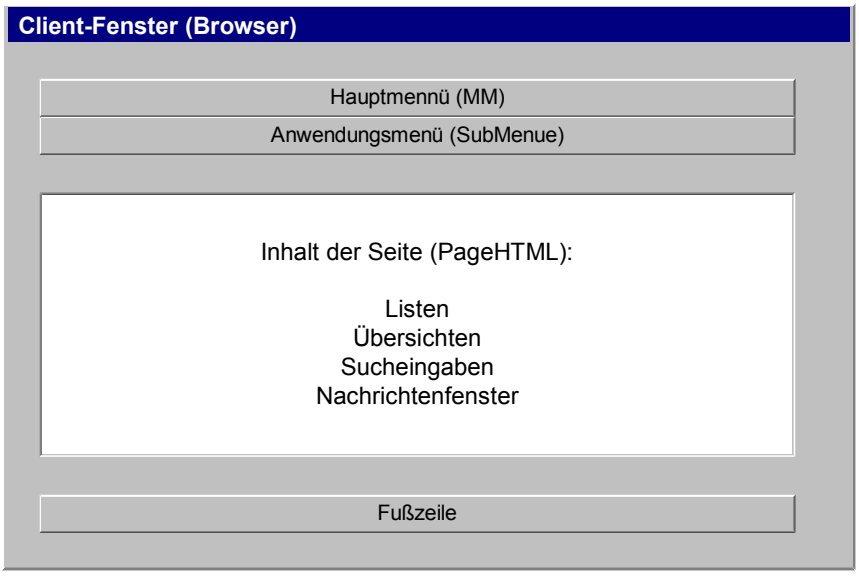

Abbildung 20: Standardisierter Dialogaufbau

Wie bereits beschrieben, sind die Menüstrukturen mittels XML-Beschreibungen vorkonfiguriert und müssen in jeden neuen Dialog lediglich eingebunden werden. Die Entwicklung eines neuen Dialoges besteht folglich darin, den Inhaltsbereich mit Informationen zu füllen.

Erzeugen des Inhalts durch PageHTML-Klassen

Für jeden Dialog existiert im Package „PageHTML“ eine Klasse. Sie enthält Methoden, um die Anfrage der aufrufenden Seite auszuwerten, die für den Dialog relevanten Daten abzurufen und für die Darstellung aufzubereiten.

Tabelle 17: Aufgaben einer Klasse des Package PageHTML

- Auswerten des Requests der aufrufenden Seite
- Erzeugen der grundlegenden HTML-Struktur
- Verbinden mit EJB
- Abrufen der angefragten Daten
- Erzeugen des HTML-Codes der Liste(n)
- Rückgabe des gesamten HTML-Codes des Inhaltsbereiches

Allen Klassen in PageHTML liegt ein gleicher Aufbau zu Grunde. Als Beispiel sei die Klasse FahrzeugHTML angeführt:

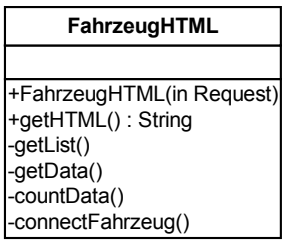

Abbildung 21: Klassendiagramm FahrzeugHTML

Nach außen hin sichtbar sind der Konstruktor und die Methode zum Abrufen des gesamten HTML-Codes. Alle Methoden zum Verbinden mit den EJB, Abrufen der Daten aus der Datenbank oder Zusammenstellen der Listen sind privat.

Datenhaltung durch „Details“-Klassen

Einzelne aus der Datenbank abgerufenen Datensätze werden in Instanzen der Klassen im Package „Details“ abgelegt. Für jedes aus der Datenbank abgerufene Attribut besitzt die jeweilige Klasse ein Feld. Für die spätere Darstellung in Listen existiert weiterhin eine Methode, um den enthaltenen Datensatz in einem für das Listen-Steuerelement verständlichen textuellen Format, beispielsweise durch Semikolons getrennt, darzustellen.

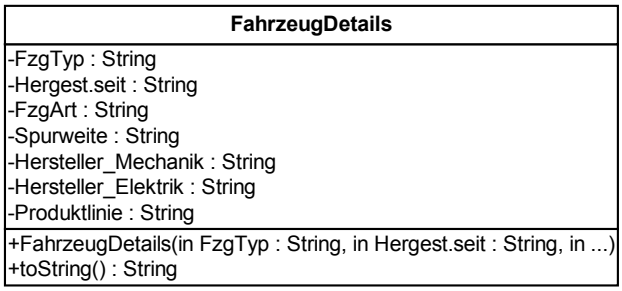

Abbildung 22: Klassendiagramm FahrzeugDetails

Eine Erweiterung dieser Klassen wäre beispielsweise möglich durch Methoden

- zur Prüfung des Inhalts auf Korrektheit
- zur Umwandlung in alternative Darstellungsformen (Grafik, Datenstrom,...)
- zur Rückumwandlung in eine SQL-Anweisung

Interaktion der Klassen

Die nachfolgenden Abbildungen verdeutlichen das Zusammenspiel der Klassen und deren erzeugte Instanzen und Dokumente von der Anfrage des Clients bis zur Ausgabe des vollständig erzeugten HTML-Dokuments.

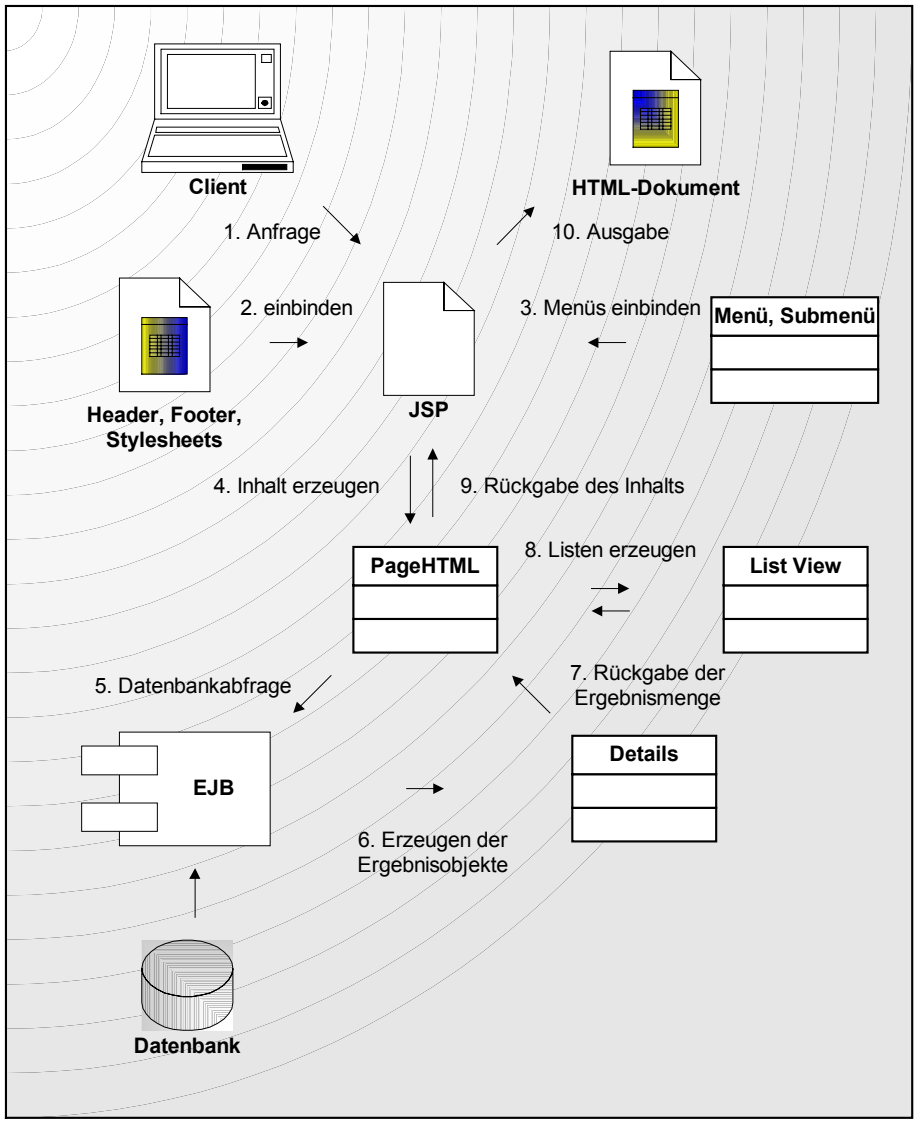

Abbildung 23: Seitenerzeugung

Die für jeden Dialog vorhandene JSP dient zur Zusammenstellung der HTML-Seite. In ihr werden vorhandene Header und Footer sowie die Stylesheets eingebunden. Je nach aufgerufener Seite werden die entsprechenden Menüs erzeugt und dargestellt. Schließlich wird über eine Instanz der entsprechenden Klasse aus dem Package PageHTML der Inhalt der Seite, d.h. die Informationen in Listenform, Bilder u.a. erzeugt und an das HTML-Dokument angefügt.

Die Generierung des Inhalts erfolgt durch Aufruf einer Methode der entsprechenden Klasse aus PageHTML. In dieser Klasse wird zunächst eine Verbindung zur entsprechenden EJB aufgebaut und die Datenbankabfrage angestoßen.

Die EJB baut hiernach eine Verbindung zur Datenbank auf und führt die Abfrage aus. Zum zurückgegebenen Abfrageergebnis erzeugt sie anschließend Objekte aus dem Package Details, die die einzelnen Ergebnisse darstellen. Die Menge dieser Ergebnisobjekte wird an die HTML erzeugende Klasse aus PageHTML zurückgegeben.

Diese Klasse erzeugt nun wiederum mittels dieser Ergebnismenge und des List View Steuerelements den HTML-Code des Ergebnisses. Überschriften, Platzhalter, Bilder u.a. ergänzen den Inhalt in Listenform.

Schließlich wird der gesamte Inhalt in HTML-Code an die JSP zurückgegeben. Das HTML-Dokument ist nun komplett und kann an den Anfragenden übermittelt und im Browser dargestellt werden.
Ein UML-Sequenzdiagram verdeutlicht die Erzeugung einer kompletten Seite wie folgt. Menüs, Kopf- und Fußzeilen sowie die Datenbank wurden aus Gründen der Übersichtlichkeit nicht dargestellt.

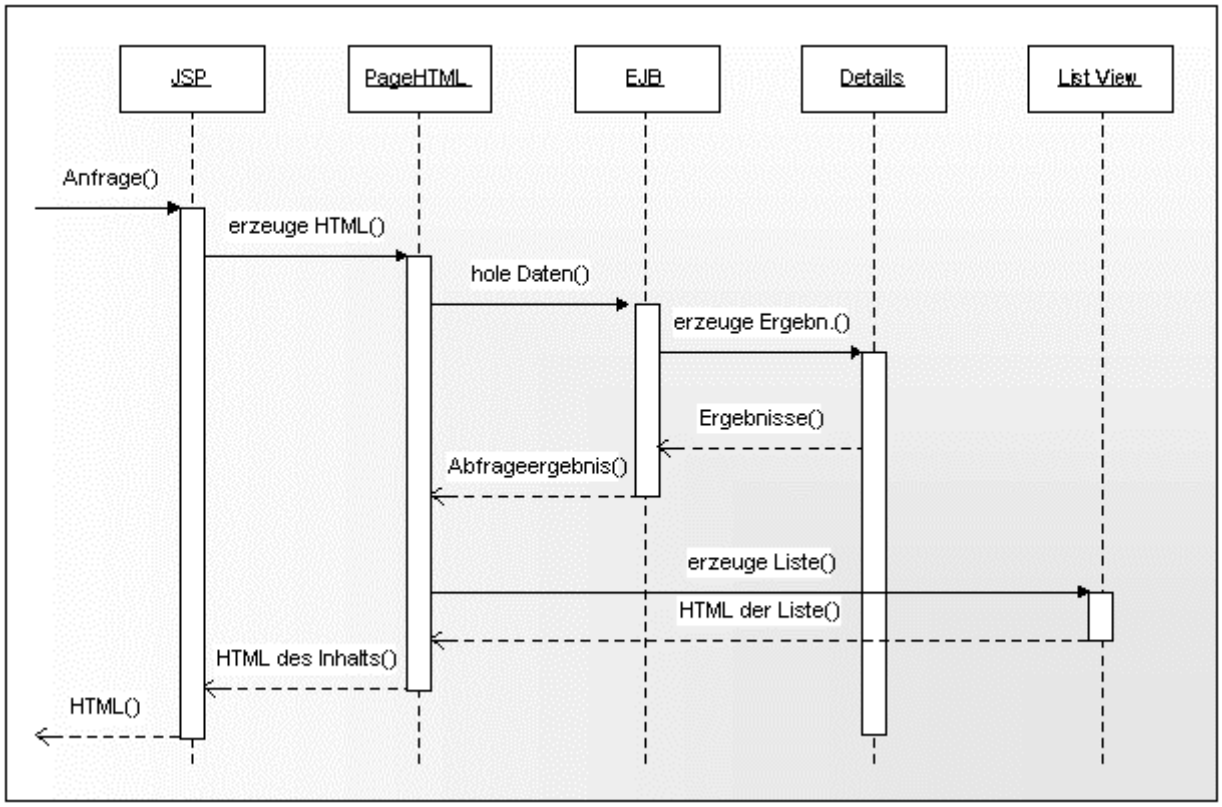

Abbildung 24: Sequenzdiagramm Seitenerzeugung

Hier wird noch einmal die strikte Trennung zwischen Datenhaltung, Anwendungslogik und Darstellung deutlich.
Einzelne abgefragte Datensätze werden mittels einer Klasse aus dem Package Details dargestellt. Eine EJB stellt eine Anfrage an die Datenbank und erzeugt zu den gefundenen Datensätzen entsprechende Objekte. Die Darstellung erfolgt durch eine HTML generierende Klasse mittels dieser Ergebnisobjekte und unter Zuhilfenahme allgemeiner, wiederverwendbarer Steuerelemente wie der List View.

Beispiel: Fahrzeug-Seite

Einen der grundlegenden Dialoge des Auskunftssystems stellt eine Gesamtliste der Fahrzeuge des Datenbestandes dar. Beispielhaft ist im folgenden dargestellt, welche Klassen und Dokumente für ihre Erstellung benötigt werden.

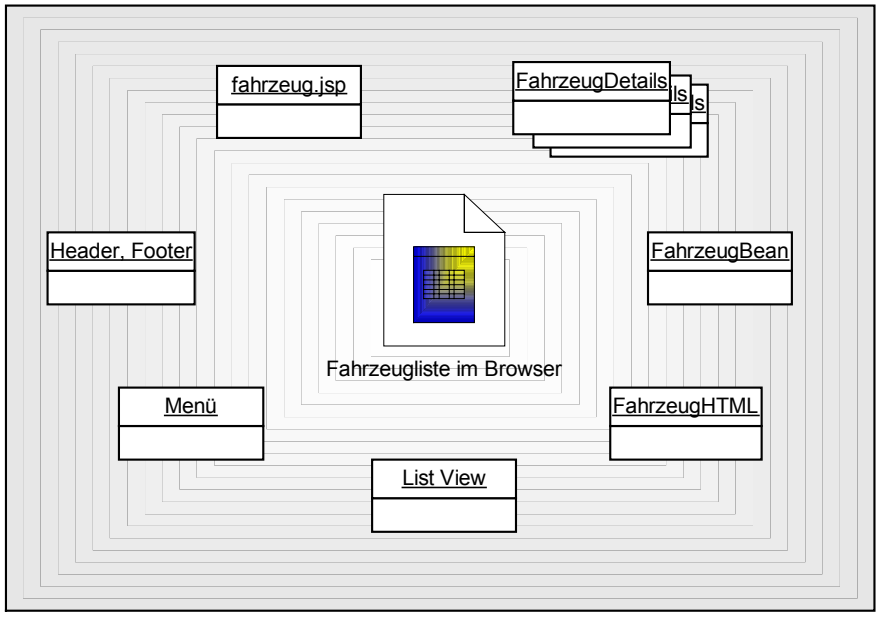

Abbildung 25: Klassen zum Erzeugen der Fahrzeug-Seite

An den Namen der Elemente lässt sich weiterhin erkennen, ob die Klassen oder Dokumente wiederverwendet oder angepasst werden oder ob sie nur für diesen speziellen Dialog existieren. Die Elemente, die den Begriff „Fahrzeug“ enthalten, sind dialogspezifisch, alle weiteren sind allgemein gehalten bzw. wiederverwendet.

Tabelle 18: Klassen zum Erzeugen der Fahrzeug-Seite

Klasse / Dokument	Funktion	Einsatz
fahrzeug.jsp	stellt HTML für Fahrzeugseite zusammen	nur für diesen Dialog
FahrzeugHTML	generiert HTML der Fahrzeugliste	nur für diesen Dialog
FahrzeugDetails	enthält Grunddaten darzustellender Fahrzeuge	nur für diesen Dialog
FahrzeugBean	Abfrage der Datenbank nach Fahrzeugen	für Dialoge mit Datenbezug zu Fahrzeugen
List View	flexible Darstellung von Listen	in allen Dialogen mit Listen
Menü	Menüs und Untermenüs, Funktionalität der Menüs	in allen Dialogen
Header, Footer	Kopf-, Fußzeilen einer Seite	in allen Dialogen

Dialogstruktur

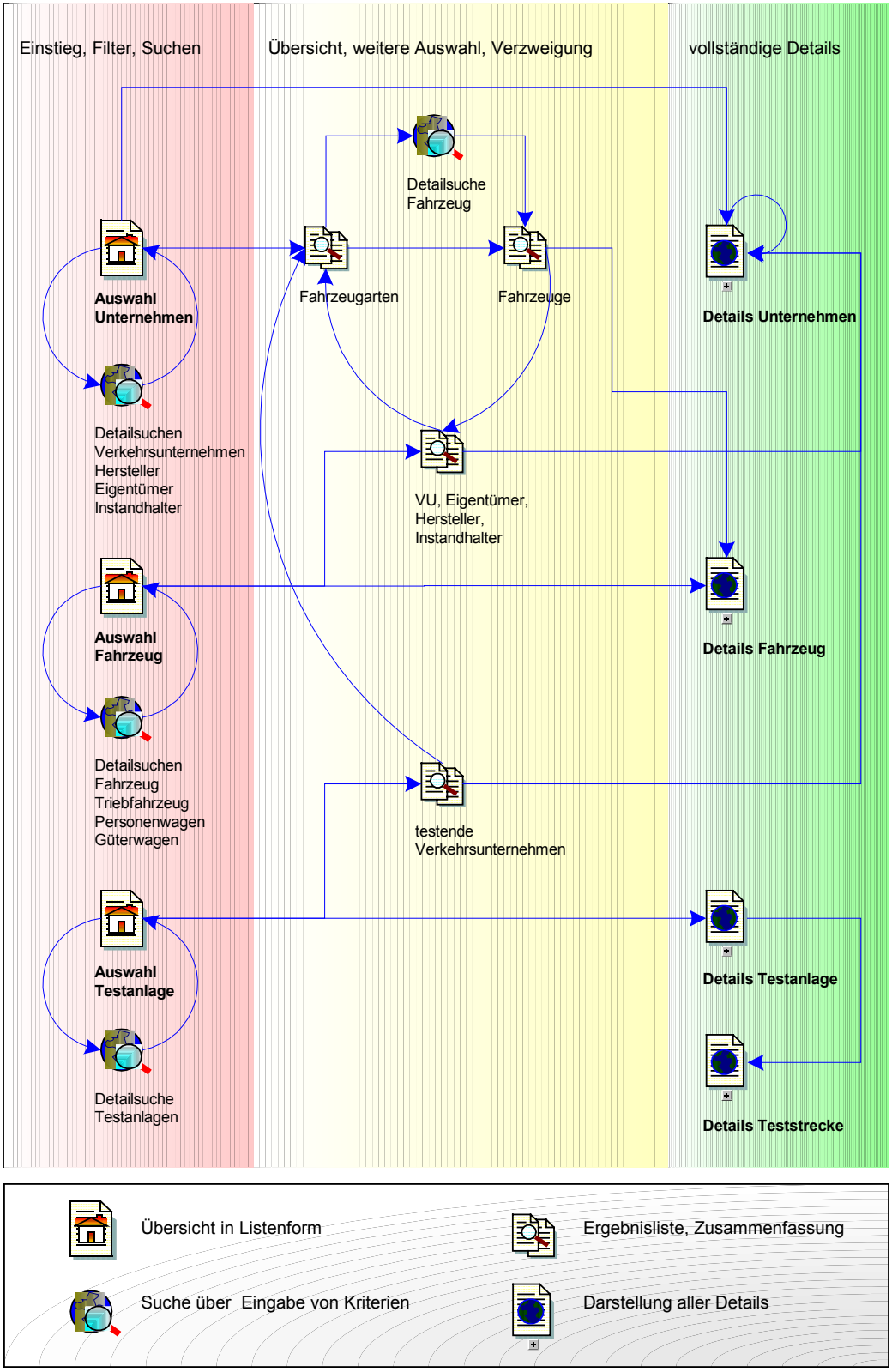

Abbildung 26: Dialogstruktur des Auskunftssystems

4.3.4 Sicherheit

Ein wichtiger Aspekt bei der Entwicklung der Webanwendung ist deren Sicherheit. Die Anwendung steht zur Laufzeit im Intranet oder gar im Internet zur Verfügung. Hierbei sollen die im Auskunftssystem recherchierbaren Informationen nur einem bestimmten Personenkreis zur Verfügung stehen. Mittels Verschlüsselungstechnologien, Passwortschutz und anderen Mechanismen soll verhindert werden, dass Dritte sich Zugang zu den Daten verschaffen oder diese sogar manipulieren können.

Datenübertragung mit SSL

Der Secure Socket Layer ist ein auf HTTP aufbauendes Protokoll und bietet sichere Verbindungen in einem Netzwerk. SSL ermöglicht nicht nur die verschlüsselte Übertragung der Daten im Netz sondern stellt auch die Identität des Gegenüber sicher.
Der bei der Entwicklung verwendete Applikationsserver BEA Weblogic in der Version 7 unterstützt SSL 3.0. Eine Einführung in die Verwendung von SSL auf dem BEA Weblogic Server findet sich auf [Bea04]. Im Wesentlichen sind zur Einrichtung der SSL-Kommunikation folgende Schritte nötig:

Tabelle 19: SSL in BEA Weblogic einrichten

1. Für jeden mit SSL einzusetzenden Weblogic Server muss ein privater Schlüssel und ein digitales Zertifikat bezogen werden. Um bei einer der Autorisierungsbehörden wie beispielsweise VeriSign (http://www.verisign.com/) oder dem DFN (http://www.pca.dfn.de/certify/ssl/serverca.html) ein Zertifikat zu beantragen, kann die mit Weblogic ausgelieferte Webapplikation „certificate“ genutzt werden[6].
2. Die erhaltenen Zertifikate und Schlüssel sind in Weblogic an den entsprechenden in [Bea04] angegebenen Stellen abzulegen.
3. Der SSL-Port von Weblogic ist über die Administrationskonsole zu aktivieren. Hier sind auch entsprechende Einstellungen wie der Speicherort der Zertifikate zu tätigen.

EJB-Sicherheit

Neben der verschlüsselten Übertragung der Daten im Netzwerk ist ein Sicherheitsmechanismus für die eingesetzten verteilten Komponenten nötig. Über den Java Namens- und Verzeichnisdienst JNDI wäre es sonst jeder Anwendung möglich, die EJB-Komponenten aufzufinden und deren Methoden aufzurufen. Die EJB-Spezifikation ermöglicht daher, vor der Installation (Deployment) der Anwendung festzulegen, welche Nutzerrollen bei der Anwendung existieren und welche Zugriffsrechte die einzelnen Rollen haben.

[6] Aufruf durch https://hostname:port/certificate

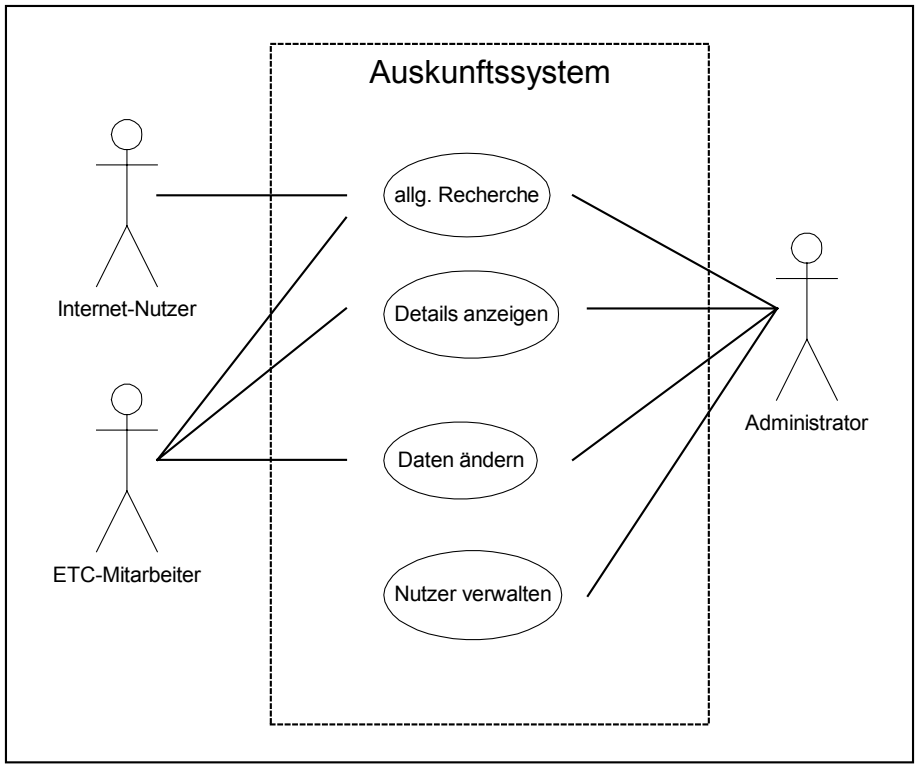

Abbildung 27: Rollen und deren Rechte bei der Nutzung des Auskunftssystems

Das abgebildete Use Case Diagramm zeigt die bei der Benutzung des Auskunftssystems möglichen Rollen. Die Anwendung soll hauptsächlich den Mitarbeitern in der ETC zur Verfügung stehen. Diese haben uneingeschränkten Zugriff auf die Recherchemöglichkeiten und können fachliche Änderungen an den Daten vornehmen. Der Administrator hat zudem die Möglichkeit, Nutzer zu verwalten. Sollte das Auskunftssystem auch im Internet zur Verfügung stehen, so haben dortige Nutzer nur eine eingeschränkte Sicht auf die Daten und können nicht sämtliche Details abrufen.

Diese verschiedenen Rollen werden in der Installationsbeschreibung (Deployment Descriptor) der Anwendung definiert.

Tabelle 20: Festlegung von Rollen im Deployment Descriptor

```
...
<assembly-descriptor>
      <security-role>
            <description>
             Anwender im Internet, eingeschränkte Ergebnisse
            </description>
            <role-name>internet-user</role-name>
      </security-role>
      <security-role>
            <description>
             Mitarbeiter der ETC, uneingeschränkte Recherche
            </description>
            <role-name>etc-user</role-name>
      </security-role>
</assembly-descriptor>
...
```

Der Auszug aus einer Installationsbeschreibung zeigt beispielhaft die Festlegung der Rollen für Internet-Nutzer und ETC-Mitarbeiter.

Nach der Definition der Rollen kann für jede einzelne Methode einer EJB-Komponente festgelegt werden, von Mitgliedern welcher Rollen sie aufgerufen werden kann.

Tabelle 21: Zugriffsrechte für eine Methode festlegen

```
...
<method-permission>
      <description>
       Nur ETC-Mitarbeiter dürfen alle Fahrzeug-Details abrufen
      </description>
      <role-name>etc-user</role-name>
      <method>
            <ejb-name>FahrzeugBean</ejb-name>
            <method-name>getFahrzeugDetails</method-name>
      </method>
</method-permission>
...
```

Hier wird definiert, dass nur Benutzer der Rolle „etc-user“ auf die Methode zugreifen können, die alle Details zu einem Fahrzeug liefert.
Für mehr Informationen zum Thema Sicherheit von EJB-Anwendungen siehe auch in [Kompf04, S.377ff.] und [Farley03, S.254ff.].

Login mit LDAP

Bei der Entwicklung wird weiterhin der bereits in der ETC in Betrieb befindliche LDAP-Server genutzt. LDAP (Lightweight Directory Access Protocol) ist ein Protokoll für einen Verzeichnisdienst, der beispielsweise Angaben zu möglichen Nutzern einer Webanwendung und deren Rechten enthält.

4.3.5 Suche

Der praktische Nutzen eines Auskunftssystems hängt in der Regel wesentlich davon ab, wie gut die Suche nach Informationen umgesetzt ist.
Auf der einen Seite sollen einfache Abfragen sehr schnell und auch für neue Benutzer unproblematisch möglich sein, auf der anderen Seite soll die Suchfunktion auch umfangreichere, verschachtelte Abfragen unterstützen. Bei der Suche sollen zudem je nach Art des Suchkriteriums unterschiedliche Operatoren möglich sein. Bei einem Textfeld könnte den Nutzer beispielsweise interessieren, welche Datensätze mit einem bestimmten Begriff enden, bei Zahlenwerten kann von Interesse sein, bei welchen Datensätzen ein bestimmter Schwellenwert nicht überschritten wird.
Der Entwurf einer Suchfunktion für ein Auskunftssystem und die spätere Implementierung sind komplex, doch ähneln sich die Suchstrategien und Kriterien in verschiedenen Auskunftssystemen sehr. Eine Suche nach einem Buch in einem Bibliothekssystem ist genau genommen das Gleiche wie das Auffinden eines Fahrzeugs anhand technischer Daten: Letztendlich werden in einer Datenbank Tupel anhand bestimmter Suchkriterien gefunden.

Allen Suchfunktionen liegt also folgendes Muster zugrunde:

Welche Tupel in bestimmten Tabellen einer Datenbank entsprechen in bestimmten Attributen den eingegebenen Werten?

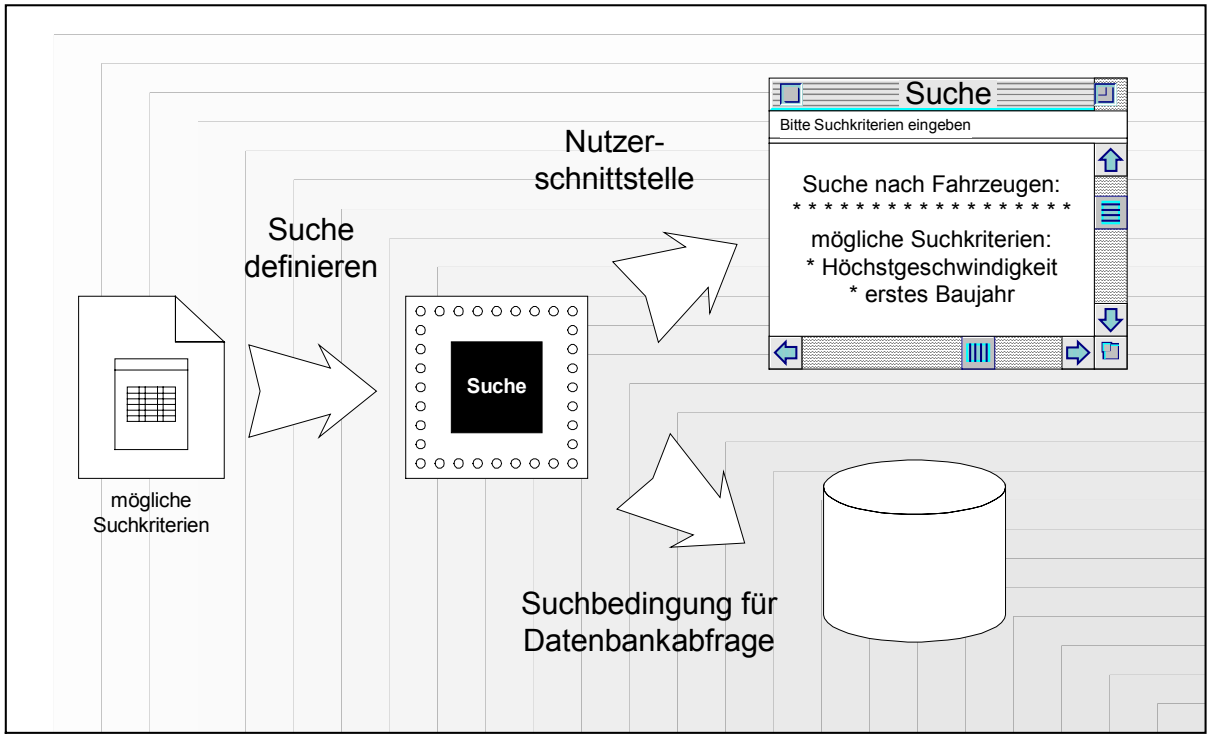

Abbildung 28: Eine flexible Suchfunktion

Der Entwurf der Suchfunktion für das Auskunftssystem erfolgt in der Weise, dass die Suche auch in anderen Auskunftssystemen wiederverwendet werden kann. Dazu wird die Logik einer allgemeinen Suche von der Beschreibung einer Suchfunktion in einem konkreten Auskunftssystem getrennt.
Die Beschreibung der Suchfunktion mit ihren Suchkriterien sollte schnell und einfach an neue Anwendungsfälle anpassbar sein. Hierfür bieten sich XML-Dokumente geradezu an. Sie sind textbasiert und somit für jedermann lesbar, frei strukturierbar und können mit unzähligen Werkzeugen erstellt, bearbeitet und auf Wohlgeformtheit überprüft werden.
Die allgemeine Suchfunktionalität verwendet diese XML-Dokumente, liest und interpretiert die in XML festgelegte Struktur der Suche, stellt für die konkrete Suche eine intuitive Nutzerschnittstelle bereit und formuliert nach Erfassen aller Nutzereingaben die Anfrage an die Datenbank.

XML-Beschreibung

In einem XML-Dokument werden alle möglichen Suchkriterien definiert. Folglich enthält das XML-Dokument folgende Angaben:

Tabelle 22: Inhalt des XML-Dokuments der Suche

- die Bezeichnungen der möglichen Suchkriterien
- die Namen der zugehörigen Attribute der Relation in der Datenbank
- die Typen der Suchkriterien (z.B. Text, Zahl, Boolean)

Beispielsweise könnte sich die XML-Beschreibung einer Suche nach Schienenfahrzeugen folgendermaßen zusammensetzen:

Tabelle 23: Beispiel einer XML-Suchbeschreibung

```
<Suche>
  <XmlObj id="fahrzeuge">
    <Name>Fahrzeugtyp</Name>
    <Name>Spurweite</Name>
    <Name>erstes Herstellungsjahr</Name>
    <Name>Vmax</Name>
    <Name>Drehgestellbauart</Name>
    <Name>Laenge ueber Puffer</Name>

    <DBName>f.FzgTyp</DBName>
    <DBName>f.Spurweite_Fzg</DBName>
    <DBName>f.hergestellt_seit</DBName>
    <DBName>f.v_max_1</DBName>
    <DBName>f.drehgestellbauart</DBName>
    <DBName>f.laenge_ue_p</DBName>

    <Typ>Text</Typ>
    <Typ>Zahl</Typ>
    <Typ>Zahl</Typ>
    <Typ>Zahl</Typ>
    <Typ>Text</Typ>
    <Typ>Zahl</Typ>
  </XmlObj>
</Suche>
```

Klassen

Die allgemein gehaltene Suchfunktionalität wird in entsprechenden Klassen umgesetzt. Diese Klassenstruktur bildet unter anderem die Hierarchie einer Suche ab, bei der eine Suchabfrage aus verschiedenen Suchkriterien besteht.
Im Folgenden sind die Klassen der Suchfunktionalität dargestellt.

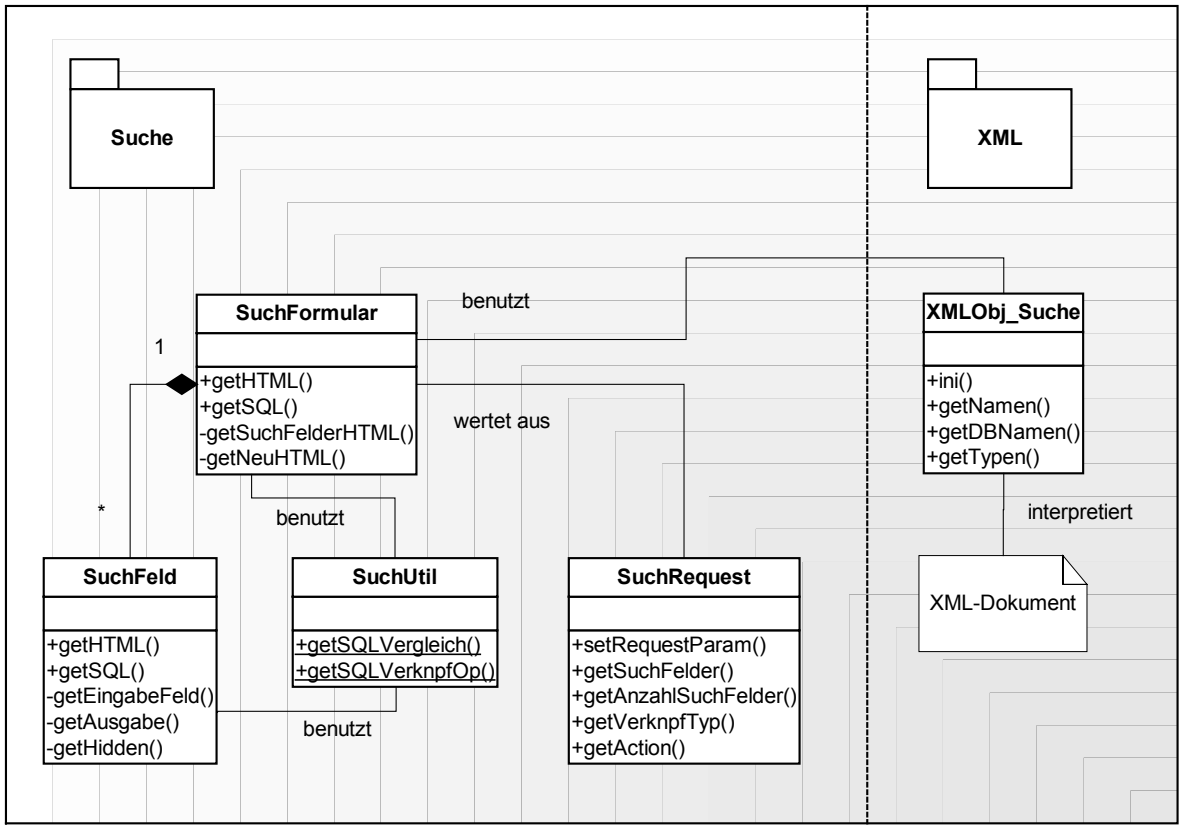

Abbildung 29: Klassen der Suche

Tabelle 24: Klassen der Suche

SuchFormular
Objekte dieser Klasse repräsentieren ein Suchformular einer Anwendung. Nach der Erzeugung kann mit zwei Methoden zum einen der gesamte HTML-Code der Nutzerschnittstelle sowie, wenn die Suche angestoßen wurde, der SQL-Code der erfassten Suchkriterien abgerufen werden.

SuchFeld
Ein Objekt dieser Klasse entspricht einem Suchkriterium. Das SuchFeld kann bereits mit Werten belegt oder nur anhand Name und Typ erzeugt werden. Eine HTML-Code erzeugende Methode liefert bei bereits erfassten Werten eine Beschreibung des Suchkriteriums. Bei einem noch leeren SuchFeld liefert sie stattdessen die Eingabefelder der Nutzerschnittstelle. Eine weitere Methode liefert, wenn im übergeordneten SuchFormular die Suche angestoßen wurde, das erfasste einzelne Suchkriterium als SQL-Bedingung.

SuchRequest
Ein Objekt dieser Klasse wertet die Anfrage aus, die ein Suchformular bei der Interaktion mit dem Nutzer an sich selbst sendet. Da in der Webandwendung das zustandslose Protokoll HTTP zum Einsatz kommt, müssen bereits eingegebene Suchkriterien mittels versteckter Felder weitergegeben und durch die Klasse SuchRequest wieder auf die SuchFelder abgebildet werden.

SuchUtil
Diese Klasse enthält mehrere Klassenmethoden, um die erfassten Suchkriterien in SQL-Code zu übersetzen. Dies ermöglicht eine schnelle Anpassung der Suche an verschiedene Datenbankbetriebssysteme, da sich gegebenenfalls Operatoren und Platzhalter unterscheiden und hier schnell geändert werden können.

XMLObject_Suche
Diese Klasse interpretiert das die Suche beschreibende XML-Dokument. Über Methoden kann auf die Elemente des XML-Dokuments zugegriffen und somit die Struktur der Suche abgerufen werden.

Integration in Anwendung

Die Suche ist sehr leicht an verschiedene Einsatzzwecke anpassbar und ist einfach in Anwendungen integrierbar.

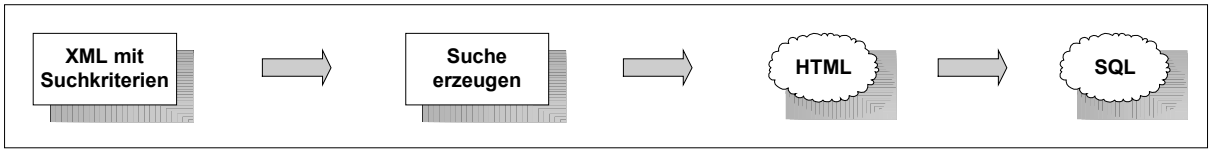

Abbildung 30: Einfacher Einsatz der Suche

Zunächst ist ein XML-Dokument zu erstellen, in welchem die möglichen Suchkriterien festgelegt werden. In der Webanwendung ist ein Dialog für die spätere Suche zu erstellen und eine Instanz von SuchFormular zu erzeugen. Eine Methode daraus liefert den HTML-Code der Nutzerschnittstelle. Jetzt kann der Nutzer seine gewünschten Suchbedingungen zusammenstellen. Stößt er schließlich die Suche an, liefert eine Methode des SuchFormular die Suchbedingungen in SQL-Notation. Der so erzeugte SQL-Anweisungsteil wird an die Komponente zur Datenbankabfrage übergeben und das Suchergebnis in geeigneter Form dargestellt.
An dieser Stelle wird auch deutlich, dass die Zusammenstellung der Suchbedingungen durch den Anwender sehr performant ist. Während der Eingabe der Suchkriterien erfolgen keinerlei Datenbankzugriffe, beispielweise um mögliche durchsuchbare Attribute zu finden. Alle benötigten Informationen über die Suchmöglichkeiten sind im XML-Dokument enthalten. Der Datenbankzugriff erfolgt nur ein einziges Mal – nach Anstoßen der Suche zur Abfrage der Suchergebnisse.

4.4 Implementierung

Bei der Implementierung des Systems wird der Entwurf aus 4.3 in Software umgesetzt.

4.4.1 Vorgehensweise

In 4.3.3 ist dargestellt, wie eine Seite der Webapplikation erzeugt wird. Durch einen hohen Wiederverwendungsgrad beispielsweise bei Menüs und Steuerelementen wie List Views beschränkt sich die Implementierung eines neuen Dialoges auf das Kodieren weniger Klassen und Datenbankabfragen.

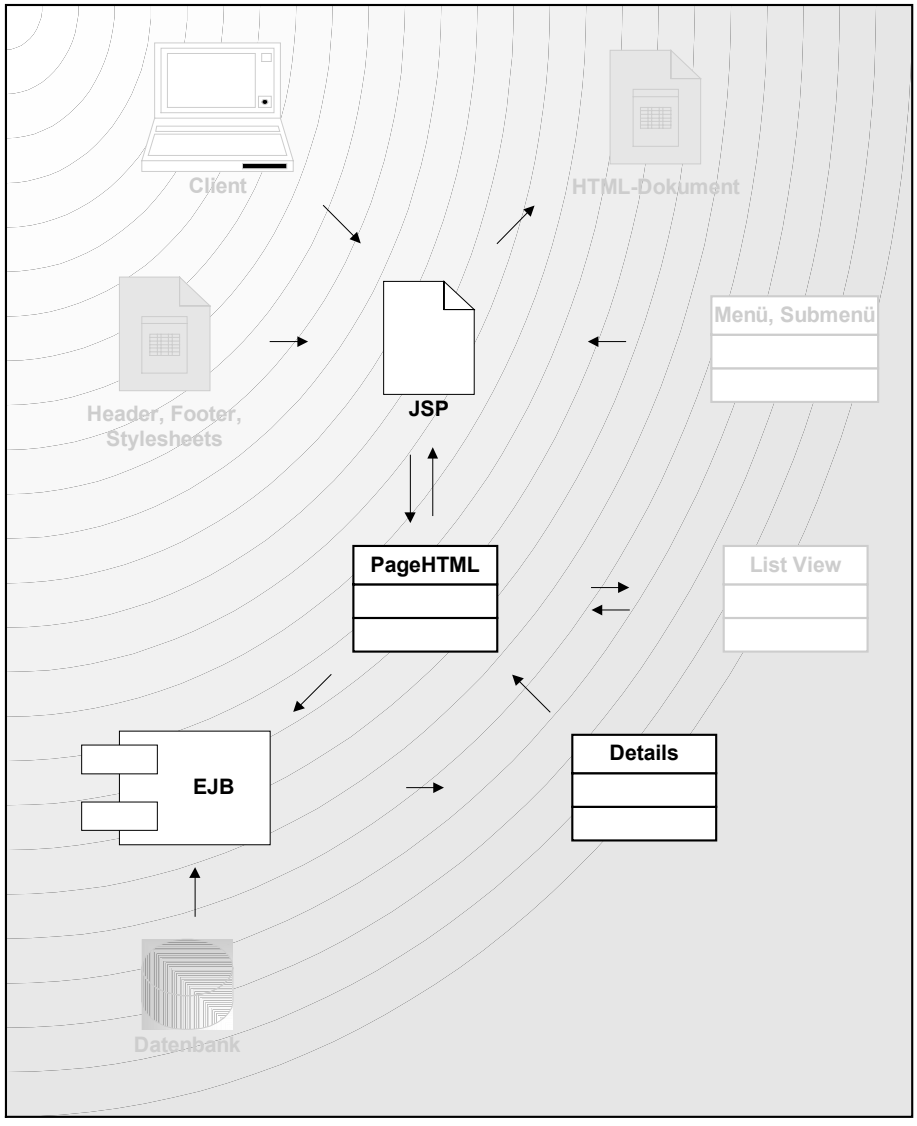

Abbildung 31: Zu implementierende Klassen und Dokumente

Nach der Implementierung der ersten Dialoge stellte sich eine sehr effektive Reihenfolge der Kodierung heraus. Da in der Implementierungsphase sämtliche Schnittstellen und Interaktionen bereits feststehen und an der Struktur der Software nur noch marginale Änderungen vorgenommen werden, ist die Reihenfolge der Kodierung der Klassen eigentlich nicht von Bedeutung.
Es gibt jedoch Gründe, die für die hier beschriebe Abfolge sprechen. Beispielsweise unterstützen viele IDE[7] heutzutage die automatische Ergänzung von Parametern bei Methodenaufrufen oder importieren vor dem Übersetzen einer Klasse alle weiteren benötigten Klassen und Packages. Auch stellt die vorgestellte Reihenfolge eine gewisse logische Abfolge dar.
Die wesentlichen Schritte der Implementierung eines Dialogs sind:

Tabelle 25: Implementierungsschritte

1. Detailklasse für Haltung der Daten implementieren
2. Datenbank abfragen, Instanzen der Detailklasse erzeugen
3. Ergebnisse in Listenform in HTML darstellen
4. HTML mit Menüs, Headern, Footern in JSP zusammenstellen

Bezogen auf die gewählte 3-Schichten-Architektur, die in 3.3.3 dargestellt wurde, bedeutet dies eine Implementierung von der Schicht der Anwendungslogik zur Datenhaltung und schließlich zur Darstellung.

[7] „integrated developer's environment": integrierte Entwicklungsumgebung, d.h. eine Software, die die Phasen des Entwicklungsprozesses unterstützt und Werkzeuge für Entwurf und Implementierung bereitstellt

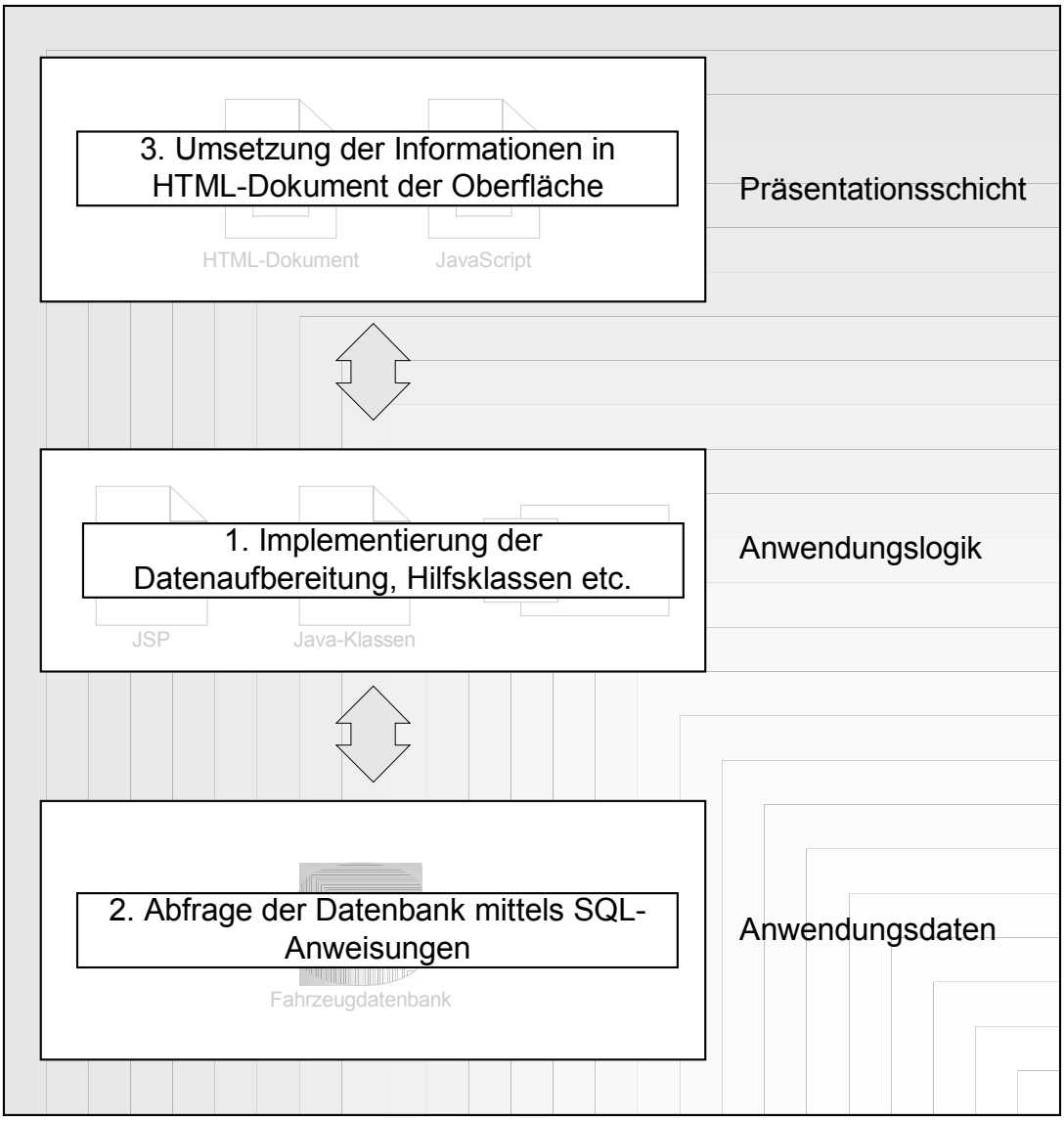

Abbildung 32: Reihenfolge der Implementierung in der 3-Schichten-Architektur

Im folgenden werden einzelne Schritte der Implementierung näher erläutert und anhand von Quellcode-Auszügen verdeutlicht.

4.4.2 ausgewählte Implementierungsdetails

SQL der EJB

Der Entwurf der Anwendung beschreibt den Datenzugriff mittels zustandsloser Session Beans[8]. Bei der Implementierung der Zugriffsmethoden des Remote Interfaces sind folglich für jede Abfrage „von Hand" entsprechende SQL-Anweisungen zu formulieren.

In diesen SQL-Anweisungen werden die abzufragenden Daten bereits nach verschiedenen Kriterien gefiltert bzw. sortiert. So wird beispielsweise die „Schnellsuche" des List View Steuerelements direkt in

[8] siehe auch 4.3.2

der Abfrage verarbeitet. Auch wird die in der Liste durch den Anwender gewählte Sortierung direkt in der SQL-Anweisung durchgeführt.
Weiterhin werden, um die Geschwindigkeit der Anwendung zu erhöhen, nur so viele Datensätze abgefragt, wie für die Darstellung aktuell benötigt werden. So werden weniger Ergebnisobjekte erzeugt und folglich findet weniger Übertragung über das Netz statt. Leider bietet das bei der Entwicklung zum Einsatz gekommene Datenbankbetriebssystem Oracle 9i im Gegensatz zu anderen Produkten nicht die Möglichkeit, mit einem einfachen Schlüsselwort die Ergebnismenge einzugrenzen[9].
Oracle 9i bietet jedoch die Möglichkeit, die Zeilennummer des Ergebnisses mit Hilfe des Schlüsselwortes ROWNUM abzufragen. Eine 2-fach geschachtelte Unterabfrage ermöglicht unter Verwendung dieses ROWNUM, Anzahl und Bereich der Ergebnisse einzugrenzen.

Die Struktur einer typischen Abfrage von Daten für die Übersichtslisten zeigt folgende SQL-Anweisung. Mit ihr werden alle Triebfahrzeuge abgefragt, die ein konkreter Nutzer in einem Dialog sieht.

Tabelle 26: SQL-Anweisung zur Auflistung aller Triebfahrzeuge

```
select
fzgtyp, produktlinie, tfzgart,
antriebsart, stromsystem,
leistungsklasse, id_fzg
from (
     select rownum rn,
     fzgtyp, produktlinie, tfzgart,
     antriebsart, stromsystem,
     leistungsklasse, id_fzg
     from (
          select
          fzgtyp, produktlinie, tfzgart,
          antriebsart, stromsystem,
          leistungsklasse, f.id_fzg id_fzg
          from fis.fzg f,
          fis.triebfahrzeug t,
          fis.tfzgart ta,
          fis.antriebsart a,
          fis.stromsystem s
          where f.id_fzg = t.id_fzg
          and t.id_tfzgart = ta.id_tfzgart
          and t.id_antrieb = a.id_antrieb
          and t.id_stromsystem = s.id_stromsystem
          and (upper(fzgtyp) like ?
               or upper(produktlinie) like ?
               or upper(tfzgart) like ?
               or upper(antriebsart) like ?
               or upper(stromsystem) like ?
               or upper(leistungsklasse) like ? )
          order by orderBy ?, id_fzg asc)
     where rownum <= ?
     )
where  rn >= ?
```

[9] vgl. beispielsweise das LIMIT-Statement in [MySQL04]

In der inneren SQL-Anweisung werden die für die Ansicht wichtigen Attribute von allen Triebfahrzeugen abgefragt. Über vier Gleichverbünde werden Domänen wie Antriebsart und Stromsysteme verbunden. Durch Reihe von LIKE-Vergleichen werden die Attribute auf den Suchbegriff der Schnellsuche der List View durchsucht, außerdem erfolgt bereits hier die Sortierung nach dem durch den Nutzer gewählten Kriterium. Da es sich aus Geschwindigkeitsgründen um eine so genannte vorbereitete SQL-Anweisung[10] handelt, sind die Parameter als Fragezeichen dargestellt. Sie werden vor Ausführen der Anweisung durch den entsprechenden Wert ersetzt. Dies spart Rechenzeit beim wiederholten Ausführen der selben Anweisung mit verschiedenen Parametern.
Die nächste SELECT-Anweisung fragt nun das so erzeugte Ergebnis ab und ergänzt es durch eine von Oracle erzeugte Zeilennummer. Gleichzeitig schneidet eine WHERE-Anweisung die nicht benötigten Ergebnisse am Ende ab.
Die äußere SELECT-Anweisung fragt wiederum das vorige Ergebnis ab und schneidet durch einen WHERE-Vergleich der vorher generierten Zeilennummern die nicht benötigten Ergebnisse am Anfang ab.
Die sich vielleicht an dieser Stelle anbietende Lösung

```
SELECT * FROM a WHERE rownum > ? AND rownum < ?
```

funktioniert leider nicht, da Oracle die Zeilennummern für „rownum“ erst nach Erzeugung des Ergebnisses generiert und dann der Vergleich mit der unteren Grenze nicht mehr funktioniert. Daher ist die zweistufige Unterabfrage nötig.

Datenzusammenstellung für List Views

Nachdem die Datenbankabfrage erzeugt, mit ihren Parametern versehen und ausgeführt wurde, werden aus den von der Datenbank zurückgegebenen Tupeln die Ergebnisobjekte zusammengestellt.
Der Konstruktor beispielsweise eines TriebfahrzeugDetails-Objektes übernimmt Angaben zum Typ des Triebfahrzeugs und spezifischen technischen Details wie die Art des Antriebs.

Tabelle 27: Konstruktor eines Ergebnisobjektes

```
    public TfzDetails ( long id_fzg,
                        String FzgTyp,
                        String Produktlinie,
                        String TfzgArt,
                        String Antrieb,
                        String Stromsystem,
                        String Leistung)
```

Dieser Konstruktor wird mit den durch die Datenbankabfrage gewonnenen Informationen aufgerufen. Beispielsweise werden für ein Triebfahrzeug die durch die Abfrage gewonnen Informationen über den Typ und wichtige technische Eigenschaften an den Konstruktor des TriebfahrzeugDetails-Objektes übergeben.

[10] zum Thema „Prepared SQL“ siehe auch sehr ausführlich in [Reese00, S.57ff.]

Tabelle 28: Erzeugen der Triebfahrzeug-Objekte aus dem Abfrageergebnis

```
            while(result.next())
            {

              TfzDetails details = new TfzDetails (result.getLong(7),
                                                   result.getString(1),
                                                   result.getString(2),
                                                   result.getString(3),
                                                   result.getString(4),
                                                   result.getString(5),
                                                   result.getString(6));
              tfzListe.add(details);
            }
```

Das Ergebnis ist eine Menge von Ergebnisobjekten, welche nun für die Darstellung weiterverwendet werden können. Dazu müssen diese Ergebnisobjekte in eine Darstellungsform umgewandelt werden, welche vom List View Steuerelement interpretiert und dargestellt werden kann. Bei der im Prototyp verwendeten Liste ist dies eine textuelle Darstellung, getrennt durch das Pipe-Sonderzeichen ‚|'.
Hier bot sich an, für diese Umwandlung die von der Klasse java.lang.Object geerbte Methode toString zu überladen und somit eine bequeme Umwandlung der Ergebnisse in eine Textdarstellung zu ermöglichen. Das TriebfahrzeugDetails-Objekt beispielsweise überlädt seine toString-Methode wie folgt.

Tabelle 29: Umwandeln eines Triebfahrzeugs in eine Textdarstellung

```
    public String toString() {
        String s =  id_fzg + "|" + FzgTyp + "|" + Produktlinie + "|"
                           + TfzgArt      + "|"
                           + Antrieb      + "|"
                           + Stromsystem  + "|" + Leistung;
        return s;
    }
```

Beim Erzeugen des HTML-Codes der Oberfläche kann nun die erzeugte Datenstruktur der Ergebnisobjekte direkt an das List View Steuerelement weitergegeben werden.

Tabelle 30: Erzeugen des HTML-Codes der Triebfahrzeugliste

```
          html += "Gesamtbestand Triebfahrzeuge: <p>";
          html += list.SetList(tfzListe, xmlObjList, ...);
```

Mehr Code ist im allgemeinen für die Erzeugung der flexibel anpassbaren und mit Sortier- und Suchfunktionen ausgestatteten Liste nicht nötig.

Implementierung der Suche

Wie in 4.3.4 beschrieben, benötigt die flexibel an verschiedene Auskunftssysteme anpassbare Suchfunktion im Wesentlichen eine XML-Beschreibung und Klassen, die die gesamte Suche und einzelne Bedingungen darstellen.
Im Mittelpunkt einer Suche steht eine einzelne Suchbedingung. Je nachdem ob der Anwender bereits Werte für dieses Suchfeld eingegeben oder er gerade ein neues Eingabefeld angefordert hat, wird die Klasse SuchFeld mit verschiedenen Parametern instanziiert.

Tabelle 31: Konstruktoren der Klasse SuchFeld

```
public SuchFeld(int index, String name, String dbname, String typ)
{
  this.index = index;
  this.name = name;
  this.dbname = dbname;
  this.typ = typ;
}

public SuchFeld(int index, String name, String dbname, String typ,
                Vector sucharten, Vector werte, String verknTyp,
                boolean mitVerknEingabe)
{
  this.index = index;
  this.name = name;
  this.dbname = dbname;
  this.typ = typ;
  this.sucharten = sucharten;
  this.werte = werte;
  this.verknTyp = verknTyp;
  this.mitVerknEingabe = mitVerknEingabe;
}
```

Auch die entsprechende Repräsentation einer Suchbedingung in HTML-Code unterscheidet sich. Das aufrufende gesamte Suchformular lässt sich einfach mit getHTML den HTML-Code des SuchFeldes erzeugen.

Tabelle 32: Methode getHTML eines SuchFeldes

```
public String getHTML()
{
 String html = null;
 if (this.name!=null && this.typ!=null && this.werte==null)
 {
    html = getEingabeFeld();
 }
 else if (this.name!=null && this.sucharten!=null && this.werte!=null)
 {
   html = getAusgabe();
   html += getHidden();
   if (mitVerknEingabe)
   {
      html += getVerknEingabe();
   }

 }
 return html;
}
```

Hier werden die beiden unterschiedlichen Möglichkeiten deutlich, ein SuchFeld in HTML darzustellen. Hat der Anwender noch keine Werte eingegeben, erhält er ein Eingabefeld und die Möglichkeit, die Suchmodalitäten (z.B. „ist kleiner als“ bei Zahlenwerten oder „enthält nicht“ bei Textfeldern) einzustellen. Sind bereits Suchkriterien eingegeben und soll das Feld lediglich als bereits erfasste Suchbedingung dargestellt werden, werden eine textuelle Darstellung der Bedingung und die zur Datenweitergabe nötigen versteckten Felder erzeugt.
Das übergeordnete SuchFormular erzeugt nun durch Aufrufen der getHTML-Methoden der einzelnen Suchkriterien den gesamten HTML-Code, der dem Anwender im Browser dargestellt wird.

Tabelle 33: HTML-Erzeugung des gesamten SuchFormular

```
    Vector felder = request.getSuchFelder();
    if (felder!=null)
    {
        SuchFeld sf = null;
        Enumeration e = felder.elements();
        while (e.hasMoreElements())
        {
          sf = (SuchFeld)e.nextElement();
          html += sf.getHTML();
          html += "<br>";
        }
    }
```

Die Einbindung in die Webanwendung geschieht nun durch einen simplen Aufruf dieser getHTML-Methode einer Instanz von SuchFormular:

Tabelle 34: Einfache Einbindung der Suche in die Webanwendung

```
   SuchFormular f = new SuchFormular (xmlID,request);
   html += "<b>Suche nach Fahrzeugen:</b>";
   html += "<p><center>";
   html += f.getHTML();
   html += "</center>";
```

Dieser Methodenaufruf ist alles, was bei einer Wiederverwendung der Suche in weiteren Auskunftssystemen an Java-Code geschrieben werden muss. Hiervor muss lediglich noch das die Suche beschreibende XML-Dokument erstellt werden.

Nur in Einzelfällen ist eine Anpassung an das verwendete Datenbankbetriebssystem nötig. Beispielsweise halten sich nicht alle Hersteller an die bekannten SQL-Standards[11] hinsichtlich Vergleichen mit dem LIKE-Operator. Diese Anpassung geschieht durch Ändern der entsprechenden Zeichenketten in der Klasse SuchUtil. Hier ist genau festgeschrieben, welche Suchmodalität durch welche Kombination von SQL-Schlüsselwörtern dargestellt wird.

Tabelle 35: Umwandeln von Vergleichen in SQL

```
public static String getSQLVergleich (String feld, String verglArt,
                                                    String wert)
  {
    if (verglArt.compareTo("enthaelt")==0)
    {
        return new String("UPPER("+feld+") LIKE '%" + wert + "%'");
    }
    if (verglArt.compareTo("beginnt mit")==0)
    {
        return new String("UPPER("+feld+") LIKE '" + wert + "%'");
    }
    if (verglArt.compareTo("endet mit")==0)
    {
        return new String("UPPER("+feld+") LIKE '%" + wert + "'");
    }
    if (verglArt.compareTo("ist exakt")==0)
    {
        return new String("UPPER("+feld+") = '" + wert + "'");
    }
    if (verglArt.compareTo("=")==0)
```

[11] eine Kurzübersicht findet sich z.B. auf [Kalis04]

```
    {
        return new String(feld + " = " + wert);
    }
    if (verglArt.compareTo(">")==0)
    {
        return new String(feld + " > " + wert);
    }
    if (verglArt.compareTo("<")==0)
    {
        return new String(feld + " < " + wert);
    }
    return "1=1";
  }
```

Die hier dargestellten SQL-Vergleichsmöglichkeiten beziehen sich auf das im Prototyp verwendete Datenbankbetriebssystem Oracle 8 und entspricht dem aktuellen Standard SQL99.

5 Bewertung und Ergebnis

5.1 Zusammenfassung

Zusammenfassend stellt der Autor dieser Arbeit fest, dass die Entwicklung des webbasierten Auskunftssystems für Schienenfahrzeuge zu einem lauffähigen und funktional umfangreichen Prototyp geführt hat. Die Entwicklung der Software dauerte etwas über drei Monate, wobei etwas über die Hälfte der Zeit für die Datenaufbereitung und –validierung verwendet werden musste (s.a. 5.2). Der eigentliche Entwurf und die Implementierung des Systems kosteten am Ende merklich weniger Zeit, was auf die guten Arbeitsbedingungen, die Unterstützung durch die eingesetzte moderne Entwicklungsumgebung und die aus der ETC wiederverwendeten Komponenten zurückzuführen ist.

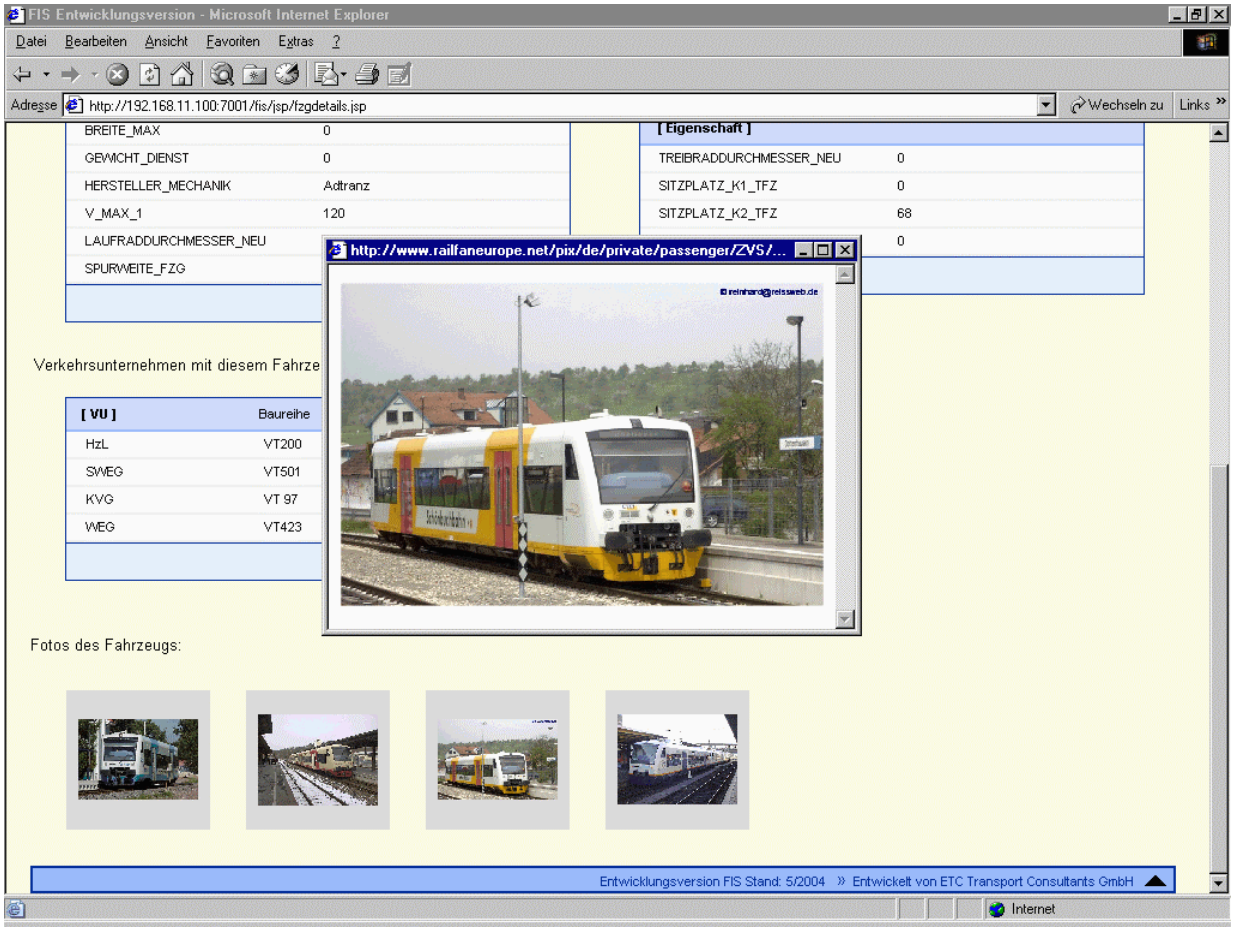

Abbildung 33: Screenshot des Auskunftssystems

Ergebnis der Entwicklung ist nicht nur ein sehr gutes Beispiel für die Entwicklung einer modernen Webapplikation sondern auch eine Software, die den gestellten Diskursbereich Auskunftssystem für Schienenfahrzeuge bereits als Prototyp sehr gut umsetzt.
Im Rahmen einer Präsentation wurde das System mit den bereits realisierten Funktionen den Mitarbeitern der ETC mit fachlichem Bezug und weiteren an der Entwicklung beteiligten Personen demonstriert. Hierbei zeigte sich sehr deutlich, welche Vorteile das Auskunftssystem gegenüber den bisher verwendeten Recherchemethoden über Excel-Dateien und ausgedruckte Listen bietet. Die Anwendung ermöglicht letztendlich nicht nur ein schnelles Auffinden von Fahrzeugen oder Unternehmen, sondern bietet auch die Möglichkeit, schnell Querverweise zu weiteren Informationen zu finden („Wer setzt genau diesen Fahrzeugtyp noch ein?“) und gedankliche Brücken in der Datenbasis zu bauen.

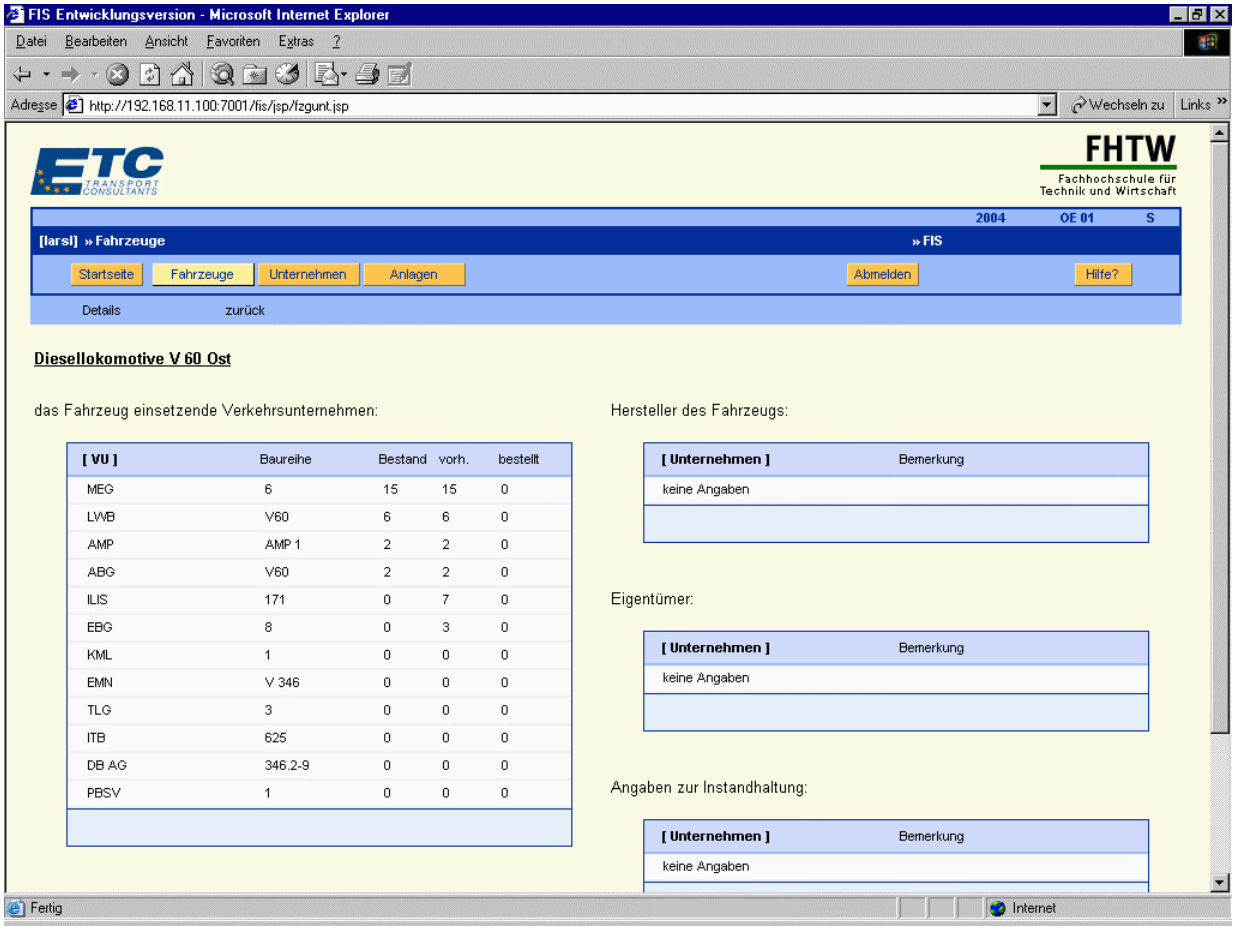

Abbildung 34: Analyse des Fahrzeugeinsatzes mit Hilfe des Auskunftssystems

5.2 Aufgetretene Probleme

Datenaufbereitung

Wie bereits in 5.1 festgestellt, wurde ein Großteil der für die Entwicklung zur Verfügung stehenden Zeit für die Aufbereitung und Übernahme der Daten verbraucht. Dies ist jedoch bei der Entwicklung eines Auskunftssystems durchaus üblich beziehungsweise nötig. [Dumke03, S.245ff.] nennt beispielsweise die spezifischen Anforderungen an Informations- und Datenbanksysteme und räumt dabei der effizienten Datenhaltung und vor allem der Datenkorrektheit Priorität ein.
Im Einzelnen traten bei der Datenaufbereitung folgende Probleme auf:

Tabelle 36: Aufgetretene Probleme bei der Datenaufbereitung

Gruppieren der Fahrzeuge schwierig
Die zu Beginn des Projekts als Excel-Datei vorliegenden Daten bestanden aus unüberschaubaren, nicht normalisierten Listen, die bereits konkrete Zuordnungen von Fahrzeugen zu Unternehmen enthielten. Baugleiche Fahrzeuge erschienen hierbei mehrfach. Zur Normalisierung der Daten mussten diese gleichen Fahrzeuge erkannt und zusammengefasst werden, was jedoch insbesondere bei Fahrzeugen mit fehlenden oder nur sehr wenigen technischen Angaben schwierig war. Nötig waren deswegen nicht nur komplexe Datenbankabfragen zur Gruppierung sondern auch Plausibilitätsprüfungen „von Hand“.

H:\s0379692\Uni\diplom\Teile\gruppieren.sql

```
-------------- Top of File --------------
SELECT Last(fzg_temp_zweiterversuch.ID_FZG) AS [LetzterWert von ID_FZG],
First(fzg_temp_zweiterversuch.ID_FZG) AS [ErsterWert von ID_FZG]
INTO abbildung
FROM fzg_temp_zweiterversuch, triebfahrzeug_temp
WHERE (((fzg_temp_zweiterversuch].[ID_FZG_NEU]) IS NULL))
GROUP BY triebfahrzeug_temp.ID_ANTRIEB, triebfahrzeug_temp.ID_STROMSYSTEM,
triebfahrzeug_temp.ID_TFZGART, triebfahrzeug_temp.DREHGESTELLACHSSTAND_TFW,
triebfahrzeug_temp.HOEHE_FUSSBODEN_TFZ, triebfahrzeug_temp.HOEHE_EINSTIEG_UE_SO_TFZ,
triebfahrzeug_temp.BREITE_EINSTIEG_TFZ, triebfahrzeug_temp.VOLUMEN_KRAFTSTOFF,
triebfahrzeug_temp.VERBRAUCH_KRAFTSTOFF, triebfahrzeug_temp.ANHAENGELAST,
triebfahrzeug_temp.ANFAHRZUGKRAFT, triebfahrzeug_temp.LEISTUNGSKLASSE,
triebfahrzeug_temp.STUNDENLEISTUNG, triebfahrzeug_temp.HERSTELLER_MOTOR,
triebfahrzeug_temp.HERSTELLER_GETRIEBE, triebfahrzeug_temp.KRAFTUEBERTRAGUNG,
triebfahrzeug_temp.A_MAX, triebfahrzeug_temp.TREIBRADDURCHMESSER_NEU,
triebfahrzeug_temp.SITZPLATZ_K1_TFZ, triebfahrzeug_temp.SITZPLATZ_K2_TFZ,
triebfahrzeug_temp.STEHPLATZ_K1_TFZ, triebfahrzeug_temp.BEMERKUNG_TFZ,
fzg_temp_zweiterversuch.HERGESTELLT_SEIT, fzg_temp_zweiterversuch.DREHGESTELLBAUART,
fzg_temp_zweiterversuch.PREIS, fzg_temp_zweiterversuch.DREHGESTELLACHSSTAND_LFW,
fzg_temp_zweiterversuch.DREHZAPFENABSTAND, fzg_temp_zweiterversuch.ACHSSTAND,
fzg_temp_zweiterversuch.GESAMTRADSTAND, fzg_temp_zweiterversuch.ACHSFOLGE,
fzg_temp_zweiterversuch.GEWICHT_DIENST, fzg_temp_zweiterversuch.VOLUMEN_SAND,
fzg_temp_zweiterversuch.ZUGHEIZUNG, fzg_temp_zweiterversuch.GEWICHT_EIGEN,
fzg_temp_zweiterversuch.HERSTELLER_MECHANIK, fzg_temp_zweiterversuch.HERSTELLER_MECHANIK,
fzg_temp_zweiterversuch.HERSTELLER_ELEKTRIK, fzg_temp_zweiterversuch.V_MAX_1,
fzg_temp_zweiterversuch.V_MAX_2, fzg_temp_zweiterversuch.LAUFRADDURCHMESSER_NEU,
fzg_temp_zweiterversuch.SPURWEITE_FZG
HAVING (((First([fzg_temp_zweiterversuch].[BAUREIHE]))<>"0")
AND ((Last([fzg_temp_zweiterversuch].[BAUREIHE]))<>"0")
AND ((triebfahrzeug_temp].[LEISTUNGSKLASSE])<>"0")
AND ((fzg_temp_zweiterversuch].[HERGESTELLT_SEIT])<>"0"));
```

Abbildung 35: SQL-Anweisung zum Gruppieren von baugleichen Fahrzeugen

Fehlerhafte Aufbereitung durch Dritte

Einige routinehafte Aufbereitungsschritte wurden durch externe studentische Hilfskräfte durchgeführt, die jedoch nicht in der Arbeit mit komplexen Datenbanken ausgebildet waren. Dies führte nicht nur zu einigen fehlerhaften Daten sondern auch zum irrtümlichen Verlust von Fremdschlüsselbeziehungen, die im Anschluss aufwendig durch komplexe Abfragen wiederhergestellt werden mussten.

Überarbeitung von Domänen

Die im Verlauf des Normalisierungsprozesses entstandenen Domänen mussten an vielen Stellen überarbeitet und angepasst werden. Oftmals standen datenbanktechnisch verschiedene Werte für den gleichen Sachverhalt. Beispielsweise mussten in der Tabelle der möglichen Stromsysteme für Triebfahrzeuge die Einträge „1 kV + 15 kV“, „1000 Volt, 15 kV“ und „1+15 kV“ auf nur einen Eintrag abgebildet und die entsprechenden Fremdschlüssel in der Triebfahrzeugtabelle angepasst werden.

Probleme bei der Wiederverwendung von Komponenten

Die von der ETC bereitgestellten, wiederverwendeten Komponenten (vgl. 4.3.3) mussten an einigen Stellen überarbeitet werden. Beispielsweise wurde das List View Steuerelement bereits in einem anderen Projekt zur Verwaltung von Nahverkehrslinien eingesetzt und enthielt verschiedene projektspezifische Details wie etwa das Anzeigen eines Liniensymbols. Auch waren in den Menü-Steuerelementen viele mit der Linienverwaltung zusammenhängende Begriffe fest kodiert und mussten „von Hand“ gesucht und ersetzt werden.

5.3 Ausblick und mögliche Erweiterungen

Ein entscheidendes Kriterium bei der Bewertung einer Software ist deren Flexibilität und Erweiterbarkeit. Das entwickelte Auskunftssystem ist in vieler Hinsicht erweiterbar und an neue Einsatzbereiche anpassbar.

Erweitern des Prototyps

Der bei der Entwicklung entstandene Prototyp besitzt bereits eine umfangreiche Funktionalität. Trotzdem wurden aus Zeit- und Effizienzgründen nicht alle Dialoge, Suchfunktionen und Datensichten realisiert. Beispielsweise unterscheidet sich die Suche nach einem Hersteller von Fahrzeugen von der Suche nach einem Instandhalter nur durch eine unterschiedliche XML-Beschreibung der Suchkriterien. Nach Festlegen der Suchkriterien durch fachliche Mitarbeiter der ETC müssten die entsprechenden XML-Beschreibungen erstellt und die Suchen in die dazugehörigen Dialoge eingebaut werden.
Diese und andere Funktionalitäten sind vergleichsweise einfach durch „Fleißarbeit“ zu realisieren. In einem ersten Erweiterungsschritt könnten folglich diese noch fehlenden Funktionen umgesetzt werden:

Tabelle 37: Noch nicht umgesetzte Funktionalitäten

- Detailsuchen nach Unternehmen und Unternehmensbereichen
- Detailsuchen nach Personen- und Güterwagen
- Auswahl von bzw. Suche nach Testanlagen
- Zuordnung von Testanlagen zu Verkehrsunternehmen

Daneben ist es denkbar, den Prototypen durch Funktionalitäten zu erweitern, die die Praxistauglichkeit erhöhen sowie die Anwendung einer größeren Zielgruppe zugänglich machen. Einige denkbare Erweiterungen seien hier aufgezählt:

Druckausgaben und andere Medien

Der Prototyp bietet bisher keine explizite Möglichkeit der Druckausgabe oder der Erzeugung von Printmedien. Ein teilweise zufrieden stellender Ausdruck ist zwar bisher über die Druckfunktionalität des eingesetzten Browsers möglich, denkbar wäre jedoch eine Erweiterung um folgende Funktionalitäten:

Tabelle 38: Erweiterungen um Druckausgaben und Exportmöglichkeiten

- Druck von Fahrzeuglisten
- Druck von Datenblättern zu einzelnen Fahrzeugen
- Druck von Fahrzeugbestandslisten
- Druck einer „Visitenkarte“ zu einem Unternehmen
- Erzeugen von PDF-Typenblättern zu Fahrzeugen
- grafische Visualisierung beispielsweise der Fahrzeugtyp-Verteilung in einem Unternehmen oder von Bestandslisten
- Export von Daten in Fremdformate

Die nachfolgende Abbildung zeigt eine mögliche Erweiterung der Anwendung um eine grafische Darstellung, bei welchen Unternehmen ein Fahrzeug in welchen Stückzahlen vorhanden ist.

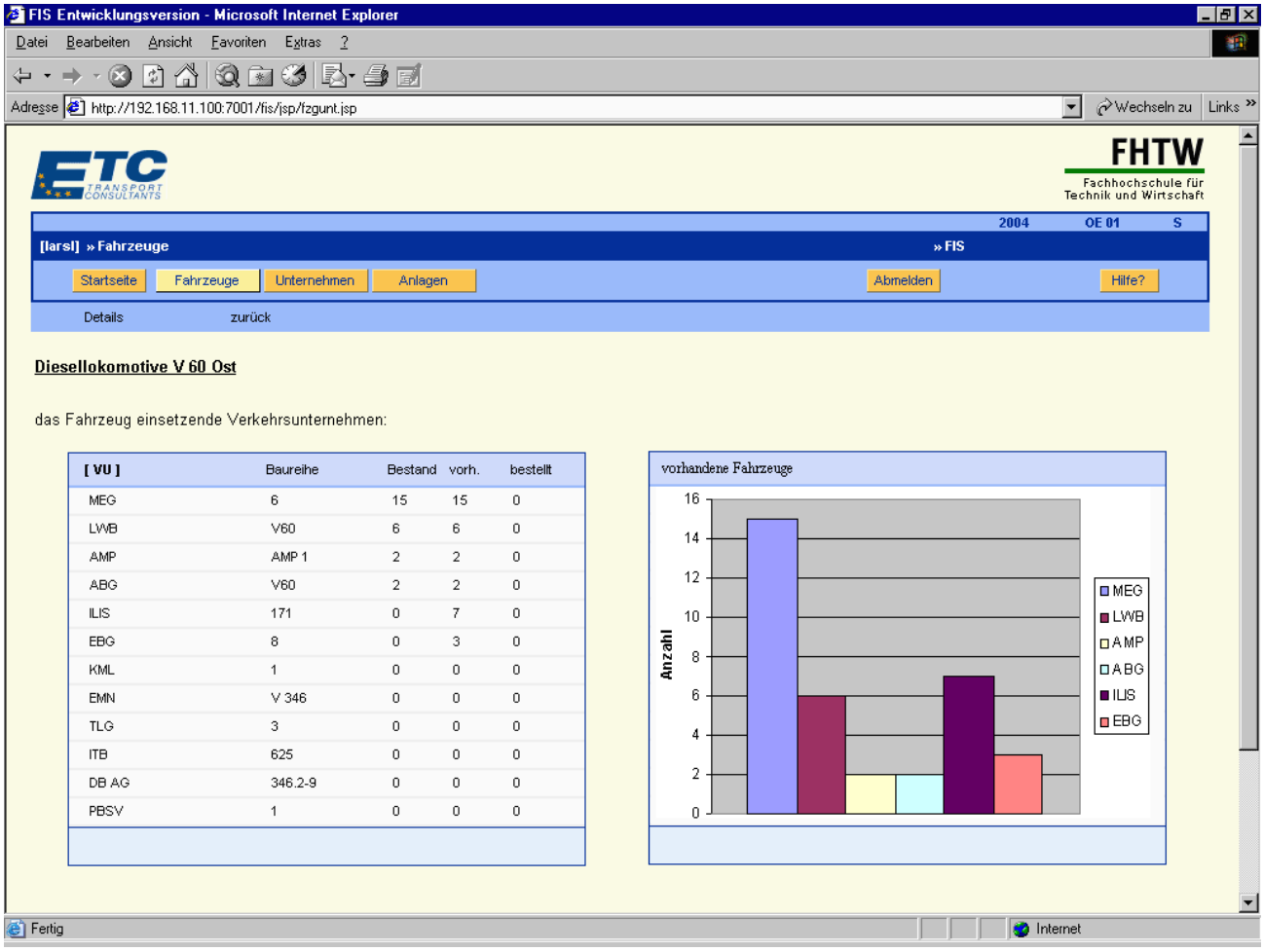

Abbildung 36: Mögliche Erweiterung um grafische Darstellungen

Diese Erweiterungen ermöglichen einen flexiblen Einsatz des Auskunftssystems vor allem in Hinblick auf Wünsche von Kunden der ETC. So könnten beispielsweise Verkehrsunternehmen für die Teilnahme an Ausschreibungen von Verkehrsleistungen gedruckte Listen möglicher Fahrzeuge erhalten.

Datenmanipulation

Die in der Datenbasis enthaltenen Informationen zu Fahrzeugen und Unternehmen unterliegen einer ständigen Änderung. Neue Fahrzeuge werden entwickelt und von Unternehmen eingesetzt, Altbaufahrzeuge werden ausgemustert oder weiter verkauft. Die Datenbank verliert folglich binnen kurzer Zeit an Aktualität.

Eine wichtige Erweiterung bestünde also in einer Möglichkeit, die vorhandenen Daten zu aktualisieren, neue Datensätze aufzunehmen oder nicht mehr aktuelle Daten zu löschen. Dies kann sowohl das Einspielen einer komplett neuen, extern bezogenen und aufbereiteten Datenversion als auch das einzelne Manipulieren von Datensätzen bedeuten.

Die Erweiterung um Datenmanipulationsfunktionen erfordert eine Reihe von Änderungen und Ergänzungen im Entwurf, etwa zur Behandlung konkurrierender Änderungen, Versionsverwaltung und Schreibrechten von Nutzern.

Internetversion

Eine eingeschränkte Version des Auskunftssystems könnte im Internet verfügbar und somit einer breiten Öffentlichkeit zugänglich gemacht werden.
Die folgende Abbildung zeigt eine mögliche Internetversion der Anwendung.

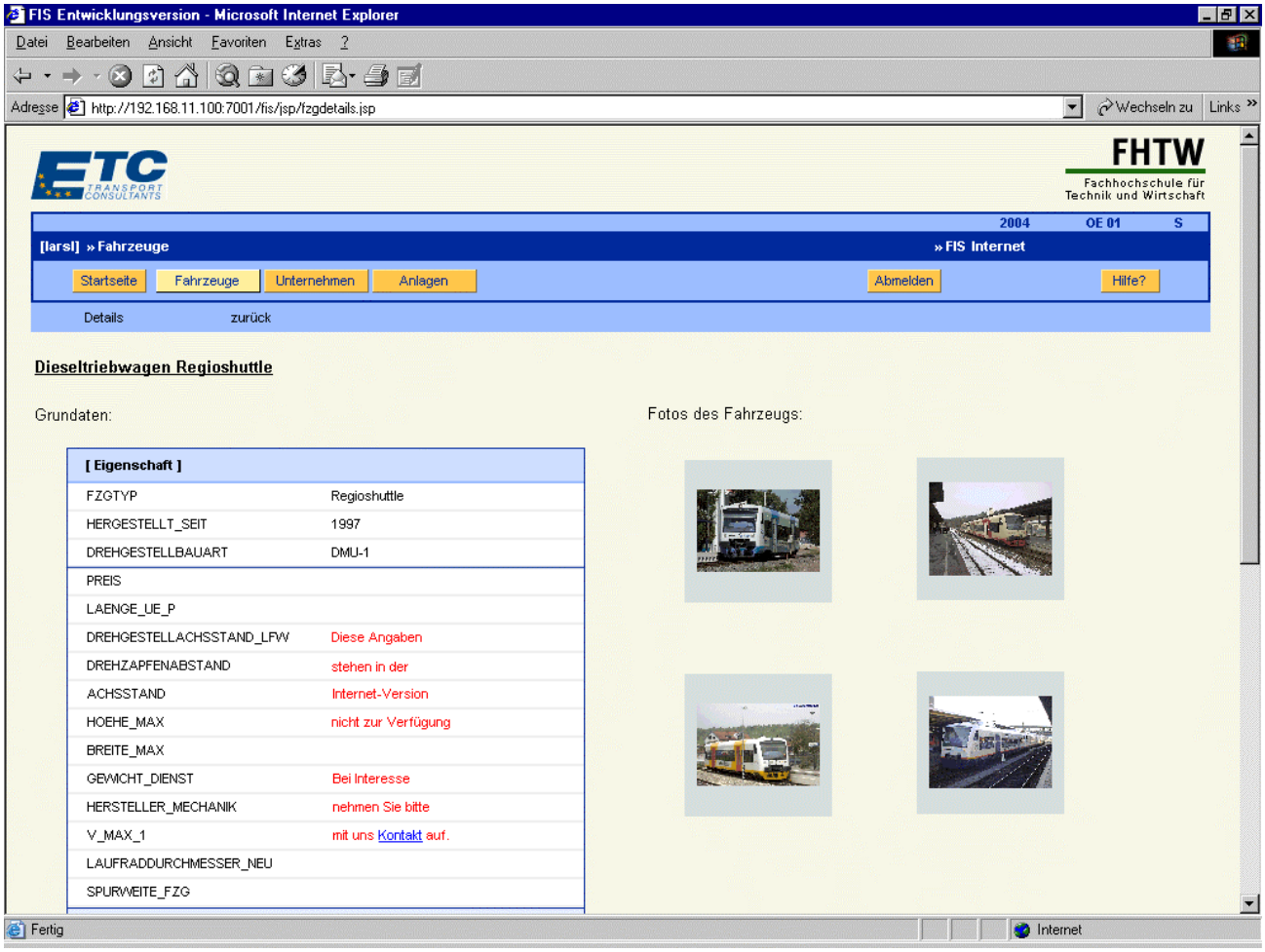

Abbildung 37: Eingeschränktes Auskunftssystem zur Verwendung im Internet

Hierbei würden der Großteil der in der Datenbasis enthaltenen Informationen ausgeblendet werden, um eine Verwendung dieser durch konkurrierende Unternehmen mit Bezug zum Schienenverkehr zu verhindern. Eine Auswahl von Informationen bietet jedoch großen Nutzen für Anwender wie Eisenbahnfans, Fahrgäste und anderen Nutztern mit allgemeinem Interesse an Schienenfahrzeugen und Verkehren.

5.4 Fazit

Nach Anfertigung dieses Dokuments und der Präsentation der Anwendung in der ETC wurde die Bearbeitung des gestellten Themas erfolgreich abgeschlossen.
Auf der einen Seite hat ein klar strukturierter und gut geplanter Software-Entwicklungsprozess zu einer praxistauglichen und modernen Internetanwendung geführt.
Auf der anderen Seite konnte der Verfasser durch die Entwicklung und die Arbeit in der ETC persönlich viele Erfahrungen sammeln. Im praktischen Studiensemester hatte der Autor bereits an der Entwicklung eines Auskunftssystems mit klassischen Herangehensweisen (C, C++, Windows-Anwendung) mitgewirkt, und in der Projektarbeit des siebten Semesters arbeitete er in einem Entwicklungsteam für eine gastronomische Geschäftsanwendung mit Open Source Technologien. Die Entwicklung einer Internetanwendung mittels kommerzieller Anwendungsserver und moderner Komponententechnologie bedeutete daher eine logische und ernorme Erweiterung seiner Kenntnisse. Auch war das Auskunftssystem für den Autor die erste praktische Verwendung der EJB-Technologie, die ihm in der Vorlesung noch als „Zukunftsmusik“ dargestellt worden war.
Insgesamt verlief die Arbeit ohne nennenswerte Probleme und Verzögerungen und bot einen guten Einstieg ins Berufsleben.

Anhang

Pflichtenheft

Pflichtenheft

Fahrzeuginformationssystem FIS

A1 Zielbestimmung

Zu erstellen ist ein Fahrzeuginformationssystem (FIS).
Die Software ist ein Auskunftssystem zur Abfrage der in der ETC Transport Consultants GmbH vorhandenen Schienenfahrzeugdatenbank. Sie ermöglicht eine Suche über Fahrzeuge und deren technische und wirtschaftliche Details und bietet Abfragemöglichkeiten über Unternehmen mit Bezug zum Schienenverkehr wie z.B. Verkehrsunternehmen und Fahrzeughersteller.
Das Auskunftssystem ermöglicht eine Abfrage, jedoch keine Manipulation der Daten.

A1.1 Musskriterien

Die Anwendung muss die Recherche gemäß nachfolgender Anforderungen über Fahrzeuge, Unternehmen und damit zusammenhängender Daten ermöglichen.

A1.1.1 Prinzip der Recherche

Die Anwendung FIS ermöglicht mit wenigen Abfragen und Suchen eine gezielte Recherche in der Fahrzeugdatenbank. Von den Rechercheergebnissen gibt es sinnvolle Querverweise zu dazu gehörigen Daten.
Eine Recherche ist möglich über Unternehmen mit Bezug zum Schienenverkehr (Verkehrsunternehmen, Hersteller, Eigentümer, Instandhalter), Fahrzeugen (Triebfahrzeuge, Personen- und Güterwagen) sowie Testanlagen für Schienenfahrzeuge.

A1.1.2 Suche nach Unternehmen

Alle in der Datenbasis enthaltenen Unternehmen werden aufgelistet. Über Filter und Stichwortsuchen kann nach allgemeinen Kriterien ein Unternehmen schnell gefunden werden. Weiterhin sollen für die vier Unternehmensarten Detailsuchen zur Verfügung stehen, die eine von der Unternehmensart abhängige ausführliche Suche nach bekannten Unternehmensdaten wie beispielsweise Jahresumsatz, Personenkilometer oder der Größe des Fahrzeugpools ermöglicht.
Zu einem so gefundenen Unternehmen soll es möglich sein, die zu dem Unternehmen gehörigen Fahrzeuge strukturiert zu suchen und die Fahrzeugdetails einzusehen.

A1.1.3 Suche nach Fahrzeugen

Alle in der Datenbasis vorhandenen Fahrzeuge sind aufgelistet. Über Filter und Stichwortsuchen kann nach allgemeinen Kriterien ein Fahrzeug schnell gefunden werden. Weiterhin sollen für die drei Fahrzeugarten Triebfahrzeug, Güter- und Personenwagen Detailsuchen zur Verfügung stehen, die eine differenzierte Suche mittels bekannter Daten des Fahrzeugs wie z.B. Stromsystem, Sitzplatzkapazität oder Lademenge ermöglicht.

Zu einem gefundenen Schienenfahrzeug sollen Informationen zu verknüpften Unternehmen (einsetzendes Verkehrsunternehmen, Hersteller, Eigentümer, Instandhalter) und sämtliche Daten des Fahrzeuges abrufbar sein.

A1.1.4 Suche nach Testanlagen

Alle in der Datenbasis enthaltenen Anlagen zum Testen von Schienenfahrzeugen werden aufgelistet. Filter und eine Stichwortsuche ermöglichen ein schnelles Finden einer Testanlage. In einer Detailsuche soll es möglich sein, ausführlich nach verschiedenen Kriterien wie der Anzahl der testenden Unternehmen, vorhandene Stromsysteme etc. eine Testanlage zu finden.
Zu gefundenen Testanlagen sind hiernach die zur Anlage gehörigen technischen und wirtschaftlichen Daten sowie auf der Anlage testende Unternehmen recherchierbar.
Zu den auf der gefundenen Testanlage testenden Unternehmen sollen in einem weiteren Verweis die Daten des Unternehmens und die zugehörigen Fahrzeuge ermittelt werden können.

A1.2 Wunschkriterien

A1.2.1 Sicherheit

Die Übertragung der Daten ist mit einer geeigneten Verschlüsselungstechnologie vor Fremdzugriff zu sichern.

A1.2.2 Versionierung

In der Datenbank erfolgt eine Versionierung der Datenbestände. Diese Versionierung ist vorerst nicht im Auskunftssystem abrufbar. Die Recherche erfolgt über die jeweils neuesten Daten. Eine Möglichkeit, auch in früheren Datenständen zu recherchieren, wäre jedoch wünschenswert.

A1.3 Abgrenzungskriterien

Die Software ermöglicht vorerst keine Manipulation der Daten oder das Erstellen von neuen Datensätzen und Beziehungen. Die Eingabe und Pflege der Daten geschieht durch externe Software, direkte Datenbankmanipulation sowie Import auf Datenbankebene.

A2 Produkteinsatz

A2.1 Anwendungsbereich

Die Software wird vorrangig im Intranet der ETC Transport Consultants GmbH eingesetzt. Es ist jedoch angedacht, die Applikation auch extern z.B. Kunden über das Internet zur Verfügung zu stellen.

A2.2 Zielgruppen

Zielgruppe der Anwendung Fahrzeuginformationssystem sind die Ingenieure der ETC Transport Consultants GmbH. Erforderlich sind Grundkenntnisse in der Bedienung einer Onlineanwendung sowie zur Nutzung der Ergebnisse entsprechende fachliche Voraussetzungen aus dem Bereich des Schienenverkehrs.

A2.3 Betriebsbedingungen

Die Serverkomponente der Anwendung läuft unbeaufsichtigt im 24h-Betrieb auf einem Applikationsserver der ETC.

A3 Produktumgebung

A3.1 Software

A3.1.1 Server:

Die Anwendung wird als Webapplikation unter Verwendung der J2EE Technologie
erstellt, wobei die Daten in einer relationalen Oracle Datenbank gehalten werden. Erforderlich ist ein dafür ausgelegter Webserver mit EJB-Container und ein Oracle Datenbankserver ab Version 8.
Durch die Verwendung von Java ist die Anwendung plattformunabhängig. Die Anforderungen an das jeweilige Betriebssystem ergeben sich aus denen des Applikationsservers.
Der Einsatz in der ETC erfolgt auf einem mit BEA Weblogic Version 7 und Oracle 8.1.7 ausgestatteten Server.

A3.1.2 Clients:

Auf den Client-Rechnern ist ein installierter Standard-Webbrowser nötig (Microsoft Internet Explorer ab Version 5.0).

A3.2 Hardware

A3.2.1 Server:

Die Anforderungen an die Hardware ergeben sich aus denen der installierten Serversoftware (Applikationsserver und Oracle Datenbank)

A3.2.2 Client:

Es wird ein Standard-PC oder vergleichbarer Rechner benötigt, auf dem die Installation und der Betrieb des Microsoft Internet Explorer ab Version 5.0 möglich ist.

A3.3 Orgware

Für die Nutzung des Auskunftssystems im Intranet ist die Anbindung von Server und Clients an ein LAN erforderlich.

A3.4 Schnittstellen

In der ersten Version sind keine Schnittstellen zu anderen Anwendungen vorgesehen.

A4 Produktfunktionen

A4.1 Suche ausgehend von Unternehmen

Es erfolgt eine Auswahl bzw. eine Suche nach Unternehmen. Neben einer Liste aller im Datenbestand befindlichen Unternehmen ermöglichen Detailsuchen die gezielte Recherche ausgehend vom Unternehmenstyp. Nach Wahl des Unternehmens erscheint eine Liste mit allen Fahrzeugarten und der Anzahl der zu dem Unternehmen gefundenen Datensätze. Durch Auswählen der Fahrzeugart gelangt man zur Übersicht der Fahrzeugtypen und deren Anzahl. Zu jedem Fahrzeugtyp ist eine Detailseite mit allen Attributen des Fahrzeugs sowie eine Übersicht mit den zu dem Fahrzeug gehörenden Unternehmen und deren Detailinformationen abrufbar.
Somit werden folgende grundlegende Recherchen ermöglicht:

- zu einem Verkehrsunternehmen sind alle eingesetzten Fahrzeugtypen, deren Eigentümer, Baureihen und technische Daten anzuzeigen.
- zu einem Verkehrsunternehmen sind die Leitzentrale, die Betriebshöfe, die Instandhalter anzuzeigen
- zu einem Hersteller ist anzuzeigen, welche Fahrzeugtypen und Produktlinien er anbietet, seit wann diese gebaut wurden, welche technischen Eigenschaften wie Stromsystem, Antriebsart und Maximalgeschwindigkeit sie besitzen und welche Verkehrsunternehmen diese Fahrzeuge einsetzen

- zu einem Eigentümer sind alle im Besitz befindlichen Fahrzeuge, deren technische Details und die Fahrzeuge einsetzenden Verkehrsunternehmen anzuzeigen
- zu einem Instandhalter ist anzuzeigen, welche Verkehrsunternehmen dort Fahrzeuge welchen Typs instandhalten lassen

A4.2 Suche ausgehend von Fahrzeugen

Es erfolgt eine Auswahl eines Fahrzeuges aus dem Gesamtdatenbestand bzw. eine Suche anhand bestimmter Attribute. Nach erfolgter Auswahl ist eine Detailseite mit allen zum Fahrzeug gehörenden Details verfügbar. Weiterhin kann eine Übersicht darüber aufgerufen werden, in welchen Fahrzeugpools dieses Fahrzeug enthalten ist und welche Unternehmen am Einsatz des Fahrzeugs beteiligt sind. Von hier kann direkt zu den Details der entsprechenden Unternehmen gesprungen werden.
Folgende grundlegende Recherchen werden ermöglicht:

- zu einem Fahrzeugtyp sind die das Fahrzeug einsetzenden Verkehrsunternehmen sowie die vorgehaltene Anzahl, das Baujahr, der Instandhalter und weitere technische Details anzuzeigen
- zu einem Fahrzeugtyp sind der Hersteller und der Eigentümer anzuzeigen
- Suche nach Fahrzeugtypen bestimmter Breite und/oder Höhe
- Suche nach Fahrzeugtypen mit bestimmter Radsatzlast
- Suche nach Fahrzeugtypen innerhalb bestimmter Altersgrenzen
- Suche nach Triebfahrzeugen bestimmer Leistungsklassen
- Suche nach Fahrzeugtypen anhand Maximalgeschwindigkeit

A4.3 Suche über Testanlagen

Aus dem Gesamtdatenbestand erfolgt die Auswahl einer Schienenfahrzeug-Testanlage. Hiernach sind alle relevanten Daten zur Anlage einsehbar. Weiterhin kann zu einer Übersicht verzweigt werden, auf der alle auf der Anlage testenden Verkehrsunternehmen gelistet sind. Analog zu A4.1 kann zu den hierdurch gefundenen Unternehmen eine Seite mit Unternehmensdetails eingesehen und direkt recherchiert werden, welche Fahrzeugarten, -typen und Fahrzeuge mit ihnen verknüpft sind.

A5 Produktdaten

Die in der Anwendung recherchierbaren Daten sind in einer relationalen Datenbank auf einem Oracle Datenbankserver abgelegt. Die Datenbank enthält insbesondere Informationen zu folgenden Bereichen:

- Fahrzeuge: detaillierte Angaben, technische Details und Nutzinformationen zu Schienenfahrzeugen, d.h. Triebfahrzeuge, Personen- und Güterwagen
- Unternehmen: detaillierte Informationen über Unternehmen mit Bezug zum Schienenverkehr, d.h. Hersteller von Fahrzeugen, Verkehrsunternehmen, Instandhaltungsbetriebe usw.
- Testanlagen: Angaben zu verfügbaren Teststrecken und Systemen, Kunden
- verknüpfende Informationen zwischen o.g. Bereichen

A6 Nutzeroberfläche

A6.1 Layout, Corporate Design

Die Nutzeroberfläche wird unter Verwendung des Corporate Design der ETC GmbH erstellt. Die zu verwendenden Farben des Unternehmens sind HKS 05 und HKS 43. Bei der Erstellung des Layouts werden vorhandene Komponenten wie List Views, Combo Boxen und Message Boxen möglichst wiederverwendet und neue Komponenten an deren Bedienlogik und „Look and Feel“ angeglichen.

A6.2 Dialogstruktur

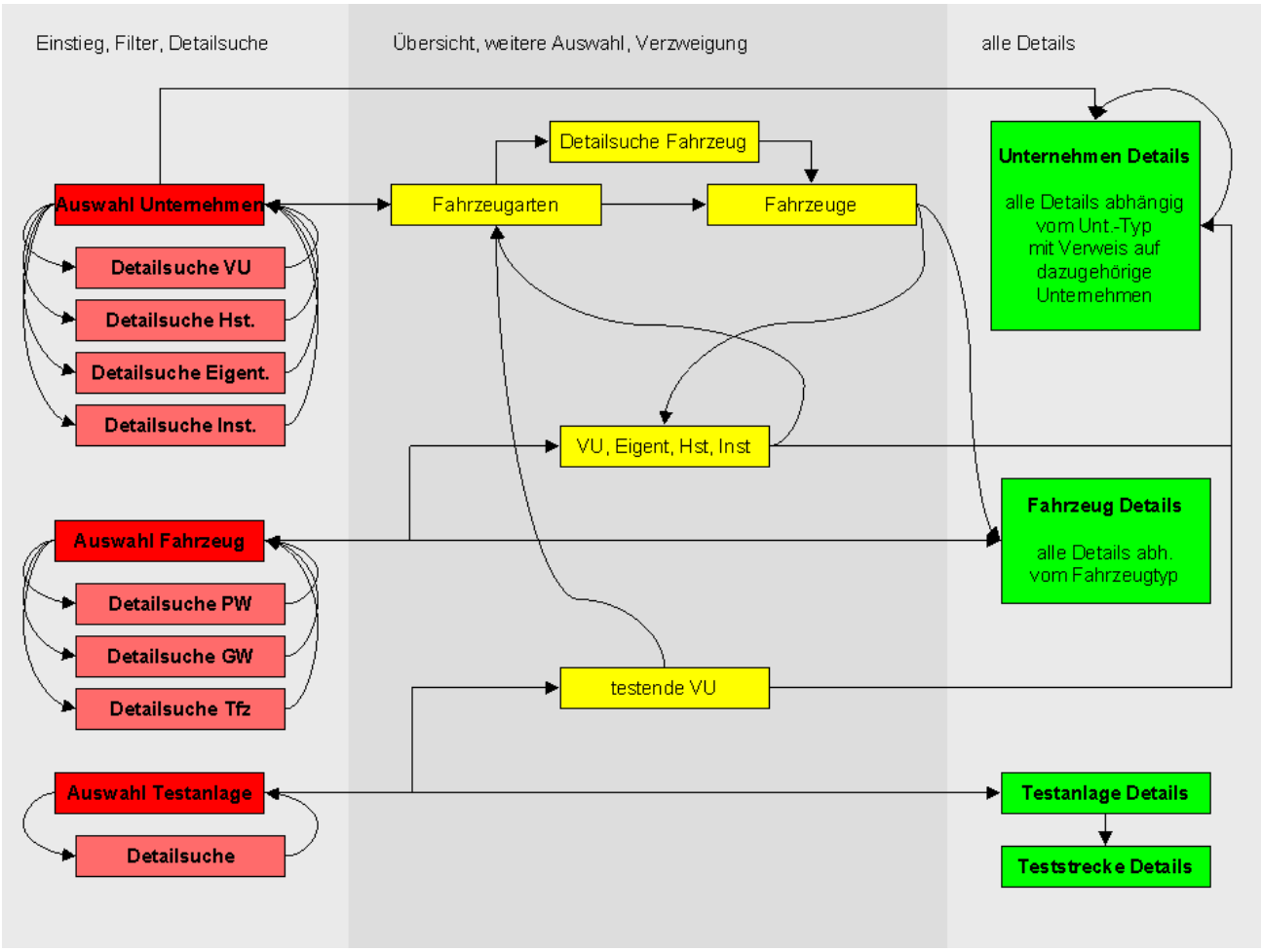

Abbildung A1: Dialogstruktur der Anwendung

A7 Qualitätsmerkmale

A7.1 Nutzerfreundlichkeit

Die Anwendung wird im Corporate Design der ETC realisiert. Aus vorhandenen Anwendungen werden Struktur- und Steuerelemente der Oberfläche übernommen. Dadurch fällt es den angestrebten Nutzergruppen auf Anhieb leichter, mit der neuen Software zu arbeiten. Die Verwendung eines Webbrowsers erfordert nur die für das Recherchieren im Internet notwendigen Vorkenntnisse und keine weitere Einarbeitungszeit.

A7.2 Portierbarkeit

Durch Verwendung von Java, insbesondere der J2EE-Technologie, ist die plattformunabhängig. Eine Portierung ist auf alle Systeme möglich, auf denen entsprechende Applikationsserver verfügbar sind.

A7.3 Wiederverwendbar- und Erweiterbarkeit

Die Software ist durch Komponententechnologie und einen objektorientierten Ansatz offen für einfache Erweiterungen wie beispielsweise Module zur Dateneingabe und –manipulation oder die Einbindung in weitere Auskunftssysteme.

A8 Entwicklungsumgebung

A8.1 Software

Die Entwicklung erfolgt unter Verwendung des Oracle JDeveloper 9.0.3 auf einem mit Microsoft Windows NT Version 4 ausgestatteten Arbeitsplatz und im LAN auf dem Server ETCBANK2.

A8.2 Hardware

Der Entwicklungsarbeitsplatz ist mit einem Pentium 4 (1,7 GHz) mit 512 MB Arbeitsspeicher und 20 GB Festplatte ausgestattet.

A8.3 Orgware

Für die Entwicklung auf dem Server und die Speicherung der Daten auf einem durch Backup gesicherten Netzlaufwerk ist ein LAN eingerichtet. Zur Dokumentation ist das Microsoft Office Paket installiert.

Verzeichnisse

Abkürzungen

API	Application Programming Interface
BMP	Bean Managed Persistence
CMP	Container Managed Persistence
COTS	Commercial Off The Shelf
DFN	Deutsches Forschungsnetz
DLL	Dynamic Link Library
EJB	Enterprise Java Beans
FIS	Fahrzeuginformationssystem
HTML	Hypertext Markup Language
IIOP	Internet Inter-Orb Protocol
J2EE	Java 2 Enterprise Edition
JDBC	Java DataBase Connectivity
JRMP	Java Remote Message Protocol
JSP	Java Server Pages
LAN	Local Area Network
LDAP	Leightweight Directory Access Protocol
PDF	Portable Document Format
PHP	Personal HomePage Tools
RMI	Remote Method Invocation
SQL	Structured Query Language
SSL	Secure Socket Layer
UML	Unified Modelling Language
XML	Extensible Markup Language

Literaturverzeichnis

[Abbey02] Abbey, M.; Corey, M. et al.:
Oracle 9i für Einsteiger
The McGraw-Hill Companies, 2002

[Balzert00] Balzert, H.:
Objektorientierung in 7 Tagen
Berlin: Spektrum, Akad. Verlag, 2000

[Bea04] Bea Systems:
Managing WebLogic Security
http://e-docs.bea.com/wls/docs70/secmanage/ssl.html
zuletzt besucht: 16. August 2004

[Boger99] Boger, M.:
Java in verteilten Systemen
Heidelberg: dpunkt-Verlag 1999

[Brown01] Brown, B.:
Oracle 8i Web Development
The McGraw-Hill Companies, 2000

[Brown98] Brown, A. W.; Wallnau, K. C.:
The Current State of CBSE
IEEE Software, 1998, S.37-46

[Dumke03] Dumke, R.:
Software Engineering
Wiesbaden: Vieweg & Sohn, 2003

[Farley03] Farley, J.; Crawford, W. et al.:
Java Enterprise in a Nutshell
Köln: O'Reilly Verlag, 2003

[Flanagan01] Flanagan, D.:
Java in a Nutshell
Köln: O'Reilly Verlag, 2001

[J2EE-Blue04] Sun Microsystems, Inc.:
Guidelines, Patterns, and code for end-to-end Java applications
http://java.sun.com/blueprints/patterns/
zuletzt besucht 23. Juli 2004

[J2EE-Tut04] Sun Microsystems, Inc.:
The J2EE 1.4 Tutorial
http://java.sun.com/j2ee/1.4/docs/tutorial/doc/index.html
zuletzt besucht 22. Juli 2004

[Kalis04] Kalis, Frank:
Die Entwicklungsgeschichte von SQL
http://www.insidesql.de/content/view/197/29/
zuletzt besucht 13. August 2004

[Klinger04] Klinger, C.; Segert, R.:
Das Online-ABC
http://www.webwunder.de/asp/abc.asp
zuletzt besucht 21. Juli 2004

[Kompf04] Kompf, M.:
Enterprise JavaBeans 2.1
Bonn: Galileo Press, 2004

[MySQL04] MySQL GmbH:
Der Datenbankserver MySQL
http://www.mysql.de/
zuletzt besucht 12. August 2004

[Petkovic02] Petkovic, D..; Brüderl, M.:
Java ins Datenbanksystemen
München: Addison Wesley Verlag, 2002

[Piemont99] Piemont, C.:
Komponenten in Java
Heidelberg: dpunkt-Verlag, 1999

[Reese00] Reese, G.:
Database Programming with JDBC and Java
Sebastopol: O'Reilly & Associates, Inc., 2000

[Roßbach00] Rossbach, P.; Schreiber, H.:
Java Server und Servlets
Addison Wesley Verlag, 2000

[Shirazi04] Shirazi, J.:
Java Performance Tuning
http://www.javaperformancetuning.com/
zuletzt besucht 23. Juli 2004

[Sommerv01] Sommerville, I.:
Software Engineering
München: Pearson Education Deutschland, 2001

[Stevens00] Stevens, P.; Pooley, R.:
UML – Softwareentwicklung mit Objekten und Komponenten
München: Pearson Education Deutschland, 2000

Abbildungen

Tabellen

Eigenständigkeitserklärung:

Hiermit versichere ich, dass ich die vorliegende Diplomarbeit selbstständig und nur unter Verwendung der angegebenen Quellen und Hilfsmittel verfasst habe. Die Arbeit wurde bisher in gleicher oder ähnlicher Form keiner anderen Prüfungsbehörde vorgelegt.