INSTITUT FÜR INFORMATIK

DER TECHNISCHEN UNIVERSITÄT MÜNCHEN

Diplomarbeit

Entwurf und Implementierung der WWW-Anwendung „Jahreswagenbörse“ bei der BMW AG

Nedo Haubelt

Aufgabensteller:	Prof. Dr. Heinz-Gerd Hegering
Betreuer:	Herr Rauber, BMW AG
	René Wies
	Stephen Heilbronner
Abgabedatum:	15. August 1995

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

München, den 15. August 1995

Inhaltsverzeichnis

Kapitel 1

Einleitung

1.1 Motivation

Das Internet, als weltweit größtes Computernetz [Wei93], wurde früher hauptsächlich zum Austausch elektronischer Nachrichten, zum interaktiven Zugriff auf entfernte Rechner sowie für den Transfer von Daten und Texten zwischen einzelnen Rechnern verwendet. Es birgt mittlerweile ein riesiges Potential an Daten und Informationen, die bisher größtenteils kostenlos zu Verfügung gestellt werden. Aufgrund neuer Internet-Dienste wie Gopher und World Wide Web (WWW), die mit Hilfe komfortabler graphischer Oberflächen einen bequemen direkten Zugriff auf Informationen in der ganzen Welt ermöglichen, ist in der letzten Zeit ein explosionsartiger Anstieg in der Nutzung des Internets zu verzeichnen. Ursprünglich im Bereich der Hochschulen verbreitet, erkennen heute auch immer mehr Firmen den Nutzen eines Anschlusses ans Internet [Wei93].

Einer der Gründe hierfür ist das WWW. Es ist nicht nur in Bezug auf die Anzahl der Benutzer, sondern auch in Bezug auf die Anzahl der Server, der Internet-Dienst mit den größten Wachstumsraten. Das WWW ist ein hypertext-basierter Internet-Dienst, der dem Benutzer die Möglichkeit bietet, auf weltweit vernetzte Informationen im Internet bequem zuzugreifen. Eine komfortable und einfach zu bedienende Benutzerschnittstelle ermöglicht das problemlose Navigieren im Internet, ohne grundlegende Kenntnis von der Art des Dienstes bzw. der Lokation der Objekte haben zu müssen. Für die gängigen Internet-Dienste wie FTP, Gopher, NEWS usw., stellt das WWW eine einheitliche Oberfläche zur Verfügung. Dies hat für den Benutzer den Vorteil, daß er nur noch eine Schnittstelle für alle gängigen Internet-Dienste kennen muß, um auf die im Internet verfügbaren Informationen zugreifen zu können. Zudem bietet das WWW die Möglichkeit, Gateways zu anderen Informationsquellen, wie beispielsweise Datenbanken und Dokumentenarchiven, zu entwickeln und in das WWW zu integrieren. Damit entwickelt sich das WWW zum globalen, interaktiven Informationssystem. Ein weiterer Grund für die hohen Wachstumsraten des WWW ist die Fülle an Software, die derzeit für alle gängigen Plattformen und Systemumgebungen größtenteils kostenlos verfügbar ist.

Neben diesen technischen Vorteilen ergeben sich auch wirtschaftliche Vorteile durch den Einsatz von WWW. Der Informationsanbieter kann mit verhältnismäßig geringem Aufwand Informationen für eine große Anzahl global verteilter Benutzer einfach und kostengünstig bereitstellen. Der Benutzer hat die Möglichkeit, die zur Verfügung gestellten Informationen bei Bedarf täglich aktuell und online abzurufen. Daneben können durch den Einsatz von WWW die Kosten für die Bereitstellung von Informationen gesenkt werden, da die bisher anfallenden Kosten für das Vervielfältigen und Verteilen von Informationen entfallen.

1.2 Analyse des Themenkreises

Im Zuge der zunehmenden Kommerzialisierung des Internets ist auch bei der BMW AG das Interesse an dessen Nutzungsmöglichkeiten gestiegen. Aus den oben genannten Gründen wird dabei besonderes Augenmerk auf den Internet-Dienst WWW gelegt. Um die im Internet angebotenen Dienste nutzen zu können, wird derzeit ein Internet-Zugang geschaffen, an dem auch ein WWW-Server seine Dienste nach außen anbieten wird.

Um als Informationsanbieter im WWW auftreten zu können, soll im Rahmen dieser Diplomarbeit untersucht werden, wie derzeit Btx-basierende sowie neue Informationsdienste der BMW AG in das WWW integriert werden können. Es soll zunächst ein „WWW-Development Guide“ für WWW-Anwendungen entwickelt werden, wobei unterschiedliche Anforderungen zu beachten sind.

Da sich die im World Wide Web zur Verfügung gestellten Informationen schnell ändern können, ist der Aufwand relativ hoch, die Informationen aktuell zu halten. Der Benutzer eines Informationssystems ist unter Umständen nicht nur daran interessiert, Dokumente abzurufen, die statisch auf dem Server liegen, sondern möchte mit dem Informationsanbieter interaktiv kommunizieren. Daraus ergibt sich die Anforderung, daß die angefragten Dokumente zum Zeitpunkt der Anfrage dynamisch generiert werden müssen. Um die Verwaltung der bereitgestellten Informationen zu vereinfachen, empfiehlt es sich daher, wie bei vielen anderen Informationsdiensten, diese in Datenbanken zu halten. Aus diesem Grund muß untersucht werden, welche Anbindungsmöglichkeiten für einen WWW-Server an diese Art von Informationsquellen bestehen. Im Einzelnen werden dabei die Anbindungsmöglichkeiten an die hierzu relevanten BMW-Datenbanken analysiert.

Um unerlaubte Zugriffe vom Internet aus auf das lokale Netz zu unterbinden, werden sog. „Firewalls“ am Internet-Zugang eingesetzt, um diesen entsprechend zu sichern. Da ein WWW-Server seine Dienste im gesamten Internet möglichst uneingeschränkt anbieten soll, müssen die Konsequenzen, die sich durch die Verwendung eines Firewalls für den Einsatz sowie den Betrieb eines WWW-Servers ergeben, untersucht werden.

Das WWW als multimedialer Informationsdienst ermöglicht neben der Übertragung von normalem Text auch die Übertragung von Bildern, Audio- und Videosequenzen, die in

WWW-Dokumente eingebettet sein können. Da diese Dokumente sehr große Datenmengen enthalten können, bewirkt das Aufkommen von WWW daher einen drastischen Anstieg der Netzlast im Internet. Der einzelne Informationsanbieter muß beim Aufbau seines Servers bedenken, daß der Vielzahl von Benutzern unterschiedliche Datenübertragungsraten zur Verfügung stehen und, daß Dokumente, die mit unnötigen multimedialen Elementen „aufgebläht“ sind, die Netzlast erhöhen und dadurch lange Wartezeiten erzeugen. Die Präsentation der einzelnen Dokumente wird somit unter anderem durch die möglichen Datenübertragungsraten auf dem Internet beeinflußt.

Anhand des entwickelten Development Guides soll dann der Protoyp der „BMW Exklusivbörse für Jahreswagen“, kurz „Jahreswagenbörse“ oder „JAWA“, als WWW-Anwendung entwickelt, implementiert und als Informationsdienst in das WWW integriert werden. Die Anwendung soll neben einer Benutzerschnittstelle auch die Anbindung an die relevanten BMW-Datenbanken enthalten, in denen die für den Informationsdienst notwendigen Daten, wie beispielsweise die angebotenen Jahreswagen, gespeichert sind. Die Grundlage für die Entwicklung der WWW-Anwendung wird dabei die Btx-basierende Anwendung „BMW Exklusivbörse für gebrauchte Automobile“ [BMW92] bilden.

Kapitel 2

Konzepte des World Wide Web

2.1 Einführung

Das World Wide Web, oftmals als WWW oder Web bezeichnet, stellt ein globales, verteiltes, hypertextbasiertes Informationssystem dar. Obwohl noch sehr jung hat es sich zum meistgenutzten Dienst im Internet entwickelt. Das WWW ist ein „Point and Click“ Hypertext-Informationssystem, das sich für den Benutzer wie eine riesige Ansammlung von Dokumenten und Seiten darstellt, die neben normalem Text auch multimediale Daten und Verweise auf andere Dokumente enthalten können [BLCGP93]. Mit Hilfe einfach zu bedienender Benutzeroberflächen ist es dem Benutzer möglich, im Web zu navigieren und auf weltweit verteilte Informationen zuzugreifen. Um die Funktionsweise dieses Internet-Dienstes besser verstehen zu können, werden in diesem Kapitel die dem WWW zugrundeliegenden Konzepte beschrieben.

2.2 Client/Server-Architektur

Das WWW basiert auf einer Client/Server-Architektur. Der WWW-Server verwaltet das Informationsangebot, stellt Informationen zur Verfügung, nimmt Dokumentenanfragen entgegen und schickt die angefragten Dokumente an die WWW-Clients zurück. Der WWW-Client, der auch als „WWW-Browser“ bzw. „Hypertext-Browser“ bezeichnet wird, ruft Dokumente vom WWW-Server ab und stellt sie dem Benutzer dar [Klu95a]. WWW-Server und WWW-Client verständigen sich dabei über das Hypertext Transfer Protocol (Kapitel 2.6).

2.3 Hypertext

Bei den vom WWW-Server zur Verfügung gestellten Dokumenten handelt es sich um Hypertext- bzw. Hypermedia-Dokumente. Hypertext ist normaler Text, der Verweise, sog. „Links“ bzw. „Hyperlinks“, auf andere Hypertext-Dokumente enthält, die beispielsweise nähere Erläuterungen oder weitere Informationen zu einem im Dokument vorkommenden Begriff enthalten [Wei93]. Hypertext-Dokumente sind in der Hypertext Markup Language (Kapitel 2.5) verfaßt. Die in den Dokumenten enthaltenen Hyperlinks generieren ein komplexes, weltweites, virtuelles Netz von Verbindungen zwischen den einzelnen Dokumenten. Der Leser dieser Dokumente kann den Links folgen, ohne zu wissen, auf welchem Server sich das gerade angezeigte Dokument befindet.

2.4 Hypermedia

Hypermedia ist Hypertext, der neben Verweisen auf andere Hypertext-Dokumente auch Hyperlinks auf Bilder, Ton- und Videosequenzen enthält [Wei93]. Das WWW wird somit zum multimedialen Informationsdienst, der es erlaubt, Bild, Ton, Video und beliebig weitere Dokumentenarten, wie beispielsweise Postscript und RTF, zu integrieren. Hypermedia kombiniert somit Hypertext und Multimedia.

Die Multimedia-Funktionalität ist eine Eigenschaft der WWW-Clients. Enthält ein WWW-Dokument multimediale Elemente, entscheidet der WWW-Client mit Hilfe des MIME-Mechanismus [BF93], ob ein externes Programm zur Anzeige der Daten verwendet werden soll oder nicht.

2.5 Hypertext Markup Language (HTML)

2.5.1 Überblick

Hypertext-Dokumente werden in der Hypertext Markup Language (HTML) verfaßt. HTML basiert auf SGML, der Standard Generalized Markup Language, einer ISO-Norm (ISO 8879) zur Definition von strukturierten Dokumententypen. HTML legt, entsprechend dem Grundgedaken von SGML, ausschließlich die logische Struktur und den Inhalt eines Dokuments fest.

Das konkrete Layout und somit die visuelle Darstellung für den Betrachter wird dem WWW-Client überlassen. Hypertext-Browser (WWW-Clients) interpretieren ein HTML-Dokument und stellen die Layout-Informationen, wie beispielsweise Seitenformat, Seitengröße, Abstand zwischen den Absätzen sowie Zeichensätze für Text und Überschriften, entsprechend den Wünschen des Benutzers dar.

HTML weist folgende Merkmale auf:

(a) HTML basiert auf SGML

(b) Mittels HTML lassen sich strukturierte Dokumente mit Überschriften verschiedener Ordnung, Aufzählungslisten, Absätzen und Texthervorhebungen erstellen

(c) HTML-Dokumente können Hyperlinks auf weitere Dokumente beliebigen Formats enthalten

(d) HTML-Dokumente können Bilder, die im laufenden Text erscheinen, sog. „Inline Images“, enthalten

HTML ist derzeit noch nicht standardisiert und liegt nur als Internet-Draft [BLC95] vor. Der gegenwärtige De-facto-Standard soll als HTML 2.0 normiert werden.

2.5.2 Dokumentenaufbau

HTML-Dokumente sind strukturierte Dokumente. Das bedeutet, daß ein Dokument nicht nur eine Folge von Zeichen ist, sondern eine strukturierte Zusammenstellung verschiedener Elemente, die gemeinsam ein Dokument ausmachen. HTML stellt Konstrukte bzw. Steueranweisungen bereit, mit denen man die Strukturelemente eines Dokumentes festlegt. HTML-Anweisungen, auch „Tags“ oder „HTML-Tags“ genannt, werden dabei in den darzustellenden Text eingebettet. Der Hypertext-Browser interpretiert diese HTML-Anweisungen und stellt daraufhin den Text entsprechend dar.

In der Regel beginnt eine HTML-Anweisung mit einem Start-Tag und endet mit einem End-Tag, wobei einige Anweisungen nur aus einem Start-Tag bestehen können. Start-Tags werden durch < und > , End-Tags durch </ und > begrenzt. Die Zeichenfolge zwischen < und > im Start-Tag bzw. zwischen </ und > im End-Tag gibt den Elementtyp der HTML-Anweisung an.

Wie im folgenden Beispiel dargestellt, wird jedes HTML-Dokument durch die HTML-Elemente <`HTML`> und </`HTML`> begrenzt. Das Dokument ist weiterhin zweigeteilt und besteht aus einem `HEAD`- und einem `BODY`-Element. Das `HEAD`-Element, das Bestandteil des Dokumenteskopfes ist, enthält Informationen über das Dokument. Dort darf, außer bestimmten HTML-Elementen, wie beispielweise dem `TITLE`-Element, kein normaler Text enthalten sein. Das `BODY`-Element, das den Dokumentenrumpf begrenzt, enthält den eigentlichen Inhalt eines Dokumentes. Innerhalb des `BODY`-Elementes können, neben dem eigentlichen Text, HTML-Elemente zur weiteren Strukturierung des Dokumentes enthalten sein.

Der Aufbau eines eines HTML-Dokumentes soll am folgenden Beispiel verdeutlicht werden:

```
<HTML> <HEAD>
<TITLE>Dies ist der Dokumententitel</TITLE>
</HEAD>
<BODY>
<H1>Dies ist eine Überschrift</H1>
Dies wird ein Paragraph <P>
<UL>
<LI>Hier steht das erste Listenelement
<LI>Hier steht das zweite Listenelement
</UL>
<A HREF=``Name des Links``>Dies ist ein Hyperlink</A> auf ein
anderes Dokument
</BODY> </HTML>
```

2.5.3 HTML+

HTML+ ist eine Erweiterung zu HTML, die sich derzeit noch in der Spezifikationsphase befindet und als HTML 3.0 [Rag95] normiert werden soll. HTML+ bietet gegenüber HTML wesentlich mehr Möglichkeiten der Textdarstellung. Grunsätzliche Neuerungen sind dabei:

- Mehr Möglichkeiten der Textdarstellung

  HTML+ bietet eine Vielzahl neuer Möglichkeiten Text darzustellen. Beispiel hierfür sind das Hoch- und Tiefstellen von Text, das Verwenden von mathematischen Formeln sowie das Einfügen von einfachen horizontalen Trennlinien um Text zu strukturieren.

- Darstellung von Tabellen

  Für die Darstellung von Tabellen, die unter HTML 2.0 mit Hilfe des Elements `<PRE>` dargestellt werden mußten, steht das HTML-Element `<TABLE>` zur Verfügung.

- Verwendung von Bildern als Hyperlinks

  Es werden sensitive Bilder, sog. „Inline Clickable Images“, unterstützt. Das bedeutet, daß Hyperlinks in Bilder und Graphiken integriert werden können. Ein Bild enthält dabei mehrere verschiedene Hyperlinks zu anderen Dokumenten. Je nachdem welche Koordinaten innerhalb des Bildes angewählt werden, wird zu dem entsprechenden WWW-Dokument verzweigt.

- Unterstützung von interaktiven Formularen

  Dies ist eine der wichtigsten Neuerungen von HTML+. Dadurch wird eine bessere

Interaktion zwischen Anwender und Informationsanbieter gewährleistet. Die WWW-Clients können dadurch beispielweise zum Online-Shopping verwendet werden oder als Benutzerschnittstelle für Datenbankanwendungen dienen.

Obwohl sich HTML 3.0 noch in der Spezifikationsphase befindet und noch nicht vollständig normiert ist, werden Teile davon bereits von einigen Browsern implementiert. Die neuesten Versionen der WWW-Clients von Netscape Communications sowie von NCSA beispielsweise implementieren bereits HTML-Tabellen und HTML-Formulare (Kapitel 3.2). Außerdem sind sie in der Lage horizontale Trennlinien darzustellen, um Text zu strukturieren. Doch nicht alle WWW-Clients haben bereits so viele HTML 3.0-Elemente implementiert. Der bildschirmorientierte WWW-Client Lynx beispielsweise implementiert zwar HTML-Formulare, doch ist er nicht in der Lage HTML-Tabellen darzustellen,

2.6 Hypertext Transfer Protocol (HTTP)

2.6.1 Überblick

Das Hypertext Transfer Protocol (HTTP) regelt die Kommunikation und Übertragung von Dokumenten zwischen dem WWW-Server und dem WWW-Client. Das HTTP-Protokoll ist einfach gehalten und belastet den WWW-Server nur minimal. Es handelt sich, im Gegensatz zum File Transfer Protocol (FTP), um ein zustandsloses Protokoll. Das bedeutet, daß sich der WWW-Server nicht merkt, was eine HTTP-Anforderung des Clients bewirkt hat. Der Server gibt nach dem Absenden der Antwort alle Resourcen, die zur Bearbeitung der Client-Anfrage nötig waren, wieder frei. Dies entlastet den Server gegenüber einem zustandsbehaftetem Protokoll wie FTP entscheidend.
Die Kommunikation zwischen Client und Server setzt für die Datenübertragung ein sicheres Transportprotokoll voraus. Die gegenwärtigen Implementierungen des HTTP-Protokolls unterstützen ausschließlich das im Internet verwendete TCP/IP, wobei andere Übertragungsprotokolle, die eine sichere Übertragung gewährleisten, in Frage kommen könnten. Die Übertragung der Daten erfolgt mit 8 Bit pro Zeichen. Als Zeichensatz wird dabei ISO Latin-1 verwendet.
Das HTTP-Protokoll ist wie HTML derzeit noch nicht standardisiert, sondern liegt als Internet-Draft [BLFN95] vor. Es kann aber davon ausgegangen werden, daß es als RFC verabschiedet wird. Die aktuelle Version ist `HTTP/1.0`, die zur Vorläuferversion `HTTP/0.9` kompatibel ist.

2.6.2 Transaktionsmodell

Der Ablauf einer HTTP-Transaktion ist in Abbildung 2.1 dargestellt und besteht aus folgenden Operationen:

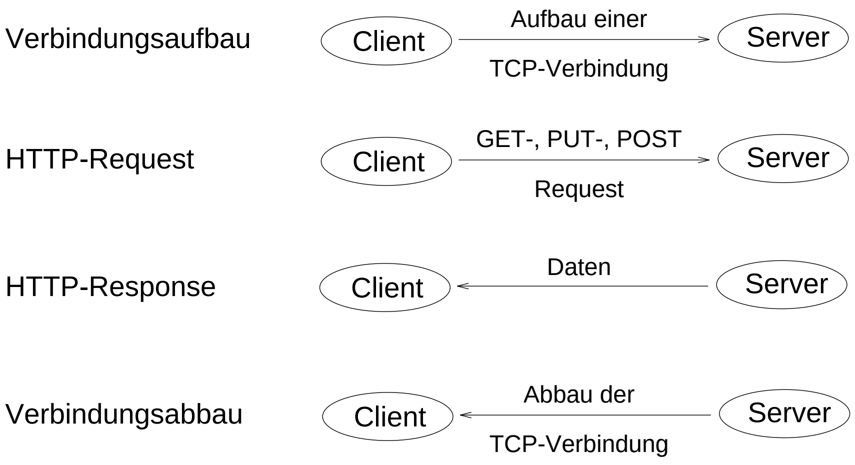

Abbildung 2.1: Ablauf einer HTTP-Transaktion

(a) Verbindungsaufbau

Der WWW-Client initiiert eine TCP/IP-Verbindung zum WWW-Server. Der Verbindungsaufbau erfolgt dabei über den „well known" Port 80 oder über jeden beliebigen, nicht reservierten Port, der dann durch den Client explizit angegeben werden muß.

(b) HTTP-Request

Nachdem eine Verbindung zwischen Client und Server aufgebaut wurde, stellt der WWW-Client eine Anfrage in Form eines HTTP-Requests an den Server. Es kann sich dabei um einen `GET`-, `POST`-, `PUT`- oder `HEAD`-Request handeln, wobei laut HTTP-Spezifikation weitere Anfragemethoden in Frage kommen können.

(c) HTTP-Response

Der WWW-Server sendet dem Client als Antwort auf die Anfrage entweder das angeforderte Dokument oder eine Fehlermeldung. Jede Antwort durch den Server enthält einen Statuscode, der dem Client mitteilt, ob ein Fehler aufgetreten ist und um welchen Fehler es sich dabei handelt.

(d) Verbindungsabbau

Nach Übertragung der Daten baut der WWW-Server bzw. der WWW-Client die Verbindung ab.

Jede Client-Anfrage wird über eine eigene Verbindung geschickt, die jeweils neu aufgebaut wird. Besteht ein Dokument aus mehreren Objekten, wird für jedes Objekt eine eigene Anfrage an den Server gestellt, da je HTTP-Request nur ein Objekt übertragen werden kann.

2.6.3 HTTP-Request

Im HTTP-Request gibt der Client an, welche Protokollversion er verwendet, welches Dokument er spezifiert und mit welcher Methode er es behandeln möchte. Man unterscheidet dabei zwischen dem *SimpleRequest* und dem *FullRequest*. Der *SimpleRequest* entspricht der Funktionalität von HTTP/0.9 , der *FullRequest* entspricht der von HTTP/1.0 .

Der *SimpleRequest* ist wie folgt aufgebaut:

```
GET URL <CR><LF>
```

URL steht für Uniform Ressource Locator (Kapitel 2.7) und enthält die genaue Lokation des angeforderten Dokuments. *Der SimpleRequest* erkennt nur die GET-Methode. <CR> steht für <Carriage Return>, <LF> steht für <Line Feed>.

Der *FullRequest* setzt sich wie folgt zusammen:

```
Method URL HTTP/1.0 <CR><LF>
[*<HTRQ Header>] [<CR><LF> <DATA>]
```

wobei als Method eine der folgenden Zugriffsmethoden verwendet werden kann:

- GET

  Diese Methode weist den Server an, das Objekt, welches durch die URL eindeutig spezifiziert ist, unabhängig vom Datenformat an den Client zu schicken.

- POST

  Mittels dieser Methode wird ein neues Objekt erzeugt, welches über einen Link zum spezifizierten Objekt in Beziehung steht und diesem untergeornet ist. Hiermit kann der Benutzer z.B. ein existierendes Objekt „Buch“, um ein neues Objekt „Kapitel“ erweitern. Der POST-Request wird zudem oftmals dazu verwendet, um die Benutzereingabe bei einem HTML-Formular an ein externes Programm auf dem WWW-Server zu übergeben (Kapitel 4.2.2).

Daneben werden im Standard weitere Methoden wie beispielsweise HEAD oder PUT spezifiziert, die aber im Rahmen dieser Diplomarbeit nicht näher betrachtet werden. HEAD beispielsweise weist den Server an, nur den HTTP-Header als Antwort zu schicken, wobei mit der Methode PUT die im Datenteil vorhandenen Daten vom Server unter der angegebenen URL gespeichert werden können.

URL steht, ebenso wie beim *SimpleRequest*, für Uniform Ressource Locator und enthält die genaue Lokation des angeforderten Dokumentes. Mit Hilfe der <HTRQ Header> (Hypertext Transfer Request Header) kann der Client dem Server zusätzliche Informationen zukommen lassen. Diese können beispielsweise sein:

- `From:`

  Dieses Feld gibt die Email-Adresse des anfragenden Benutzers an

- `Accept:`

  Anhand dieses Feldes kann der Client dem Server anzeigen, welche MIME-Typen er akzeptiert. Damit weiß der Server welches Datenformat der Client verarbeiten kann bzw. welches er an den Client schicken darf.

- `User-Agent:`

  Über diese Variable teilt der Client dem Server mit, welche Client-Software er verwendet. Wird diese Variable bei Zugriffen auf einen Informationsdienst in einer Log-Datei protokolliert, kann der Informationsanbieter beispielsweise erkennen, welche Client-Software verwendet wird, um dem von ihm angebotenen WWW-Dienst zu nutzen.

2.6.4 HTTP-Response

Die Antwort des Servers auf eine Client-Anfrage besteht aus einem Statuscode mit erläuterndem Text und weiteren Header-Informationen, die Informationen über den Server sowie Meta-Daten über das angeforderte Objekt enthalten. Danach folgen die eigentlichen Daten. Der HTTP-Response liegt folgende Syntax zugrunde:

```
<Status Line><CR><LF>
<Response Header><CR><LF>
<Data><CR><LF>
<Status Line> ::= <http version> <status code> <reason line>
```

Das folgende Beispiel, stell eine mögliche Antwort eines WWW-Servers dar.

```
HTTP/1.0 200 OK
MIME-Version: 1.0
Server: CERN/3.0
Date: Monday, 03-Apr-95 10:33:17 GMT
Content-Type: text/html
Content-Length: 311
Last-Modified: Monday, 06-Mar-95 15:10:13 GMT
```

Wie im obigen Beispiel dargestellt, enthält die Statuszeile der Antwort die HTTP-Version des Servers, einen dreistelligen Statuscode sowie eine kurze Erläuterung des Statuscodes. Der Statuszeile folgt der `Response Header`. Er enthält die MIME-Version, die Bezeichnung des Servers, das Datum der Transaktion, die MIME-Header `Content-Type` und

`Content-Length`, die den Typ und die Größe des Dokumentes anzeigen, sowie das Datum der letzten Änderung. In den meisten Fällen wird dafür der Zeitstempel der Datei auf der Platte verwendet. Zusätlich können weitere Informationen im Response Header enthalten sein. Nach dem Response Header folgen die eigentlich zu übertragenden Daten.

2.6.5 HTTP-Next Generation (HTTP-NG)

Weil bei Verwendung von HTTP als Kommunikationsprotokoll im WWW Performance-Probleme auftreten [Spe94], und weil sich HTTP nicht für den Einsatz im Bereich des elektronischen Handels eignet, wird das Bedürfnis nach einem neuen Protokoll immer größer. HTTP-Next Generation (HTTP-NG), der als Nachfolger von HTTP/1.0 angekündigt wird, soll HTTP/1.0 als Kommunikationsprotokoll im WWW ablösen [Spe95c].

HTTP-NG bietet neben einer wesentlich höheren Perfomance auch zusätzliche Eigenschaften, die von kommerziellen Applikationen benötigt werden [Spe95a]. Die höhere Performance gegenüber HTTP/1.0 wird durch ein verändertes Transaktionsmodell erreicht. Bei HTTP/1.0 wird für jede Anfrage eine neue Verbindung zum Server aufgebaut, was zu Performance-Problemen führt. Jeder Verbindungsaufbau nimmt eine Menge Zeit in Anspruch und verursacht eine hohe Last auf dem Netz. Nach erfolgtem Verbindungsaufbau schickt der Client seine Anfrage an den Server. Der Server anwortet über die gleiche Verbindung und schickt dem Client, wenn möglich die angeforderten Daten. Nach Übertragung der Daten wird die Verbindung wieder abgebaut. Um diese Probleme zu umgehen verwendet HTTP-NG ein verändertes Transaktionsmodell. HTTP-NG ermöglicht die Abwicklung mehrerer verschiedener Anfragen eines Clients über eine einzige Verbindung. Die Verbindung ist in mehrere virtuelle Kanäle aufgeteilt. Ein Kanal wird für die Übertragung von Kontrollinformationen verwendet, die anderen Kanäle dienen der Übertragung der angeforderten Objekte. Die Kommunikation zwischen Client und Server ist dabei asynchron, was bedeutet, daß der Client nicht mehr auf die Antwort einer Anfrage warten muß, bevor er eine neue Anfrage an den Server schicken kann.

Neben den Verbesserungen im Bereich der Performance enthält HTTP-NG zusätzliche Eigenschaften, die benötigt werden, um elektronischen Handel im WWW zu etablieren. HTTP-NG ermöglicht eine sichere Kommunikation zwischen Client und Server, die durch Authentifizierung und Datenverschlüsselung gewährleistet werden und unterstützt zusätzlich die Online-Abrechnung von getätigten WWW-Transaktionen.

HTTP-NG befindet sich derzeit noch in der Spezifikationsphase, doch es ist davon auszugehen, daß es zukünftig HTTP/1.0 ersetzen wird. Derzeit sind keine Implementierungen dieses Protokolls verfügbar.

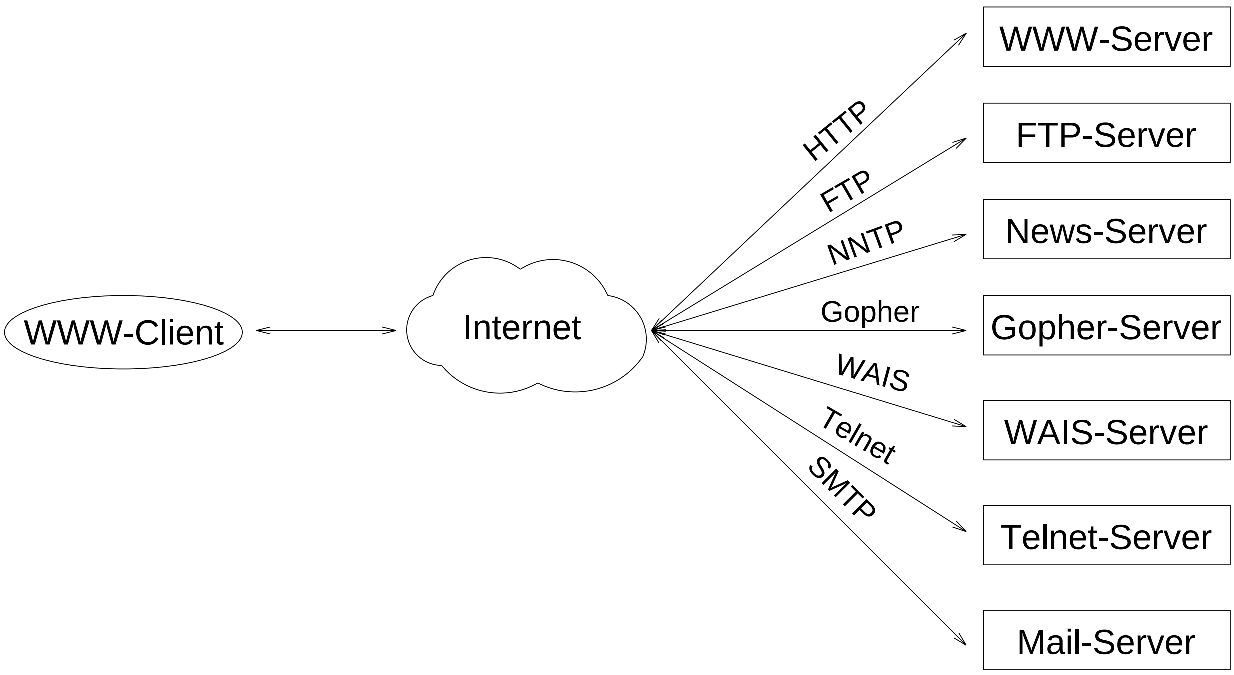

Abbildung 2.2: Intergration traditioneller Internet-Dienste im WWW

2.7 Uniform Resource Locator (URL)

2.7.1 Überblick

Der URL ist ein allgemeines Konzept zur einheitlichen Adressierung beliebiger Ressourcen im Internet. Eine Internet-Ressource ist beispielsweise eine Datei, ein News-Artikel oder eine Telnet-Sitzung auf einem entfernten Rechner, die über ein einheitliches Schema adressiert werden kann. Mit Hilfe dieses Konzeptes ist es einem WWW-Client nicht nur möglich mit einem WWW-Server über HTTP zu kommunizieren, sondern der Client kann Daten auf anderen Servern, die über FTP, Gopher, WAIS, NNTP usw., kommunizieren, abrufen. Dies bedeutet, daß die herkömmlichen Internet- Dienste wie FTP, Gopher, WAIS, NNTP, usw., wie in Abbildung 2.2 dargestellt, im WWW integriert sind [Klu95a].

URLs werden im WWW einerseits als Hyperlinks innerhalb von Hypertext-Dokumenten verwendet um auf verwandte Dokumente zu verweisen, und andererseits werden sie von WWW-Clients dazu verwendet, auf beliebige Ressourcen im Internet zuzugreifen. Die URL-Spezifikation wurde als RFC 1738 [BLMM94] verabschiedet.

2.7.2 Struktur einer URL

Die URL setzt sich aus einem allgemeinen Teil und einem protokollspezifischen Teil zusammen. Der allgemeine Teil beschreibt das Namensschema des zu adressierenden Objekts. Der protokollspezifische Teil bezeichnet das Objekt in einer bereichs- bzw. protokollspezifischen Syntax. Genauer gesagt, setzt sich eine URL aus folgenden Komponenten zusammen:

- Zugriffsmethode

  Die Zugriffsmethode bezeichnet das Namensschema bzw. die Methode, mit der auf ein Objekt zugegriffen wird (z.B. HTTP, FTP Gopher usw.). Die Zugriffsmethode wird durch „://“ vom Rechnernamen getrennt. Sie stellt den allgemeine Teil einer URL dar.

- Rechnername

  Der Rechnername kann in Form eines Domain-Namens oder einer IP-Adresse angegeben werden. Zusätzlich zum Rechnernamen kann eine Portnummer angegeben werden, falls diese von der im Protokoll vordefinierten Portnumer abweicht. Die Portummer wird dabei durch einen „:“ vom Rechnernamen getrennt.

- Verzeichnis- bzw. Dateiname

  Der Verzeichnis- bzw. Dateinname spezifiziert den vollständigen Pfad des Objektes lokal zum Rechner. Er wird dabei in protokollspezischer Syntax angegeben. Rechnername sowie Verzeichnis- bzw. Dateiname stellen den protokollspezifischen Teil einer URL dar.

Um die Stuktur einer URL zu verdeutlichen werden nun einige Zugriffsmethoden bzw. Namensschemata vorgestellt:

http: Zugriff auf Hypertext-Dokumente auf WWW-Servern

- http://home.mcom.com/home/welcome.html
- http://wwwcache.leo.org:8009/

ftp: Zugriff auf Dateien auf FTP-Servern

- ftp://ftp.germany.eu.net/pub/infosystems/www/Index

gopher: Zugriff auf Dokumente auf Gopher-Servern

- gopher://gopher.tu-clausthal.de:70/00/Gopher-de/Willkommen
- gopher://gopher.ask.uni-karlsruhe.de/

news: Zugriff auf News-Gruppen bzw. einzelne News-Artikel, die über den Gruppennamen bzw. die Message-ID spezifiert sind

- news:comp.infosystems.www.providers
- news:www/guide_792957573@rtfm.mit.edu

Beim Zugriff auf News-Gruppen und News-Artikel weicht die dazu verwendete URL von der normalen URL-Struktur ab. Der Zugriffsmethode `news:` folgt entweder die Bezeichnung einer News-Gruppe oder eines News-Artikels. Die Zugriffsmethode ist dabei nicht, wie gewohnt, durch „://“ von dem Bezeichner getrennt. Beim Zugriff auf ein News-Objekt wird der Rechnername nicht in der URL angegeben, sondern aus der Umgebungsvariablen `NNTPSERVER` ausgelesen.

2.7.3 Kodierung von Sonderzeichen

Innerhalb einer URL dürfen keine Sonderzeichen bzw. Zeichen, die eine Sonderbedeutung haben, auftreten. Aus diesem Grunde werden Sonderzeichen in der Form „%xx“ kodiert, wobei „xx“ für den ISO-8859-1 Code des Zeichens steht. Weiterhin hat das Pluszeichen eine Sonderbedeutung innerhalb einer URL. Es wird dazu verwendet, Leerzeichen zu kodieren, die beispielsweise bei der Eingabe von Formulardaten auftreten können.
Die Zeichenkette „Beispiel für Sonderzeichen“ wird beispielsweise innerhalb einer URL als „Beispiel+f%FCr+Sonderzeichen“ kodiert. Im weiteren Verlauf werden auf diese Weise kodierte Zeichenketten als „url-kodiert“ bezeichnet. Eine Tabelle der Sonderzeichenkodierung ist in [Klu94e] und [Klu94b] aufgeführt.

Die Kodierung von Sonderzeichen spielt beim Aufruf von externen Programmen (Kapitel, 4.2) eine Rolle. Sollen beispielsweise Informationen, die in der URL einer Client-Anfrage gespeichert sind, als Eingabeparameter für ein externes Programm dienen, müssen die url-kodierten Zeichen erst durch das Programm dekodiert werden, bevor dieses die Informationen weiterverarbeiten kann.

2.8 Uniform Resource Name (URN)

Neben URLs können auch Uniform Resource Names (URNs) verwendet werden, um Objekte im Internet zu spezifizieren. Ein URN identifiziert eine Ressource oder ein Objekt, wohingegen eine URL den Ort spezifiziert, an dem diese Ressource gespeichert ist [SM94]. Das Objekt, das durch einen URN identfiziert wird, kann dabei an mehreren verschiedenen Orten gespeichert sein. Dies bedeutet, daß einem URN mehrere URLs zugeordnet sein können, die jeweils verschiedene Orte beschreiben, an denen das gleiche Objekt verfügbar ist. Ein URN ist der Name für ein Objekt, der weltweit gültig ist und keine Lokation des Objekts impliziert. Der Name dieses Objekt ist eindeutig. Dies bedeutet, daß ein URN nicht für zwei verschiedene Objekte gleichzeitig verwendet werden darf. Ein URN ist persistent, was eine unendliche Lebensdauer impliziert. Der URN ist mit anderen Worten ein ortsunabhängiger, global eindeutiger, persistenter Bezeichner für Objekte im Internet.

Ein URN setzt sich aus zwei Teilen zusammen, der „Name Authority“ sowie dem „Opaque String “, die durch einen „/“ voneinander getrennt sind. Die „Name Authority“ ist der Teil des URNs, der die Person, den Dienst oder die Organisation beschreibt, die den „Opaque String“ zur Verfügung stellt. Der „Opaque String“ ist der eindeutige Name des Objekts innerhalb der „Name Authority“ [SMTW95]. Der Name bzw. die URN eines Objekts wird durch die „Name Authority“ festgelegt. Die „Name Authority“ wird entweder durch einen global eindeutigen Identifikator, die sog. „Name Authority ID“ oder durch einen vollständigen Domain-Namen bzw. eine IP-Adresse, den sog. „Resolution Host“, beschrieben. Beispiele hierfür sind die URNs `GNU/index.html`, `OCLC/paper1.1995` und `www.oclc.org/paper1.1995`. Bei den ersten beiden URNs werden die Identifikatoren `GNU` bzw. `ODLC` als „Name Authorities“ verwendet. Das letzte Beispiel verwendet den Domain-Namen `www.oclc.org`, um die „Name Authority“ zu beschreiben. `index.html` und `paper1.1995` bezeichnen die Objekte innerhalb der „Name Authority“, die identifiziert werden sollen.

Damit ein URN beispielsweise in einem WWW-Dokument als Link verwendet werden kann, muß dem WWW-Client mitgeteilt werden, auf welche Weise ein URN aufgelöst werden soll, damit der Client auf das Objekt zugreifen kann, daß über den URN identifiziert wird. Ein URN kann dabei auf einen URL bzw. auf eine Liste von URLs oder auf einen anderen URN abgebildet werden. Mit Hilfe des „Resolution Requests“ wird dem Client mitgeteilt, wie die Auflösung des URNs erfolgen soll. Wird dieser Request in ein Dokument eingebettet, ist er in spitze Klammern zu setzen, um ihn von dem übrigen Text unterscheiden zu können. Der „Resolution Request“ hat folgende Syntax:

```
<Resolution_Service:/Optional_Resolution_Path/URN>
```

Der „Resolution Service“ beschreibt den Dienst, der angewendet werden soll, um den angegebenen URN aufzulösen. Der Request `<N2L://GNU/index.html>` beschreibt beispielsweise die Abbildung des URNs auf einen URL. Es liegt dabei in der Hand des Clients, welchen Host er verwendet, um den URN in eine URL aufzulösen. Neben dem „Resolution Service“ `N2L` können auch die Dienste `N2C0` bzw. `N2N` verwendet werden, die eine Auflösung der URN in eine Liste von URLs bzw. in einen anderen URN bewirken. Der Autor eines Dokuments, kann zusätzlich zum URN über den optionalen „Resolution Path“ weitere „Resolution Hosts“ spezifizieren, die zur Auflösung des URNs verwendet werden können. Ein Beispiel hierfür ist der „Resolution Request“ `<N2L:/ncsa.uiuc.edu:120/GNU/index.html>`. Die Zeichenkette `ncsa.uiuc.edu:120` beschreibt dabei den „Resolution Host“, der vom Client verwendet werden kann, um den URN `GNU/index.html` in eine URL aufzulösen.

URNs befinden sich derzeit noch in der Spezifikationsphase. Bisher sind nur die funktionalen Anforderungen für URNs in Form eines RFCs [SM94] festgehalten. Weitere Dokumente zu diesem Thema liegen nur als Internet-Drafts [SMTW95] [LS95] [HD95] vor. Die im Kapitel 3.2 beschriebenen WWW-Clients sind bisher nicht in der Lage, URNs zu verarbeiten.

Kapitel 3

Evaluierung existierender WWW-Software

WWW-Server und -Clients stehen derzeit für die unterschiedlichsten Plattformen und Systemumgebungen zur Verfügung. Um sicherzustellen, daß sie die Anforderungen erfüllen, die sich aus der Entwicklung der WWW-Anwendung JAWA ergeben, werden aus der Fülle der zur Verfügung stehenden Software einige für die Diplomarbeit relevante WWW-Server sowie -Clients ausgewählt und anhand eines erstellten Kriterienkataloges bewertet.

3.1 WWW-Server

WWW-Server, die auch als „HTTP-Daemon“ oder „httpd“ bezeichnet werden, sind derzeit für ein Vielzahl von Plattformen und Betriebssystemen verfügbar. Es existieren Implementierungen für alle gängigen UNIX-Varianten, wie beispielsweise HP-UX, SunOS, SGI IRIX, DEC ULTRIX, IBM AIX, LINUX, sowie für VMS, NeXTStep, Macintosh, OS/2, Windows 3.1 und Windows NT. Aus der Fülle von Software werden nun drei für die Diplomarbeit relevante Server ausgewählt und auf ihre Funktionalität hin untersucht. Es handelt sich dabei ausschließlich um Server aus der UNIX-Umgebung, da im Rahmen dieser Diplomarbeit die Jahreswagenbörse bei der BMW AG ebenfalls auf einem UNIX-System zu entwickeln und zu implementieren ist. Ein weiteres Kriterium bei der Auswahl der Server, war der Grad der Verbreitung der Software, der auf einer Umfrage unter Benutzern des WWWs basiert [PR94]. Bei den WWW-Servern handelt es sich im Einzelnen um CERN httpd 3.0, NCSA httpd 1.3 sowie Netsite Commerce Server 1.0.

3.1.1 Bewertungskriterien

Die Kriterien, die zur Bewertung der drei WWW-Server herangezogen werden, sollen nun im Einzelnen erläutert werden und es soll aufgezeigt werden, warum gerade diese Kriterien ausgewählt wurden.

- Protokollimplementierung

  Die Version des HTTP-Protokolls ist wichtig für die Kommunikation zwischen Client und Server. Hat der Server beispielsweise `HTTP/0.9` und der Client `HTTP/1.0` implementiert, können sich beide zwar verständigen, doch ist die Kommunikation auf die HTTP-Methode `GET` beschränkt. Das bedeutet, daß ein WWW-Client in seinen Anfragen außer der `GET`-Methode keine weiteren Zugriffsmethoden spezifizieren kann, da der WWW-Server diese nicht verstehen würde. Da beim Zugriff auf die JAWA beide Methoden verwendet werden, bedeutet dies, daß ein Server, der `HTTP/0.9` implementiert, nicht als WWW-Server für die JAWA eingesetzt werden kann.

- Zugriffskontrolle

  Der Zugriff auf Dokumente auf dem WWW-Server soll aus Gründen der Sicherheit eingeschränkt werden können. Die Einschränkung kann für bestimmte Benutzer oder Benutzergruppen gelten, wobei die Authentifizierung der einzelnen Benutzer auf Benutzernamen oder auf IP-Adressen basieren kann. Der Benutzername bzw. die Benutzergruppe muß nicht identisch sein mit der Kennung bzw. der Gruppenzugehörigkeit auf dem Rechner. Ein WWW-Server ist in der Lage, seine eigenen Benutzer bzw. Benutzergruppen zu verwalten. Das Erstellen einer Benutzergruppe erfolgt in der Regel durch manuelles Editieren einer serverspezifischen Konfigurationsdatei.

- Datenverschlüsselung

  Werden sensitive Daten, wie beispielsweise die Kreditkartennummer beim Online-Shopping oder das Paßwort beim Anmeldevorgang über das Netz übertragen, müssen diese aus Gründen der Sicherheit vor der Übertragung verschlüsselt werden. Dies setzt voraus, daß beide Kommunikationspartner ein Verschlüsselungsverfahren verwenden, das von beiden Seiten verstanden wird.

- Logging

  Der Zugriff auf Dokumente sowie die beim Zugriff auftretenden Fehler werden vom Server protokolliert. Er merkt sich in einer entsprechenden Datei, wer wann wie auf welches Dokument zugegriffen hat. Die Protokolldatei kann zu einem späteren Zeitpunkt ausgewertet werden, um damit Rückschlüsse aus der Nutzung der auf einem WWW-Server angebotenen Dienste zu machen und um aufgetretene Fehler zu analysieren.

- Unterstützung von serverseitigen Programmen

  Ein WWW-Server, der nicht nur statische Dokumente bereitstellen, sondern diese auch zum Zeitpunkt einer Client-Anfrage durch ein Programm dynamisch generieren soll, muß dazu serverseitige Programme aufrufen können. Der Server erkennt anhand der URL in der Client-Anfrage, daß kein Dokument auszuliefern, sondern ein Programm auf dem Server zu starten ist. Soll ein WWW-Server als Gateway für eine Datenbank dienen, muß er mindestens dieses Kriterium erfüllen.

- CGI-Kompatibilität

  Die Unterstützung des Common Gateway Interfaces (Kapitel 4.2) ist ein wichtiger Aspekt bei der Auswahl eines WWW-Servers. Ist ein Server nicht CGI-kompatibel, sind CGI-kompatible Programme auf diesem Server nicht ablauffähig.

- Proxy-Fähigkeit

  Möchte ein WWW-Client innerhalb einer Firewall mit einem Informations-Server im Internet kommunizieren, kann unter anderem ein Proxy-Server hierzu verwendet werden (Kapitel 7.2). Besitzt ein WWW-Server diese Proxy-Funktionalität, kann er auch als Proxy-Server eingesetzt werden.

- Dokumenten-Caching

  Ein Proxy-Server kann unter Umständen gleichzeitig als Cache-Server verwendet werden (Kapitel 7.2.2). Der WWW-Client stellt seine Anfragen dabei nicht mehr an einen entfernten Informations-Server, sondern an den Cache-Server. Dieser entscheidet dann, ob er dem Client ein zwischengespeichertes Dokument zurückschickt oder ob er selbst das Dokument von einem entfernten Server holen muß, bevor er die Anfrage des Clients beantwortet. Durch den Einsatz eines Cache-Servers können die Zugriffe auf entfernte Server und somit der Internet-Verkehr, der sich in den Kosten für die Benutzung des Internets wiederspiegelt, drastisch reduziert werden. Das bedeutet, daß durch den Einsatz eines Cache-Servers die durch WWW erzeugt Netzlast verringert und der Zugriff auf WWW-Seiten für die Benuzter beschleunigt werden kann. Daneben treten durch das Zwischenspeichern von Dokumenten auch Nachteile auf, die in Kapitel 7.2.2 aufgeführt sind.

- Abbildung von Client-Anfragen

  Mit Hilfe von Regeln bildet ein Server virtuelle Pfade aus der Client-URL in physikalische Pfade auf dem Server ab. Das bedeutet, daß ein WWW-Client zum Zeitpunkt der Anfrage nur den virtuellen Namen eines Dokuments kennen muß. Zudem kann die physikalische Lokation von Dokumenten auf dem Server durch das Hinzufügen oder Entfernen von Abbildungsregeln jederzeit geändert werden, ohne dabei die Hyperlinks, die auf diese Dokumente verweisen, ändern zu müssen.

- Automatisches Einfügen von Text

  Bei der Auswahl eines Servers kann auch das automatische, serverseitige Einfügen von Text in auszuliefernde Dokumente eine Rolle spielen. Dabei wird in jedes Dokument an einer speziell markierten Stelle ein ganz bestimmter Text, wie beispielsweise das Datum des Zugriffs oder eine Menüleiste für die Navigation innerhalb des Servers, eingefügt. Dies erleichtert das Erstellen von HTML-Dokumenten erheblich, da Text, der in jedem Dokument enthalten sein soll, vom Server automatisch eingefügt wird.

- Verzeichnisindizierung

  Spezifiert ein Client bei seiner Anfrage nur ein Verzeichnis, liefert der Server als Ergebnis den Inhalt des Verzeichnisses zurück. Dies ist von Vorteil, wenn man die genaue Lokation eines Dokuments auf dem WWW-Server nicht kennt. Man spezifiert bei seiner Anfrage nur den Server-Namen und erhält als Ergebnis die Inhalte auf diesem. Danach kann man die Verzeichnisse nach dem jeweiligen Dokument durchsuchen.

- Konfiguration des Servers

  Bei der Installation eines WWW-Servers kann die Art der Installation bzw. Konfiguration des Servers sowie der Aufwand, der sich daraus ergibt, für den Benutzer eine Rolle spielen. Zudem bleibt zu betrachten, ob die Konfiguration bzw. Wartung eines Servers durch manuelles Editieren einer Konfigurationsdatei oder menügesteuert erfolgt. Die menügesteuerte Konfiguration des Servers hat den Vorteil, daß der Benutzer die Syntax der Konfigurationsparameter nicht zu kennen braucht, da er bei der Installation von dem System unterstützt wird.

- Start des Servers

  Ein Prozess unter Unix kann entweder über den Internet-Daemon `inetd`, über ein Shell-Skript beim Hochfahren eines Rechners oder von Hand gestartet werden. Wird der WWW-Server über den Internet-Daemon `inetd` gestartet, generiert `inetd` für jede ankommende Anfrage einen neuen httpd-Prozess, der die Anfrage bearbeitet. Wird der Server hingegen über ein Shell-Skript beim Hochfahren des Rechners bzw. von Hand gestartet, was auch als „Standalone"-Konfiguration bezeichnet wird, generiert der HTTP-Daemon selbst einen neuen Prozess um eine Client-Anfrage zu bearbeiten. Die Standalone-Konfiguration ist der ersten Methode vorzuziehen, da sie schneller und effizienter ist und dadurch zur Verkürzung der Antwortzeit des Server beiträgt. Beim Start des WWW-Servers über `inetd` wird zuviel Overhead produziert, wenn ein neuer httpd-Prozeß gestartet wird, was sich auf die Antwortzeit des Servers auswirkt. Daher wird empfohlen, den WWW-Server nicht über den Internet-Daemon, sondern über ein Shell-Skript bzw. von Hand zu starten, sofern der Server dazu ausgelegt ist.

Tabelle 3.1: Untersuchte WWW-Server

Kriterium	CERN httpd	NCSA httpd	Netsite httpd
Protokollimplementierung	HTTP/1.0	HTTP/1.0	HTTP/1.0
Zugriffskontrolle	ja	ja	ja
Datenverschlüsselung	nein	nein	ja[1]
Logging	ja	ja	ja
CGI-Kompatibilität	CGI/1.1	CGI/1.1	CGI/1.1
Unterstützung serverseitiger Skripts	ja	ja	ja
Proxy-Fähigkeit	ja	nein	nein
Dokumenten-Caching	ja	nein	nein
Abbildung von Client-Anfragen	ja	ja	ja
Automatischens Einfügen von Text	nein	ja	nein
Verzeichnisindizierung	ja	ja	ja
Konfiguration des Servers	manuell	manuell	menügesteuert
Start des Servers	standalone	standalone	standalone
Verfügbarkeit der Software	public domain	public domain	kommerziell

- Verfügbarkeit der Software

  Bei dem WWW-Server kann es sich entweder um ein kommerzielles Produkt oder um „Public Domain"-Software handeln. „Public Domain"-Software hat den Vorteil, daß sie kostenlos verfügbar ist, somit auch für Personen, die sich mit Hilfe krimineller Energie unerlaubten Zugang zu Informationssystemen verschaffen. Da diesen Personen der Source-Code eines Public Domain WWW-Servers vorliegt, können sie nach Fehlern in der Software suchen, die sie unter Umständen ausnutzen, um sich unerlaubten Zugang zu diesen Systemen zu verschaffen. Es bleibt auch zu berücksichtigen, daß die Anbieter kommerzieller Software für die Fehler in ihren Produkten haften müssen, was bei Anbietern von „Public Domain"-Software nicht der Fall ist. Zudem bieten kommerzielle Software-Hersteller oftmals bessere Unterstützung in Problemsituationen.

3.1.2 CERN-Server

Der CERN httpd 3.0 [LBLF94] wurde am CERN, dem europäischen Zentrum für Teilchenphysik in GENF, entwickelt und ist für alle gängigen UNIX-Plattformen sowie für

[1] Nur beim Einsatz von Netsite Commerce Server als WWW-Server und Netscape Navigator als WWW-Client

VMS verfügbar. Er implementiert das HTTP-Protokoll in der Version 1.0. Unter den betrachteten WWW-Servern ist der CERN httpd der einzige, der zugleich als Proxy-Server (Kapitel 7.2) verwendet werden kann. Er erlaubt das „Caching“, das Zwischenspeichern oft abgerufener Dokumente. Daneben unterstützt er das Ausführen externer Programme von WWW-Servern aus und ist CGI/1.1-kompatibel. Der CERN httpd hat eine Zugriffskontrolle auf Dokumenten implementiert. Das Authorisierungsverfahren basiert dabei auf Benutzernamen (inclusive Paßwort-Schutz) sowie auf IP-Adressen. Die Zugriffe auf Dokumente werden in einer Datei protokolliert. Wird in der Anfrage-URL des Clients kein Dateiname angegeben, liefert der Server als Ergebnis der Anfrage das Listing des Verzeichnisses zurück. Der CERN-Server steuert die Zugriffe auf seine Dokumente über Abbildungsregeln, die in einer Konfigurationsdatei definiert werden. Die Konfiguration sowie die Wartung des Servers erfolgt durch manuelles Edititeren einer Konfigurationsdatei. Der CERN httpd kann sowohl über den Internet-Daemon `inetd` als auch über ein Shell-Skript bzw. von Hand gestartet werden („Standalone“-Konfiguration). Der CERN httpd 3.0 ist als „Public Domain“-Software verfügbar.

3.1.3 NCSA-Server

Der NCSA httpd 1.3 [McC94] wurde am NCSA, dem National Center for Supercomuting Applications an der Universität von Illinois, entwickelt und ist derzeit nur für die gängigsten UNIX-Varianten verfügbar. Er ist ein „leichtgewichtiger“ Serverprozess, der ähnliche Eigenschaften besitzt wie der CERN httpd, aber dafür wesentlich kleiner ist. Der Server implementiert `HTTP/1.0` und unterstützt die Zugriffskontrolle auf Verzeichnissen, die auf Benutzer- und Rechnernamen basiert. Die Dokumente auf dem Server werden über einen virtuellen Pfad angesprochen, der, ebenso wie beim CERN httpd, über Abbildungsregeln auf einen reallen Pfad auf dem Server abgebildet wird. Der NCSA httpd unterstüzt serverseitige Programme sowie das Common Gateway Interface (Kapitel 4.2). Weiterhin wird die Indizierung von Verzeichnissen unterstützt. Das bedeutet, das der Server als Antwort auf eine Anfrage, in der nur ein Verzeichnis spezifiziert wurde, den Inhalt des Verzeichnisses an den anfragenden Client zurückschickt. Zusätzlich kann der Server so konfiguriert werden, daß die Ausgabe eines Shell-Kommandos oder jeder beliebige Text automatisch in auszuliefernde HTML-Dokumente eingefügt wird. Die Konfiguration bzw. die Wartung des Servers erfolgt durch manuelles Editieren einer Konfigurationsdatei. Der NCSA httpd unterstützt den Start über den Internet Daemon `inetd` und die „Standalone“-Konfiguration. Er ist, ebenso wie der CERN-Server, als „Public Domain“-Software verfügbar.

3.1.4 Netsite Commerce Server

Der Netsite Commerce Server 1.0 [Net95a] wurde von der Firma Netscape Communications entwickelt und ist der einzige unter den betrachteten Server, der weitreichende Sicherheitsmechanismen wie Datenverschlüsselung und Server-Authentifizierung implementiert.

Die Datenverschlüsselung basiert auf der „RSA Public Key"-Technologie. Für die Übertragung von verschlüsselten Daten wird vorausgesetzt, daß beide Kommunikationspartner das gleiche Verschlüsselungs- bzw. Entschlüsselungsverfahren verwenden. Derzeit sind nur der Netsite Commerce Server als WWW-Server sowie der Netscape Navigator als WWW-Client in der Lage, Daten verschlüsselt zu übertragen und somit eine sichere Kommunikation zu gewährleisten. Der Netsite Communication Server ist derzeit für alle gängigen Unix-Plattformen verfügbar. Weiterhin sind Implementierungen für Windows NT sowie Novell Netware angekündigt. Er implementiert `HTTP/1.0`, unterstützt den Aufruf serverseitiger Programme und ist zu CGI/1.1 kompatibel. Der Zugriff auf die HTML-Dokumente wird mit Hilfe eines Authorisierungsverfahrens kontrolliert, das auf Rechnernamen und IP-Adressen basiert. Die Client-Zugriffe werden in einer Log- Datei protokolliert, wobei angegeben werden kann, welche Zugriffe protokolliert werden sollen und welche nicht. Zusätzlich unterstützt er die Indizierung von Verzeichnissen. Der Netsite Commerce Server verwendet, ebenso wie die anderen beiden Server, Abbildungsregeln, um virtuelle Pfade in der Client-URL auf reale Pfade auf dem Server abzubilden. Die Installation und Konfiguration sowie die Wartung des Servers ist vollkommen menügesteuert und einfach auszuführen. Der Netsite Commerce Server ist als einziger unter den betrachteten Servern ein kommerzielles Produkt.

3.2 WWW-Clients

Ebenso wie die WWW-Server, sind die WWW-Clients, sog. WWW-Browser, für eine Vielzahl von Plattformen verfügbar. Es existieren Implementierungen für alle gängigen Unix-Varianten, wie z.B. HP-UX, SunOS, IBM AIX, DEC Ultrix, Linux, sowie für Macintosh, NeXTStep, MS DOS, MS Windows 3.1, MS Windows NT und IBM OS/2. Man kann dabei zwischen zeilen- und bildschirmorientierten sowie graphikorientierten Clients unterscheiden. Zeilen- und bildschirmorientierte Clients können beispielsweise auf einfachen Terminals unter Unix sowie auf Terminalemulationen unter DOS, Windows oder Macintosh eingesetzt werden. Graphikorientierte Clients werden in Windows-Umgebungen, wie z.B. X11, DEC-Windows, Microsoft Windows oder Macintosh verwendet. Wie bei den Servern werden auch bei den WWW-Clients drei Produkte aus der Unix-Umgebung sowie zwei PC-basierte Clients ausgewählt und deren Funktionalität untersucht. Es handelt sich dabei um den bildschirmorientierten WWW-Client Lynx 2.3, um die graphikorientierten, Unix-basierten Browser NCSA Mosaic 2.6 for X , Netscape Navigator 1.0N for X sowie um die PC-basierten WWW-Clients NSCA Mosaic 2.0Beta4 for MS Windows und Netscape Navigator 1.1N for MS Windows. Der bildschirmorientierte WWW-Client Lynx wurde neben den graphikorientierten X11-Clients ausgewählt, da in der BMW-Umgebung neben einer X11-Umgebung auch einfache Terminals zum Einsatz kommen. Aus diesem Grund muß untersucht werden, ob auch bildschirmorienierte bzw. zeichenorientierte Browser für den Zugriff auf die „Jahreswagenbörse" eingesetzt werden können. NCSA Mosaic sowie Netscape Navigator wurden zur Evaluierung herangezogen, da laut Umfrage unter Benutzern des WWWs [PR94] diese WWW-Clients in der Unix-Umgebung am Häufigsten verwendet werden. Da die Benutzer im Umfeld der BMW AG hauptsächlich PC-basierten Zugang besitzen, wurden zudem die beiden PC-basierten Clients von NCSA und Netscape untersucht.

3.2.1 Bewertungskriterien

Zur Untersuchung der WWW-Clients werden Bewertungskriterien herangezogen, die im Einzelnen vorgestellt werden.

- Protokollimplementierung

  Wie bei den WWW-Servern, ist auch bei den WWW-Clients von Bedeutung, welche Version des HTTP-Prokolls im Client implementiert ist. Neben der `GET-Methode` können weitere Zugriffsmethoden nur dann verwendet werden, wenn beide Kommunikationspartner `HTTP/1.0` implementieren.

- Unterstützung von Proxy- bzw. SOCKS-Server

  Wird ein Proxy-Server für die Kommunikation mit entfernten Informationsservern außerhalb der Firewall verwendet, muß der WWW-Client für die Verwendung eines Proxy-Servers ausgelegt sein. Ebenso verhält es sich mit SOCKS-Servern (Kapitel 7).

- Caching von Dokumenten

  Wird für das Zwischenspeichern von Dokumenten kein zentraler Cache-Server verwendet, ist es für den Client von Vorteil, wenn er das Zwischenspeichern der Dokumente selbst übernimmt, um dadurch die Antwortzeiten sowie die Netzlast bei Dokumentenanfragen zu reduzieren. Man kann dabei unterscheiden, ob für das Caching Primär- und/oder Sekundärspeicher verwendet wird. Der Cache auf Festplatte hat gegenüber dem Hauptspeicher-Cache den Vorteil, daß er persistent ist. Das bedeutet, daß der Hauptspeicher-Cache nach Beendigung des Client-Programms verloren geht, wobei der Festplatten-Cache erhalten bleibt. Die dort gecachten Dokumente stehen dem Client nach einem erneuten Start wieder zur Verfügung.

- Verschlüsselung der Daten

  Sollen sensitive Daten über das Netz transportiert werden, wie beispielsweise die Kreditkartennummer beim Online-Shopping oder das Paßwort beim Verbindungsaufbau zu einem anderen Rechner, ist es aus Sicherheitsgründen sinnvoll, diese Daten nur verschlüsselt zu übertragen. Dies setzt natürlich voraus, daß beide Kommunikationspartner das gleiche Verschlüsselungs- und Entschlüsselungsverfahren verwenden.

- Unterstützung von HTML-Formularen

  Soll ein WWW-Client zur interaktiven Kommunikation zwischen Informationsanbieter und Benutzer genutzt werden, beispielsweise als Benutzerschnittstelle für Datenbankanwendungen, muß er HTML-Formulare unterstützen.

- Unterstützung von HTML-Tabellen

  Werden Teile von Daten in einem WWW-Dokument mit Hilfe des HTML 3.0-Sprachelements `<TABLE>` tabellarisch dargestellt, muß der Browser dieses Sprachelement unterstützen, um die Daten korrekt darstellen zu können.

- Schnittstelle zu anderen Informationsdiensten

  Aufgrund der Adressierung von Internet-Ressourcen im URL-Format ist ein Client prinzipiell in der Lage, neben WWW, auch andere Internet-Dienste wie FTP, Gopher, WAIS, Telnet, News und Electronic Mail zu nutzen. Es ist dabei den Entwicklern der WWW-Clients überlassen, welche Zugriffsmethoden neben HTTP implementiert werden.

- Unterstützung von Inline Images

  Aufgrund der multimedialen Fähigkeiten des WWW, können HTML-Dokumente Bilder enthalten, die im laufenden Text erscheinen. Diese Bilder werden dann als „Inline Images" bezeichnet. Ein WWW-Client, der diese Art von Bildern nicht verarbeiten und anzeigen kann, fügt an dessen Stelle ein spezielles Symbol in das Dokument ein, was den Benutzer darauf hinweisen soll, daß an dieser Stelle im Dokument ein Bild erscheinen sollte. Das eigentliche Bild bleibt dem Betrachter eines Dokumentes somit verborgen, sofern nicht ein externes Programm aufgerufen wird, das die Darstellung des Bildes übernimmt.

- Präsentation von Dokumenten

  Für die Beurteilung der Performance eines WWW-Clients ist ausschlaggebend, auf welche Art die angeforderten Dokumente vom Client geladen werden. Es gibt Clients, die das Dokument anzeigen, während Teile davon noch übertragen werden. Andere Clients wiederum bauen das Dokument erst auf, wenn es vollständig übertragen ist. Im ersten Fall kann der Benutzer mit dem Dokument bereits arbeiten, während es noch geladen wird. Dies ist von Vorteil, wenn das zu übertragene Dokument sehr groß ist und viele multimediale Elemente enthält. In zweiten Fall muß der Benutzer mit der Berbeitung des Dokuments warten, bis es vollständig geladen ist, was unter Umständen zu langen Wartezeiten führen kann.

- Laden von Bildern

  Für den Benutzer eines WWW-Clients ist es von Vorteil, wenn sich das Laden von

Tabelle 3.2: Untersuchte WWW-Clients

Kriterium	Mosaic for X	Netscape for X	Lynx	Mosaic for Win	Netscape for Win
Protokollimplementierung HTTP/1.0	ja	ja	ja	ja	ja
Unterstützung von Proxy-Servern	ja	ja	ja	ja	ja
Unterstützung von SOCKS-Servern	nein	ja	nein	nein	ja
Caching von Dokumenten	ja	ja	ja	ja	ja
Datenverschlüsselung	nein	ja[1]	nein	nein	ja[1]
Schnittstelle zu anderen Internet-Diensten	ja[2]	ja[2]	ja[2]	ja[2]	ja[2]
Unterstützung von HTML-Formularen	ja	ja	ja	ja	ja
Unterstützung von HTML-Tabellen	ja	ja	nein	ja	ja
Unterstützung von Inline Images	ja	ja	nein	ja	ja
Präsentation von Dokumenten während dem Laden	nein	ja	nein	ja	ja
Laden von Bilder deaktivierbar	ja	ja	-	ja	ja
Verfügbarkeit der Software	pd[3]	k[4]	pd[3]	pd[3]	k[4]

Bilder, die in WWW-Dokumente eingebettet sind, explizit aktivieren oder deaktivieren läßt. Damit hat der Benutzer die Möglichkeit, das Laden aufwendiger Bilder zu umgehen, wodurch die Antwortzeiten für den Benutzer erheblich reduziert werden.

- Verfügbarkeit der Software

  Wie auch bei den WWW-Servern wird auch bei den WWW-Clients unterschieden, ob es sich bei dem Produkt um „Public Domain“-Software oder um kommerzielle Software handelt.

3.2.2 Netscape Navigator for X

Netscape Navigator 1.0N [Net95b] for X ist ein graphikorientierter WWW-Client für die X11-Umgebung. Er wurde wie die Netsite Server von Netscape Communications entwickelt und ist eine kommerzielles Produkt. Netscape implementiert HTTP/1.0 und unterstützt Proxy- sowie SOCKS-Server. Für das Zwischenspeichern von Dokumenten verwendet er sowohl Hauptspeicher- als auch Festplatten-Cache, deren Größe frei konfigurierbar ist. Netscape hat gegenüber den anderen beiden WWW-Clients den Vorteil, daß er bei einem Neustart auf Dokumente zurückgreifen kann, die sich im Festplatten-Cache befinden. Wird ein solches Dokument in einer neuen Sitzung angefordert, zeigt der Client das gespeicherte Dokument an. Da die Größe der Caches frei konfigurierbar ist, ist er in der Lage den Zugriff auf Dokumente zu beschleunigen sowie die durch WWW erzeugte Netzlast zu reduzieren. Netscape verfügt über die Fähigkeit, Dokumente anzuzeigen, während dem sie noch übertragen werden, was ebenfalls zu einem beschleunigten Zugriff auf WWW-Dokumente führt. Dieser WWW-Client hat weitreichende Sicherheitsmechanismen wie Server-Identifizierung und Datenverschlüsselung implementiert. Server-Identifizierung bedeutet, daß bei einer Kommunikation zwischen zwei Rechnern, keiner der beiden vorgeben kann, ein anderer Rechner zu sein. Für die Verschlüsselung der Daten wird die „RSA Public Key„-Technologie verwendet. Um diese Art von Sicherheit zu erreichen, wird das von Netscape Communications eigens entwickelte Secure Socket Layer (SSL)-Protokoll verwendet. Sichere Kommunikation zwischen Client und Server ist aber nur dann gewährleistet, wenn beide Kommunikatiospartner das SSL-Protokoll verwenden, was derzeit nur bei Netscape Navigator sowie Netsite Commerce Server der Fall ist. Mit Hilfe der oben beschriebenen Maßnahmen können kommerzielle Transaktionen sowie jede andere Art von Kommunikation vor unerlaubten Zugriffen geschützt werden. Neben den weitreichenden Sicherheitsmaßnahmen bietet Netscape auch eine Schnittstelle zu weiteren Internet-Diensten wie FTP, Gopher, WAIS, Telnet, News sowie Electornic Mail und unterstützt Inline Images. Netscape Navigator ist außerdem in der Lage HTML-Formulare sowie HTML-Tabellen darzustellen.

3.2.3 NCSA Mosaic for X

NCSA Mosaic 2.4 for X [Nat95b] ist ein graphikorientierter WWW-Client für das Window-System X11, der ebenso wie NCSA httpd am National Center for Supercomputing Applications (NCSA) an der Universität von Illinois entwickelt wurde. Er implementiert HTTP/1.0 und unterstützt Proxy-Server. Für das Zwischenspeichern von Dokumenten wird nur ein Hauptspeicher-Cache verwendet, dessen Größe fest vorgegeben ist. NSCA Mosaic bietet eine Schnittstelle zu anderen Internet-Diensten und erlaubt die Konfiguration von externen

[1] Nur beim Einsatz von Netsite Commerce Server als WWW-Server und Netscape Navigator als WWW-Client

[2] Schnittstelle zu FTP, Gopher, WAIS, Telnet, News und Electronic Mail

[3] public domain

[4] kommerziell

Programmen. Dokumente werden von diesem Client erst angezeigt, wenn sie vollständig übertragen wurden, was im Vergleich zu Netscape zu längeren Wartezeiten beim Laden eines Dokuments führt. Mosaic unterstützt HTML-Formulare sowie HTML-Tabellen und ist „Public Domain“-Software. Bei der Verwendung von NCSA Mosaic ist keine sichere Kommunikation gewährleistet, da dieser Client keine Sicherheitsmaßnahmen wie Server-Identifizierung oder Datenverschlüsselung bei der Übertragung der Daten verwendet.

3.2.4 Lynx

Lynx 2.3 [BMGW] ist ein bildschirmorientierter WWW-Client, der an der Universität von Kansas entwickelt wurde. Er ist für die Betriebssysteme Unix sowie VMS verfügbar und setzt ein VT-100 Terminal oder eine VT-100 Terminalemulation varaus. Lynx verwendet die Cursortasten für die Navigation innerhalb von Dokumenten bzw. um neue Dokumente zu laden. Die Bedienung erfolgt über einfache Tastaturkommandos. Lynx implementiert HTTP/1.0 und ermöglicht das Zwischenspeichern von Dokumenten im Hauptspeicher, wobei die Größe des Caches definiert werden kann. Mit Lynx lassen sich, neben WWW, weitere Internet-Dienste wie FTP, Gopher, WAIS, News und Electonic Mail nutzen. Der Client ist zwar nicht in der Lage Bilder anzuzeigen, doch es kann ein externes Programm konfiguriert werden, das die Ausgabe der Bilder übernimmt. In wie weit dies Sinn macht bleibt zu bedenken, da ein normales Terminal nicht in der Lage ist, Graphiken und Bilder darzustellen. Alle WWW-Dokumente werden erst angezeigt, wenn das Dokument vollständig übertragen wurde, was bei großen Dokumenten zu langen Wartezeiten für den Benutzer führen kann. Lynx unterstützt HTML-Formulare und ermöglicht damit die interaktive Nutzung des WWWs. Lynx 2.3 ist als „Public Domain“-Software verfügbar.

3.2.5 NCSA Mosaic for MS Windows

NCSA Mosaic [Nat95a] ist ein graphikorientierter WWW-Client für die Microsoft Windows-Umgebung, der als „Public Domain“-Software verfügbar ist. Er implementiert das HTTP-Protokoll in der Version 1.0 und unterstützt Proxy-Server. Für das Zwischenspeichern von bereits geladenen Dokumenten verwendet der Client Hauptspeicher- sowie Festplatten-Cache, deren Größe frei definiert werden kann. NCSA Mosaic unterstützt HTML-Formulare sowie HTML-Tabellen. Weiterhin werden Inline Images unterstützt, wobei das Laden dieser Bilder jederzeit deaktiviert werden kann. Der Client ist in der Lage, die Dokumente während dem Laden anzuzeigen, was zu einem beschleunigten Zugriff auf WWW-Dokumente führt. Bei Verwendung von NCSA Mosaic als WWW-Client ist keine sichere Kommunikation gewährleistet, weil die Daten unverschlüsselt übertragen werden.

3.2.6 Netscape Navigator for MS Windows

Bei Netscape Navigator [Net95b] handelt es sich um einen graphikorientierten WWW-Client für die Microsoft Windows-Umgebung. Er ist ebenso wie der Unix-basierte Client von Netscape ein kommerzielles Produkt. Er implementiert wie alle anderen untersuchten WWW-Clients HTTP/1.0. Im Vergleich zu NCSA Mosaic unterstützt er neben Proxy-Servern auch SOCKS-Server. Für das Zwischenspeichern von Dokumenten werden sowohl Hauptspeicher- als auch Festplatten-Cache verwendet, die in ihrer Größe frei konfigurierbar sind. Netscape Navigator unterstützt HTML-Formulare, HTML-Tabellen sowie Inline Images. Der Client ist in der Lage Dokumente anzuzeigen, während sie noch geladen werden. Das Laden von Inline Images kann jederzeit durch den Benutzer deaktiviert werden. Ebenso wie der Unix-basierte Client von Netscape hat dieser WWW-Client weitreichende Sicherheitsmechanismen wie Server-Identifizierung und Datenverschlüsselung implementiert. Bei Verwendung von Netscape Navigator als WWW-Client ist somit eine sichere Kommuikation zwischen Server und Client gewährleistet.

3.3 Auswahl eines WWW-Servers und WWW-Clients für die JAWA

Von den untersuchten Servern können prinzipiell alle drei Produkte als WWW-Server für die JAWA eingesetzt werden. Da die untersuchten Server bezüglich der Bewertungskriterien sehr ähnliche Eigenschaften besitzen, ist die Auswahl eines Servers relativ schwer. Muß man sich für einen entscheiden, würde die Wahl auf den WWW-Server der Firma Netscape fallen. Dafür sprechen einerseits die einfache, menügesteuerte Konfiguration des Servers, sowie das aufwendigere und bessere Logging der Zugriffe auf den Server. Netsite httpd ist den anderen beiden Servern außerdem vorzuziehen, weil er in Verbindung mit einem WWW-Client von Netscape aufgrund der implementierten Sicherheitsmaßnahmen eine sichere Kommunikation zwischen Client und Server gewährleisten kann. Da es sich bei Netsite httd um ein kommerzielles Produkt handelt, ist davon auszugehen, daß dieses Produkt ständig weiterentwickelt wird und daß der Benutzer bessere Unterstützung in Problemsituationen erhält, als dies bei den anderen beiden Servern der Fall wäre. Netsite httpd ist zwar im Vergleich zum CERN httpd für den Einsatz als Proxy-Server nicht geeignet, doch ist dies für den Einsatz als WWW-Server für die JAWA nicht relevant.

Von den untersuchten Browsern können ausschließlich die beiden Unix-basierten Clients NCSA Mosaic for X und Netscape Navigator for X sowie der PC-basierte Browser Netscape Navigator for MS Windows als WWW-Clients für die JAWA verwendet werden. Lynx sowie NCSA Mosaic for MS Windows sind als WWW-Clients für die JAWA nicht geeignet. Sie sind nicht in der Lage, die von der JAWA generierten WWW-Seiten korrekt darzustellen. Die bei diesen WWW-Clients aufgetretenen Probleme werden in Kapitel C beschrieben.

Benötigt man einen Browser für die Unix-Umgebung, ist Netscape Navigator dem WWW-

Client NCSA Mosaic vorzuziehen. Netscape Navigator unterstützt standardmäßig neben dem Proxy-Server auch SOCKS-Server. Der Zugriff auf WWW-Dokumente erfolgt bei Einsatz von Netscape Navigator schneller, da dieser Browser die Dokumente bereits während dem Laden dem Benutzer darstellt. Zusätzlich zu seinem Hauptspeicher-Cache verwendet er im Vergleich zu NCSA Mosaic einen in der Größe frei konfigurierbaren Festplatten-Cache, was den Zugriff auf WWW-Dokumente weiterhin beschleunigt. Außerdem ist Netscape Navigator in der Lage eine sichere Kommunikation zwischen Client und Server zu gewährleisten, da er weitreichende Sicherheitsmechanismen implementiert. Möchte man hingegen einen PC-basierten WWW-Client für den Zugriff auf die JAWA verwenden, ist hierzu Netscape Navigator for MS Windows zu verwenden. Der PC-basierte Client NCSA Mosaic ist, wie schon erwähnt, hierfür nicht geeignet.

Kapitel 4

Design eines Datenbank-Gateways

4.1 Motivation

Um den WWW-Client als graphische Benutzerschnittstelle für komplexe Datenbankanfragen einsetzen zu können, wird untersucht, in welcher Weise das WWW die Möglichkeit bietet, auf Informationen zuzugreifen, die in Datenbanken abgelegt sind. Als Informationsquellen kommen dabei jegliche Art von Datenbanken, wie beispielsweise Bilddatenbanken, Dokumentenarchive sowie hierarchische oder relationale Datenbanken, in Frage.

Aus der Sicht eines Datenbank-Servers verhalten sich WWW-Datenbank-Gateways wie ganz normale Datenbank-Clients. Für den WWW-Server sind sie nichts anderes als eine Art „black box", die HTML-Text auf die Standardausgabe ausgibt. Dieser Art von Gateway liegt eine „Client-Agent-Server"-Architektur zugrunde. Das Datenbank-Gateway agiert dabei als Agent zwischen dem WWW-Client und dem Datenbank-Server.

Im weiteren Verlauf dieses Kapitels soll ein grundlegendes Design für die WWW-Anwendung „Jahreswagenbörse" entwickelt werden. Es werden dazu die Möglichkeiten beschrieben, die das WWW bietet, um ein solches Gateway zu entwickeln. Daneben werden die verschiedenen Architekturmöglichkeiten des Datenbank-Gateways sowie einige Betriebsaspekte betrachtet. Die daraus gewonnenen Erkenntnisse werden als Grundlage für die Implementierung eines Prototypen der JAWA verwendet.

4.2 Common Gateway Interface (CGI)

4.2.1 Funktionsweise

Das Common Gateway Interface (CGI) [McC93] ist ein standardisiertes Verfahren im WWW, das es ermöglicht, Dokumente zum Zeitpunkt der Client-Anfrage dynamisch durch ein Programm zu generieren.
Ein WWW-Server kann mit Hilfe des CGI als Gateway zu fast beliebigen Informationssystemen dienen. Über HTML-Formulare ist der Benutzer eines WWW-Clients beispielsweise in der Lage, komplexe Anfragen an Datenbanken zu stellen, die durch externe Programme bearbeitet, und deren Ergebnisse an den Benutzer zurückgeschickt werden.
Der Ablauf ist dabei wie folgt: Der Benutzer setzt über seinen WWW-Client eine Anfrage an den WWW-Server ab. Dieser erkennt anhand des URL, daß als Antwort auf die Anfrage nicht der Inhalt einer Datei bzw. ein statisches Dokument zurückzuliefern ist, sondern ein externes Programm ausgeführt werden muß, das ein Dokument dynamisch generiert. Der Server ruft dieses Programm auf und übergibt ihm die Parameter, die der Client in seiner Anfrage spezifiziert hat. Das externe Programm erzeugt daraufhin das gewünschte Dokument und schreibt es auf die Standardausgabe. Der WWW-Server seinerseits leitet dann das dynamisch generierte Dokument an den WWW-Client zurück.

Das CGI ist eine Schnittstellenspezifikation zum Aufruf und zur Parameterversorgung von externen Programmen. Der WWW-Server und das aufgerufene Programm kommunizieren über Umgebungsvariablen (Anhang A), die Informationen über den WWW-Server und die vom Client empfangene Anfrage enthalten. Der CGI-Standard legt dazu Namen und Inhalt dieser Variablen fest. Daneben besteht die Möglichkeit, externe Programme über die Kommandozeile oder über die Standardeingabe mit Parametern zu versorgen. Programme, die dem CGI-Standard entsprechen, werden als CGI-Skripten bezeichnet und sind ohne Änderung auf unterschiedlichen CGI-kompatiblen WWW-Servern ablauffähig. Das CGI sorgt somit für Kompatibilität zwischen WWW-Server einerseits und externen Programmen andererseits. Die Entwicklung solcher CGI-Skripten ist nicht an eine bestimmte Sprache gebunden. Sie können in jeder beliebigen Sprache, wie beispielsweise *C/C++*, *perl* oder *C Shell*, implementiert werden, mit der man ausführbare Programme erstellen kann.

4.2.2 Parameterübergabe an CGI-Skripten

Verfahren zur Parameterübergabe

Der CGI-Standard sieht vor, externe Programme mit Parametern zu versorgen. Die Parameterübergabe erfolgt dabei in Abhängigkeit der HTTP-Methode, mit der das externe Programm referenziert wird.

Verwendung der Zugriffsmethode `GET`: Gemäß CGI-Spezifikation gibt es drei Möglichkeiten, um Parameter beim Einsatz der HTTP-Methode `GET`, an ein CGI-Skript zu übergeben. Die zu übergebenden Parameter werden dabei in der URL-spezifischen Syntax kodiert.

- Die zu übergebenden Parameter werden durch einen Schrägstrich getrennt, an die URL der Client-Anfrage gehängt. Die Umgebungsvariable `PATH_INFO` wird in diesem Fall mit dem Wert der zu übergebenden Parameter belegt. Sie enthält zusätzlich den führenden Schrägstrich.
  Beispiel: `http://Servername/cgi-bin/cgi/Paramter+f%FCr+CGI-Skript`
- Die zu übergebenden Parameter werden, durch ein Fragezeichen getrennt, an die URL der Client-Anfrage gehängt. In diesem Fall wird die Umgebunsvariable QUERY_STRING mit dem Wert der Parameter belegt.
  Beispiel: `http://Servername/cgi-bin/cgi?Paramter+f%FCr+CGI-Skript`
- Der dritte Fall ist eine Kombination der ersten beiden Verfahren. Hierbei werden sowohl PATH_INFO als QUERY_STRING mit den entsprechenden Werten belegt.
  Beispiel: `http://Servername/cgi-bin/cgi/Paramter?f%FCr+CGI-Skript`

Verwendung der Zugriffsmethode `POST`: Neben der `GET`-Methode kann ebenso die HTTP-Methode `POST` verwendet werden, um CGI-Skripten zu referenzieren. Diese Methode kommt hauptsächlich bei HTML-Formularen zum Einsatz. Bei dieser Art von Zugriff werden die Werte der Eingabefelder aus dem Formular dem CGI-Skript nicht über die Umgebungsvariablen `PATH_INFO` oder `QUERY_STRING`, sondern über die Standardeingabe übergeben. Die Umgebungsvariable `CONTENT_TYPE` enthält dabei den MIME-Typ der übertragenen Daten, die Variable `CONTENT_LENGTH` ist mit der Länge der Daten, die auf der Standardeingabe übermittelt werden, belegt.

Parameterübergabe auf der Kommandozeile

Werden die zu übergebenen Parameter, durch ein Fragezeichen separiert, an die URL der Client-Anfrage gehängt und an den Server geschickt, dekodiert dieser die in der URL auf das „?“-Zeichen folgende Zeichenkette. Setzt sich die Zeichenkette aus mehreren Wörtern zusammen, wird sie in jeweils einzelne Worte aufgeteilt, bevor diese dem CGI-Skript auf der Kommandozeile übergeben werden. Die einzelen Worte stehen dann in den Variablen `argv[i]`, wobei „i“ für die Anzahl der Worte steht.

Ein Beispiel hierfür ist die Verwendung des HTML-Elements <`ISINDEX`> im Dokumentenkopf. Dem Benutzer wird dadurch ermöglicht einen Suchbegriff in ein zur Verfügung gestelltes Textfeld einzugeben. Der eingegebene Suchbegriff wird vom Client, durch ein „?“-Zeichen getrennt, an die URL gehängt.
Diese Art der Paramterübergabe ist aber nicht nur auf Dokumente beschränkt, die das

HTML-Element <ISINDEX> verwenden. Der Benutzer kann eine entsprechende URL direkt am Client eingeben oder in einem Dokument als Hyperlink einbetten.

Zusätzlich zur Kommandozeile erscheinen die zu übergebenen Parameter als kodierte Zeichenkette in der Umgebungsvariablen `QUERY_STRING`. Das ganze soll nun an einem Beispiel verdeutlicht werden. Sei die URL der Client- Anfrage gegeben durch

```
http://Servername/cgi-bin/cgi?Parameter+f%FCr+CGI-Skript
```

werden folgende Variablen belegt:

- **argv[1]**: `Parameter`
- **argv[2]**: `für`
- **argv[3]**: `CGI-Skript`
- **QUERY_STRING**: `Parameter+f%FCr+CGI-Skript`

Die Parameterübergabe auf der Kommandozeile ist mit Vorsicht zu genießen. Der Server stellt zwar die Parameter einzeln und in dekodierter Form dem Skript zur Verfügung, doch kann es manchmal vorkommen, daß der Server die Kommandozeile leer läßt, wenn Beschränkungen im Betriebssystem die Übergabe sehr langer Zeichenketten auf der Kommandozeile verhindern. Die Übergabe der Parameter erfolgt dann ausschließlich in der Umgebungsvariablen `QUERY_STRING`. Ein CGI-Skript, das neben der Kommandozeile auch diese Umgebungsvariable auswertet, ist somit immer auf der sicheren Seite.

Parameterübergabe über CGI-Umgebungsvariablen

Wie oben dargestellt belegt der WWW-Server einzelne Umgebungsvariablen in Abhängigkeit der Zugriffsmethode und der verschieden Verfahren zur Parameterübergabe. Die Belegung der einzelnen Umgebungsvariablen soll anhand von Beispielen verdeutlicht werden. Dazu wurde ein Shell-Skript verwendet, daß sowohl die auf der Kommandozeile übergebenen Parameter als auch die im CGI-Standard definierten Umgebungsvariablen ausgibt. Weiterhin werden durch dieses Skript die Parameter ausgegeben, die dem Skript auf der Standardeingabe übergeben werden. Im folgenden werden nur die für die Darstellung relevanten Umgebungsvariablen aufgeführt.
Nehmen wir an, die URL zum Aufruf des oben beschriebenen CGI-Skripts sei

```
http://Servername/cgi-bin/showenv
```

wobei „`showenv`“ der Name des aufzurufenden Skripts sei. Die Zeichenkette, die an das Skript zu übergeben ist, sei

„`Parameter für CGI-Skript`“.

- Verwendung der HTTP-Methode `GET` und Anhängen der Parameter, die durch einen Schrägstrich von der URL getrennt sind.
  URL: `http://Servername/cgi-bin/cgi/Parameter+f%FCr+CGI-Skript`
  Umgebungsvariablen:
  - **REQUEST_METHOD**: `GET`
  - **argv[1]**:
  - **argv[2]**:
  - **argv[3]**:
  - **PATH_INFO**: `Parameter+für+Skript`
  - **PATH_TRANSLATED**: `/usr/local/pub/WWW/Parameter+für+Skript`
  - **QUERY_STRING**:
- Verwendung der HTTP-Methode `GET` und Anhängen der Parameter, die durch ein Fragezeichen von der URL getrennt sind.
  URL: `http://Servername/cgi-bin/cgi?Parameter+f%FCr+CGI-Skript`
  Umgebungsvariablen:
  - **REQUEST_METHOD**: `GET`
  - **argv[1]**: `Parameter`
  - **argv[2]**: `für`
  - **argv[3]**: `CGI-Skript`
  - **PATH_INFO**:
  - **PATH_TRANSLATED**:
  - **QUERY_STRING**: `Parameter+f%FCr+CGI-Skript`
- Verwendung der HTTP-Methode `GET` und Anhängen der Parameter, die durch Schrägstrich und Fragezeichen von der URL getrennt sind.
  URL: `http://Servername/cgi-bin/cgi/Parameter?f%FCr+CGI-Skript`
  Umgebungsvariablen:
  - **REQUEST_METHOD**: `GET`
  - **argv[1]**: `für`
  - **argv[2]**: `CGI-Skript`
  - **argv[3]**:
  - **PATH_INFO**: `Parameter`
  - **PATH_TRANSLATED**: `/usr/local/pub/WWW/Parameter`
  - **QUERY_STRING**: `f%FCr+CGI-Skript`

Parameterübergabe über Standardeingabe

Wird die HTTP-Methode `POST` verwendet, wird das referenzierte CGI-Skript über die Standardeingabe mit Parametern versorgt. Um dies an einem Beispiel zu verdeutlichen, wurde die im vorherigen Abschnitt beschriebene Umgebung verwendet.

- Verwendung der HTTP-Methode `POST`

  Bei einem Formular, das aus einem Texteingabefeld mit dem Namen „`Eingabefeld`" besteht, setzt der Server folgende Umgebungsvariablen:

  - **REQUEST_METHOD**: `POST`
  - **argv[1]**:
  - **argv[2]**:
  - **argv[3]**:
  - **PATH_INFO**:
  - **PATH_TRANSLATED**:
  - **QUERY_STRING**:
  - **CONTENT_TYPE**: `application/x-www-form-urlencoded`
  - **CONTENT_LENGTH**: `38`

Bei der Zugriffsmethode `POST` werden die Werte der Eingabefelder aus dem Formular dem CGI-Skript über die Standardeingabe übermittelt. Die Umgebungsvariable `CONTENT_TYPE` verrät, daß es sich um ein Formular handelt. Die Variable `CONTENT_LENGTH` enthält die Anzahl der Bytes, die an das Skript über die Standardeingabe übermittelt wurden. Die Standardeingabe enthält die Zeichenkette „`Eingabefeld=Parameter+f%FCr+CGI-Skript`".

4.2.3 Ausgabe der CGI-Skripten

Bevor ein CGI-Skript die eigentlichen Nutzdaten generiert, schreibt es zunächst Meta-Informationen, die das Dokument näher beschreiben, auf die Standardausgabe. Diese Informationen gehen als Header den im Body enthaltenen Daten voraus und bestehen aus HTTP-Informationen.

Das aufgerufene CGI-Skript kann Dokumente beliebigen Formats generieren, wie beispielsweise HTML-Dokumente, einfache ASCII-Dokumente oder sogar Dokumente, die Audio-Sequenzen enthalten. Damit der Client, an den dieses Dokument geschickt wird, weiß, um welche Art von Dokument es sich handelt, muß das CGI-Skript vor den eigentlichen Nutzdaten einen Header generieren. Der Header besteht aus Textzeilen die das gleiche Format besitzen wie HTTP-Header-Informationen. Er wird durch eine Leerzeile abgeschlossen. Jeder Header besteht aus mindestens zwei Zeilen, die für jedes Skript zwingend erforderlich sind.

- Die erste Zeile spezifiert den MIME-Typ des Dokuments, falls das Skript ein vollständiges Dokumente generiert.
  Format:`Content-Type:` *type/subtype*
  Beispiel: `Content-Type: text/html`

- Sollte das Skript nur einen Verweis auf ein anderes Dokument auf dem eigenen WWW-Server oder auf einem anderen Informationsserver generieren, spezifiziert die erste Zeile die Lokation dieses Dokuments. Diese Zeile wird dann vom Server ausgewertet. Handelt es sich bei dem Verweis um ein Dokument auf dem lokalen Server, holt der WWW-Server dieses Dokument und schickt es an den Client zurück. Handelt es sich um ein Dokument auf einem entfernten Server, schickt der WWW-Server nur die im Header angegebene URL an den Client zurück.
  Format: `Location` *URL*
  Beispiel: `Location: http://www.mcom.com/home/welcome.html`

- Die zweite Zeile ist eine Leerzeile, die den Header von den eigentlichen Nutzdaten trennt. Diese Zeile ist zwingend erforderlich.

Daneben kann das CGI-Skript weitere Header-Informationen generieren [McC93]. Für die Diplomarbeit von Interesse ist dabei noch die Header-Information „`Expires: date`“, die angibt, wann die Gültigkeit eines Dokuments abgelaufen ist. Dieser Sachverhalt ist für Proxy-Server mit Cache (Kapitel 7.2.2) von großem Interesse. Anhand der im Dokument enthaltenen Datumsangabe, weiß der Proxy-Server, ob er einem anfragenden WWW-Client das zwischengespeicherte Dokument schicken soll, oder ob er eine neue Version vom entsprechenden Informations-Server anfordern muß, bevor er das Dokument an den Client zurückschickt.

4.3 Die Btx-basierende Anwendung „BMW Exklusivbörse für gebrauchte Automobile“

Die Btx-basierte Anwendung „BMW-Exklusivbörse für Gebrauchtwagen“ der BMW AG [BMW92] ist ein Gebrauchtwagenangebot der BMW AG für den öffentlichen und privaten Btx-Bentuzer. Der von der BMW AG betriebene Informationsdienst ist derzeit erfolgreich im Einsatz. Der Btx-basierenden Anwendung liegt folgende Architektur zugrunde: Die Daten der Gebrauchtwagenbörse, d.h. die Gebrauchtwagendaten werden in einer IMS-Datenbank auf einem Großrechner gespeichert und gepflegt. Unter Verwendung eines BMW-spezifischen Filetransferprogramms (NWP bzw. DTS) werden die Gebrauchtwagendaten in eine relationale Oracle-Datenbank in einer Unix-Umgebung transferiert und dort dem Anwender des Dienstes zur Verfügung gestellt. Der Benutzer der Gebrauchtwagenbörse greift über seine Btx-Schnittstelle auf die in der relationalen Datenbank bereitgestellten Daten zu. Die Zugriffe der Anwendung sind damit auf Unix-basierte, relationale Datenbanken beschränkt.

Die oben beschriebene Architektur der Btx-basierenden Anwendung „Gebrauchtwagenbörse“ kann als Grundlage für die Entwicklung der WWW-Anwendung „Jahreswagenbörse“ verwendet werden. Da die für die Anwendung benötigten Daten sowohl auf einem Großrechner als auch auf einem Unix-basierten System zur Verfügung stehen, beschränkt sich die Evaluierung der Anbindungsmöglichkeiten der JAWA somit auf Datenbanken auf Großrechnern und auf Unix-basierte relationale Datenbanksysteme.

4.4 Die JAWA-Datenbank

Da die Jahreswagendaten nur ein Teil der Gebrauchtwagendaten auf dem Großrechner sind, wird untersucht, welche Möglichkeiten zur Verfügung stehen, um auf diese Daten zugreifen zu können. Als Grundlage der Untersuchung wird das in Kapitel 4.3 beschriebene Szenario verwendet.

Aus dem Gesamtbestand der Gebrauchtwagendaten in der hierarchischen IMS-Datenbank auf dem Großrechner wird einmal pro Nacht durch einen Batch-Auftrag ein Extrakt der Jahreswagendaten erzeugt und in einer sequentiellen Datei auf dem Großrechner abgelegt. Danach wird diese Datei vom Großrechner auf ein Unix-basiertes System übertragen (Abbildung 4.1). Der Datentransfer wird über das File Transfer Program FTP abgewickelt und erfolgt zeitgesteuert einmal am Tag. Dei Initiative geht dabei vom Unix-basierten Host-System aus. Der Mechanismus der Übertragung ist ein Shell-Skript, das mit Hilfe des Unix-Mechanismus „crontab“ gestartet wird. Ist die sequentielle Datei auf das Unix-System transferiert, können die Jahreswagendaten in die relationale Datenbank übertragen werden und stehen dort zur weiteren Verarbeitung zur Verfügung. Es ist vorgesehen das relationale Datenbanksystem Oracle als JAWA-Datenbank einzusetzen, wobei auch jede andere relationale Datenbank hierfür verwendet werden kann.

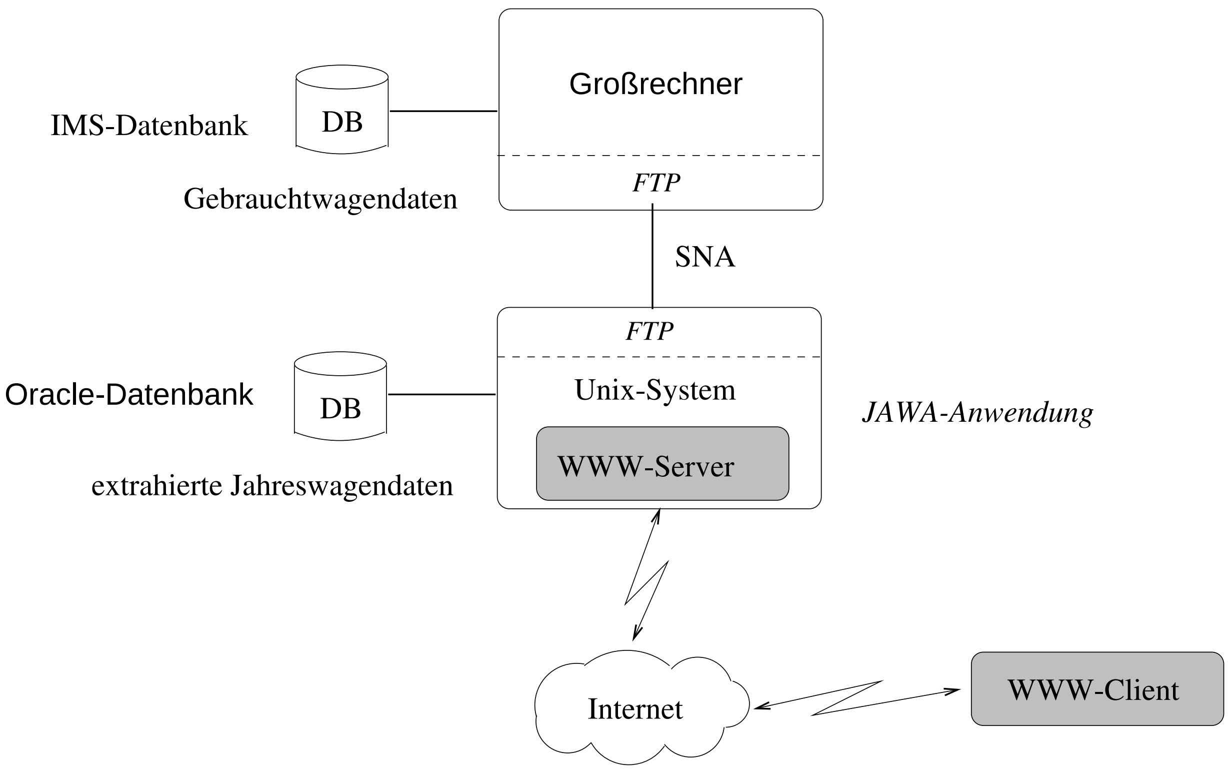

Abbildung 4.1: Transport der JAWA-Daten auf ein Unix-basiertes System

Durch das Zurverfügungstellen der Daten in einer Unix-basierten, relationalen Datenbank ergeben sich sowohl für den Benutzer der JAWA als auch für den Informationsanbieter zahlreiche Vorteile. Mit der „Standard Query Language“ (SQL) steht ein mächtiger Standard zur Verfügung, mit dem sich Zugriffe auf relationale Datenbanksysteme einfach und komfortabel realisieren lassen. Die Datenbankzugriffe beschränken sich auf SQL-Anweisungen, die in das Datenbank-Gateway eingebettet sind. Die Daten der Anwendung lassen sich mit Hilfe des relationalen Modells, im Gegensatz zu den hierarchischen Datenbanksystemen, einfach modellieren. Es sind keine Zugriffe über SNA auf den Großrechner nötig, da sowohl Datenbank-Gateway als auch Datenbank-Server auf einem Unix-basierten System laufen. Die Datenbankzugriffe erfolgen dadurch sehr schnell und einfach.

Aufgrund der beschrieben Architektur beschränkt sich die Evaluierung der Anbindungsmöglichkeiten des Datenbank-Gateways an die JAWA-Datenbank auf Unix-basierte, relationale Datenbanksysteme.

4.5 Gateway-Architekturen

Für die Anbindung an SQL-basierte relationale Datenbanksysteme werden im Folgenden Architekturen von Datenbank-Gateways beschrieben und ihre Vor- und Nachteile gegeneineinander abgewägt.

4.5.1 Standalone-Gateway

Wie in Abbildung 4.2 dargestellt erfolgt beim Standalone-Gateway der Zugriff auf die Datenbank direkt über ein externes CGI-kompatibles Programm, das auch als CGI-Skript bezeichnet wird (Kap. 4.2). Das CGI-Skript wird von einem WWW-Client aus referenziert. Der WWW-Server erkennt anhand der in der Client-Anfrage angegebenen URL, das in diesem Fall kein statisches Dokument zurückzuliefern ist, sondern ein externes Programm augerufen werden muß. Der WWW-Server versorgt das externe Programm über das CGI mit Parameter. Das Programm dekodiert, falls notwendig, die übergebenen Parameter und generiert daraus dynamisch eine Datenbankanfrage. Danach wird eine Verbindung zum Datenbank-Server aufgebaut und das Programm meldet sich dort an. Die Kommunikation zwischen dem externem Programm und dem relationalen Datenbanksystem (RDBMS) erfolgt über die Structured Query Language (SQL), wobei hierzu auch andere Sprachen verwendet werden können. Nach erfolgreicher Anmeldung wird die SQL-Anfrage an den Datenbank-Server geschickt, der die Anfrage verarbeitet. Der Datenbank-Server schickt die extrahierten Daten an das CGI-Skript zurück, das daraus ein HTML-Dokument generiert und auf die Standardausgabe schreibt. Der WWW-Server schickt das erstellte HTML-Dokument an den Client zurück. Die zuvor erstellte Verbindung zum Datenbank-Server wird vom CGI-Skript wieder abgebaut.

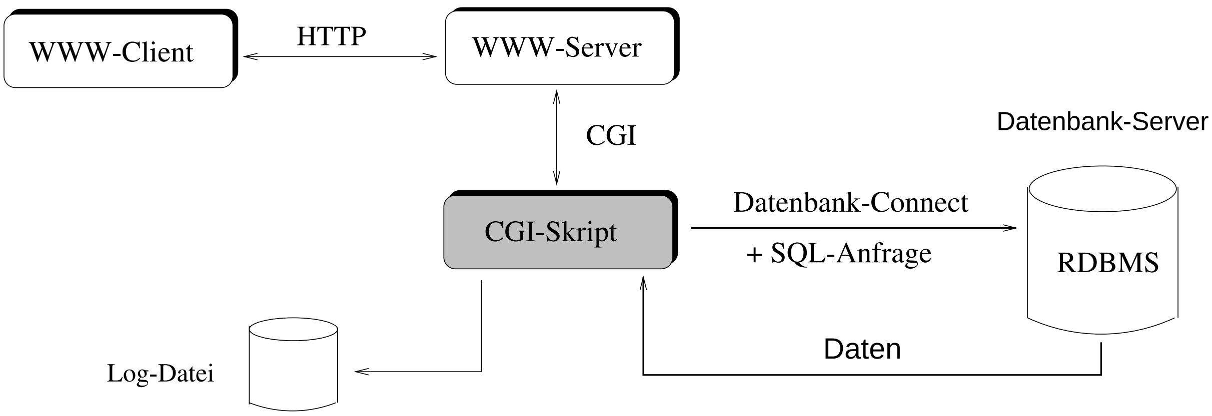

Abbildung 4.2: Standalone-Gateway

Bei dieser Art der Anbindung wird für jede Client-Anfrage durch den WWW-Server ein neuer lokaler Prozeß gestartet. Dieser ist in Bezug auf die Zeit sehr teuer und belastet den Server zusätzlich, da er das externe Programm aufrufen und über das CGI mit Parametern versorgen muß. Für jede Client-Anfrage muß außerdem eine neue Verbindung zum Datenbank-Server aufgebaut werden, wodurch die Antwortzeiten für den Benutzer verlängert werden. Tests haben ergeben, daß die Zeit für den reinen Verbindungsaufbau, je nach Auslastung der Datenbank, im Bereich von eins bis fünf Sekunden liegt.

Betrachtet man die möglichen Fehlerquellen, die den Betrieb der Jahreswagenbörse beeinflussen könnten, wird man feststellen, daß bei diesem Datenbank-Gateway hierfür nur das Übertragungsmedium sowie das externe Programm selbst in Frage kommen. Ist die Verbindung zwischen dem CGI-Skript und dem Datenbank-Server aufgrund eines Fehlers im Übertragungsmedium unterbrochen, kann der Informationsdienst JAWA nicht genutzt werden. Diese Fehlerquelle kann jedoch ausgeschlossen werden, wenn Datenbank-Server, WWW-Server und externes Programm auf dem gleichen Rechner laufen. Sollte ein Fehler während der Ausführung des Programm auftreten, wird es beendet und der Benutzer darüber informiert. Dieser hat dann die Möglichkeit, die zuvor gestellte Anfrage zu wiederholen.

Vorteile des Standalone-Gateways:

- Geringe Komplexität
- Einfache Implementierung
- Geringe Fehleranfälligkeit
- Gute Wartbarkeit

Nachteile:

- Ein lokaler Prozeßaufruf je Client-Anfrage
- Ein Verbindungsaufbau zum Datenbank-Server je Anfrage

4.5.2 Client/Server-Gateway

Bei der Client/Server-Lösung erfolgt der Zugriff des externen Programms, das im Folgenden auch als JAWA-Client bezeichnet wird, indirekt über einen weiteren Prozess, den Connect-Daemon bzw. den JAWA-Server, der die Verbindung zur Datenbank hält (Abbildung 4.3). Der JAWA-Client übermittelt die vom WWW-Server erhaltenen Daten an den JAWA-Server, der daraus dynamisch eine SQL-Anfrage aufbaut und diese an den Datenbank-Server weiterleitet. Nach der Verarbeitung der Anfrage durch den Datenbank-Server, schickt dieser die extrahierten Daten an den JAWA-Server zurück, der sie an den JAWA-Client weiterleitet. Nach Erhalt der Daten generiert der JAWA-Client daraus ein

HTML-Dokument und schreibt es auf die Standardausgabe, von der es durch den WWW-Server an den Benutzer zurückgeschickt wird.

Um den JAWA-Client zu starten, ist wie bei dem Standalone-Gateway jeweils ein lokaler Prozeßaufruf notwendig, Für die Interprozesskommunikation zwischen JAWA-Client und JAWA-Server ist ein Mechanismus zu verwenden, der entweder auf Remote Procedure Calls (RPCs), Unix-Sockets, Shared Memory oder auf einem Batch-System basiert. Kommen JAWA-Client und JAWA-Server auf unterschiedlichen Maschinen bzw. Systemumgebungen zum Einsatz, müssen die zu transferierenden Daten vor der Übertragung serialisiert bzw. in ein einheitliches Datenrepräsentationsmodell umgewandelt werden. Um weitere Anfragen von JAWA-Clients zwischenspeichern zu können, muß der JAWA-Server einen Warteschlangenmechanismus implementieren, da er während der Bearbeitungsphase für alle weiteren Anfragen blockiert ist.

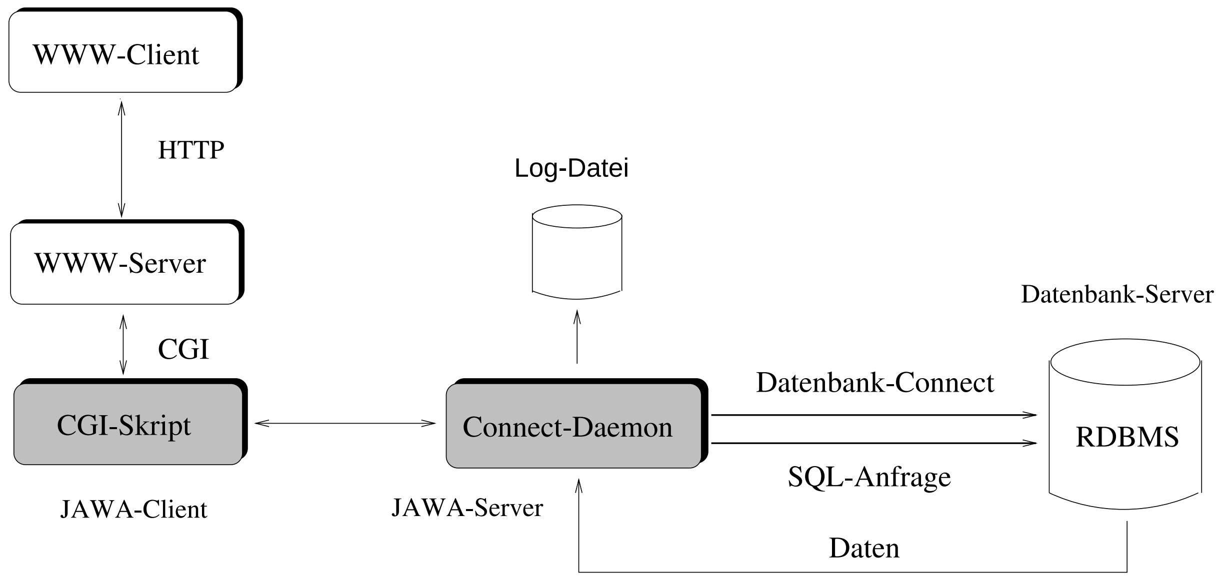

Abbildung 4.3: Client/Server-Gateway

Beim Start des JAWA-Servers wird eine Verbindung zum Datenbank-Server aufgebaut und für alle weiteren Transaktionen aufrecht erhalten. Im Vergleich zum Standalone-Gateway hätte dies primär betrachet eine Verkürzung der Antwortzeiten zur Folge. Da durch die Blockierung des JAWA-Servers Wartezeiten für andere JAWA-Clients entstehen und weil die Antwortzeiten im WWW hauptsächlich von der Netzlast, den verfügbaren Übertragungsraten sowie der Auslastung der WWW-Server abhängen, kann der durch die Client/Server-Architektur erlangte Zeitvorteil vernachlässigt werden. Der gewonnene Zeitvorteil wird weiterhin durch zusätzlichen Bearbeitungsaufwand im JAWA-Client und JAWA-Server sowie durch die Zeit, die für die Interprozesskommunikation in Anspruch genommen wird, geschmälert.

Betrachtet man die möglichen Fehlerquellen, die den Betrieb des Informationsdienstes JAWA beeinflussen könnten, wird man feststellen, daß bei dieser Gateway-Architektur neben dem Übertragungsmedium und dem JAWA-Client nun auch der JAWA-Server und der Kommunikationsmechanismus zwischen JAWA-Client und -Server als Fehlerquellen in Frage kommen. Das Übertragungsmedium kann als Fehlerquelle ausgeschlossen werden, wenn sowohl WWW-Server und JAWA-Client als auch JAWA-Server und Datenbank-Server auf der gleichen Maschine laufen. Tritt ein Fehler im JAWA-Client bzw. während der Kommunikation zwischen Client und Server auf, kann dies durch den Client festgestellt und an den Benutzer weitergeleitet werden. Dieser kann daraufhin seine Anfrage an den Datenbank-Server wiederholen. Bei einem Ausfall des JAWA-Servers, der nicht immer sofort erkannt und behoben werden kann sowie bei einem Ausfall der Maschine, auf dem der Server läuft, ist die gesamte Jahreswagenbörse nicht mehr verfügbar. Um in diesem Fall die Verfügbarkeit der Anwendung möglichst hoch zu halten, wäre es denkbar einen weiteren Server-Prozess bereitzustellen, der die Verfügbarkeit des JAWA-Servers überprüft und im Fehlerfall die zuständige Person alarmiert.

Eine weitere Alternative wäre der Einsatz mehrerer JAWA-Server (Pool von JAWA-Server), die über eine Art „Directory-Server“ verwaltet werden. Der Directory-Server überprüft ständig die Verfügbarkeit der JAWA-Server und weist den anfragenden JAWA-Clients den JAWA-Server zu, der gerade verfügbar ist. Damit könnte auch das Problem der Wartezeiten der JAWA-Clients bei Blockierung des JAWA-Servers gelöst werden.

Diese Ansätze könnten zwar angewandt werden, um die in der Lösung auftretenden Fehlerquellen zu minimieren, sie würden aber zugleich die Komplexität des gesamten Systems erhöhen und eine Implementierung wesentlich erschweren. Auch würde dadurch der Zeitgewinn, den man sich durch den Einsatz der Client/Server-Architektur erhofft hat, weiterhin geschmälert.

Die höhere Komplexität des Systems und die aufwendigere Implementierung würden zugleich eine höhere Fehleranfälligkeit als beim Standalone-Gateway bedeuten. Kommt beim Standalone-Gateway nur das CGI-Skript als Fehlerquelle in Frage, können beim Client/Server-Gateway sowohl Fehler im JAWA-Client und -Server als auch Fehler während der Interprozesskommunikation zwischen beiden Partnern auftreten. Werden zwischen JAWA-Server und -Client beispielsweise Unix-Sockets als Kommunikationsmechanismus eingesetzt, könnten zum Beispiel während der Serialisierung der Daten Fehler auftreten, die eine Kommunikation unterbrechen würden.

Vorteile des Client/Server-Gateways:

- Reduzierung der Antwortzeiten gegenüber Standalone-Gateway, die aus oben genannten Gründen in der Realität jedoch fraglich sein wird

Nachteile:

- Ein lokaler Prozessaufruf je Client-Anfrage

- Höhere Fehleranfälligkeit als Standalone-Gateway
- Hohe Komplexität der Anwendung und somit aufwendigere Implementierung als bei Standalone-Gateway

4.5.3 WWW-Server mit SQL-Funktionalität

Betrachtet man nur den Faktor Antwortzeit, stellt dieser Ansatz die optimale Lösung für die Anbindung eines WWW-Servers an eine relationale Datenbank dar. Der Zugriff auf die Datenbank erfolgt, wie in Abbildung 4.4 dargestellt, direkt vom WWW-Server aus, ohne dabei ein externes Programm zu verwenden. Der HTTP-Server baut beim Start eine Verbindung zum Datenbank-Server auf und hält diese für alle weiteren Client-Anfragen aufrecht. Erhält der Server eine Anfrage eines Clients, generiert er daraus eine SQL-Anfrage und leitet sie direkt an den Datenbank-Server weiter. Im Gegensatz zu den anderen beiden Lösungen ist hier nur noch ein lokaler Prozeduraufruf notwendig, um die Daten zum Datenbank-Server zu transferieren. Da der Prozeßaufruf zum Start des externen Programms und der Verbindungsaufbau je Client-Anfrage entfallen, könnte man sich dadurch eine erhebliche Reduzierung der Antwortzeiten erhoffen. Da, wie schon erwähnt, die Antwortzeiten im WWW hauptsächlich von der Netzlast, den verfügbaren Übertragungsraten sowie der Auslastung der WWW-Server abhängig sind, ist fraglich, ob sich in der Realität der aus dieser Lösung enstandene Zeitvorteil auf die Antwortzeiten im WWW auswirkt.

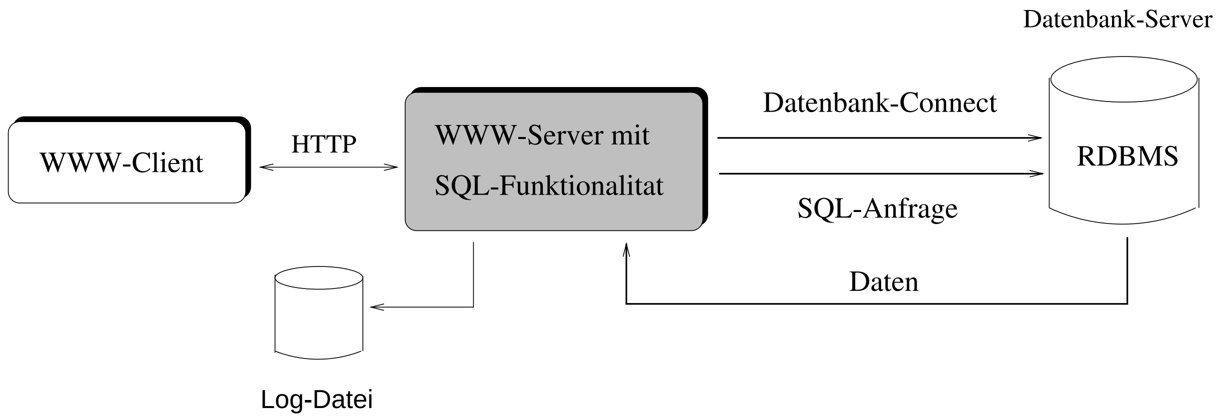

Abbildung 4.4: WWW-Server mit SQL-Funktionalität

Derzeit gibt es Ansätze von CERN sowie von Netscape Communications, die diese Art von Datenbankzugriff erlauben. Am CERN wurde ein Prototyp eines WWW-Servers entwickelt, der als Source Code verfügbar ist und den eigenen Anforderungen entsprechend angepaßt werden kann. Netscape Communications bietet eine Server-API (NSAPI) an, mit der man die Netsite Server um die entsprechende Funktionalität erweitern kann.

Der große Nachteil dieser Lösung liegt darin, daß man an einen bestimmten WWW-Server gebunden ist. Da das WWW ein sehr junger Internet-Dienst ist, werden ständig neue Standards sowohl im Bereich der WWW-Server als auch im Bereich des HTTP-Protokolls entwickelt. In der Tat gibt es derzeit Bestrebungen [Spe95c], die darauf zielen, das HTTP-Protokoll im Bezug auf die Performance zu verbessern und neue Sicherheitsmechanismen zu integrieren. Hat man einmal einen WWW-Server um die SQL-Funktionalität erweitert, ist man an diesen gebunden, will man sich die Arbeit nicht erneut machen und einen neuen Server um eine derartige Funktionalität erweitern. Bei dem Standalone- sowie Client/Server-Gateway verhält es sich anders. Dort können die WWW-Server beliebig ausgetauscht werden, sofern sie dem CGI-Standard entsprechen. Da das CGI selbst ein standardisiertes Verfahren ist, ist davon auszugehen, daß auch zukünftige WWW-Server das CGI unterstützen.

Betrachtet man wie bei den anderen beiden Lösungsansätzen die möglichen Fehlerquellen, welche die Verfügbarkeit der Jahreswagenbörse einschränken könnten, kommen hierfür nur das Übertragungsmedium sowie der WWW-Server in Frage. Das Übertragungsmedium kann als Fehlerquelle ausgeschlossen werden, falls WWW-Server und Datenbank-Server auf der gleichen Maschine laufen. Fällt der WWW-Server aus, ist der gesamte Dienst WWW nicht mehr verfügbar.

Vorteile der Lösung:

- Kein Prozeßaufruf für externes Programm mehr notwendig
- Reduzierung der Antwortzeit durch direkte Kommunikation und permanente Verbindung zur Datenbank
- Geringe Komplexität
- Geringe Fehleranfälligkeit
- geringe Netz- und Systemlast durch direkte Kommunikation und weil kein Prozeßaufruf für externes Programm mehr notwendig ist

Nachteile:

- Der WWW-Server wird aufgebläht
- Blockierung des WWW-Servers für die Zeit der Anfrage an die Datenbank
- Erweiterung eines WWW-Servers ist nur möglich, wenn dessen Source Code verfügbar ist oder wenn der Hersteller eine API bereitstellt, über die der Server erweitert werden kann
- Bindung an den erweiterten WWW-Server

4.6 Alternative Datenhaltung für die JAWA

Neben der in Kapitel 4.4 beschriebenen Speicherung der Jahreswagendaten in einem relationalen Datenbanksystem bietet sich die Möglichkeit an, eine sequentielle Datei als Datenbasis für die JAWA zu verwenden. Anstatt die Jahreswagendaten nach dem Transfer vom Großrechner auf das Unix-basierte System in die dortige relationale Datenbank einzuspielen, könnten die Daten als sequentielle Datei („flat file“) dem CGI-Skript zur Verfügung gestellt werden. Das bedeutet, daß die sequentielle Datei mit dem Extrakt der Jahreswagendaten nur noch vom Großrechner auf das Unix-basierte System übertragen wird und dort als Datenbasis für die JAWA verwendet werden kann (Abbildung 4.5). Damit beschränken sich die Zugriffe des CGI-Skripts nur noch auf Dateizugriffe.

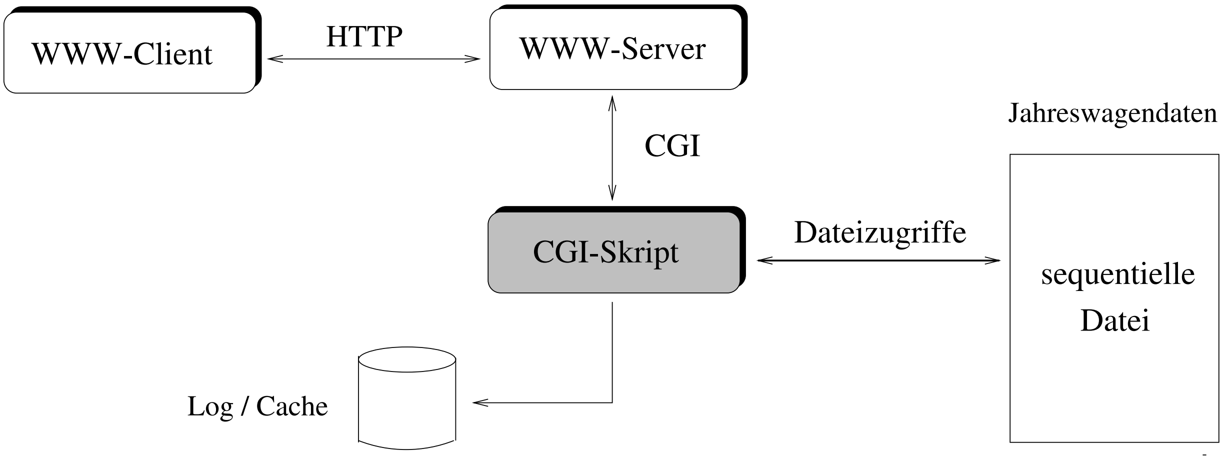

Abbildung 4.5: Sequentielle Datei als Datenbasis für die JAWA

Dies bedeutet, daß keine teuren Datenbankzugriffe mehr notwendig sind, um die gewünschten Daten aus dem Datenbestand zu extrahieren. Anstatt aufwendiger SQL-Anweisungen verwendet das CGI-Skript ausschließlich einfache Dateioprationen für den Zugriff auf die Daten. Zudem muß nicht explizit ein relationales Datenbanksystem angeschafft und installiert werden, falls es nicht schon vorhanden ist und für die JAWA verwendet werden kann. Betrachtet man die mögliche Archtiektur des CGI-Skripts, kommt dafür nur das „Standalone-Gateway“ in Frage. Anstatt mit dem relationalen Datenbanksystem zu kommunizieren, operiert das CGI-Skript ausschließlich auf einer Datei. „Client/Server-Gateway“ und „WWW-Server mit SQL-Funktionalität“ kommen als Architekturmodell bei dieser Alternative nicht in Frage, weil das CGI-Skript keine explizite Verbindung zu einem weiteren Kommunikationspartner, wie beispielsweise dem Datenbanksystem, aufrechterhalten muß. Weitere Vorteile dieses Ansatzes sind die geringe Komplexität der Anwendung und die daraus resultierende einfache Implementierung. Betrachtet man die Fehleranfälligkeit des Systems, stellt man fest, daß nur das CGI-Skript selbst als Fehlerquelle in Frage kommt. Bei der Implementierung des CGI-Skripts würde sich *perl* [WS91] als Programmiersprache anbieten. Es zeichnet sich durch das Bereitstellen einfacher und sehr effizienter Möglichkeiten aus, auf Dateien zuzugreifen und diese zu modifizieren. Außerdem enthält *perl* eine gute „Pattern Matching"-Funktionalität.

Neben den vielen Vorteilen stehen diesem Ansatz auch einige Nachteile gegenüber. Enthält die sequentielle Datei sehr viele Datensätze, erweist sich die Wartung der Datei als schwierig, da im Gegensatz zu relationalen Datenbanksystemen kein Standard wie beispielsweise SQL existiert, um die Datensätze einfach und komfortabel zu modifizieren. Das Ändern von Datensätzen, wie beispielsweise das Entfernen, Hinzuzufügen oder Modifizieren, erfordert das manuelle Editieren der Datei. Ein weiteres Problem stellt die Modellierung der Daten dar. Einfache Datenstrukturen lassen sich ohne größeren Aufwand auf Daten in Dateien abbilden. Problematischer erweist sich die Lage bei komplexen Datenstrukturen. Lassen sich beispielsweise Daten im relationalen Modell durch eine Tabelle modellieren, ist die Abbildung des relationalen Modells auf eine Datei einfach. Möchte man aber beispielsweise komplexere Datenstrukturen in einer Datei abbilden, können dabei Probleme auftreten. Betrachtet man die Btx-basierende Anwendung „Gebrauchtwagenbörse" [BMW92], sind dort mehrere Tabellen notwendig, um die Daten im relationalen Modell zu modellieren. Eine Abbildung dieses Modells auf eine Datei würde sich als äußerst schwierig erweisen. Je mehr Datensätze eine Datei enthält desto zeitaufwendiger gestaltet sich das Suchen innerhalb dieser Datei. Der Zeitvorteil, der sich durch den Zugriff auf eine Datei ergibt, wird zunehmend kleiner, je größer die Datei ist, auf die zugegriffen wird. Ab einer bestimmten Dateigröße ist es sogar vorteilhafter, die Daten in einer relationalen Datenbank zu speichern, weil die Datenbankzugriffe dann trotz des zeitaufwendigen Verbindungsaufbaus schneller erfolgen als normale Dateizugriffe. Betrachtet man beim Design eines solchen Gateways den Betriebsaspekt „Caching", stellt man fest, daß man einen eigenen Cache entwickeln und implementieren muß, um Benutzeranfragen zwischenzuspeichern. Bei Verwendung von relationalen Datenbanksystemen als Datenbasis ist dies nicht erforderlich, weil dort das Datenbanksystem selbst das Zwischenspeichern von Anfragen übernimmt.

Für die JAWA besteht zwar die Möglichkeit eine sequentielle Datei mit den Jahreswagendaten als Datenbasis für die Anwendung zu verwenden. Da aber der zu entwickelnde Prototyp der JAWA als Grundlage für die Entwicklung weiterer WWW-Anwendungen dienen soll, beispielsweise als Grundlage einer möglichen WWW-basierten „Gebrauchtwagenbörse", wird ein relationales Datenbanksystem als Datenbasis für die Anwendung vorgezogen. Anhand des Prototypen der JAWA soll der Zugriff auf Datenbanken aus dem WWW heraus exemplarisch dargestellt werden.

4.7 Gateway-Strategien

Es gibt verschiedene Möglichkeiten, um ein CGI-Skript zu implementieren, das als Gateway zwischen dem WWW und einem Datenbanksystem agiert. Die einzelnen Ansätze sowie deren Vorteile und Nachteile werden im Folgenden beschrieben.

4.7.1 Statische Gateways

Bei statischen Datenbank-Gateways handelt es sich um CGI-Skripten, bei denen das Format der HTML-Eingabemasken und Ausgabeseiten fest vorgegeben ist und der Zugriff auf die Datenbank statisch erfolgt. Dies bedeutet, daß die für einen Datenbankzugriff notwendigen Daten, wie beispieslweise der Name der Datenbank und der Tabellen, auf die zugegriffen wird, die Benutzerkennung und das Paßwort, die Art des Zugriffs sowie der Name und der Typ der Tabellenspalten, im Gateway fest verankert sind. Möchte man das Format der HTML-Seiten oder die dem Datenbankzugriff zugrundeliegenden Daten ändern, muß hierzu der Source-Code modifiziert werden. Ein erneutes Übersetzen der Anwendung wird dadurch erforderlich. Die Änderung des Source-Codes setzt voraus, daß die Person, die die Änderungen vornimmt, mit der Anwendung und mit dem Source-Code vertraut ist und weiß, an welcher Stelle die entsprechenden Änderungen vorzunehmen sind. Der Vorteil dieser statischen Gateways liegt in der schnellen und einfachen Implementierung, wie das folgende Beispiel beweist.

```
#! /bin/csh
setenv ORACLE_HOME /path/to/oracle
setenv PATH ${ORACLE_HOME} /bin:${PATH}
echo 'Content-type: text/html /n/n'
echo '<title>TUM-Studenteninformation</title>'
echo '<h1>TUM-Studenteninformation</h1>'
echo '<pre>'
sqlplus -s scott/tiger@ << _eof_
select * from student;
_eof_
echo '<pre>'
```

Mit nur wenigen Zeilen Code und unter Zuhilfename des Oracle-spezifischen Tools `SQL*Plus` wurde ein statisches Datenbank-Gateway realisiert. Es greift unter der Kennung `scott/tiger` auf die Tabelle `student` zu und liefert als Ergebnis alle darin gespeicherten Daten.

4.7.2 Dynamische Gateways

Ein dynamisches Datenbank-Gateway ist ein CGI-Skript, bei dem der Zugriff auf die zugrundeliegende Datenbank dynamisch erfolgt. Man kann dabei zwei Arten von dynamischen Gateways unterscheiden.

Bei der ersten Art wird eine Art Meta-Datenbank verwendet, die dynamische Informationen über die zugrundeliegende Datenbank und deren Tabellen enthält. Anhand dieser

Informationen, die vom Datenbank-Gateway ausgelesen werden, wird dann das HTML-Formular für den Benutzer dynamisch aufgebaut. Bei dieser Meta-Datenbank handelt es sich entweder um Systemtabellen, sog. „data dictionaries“, der Datenbanksysteme, in denen die erforderlichen Informationen gespeichert werden oder um externe Konfigurationsdateien, die vom Administrator der Anwendung explizit erstellt und vom Datenbank-Gateway zur Laufzeit ausgewertet werden. Diese Meta-Datenbanken enthalten beispielsweise die Namen der Datenbank und der Tabellen sowie der Tabellenspalten, auf die zugegriffen werden soll, eventuelle „Joins“ zwischen den Tabellen, falls auf mehrere Tabellen zugegriffen wird, sowie die Benutzerkennung und das Paßwort, das für den Datenbankzugriff verwendet werden soll. Einem dynamischen Gateway können dabei eine oder mehrere dieser Meta-Datenbanken zugrundeliegen. Sind beispielsweise mehrere Konfigurationsdateien vorhanden, teilt der Benutzer bei seiner Anfrage dem CGI-Skript mit, welche Datei bei dem aktuellen Zugriff verwendet werden soll. Auf diese Weise lassen sich mit einem Datenbank-Gateway Zugriffe auf unterschiedliche Datenbanken sowie verschiedene Zugriffsarten, wie das Lesen, Einfügen oder Modifizieren der Daten innerhalb der Datenbank, realisieren. Die Public Domain Gateways „WDB“ [Ras95] und „GSQL“ [Ng93] beispielsweise verwenden Konfigurationsdateien als Basis für die Datenbankzugriffe. Das Tool „oraywww“, das Teil des „Oracle World Wide Web Interface Kits“ [Ora95] ist, bedient sich der Systemtabellen der zugrundeliegenden Datenbank, um daraus dynamisch das Eingabeformular für den Benutzer bzw. die Anfragen an die Datenbank zu generieren.

Eine weitere Möglichkeit dynamische Datenbank-Gateways zu implementieren besteht in der Verwendung statischer HTML-Dokumente, die Pseudo-HTML-Sprachelemente enthalten. Diese Pseudo-Sprachelemente werden durch das CGI-Skript verarbeitet. Wie bei den oben beschriebenen Meta-Datenbanken, enthalten die in das HTML-Dokumente eingebetteten Pseudo-Sprachelemente Informationen, die für die Generierung der Datenbankanfrage durch das CGI-Skript verwendet werden. Der Ablauf einer Anfrage ist dabei wie folgt: Der Benutzer adressiert das CGI-Skript und übergibt diesem zusätzlich den Pfad zu einem solchen statischen HTML-Dokument, in das die erwähnten Pseudo-HTML-Sprachelemente eingebettet sind. Das CGI-Skript verarbeitet das HTML-Dokument mit Hilfe einer Art Präprozessor, erkennt die Pseudo-HTML-Sprachelemente, baut entsprechend den darin beschriebenen Informationen dynamisch eine Anfrage auf und schickt sie an die Datenbank. Das Ergebnis der Anfrage wird anstelle der Pseudo-Sprachelemete in das HTML-Dokument eingefügt und an den Benutzer zurückgeschickt. Das Pseudo-Sprachelement

```
<SQL table=''Angestellte'' projection=''name,id,funktion,gehalt''
restriction=''abtnummer=12''>
```

beispielsweise könnte in ein HTML-Dokument eingebettet sein und durch ein entsprechendes CGI-Skript verarbeitet werden. Das Gateway erkennt anhand des Pseudo-Sprachelements `<SQL>`, daß daraus eine Datenbankanfrage zu generieren ist. Das Ergebnis der Anfrage wird dann anstelle dieses Pseudo-Sprachelements in das Dokument eingebettet und an den Benutzer zurückgeschickt. Das „DECOUX-Gateway“, das Teil des „Oracle World Wide

Web Interface Kits“ ist [Ora95], hat diesen Ansatz implementiert. Er wurde ebenfalls bei der Implementierung einer WWW-Version der OMNIS-Recherche [Cla94] für das an der TU-München existierende Literaturrecherche- und Bibliotheks-System OMNIS verwendet.

Der Vorteil der dynamischen Gateways gegenüber den statischen Gateways liegt in der Tatsache, daß die dynamischen Gateways flexibler sind gegenüber Änderungen und einfacher modifiziert werden können. Ein dynamisches Gateway kann für unterschiedliche Zugriffe auf diverse Datenbanken verwendet werden, ohne dabei den Source-Code der Anwendung ändern und neu übersetzen zu müssen. Der Administrator der Anwendung muß lediglich die Meta-Datenbanken in Form von Systemtabellen bzw. Konfigurationsdateien erstellen oder die erwähnten, vorgegebenen Pseudo-HTML-Sprachelemente in bestehende HTML-Dokumente einfügen, die dann vom Gateway für die Datenbank-Zugriffe verwendet werden. Der Administrator braucht nur noch mit der Konfiguration des Gateways vertraut sein und muß sich nicht mehr um den Source-Code der Anwendung kümmern.

4.8 Betriebsaspekte

Neben den unterschiedlichen Architekturmodellen sind auch folgende Betriebsaspekte beim Design eines WWW-Datenbank-Gatewas zu berücksichtigen.

4.8.1 Logging

Um Rückschlüsse aus den Zugriffen auf die Daten in der Datenbank ziehen zu können, müssen diese protokolliert werden. Da vom WWW-Server, wie in Kapitel 3.1 aufgeführt, nur Zugriffe auf HTML-Dokumente auf dem Server protokolliert werden, ist diese Log-Datei für solche Zwecke nicht geeignet. Aus diesem Grund ist eine eigene Protokolldatei zu führen, welche die Zugriffe auf die Datenbank protokolliert. Bei dem Standalone- sowie Client/Server-Gateway kommen hierfür sowohl der WWW-Server als auch das Gateway selbst in Frage. Weil die Gateways die vom Server übermittelte Benutzereingabe verarbeiten und mit dem Datenbank-Server kommunizieren, bietet es sich an, daß die Protokolldatei von ihnen geführt wird. Beim WWW-Server mit SQL-Funktionalität kommuniziert der Server direkt mit der Datenbank und verwendet hierzu kein externes Gateway. Aus diesem Grund muß er selbst neben den Zugriffen auf HTML-Dokumente auch die Zugriffe auf die Datenbank protokollieren.

Neben dem oben beschriebenen „Event Logging“, dem Aufzeichnen von Datenbankzugriffen, sollte auch das „Error Logging“, das Aufzeichen von aufgetretenen Fehlern, beim Design des Gateways berücksichtigt werden. Die auftretenden Fehler werden in einer eigenen Protokolldatei aufgezeichntet, die dazu verwendet werden kann, Fehlerfälle zu analysieren bzw. bei Bedarf Fehler zu erkennen und zu beheben. Das Führen einer Fehler-Protokolldatei muß Aufgabe des Gateways sein, da der WWW-Server nur Fehler beim Zugriff auf das Gateway protokolliert. Fehler die während dem Betrieb des Programms auftreten, blieben ansonsten unprotokolliert.

4.8.2 Caching

Ein Hauptspeicher- oder Festplattencache kann dazu verwendet werden, Zugriffe der externen Gateways bzw. des WWW-Servers auf die Datenbank zu beschleunigen. Die Entwicklung und Implementierung eines Caches, der direkt vom Gateway bzw. WWW-Server genutzt wird, erscheint aber sehr schwierig und aufwendig, da die vom Benutzer erstellten Anfragen sowie die aus der Datenbank extrahierten Daten sehr vielfältig sein können. Zudem stellt sich die Frage, ob bei Verwendung eines Cache die Antwortzeiten für den Benutzer reduziert werden können, da es fraglich ist, ob die Auswertung des selbst geführten Caches wesentlich schneller verläuft, als eine direkte Anfrage an den Datenbank-Server zu stellen. Da jedes Datenbankmanagementsystem selbst Anfrageoptimierungen durchführt und einen eigenen Cache unterhält, erscheint es sinnvoller, diesen zu nutzen, um Anfragen zu optimieren und zu beschleunigen. Dem Datenbankmanagementsystem stehen effizientere Möglichkeiten zur Verfügung, um gestellte Anfragen zwischenzuspeichern und auszuwerten. Außerdem können das Datenbanksystem bzw. der verwendete Cache über spezielle Parameter „getunt“ werden, was sich auf die Antwortzeit für den Benutzer zusätzlich positiv auswirkt. Für die Verwendung des Datenbank-Caches spricht auch, daß dieser bereits vorhanden ist und nicht erst entwickelt und implementiert werden muß.

4.8.3 Sicherheit

Benutzerauthentifizierung

Um den Zugriff auf die Datenbank einzuschränken und unerlaubte Zugriffe zu unterbinden, müssen Zugriffsschutzmechanismen im Datenbank-Gateway implementiert sein. Der Benutzer muß sich über eine gültige Kennung verbunden mit der Eingabe eines Paßwortes authentifizieren, bevor eine Verbindung zum Datenbank-Server aufgebaut wird. Erst nach einer gültigen Authentifizierung hat der Benutzer Zugriff auf die in der Datenbank zur Verfügung gestellten Daten. Um den Prozess der Anmeldung auszuführen, stehen dem Gateway mehrere Möglichkeiten zur Verfügung.

(a) Die Benutzerkennung und das Paßwort sind im Source Code des Gateways fest verankert.

    Das Gateway braucht die Zugriffsberechtigung des Benutzers nicht mehr zu prüfen. Diese Authentifizierungsmethode ist sehr sicher, da sowohl Benutzerkennung und Paßwort für Außenstehende nicht sichtbar sind. Ein Nachteil ist aber die geringe Flexibilität. Soll das Gateway abhängig von der Benutzerkennung gewisse Aktionen ausführen können, ist diese Methode hierfür nicht geeignet.

(b) Die Benutzerkennung und das Paßwort werden über ein HTML-Formular eingegeben.

Das Gateway muß vor dem Verbindungsaufbau zum Datenbank-Server die eingegebenen Daten überprüfen. Diese Methode ist sehr unsicher, da die eingegeben Daten unverschlüsselt übertragen werden. Wird bei dem HTML-Formular die `GET`-Methode verwendet, erscheinen sowohl die Benutzerkennung als auch das Paßwort unverschlüsselt im `Location`-Feld des WWW-Clients.

(c) Das HTML-Dokument, über das der Benutzer seine Datenbankanfragen initiiert, ist über den Zugriffskontrollmechanismus des WWW-Servers geschützt.

Bei dieser Methode ist der WWW-Server selbst für die Zugriffskontrolle verantwortlich. Nach vollzogener Authentifizierung wird der Benutzername dem CGI-kompatiblen Datenbank-Gateway in einer Umgebungsvariablen mitgeteilt.

Schutz vor Datenabzug

Um den angebotenen Informationsdienst vor unerlaubten bzw. unerwünschten häufigen Zugriffen zu schützen, müssen Maßnahmen in die Anwendung integriert werden, welche die Sicherheit der gespeicherten Daten gewährleisten und einen unerwünschten Abzug der bereitgestellten Daten unterbinden. Im Folgenden werden die verschiedenen zur Verfügung stehenden Möglichkeiten analysiert und bewertet.

Beschränkung der angezeigten Suchergebnisse Eine Möglichkeit besteht darin, die angezeigten Daten zu beschränken. Dies bedeutet, daß je Anfrage dem Benutzer nur eine bestimmte Anzahl an Daten angezeigt werden, obwohl in der Datenbank tatsächlich mehr Daten gefunden werden können. Dies hindert den Benutzer durch Formulieren allgemeiner Anfragen große Mengen an Daten mit nur einer Anfrage aus der Datenbank zu extrahieren. Diese Maßnahme ist einerseits sehr leicht zu implementieren, hindert aber andererseits den Benutzer nicht daran, durch Formulieren vieler Anfragen ein große Mengen an Daten aus der Datenbank abzuziehen. Der Benutzer braucht hierzu lediglich ein Skript entwicklen, das die Anwendung permanent aufruft und ihr unterschiedliche Suchkriterien als Parameter übergibt. Die Ergebnisse der Anfragen werden dann in einer Datei abgelegt. Der Benutzer kann sich so in relativ kurzer Zeit einen Überblick über den gespeicherten Datenbestand verschaffen.

Beschränkung der Anzahl der Zugriffe Die Beschränkung der Anzahl der Zugriffe für einen bestimmten Zeitraum ist eine weitere Möglichkeit, Daten vor unerlaubtem Abzug zu schützen. Mit Hilfe der im CGI-Standard definierten Umgebungsvariablen `REMOTE_HOST` bzw. `REMOTE_ADDR` und `REMOTE_IDENT` ließe sich diese Beschränkung realisieren. Die ersten beiden Variablen enthalten den Rechnernamen bzw. die IP-Adresse des Rechners, von dem die Client-Anfrage stammt. Die Variable `REMOTE_IDENT` wird mit der Benutzerkennung des Anwenders belegt, falls auf dem Client-Rechner ein Authentisierungs-Server läuft. Ein Benutzer wäre somit eindeutig über Benutzerkennung und Rechnername bzw. IP-Adresse identifizierbar. Damit ließe sich eine auf Benutzerkennung und Rechnername basierende Zugriffsbeschränkung realisieren. Da aber auf dem größten Teil der Unix-basierten sowie auf allen PC-basierten Systemen dieser Authentisierungsdienst nicht läuft, ist eine Zugriffsbeschränkung auf diese Weise nicht realisierbar.

Unterstellt man, daß jedem Benutzer ein eindeutiger Rechnername bzw. eine eindeutige IP-Adresse zuzuordnen ist, könnte die Zugriffsbeschränkung nur auf dem Rechnernamen oder auf der IP-Adresse basieren. Dabei tritt allerdings ein weiteres Problem auf. Wird die Kommunikation zwischen WWW-Client und WWW-Server über einen Proxy-Server abgewickelt, werden `REMOTE_HOST` und `REMOTE_ADDR` stets mit den Werten des Proxy-Servers belegt. Damit ist die rein IP-basierende Methode ebenfalls nicht sinnvoll, da der Originator einer Anfrage nicht eindeutig bestimmt werden kann.

Stellt der Betreiber der Anwendung seinen Dienst nur einer bestimmten Anwendergruppe zur Verfügung, kann innerhalb dieser Gruppe die Beschränkung der Anzahl der Zugriffe realisiert werden. Die WWW-Anwendung und somit das zugrundeliegende Skript werden dabei über den internen Zugriffschutzmechanismus des WWW-Servers vor unerlaubten Zugriffen geschützt. Nach Adressierung des Dienstes und erfolgreicher Benutzerauthentifizierung belegt der WWW-Server zusätzlich zu den anderen CGI-Umgebungsvariablen die Variable `REMOTE_USER`. Diese Variable enthält den Namen des Benutzers, der auf das geschützte CGI-Skript zugegriffen hat. Damit ließe sich dann eine Zugriffsbeschränkung realisieren. Diese Methode hätte zwar den Vorteil, die Daten vor unerlaubtem Datenabzug zu schützen, mit dem Nachteil, daß der angebotene Dienst nur für eine bestimmte, dem Informationsanbieter bekannte Anwendergruppe verfügbar ist.

Sind die Personen innerhalb der Anwendergruppe dem Informationsanbieter namentlich bekannt und weiß er auch, daß diese Personen tatsächlich existent sind, stellt diese Anwendergruppe keine Gefahr für die Anwendung dar. Da jeder Person eine eigene Kennung zugeordnet ist, ist die Anzahl der Zugriffe je Person beschränkt. Hat der Informationsanbieter jedoch keine Informationen über die Existenz einer Person, könnte sich beispielsweise ein Benutzer mehrere Kennungen unter falschem Namen verschaffen, die ihm den Zugriff auf den Informationsdienst ermöglichen. Er könnte dann das Zugriffskontingent jeder einzelnen Kennung ausschöpfen und so die Zugriffsbeschränkung umgehen. Um dies zu verhindern, könnte der Informationsanbieter bei der Registrierung einer Kennung den Namen des Benutzers, seine E-mail-Adresse, den Namen der Domäne, von der aus zugegriffen wird, sowie den Namen und die Adresse der Institution, über deren Domäne der Zugriff erfolgt, verlangen. Bevor eine Kennung an einen Benutzer vergeben wird, kann der Informationsanbieter anhand der ihm zur Verfügung stehenden Informationen überprüfen, ob der Benuzter, unter dessen Namen die Kennung für den Zugriff registriert wird, auch tatsächlich existiert. Auch wäre es denkbar, daß für jede Institution nur ein bestimmtes Kontingent an Kennungen zur Verfügung gestellt wird, um den Mißbrauch bei der Vergabe einer Kennung einzuschränken.

Soll der angebotene Informationsdienst nicht nur einer begrenzten Anwendergruppe zur Verfügung gestellt werden, ist es ratsam, die Zugriffe auf den Informationsdienst auf Basis der IP-Adressen bzw. Rechnernamen zu beschränken. Obwohl mit dieser Methode der Originator einer Anfrage nicht eindeutig bestimmt werden kann, wird er daran gehindert, sehr große Mengen an Daten in einem sehr kurzen Zeitraum abzuziehen. Es wäre beispielsweise vorstellbar, die Anzahl der Zugriffe je Stunde bzw. je Tag zu beschränken. Dies bedeutet, daß von einem Rechner aus nur eine bestimmte Anzahl an Zugriffen je Stunde bzw. je Tag erfolgen können. Wird die Beschränkung der Anzahl der Zugriffe mit der Beschränkung der der angezeigten Suchergebnisse gekoppelt, ist hier eine relativ wirksame Methode gegeben, um die gespeicherten Daten vor unerlaubtem Abzug zu schützen. Diese Methode bietet aber keine absolute Sicherheit, da der Benutzer über einen langen Zeitraum gesehen, trotzdem die Möglichkeit besitzt, große Mengen an Daten abzuziehen.

Werden die Zugriffe auf den Informationsdienst über einen Proxy-Server abgewickelt, ergibt sich aus der oben beschriebenen Methode ein Nachteil. Sind die Zugriffe beschränkt und wurde das Kontingent für einen bestimmten Zeitraum von einem oder mehrerer Benutzer bereits ausgeschöpft, hat ein weiterer Benutzer, der auch diesen Proxy-Server verwendet, nicht die Möglichkeit, den Informationsdienst in diesem Zeitraum zu nutzen. Die Anwendung erkennt, anhand der IP-Adresse bzw. des Rechnernamens, daß das zur Verfügung stehende Kontingent an Zugriffen erschöpft ist und verweigert weitere Zugriffe, obwohl die Zugriffe von verschiedenen Bentuzern initiiert wurden.

Um die IP-basierende Zugriffsbeschränkung zu implemtieren, gibt es für die verschiedenen Gateway-Architekturen unterschiedliche Ansätze. Das Standalone-Gateway muß die Zugriffe in einer Datei protokollieren, damit diese nach Beendigung der Anwendung persistent sind und somit bei einem erneuten Start zur Verfügung stehen. Das Lesen und Auswerten dieser Datei wirkt sich dabei negativ auf die Performance des Gateways aus und führt zu einer Erhöhung der Antwortzeit für den Benutzer. Das Client/Server-Gateway und der WWW-Server mit SQL-Funktionalität können die Zugriffe in einer Datenstruktur im Hauptspeicher oder in einer Datei auf Festplatte protokollieren. Aus Gründen der Performance ist die Variante im Hauptspeicher vorzuziehen, da sich dabei keine bzw. geringe Auswirkungen auf die Antwortzeit für den Benutzer ergeben.

Verschlüsselung der Datenbankzugriffe Eine weitere Möglichkeit, die Daten vor unerlaubtem Abzug zu schützen, besteht darin, die Zugriffe auf einzelne Objekte in der Datenbank, die über einen eindeutigen Schlüssel erfolgen, zu kodieren. Der Zugriff auf einzelne Datenbankobjekte erfolgt häufig über Hyperlinks aus HTML-Dokumenten heraus oder direkt durch Eingabe der Skript-URL zusammen mit dem eindeutigen Schlüssel. Die Parameterübergabe erfolgt dabei über die HTTP-Methode `GET`. Sind die Schlüssel in HTML-Dokumente eingebettet, können sie mit einfachen Mitteln extrahiert und gesammelt werden. Ist der Zugriff auf die Daten nicht beschränkt, hat damit jeder Benutzer die Möglichkeit, alle Schlüssel und somit alle in der Datenbank enthaltenen Daten abzuziehen.

Verwendet die WWW-Anwendung hingegen ein nur ihr bekanntes temporäres Verschlüsselungsverfahren, um die einzelnen Objekt-Schlüssel zu kodieren bzw. zu dekodieren, hat der Benutzer zwar weiterhin die Möglichkeit alle Objekt-Schlüssel zu sammeln, mit dem Unterschied, daß sie in diesem Fall kodiert sind und ohne die entsprechende WWW-Anwendung nicht dazu verwendet werden können, auf die einzelnen Datenbankobjekte zuzugreifen. Bei Verwendung eines temporären Schlüssels kann der Benutzer nur in der Zeit auf das dem Schlüssel zugrundeliegende Objekt zugreifen, in welcher der kodierte Schlüssel gültig ist. Dies bedeutet, daß die Zugriffe auf die Objekte in der Datenbank über einen kodierten Schlüssel nur auf einen bestimmten Zeitraum begrenzt sind.

Als Basis des Verschlüsselungsverfahrens könnte beispielsweise die aktuelle Zugriffszeit des Benutzers verwendet werden, mit Hilfe derer der neue temporäre Schlüssel berechnet wird. Ist der Gültigkeitszeitraum eines Schlüssels abgelaufen, verweigert die Anwendung den Zugriff auf das jeweilige Datenbankobjekt. Der Benutzer hat zwar die Möglichkeit, viele Schlüssel aus der Datenbank zu extrahieren, ihm steht aber nur noch ein begrenzter Zeitraum zur Verfügung, um mit Hilfe der extrahierten Schlüssel, die in der Datenbank gespeicherten Daten abzuziehen. Werden außerdem die Verschlüsselung der Datenbankzugriffe mit der Beschränkung der Suchergebnisse und der Beschränkung der Anzahl der Zugriffe kombiniert, besitzt die Anwendung eine sehr wirksame Methode, um die zugrundeliegenden Daten vor unerlaubtem Abzug zu schützen.

Werden die Zugriffe auf den Informationsdienst zusätzlich in einer Log-Datei protokolliert, kann diese dazu verwendet werden, die Wirksamkeit der implementierten Schutzmaßnahmen zu erhöhen. Hat ein Benutzer die Absicht, große Mengen an Daten abzuziehen, werden bei Beschränkung der Anzahl der Zugriffe und der angezeigten Suchergebnisse sowie einer Verschlüsselung der Datenbankzugriffe viele Zugriffe erforderlich, um große Mengen an Daten abzuziehen. Jeder einzelne Zugriff wird dabei in der Log-Datei protokolliert. Der Informationsanbieter kann mit Hilfe dieser Log-Datei das Zugriffsverhalten der einzelnen Benutzer analysieren. Erkennt er bei Auswertung der Datei, daß ein Benutzer permanent auf den Informationsdienst zugreift und sein Zugriffskontingent ständig ausschöpft, um Daten abzuziehen, kann er dies unterbinden, indem er den Zugriff für diesen Benutzer sperrt.

Kapitel 5

Style Guide für WWW-Anwendungen

Da das WWW ein sehr junger Dienst im Internet ist, gibt es noch sehr wenig Informationen und Erfahrungen über die Entwicklung von Anwendungen für das WWW. Aus diesem Grunde werden in diesem Kapitel Aspekte zusammengetragen, die bei der Entwicklung solcher Anwendungen sowohl für das äußere Erscheinungsbild als auch für die technische Realisierung als Ausgangspunkt verwendet werden können.

5.1 Layout-spezifische Aspekte

5.1.1 Seitenlayout

Eine WWW-Seite sollte in drei Teile untergliedert sein: den Kopf, den Rumpf sowie den Trailer. Der Kopf sollte das Logo der Anwendung und einen Schriftzug enthalten, damit der Benutzer, der durch Folgen eines Links auf diese Seite gelangt ist, weiß, wo er sich im Moment befindet. Das Logo sollte entsprechend dimensioniert sein. Der Rumpf enthält die eigentlichen für den Benutzer des Informationssystems bestimmten Informationen. Der Trailer bildet den Abschluß einer WWW-Seite und sollte eine Kombination aus folgenden Elementen enthalten:

- Navigationshilfen
- Name einer Kontaktperson in Form eines Hyperlinks
- Datum der letzten Änderung
- Datum des aktuellen Zugriffs

- Status bzw. Version der Anwendung

Es ist nicht sinnvoll, wenn jede dynamisch generierte Seite einer WWW-Anwendung alle oben aufgeführten Elemente enthält. Die „Home Page“ der Anwendung beispielsweise sollte die Version der Anwendung und das Datum von eventuellen Änderungen enthalten. Daneben ist es sinnvoll, den Namen einer Kontaktperson, wie beispielsweise den Namen des Verwalters der Anwendung, aufzuführen, um dem Benutzer die Möglichkeit zu gewähren, sich in bestimmten Situationen, wie beispielsweise bei Auftreten eines Fehlers, direkt an die zuständige Person zu wenden. Der Name sollte dabei als Hyperlink in das Dokument integriert sein. Etwaige Ergebnisseiten der Anwendung sollten im Trailer das Datum des aktuellen Zugriffs und Navigationshilfen enthalten, um den Benutzer die Möglichkeit zu geben, sich zwischen mehreren Seiten komfortabel bewegen zu können. Das Layout einer im WWW angebotenen Seite sollte den Richtlinien des Corporate Identity des Unternehmens entsprechen.

5.1.2 Präsentation von Daten

Darstellung von Tabellen

Für die Darstellung von Informationen, die als Tabellen organisiert sind, sieht HTML zwei Möglichkeiten vor. Es können hierfür die HTML 2.0-Elemente `<PRE>` und `</PRE>` sowie die HTML 3.0-Elemente `<TABLE>` und `</TABLE>` verwendet werden. Für eine Verwendung des Elements `<TABLE>` sprechen die dynamische Formatierung der Tabellenspalten durch den Browser sowie die kompakte Darstellunsweise der Tabelle im Dokument. Werden zur Darstellung Sprachelemente von HTML 3.0 verwendet, sollte in der „Home Page“ der Anwendung darauf hingewiesen werden, damit der Benutzer der Anwendung nicht überrascht ist, wenn er einen Browser verwendet, der nur HTML 2.0 implementiert und dadurch die WWW-Seiten nicht so erscheinen, wie es der Benutzer erwartet.

Sortieren von Daten

Werden durch eine Anwendung große Mengen an Daten angezeigt, ist es für den Benutzer von Vorteil, wenn diese sortiert dargestellt werden. Dabei können die Daten implizit durch die Anwendung oder explizit durch den Benutzer sortiert werden. Bei expliziter Sortierung muß dem Benutzer vor oder nach dem Generieren der Daten die Möglichkeit gegeben werden, die dargestellten Ergebnisse nach bestimmten Kriterien zu sortieren. Im Falle einer Datenbankanwendung würde dies bedeuten, daß der Benutzer durch das Auslösen der Sortieroption nach der Generierung der Daten, eine neue Datenbankanfrage mit den gleichen Auswahlkriterien startet, mit dem Unterschied, daß bei dieser Anfrage die Daten, entsprechend den Wünschen des Benutzers, sortiert aus der Datenbank extrahiert werden. Wird der Wunsch nach Sortierung der Ausgabe zusammen mit den Sortierkriterien eingegeben, können die Suche auf der Datenbank sowie das Sortieren der Daten mit einer Anfrage erledigt werden. Auf diese Weise können die Anzahl der Datenbankanfragen sowie die Last des Datenbanksystems reduziert werden.

Standardisierte HTML-Sprachelemente

Für die Erstellung der WWW-Seiten sollten nur standardisierte HTML-Elemente verwendet werden. Auf das Verwenden von proprietären Sprachelementen, die nur von einzelnen WWW-Clients unterstützt werden, sollte verzichtet werden, da diese Elemente von anderen Browsern nicht interpretiert und somit nicht korrekt bzw. überhaupt nicht dargestellt werden können. Ein Beispiel hierfür sind die Sprachelemente `<BLINK>` und `</BLINK>`, die nur vom Netscape Navigator implementiert werden und diesen Browser veranlassen, den markierten Text blinkend darzustellen.

Durchführung von Tests

Da HTML keine Layoutinformationen enthält, ist das konkrete Layout und somit die visuelle Darstellung einer WWW-Seite dem einzelnen WWW-Client überlassen. Die einzelnen Layoutinformationen wie Seitenformat, Seitengröße, Schriftgröße und Abstand zwischen den Absätzen werden von den jeweiligen Browsern unterschiedlich dargestellt. Es ist für den Entwickler einer Anwendung daher wichtig zu sehen, wie seine Anwendung und die darin generierten Seiten auf den einzelnen Clients dargestellt werden. Man sollte beispielsweise die erstellten WWW-Seiten sowohl auf Unix- und PC-basierten graphikorientierten WWW-Clients als auch auf bildschirmorientierten Browsern testen, um einen Überblick über die unterschiedlichen Darstellungsweisen zu erhalten.

Bei der Durchführung der Tests sollte stets beachtet werden, daß trotz der unterschiedlichen visuellen Darstellung durch die einzelnen WWW-Clients, die eigentlichen Informationen auf einer Bildschirmseite dargestellt werden, um dem Benutzer das Scrollen der Bildschirmseiten zu ersparen. Unter Umständen muß das Layout einer WWW-Seite so angepaßt werden, daß auch der Browser mit der „schlechtesten" Auflösung, die Informationen auf einer Bildschirmseite darstellen kann.

5.1.3 Vermeidung von Graphikflut

Das WWW als multimedialer Informationsdienst bietet die ideale Möglichkeit textbasierende Informationen zusammen mit Grafik, Audio und Video in ein Dokument einzubetten und diese Informationen schnell und einfach weltweit zugänglich zu machen. Im Gegensatz zur reinen Übertragung von Texten ist die Übertragung von Grafiken wesentlicher aufwendiger. Damit steigt nicht nur die Netzlast, sondern es hat auch eine Verlängerung der Anwortzeiten für den Benutzer sowie eine Erhöhung der Übertagungskosten zur Folge. Bei der Erstellung einer WWW-Anwendung ist zu beachten, daß den Nutzern des angebotenen Informationsdienstes in der Regel Übertagungsraten zwischen 2 Mbit/s und 9600 Baud zur Verfügung stehen, was sich bei der Übertragung einer WWW-Seite in unterschiedlichen Wartezeiten für den einzelnen Benutzer wiederspiegelt. Aus diesem Grund sollten bei der Entwicklung einer solchen Anwendung bzw. der daraus generierten WWW-Seiten folgenden Richtlinien beachtet werden:

- Keine Verwendung von Grafiken als Aufzählungspunkte

  Für Aufzählungen sind die dafür vorgesehenen HTML-Elemente sowie die von den WWW-Clients dargestellten Aufzählungspukte zu verwenden.

- Keine Verwendung von Grafiken als horizontale Trennlinien

  Soll die darzustellende Information durch horizontale Trennlinien strukturiert werden, ist dazu das HTML-Element `<HR>` zu verwenden.

- Vermeidung von Bildmaterial, das für den eigentlichen Informationsgehalt unbedeutend und viel zu groß dimensioniert ist

- Vermeidung von Inline Clickable Images als Home Pages

  Die Einstiegsseite einer Anwendung sollte nicht als aufwendige sensitive Grafik realisiert sein, da aufwendig gestaltete Inline Clickable Images sogar bei Benutzern, denen schnelle Übertragungsraten zur Verfügung stehen, zu langen Wartezeiten führen.

- Zwei Versionen der gleichen Seite anbieten

  Will ein Informationsanbieter nicht auf die graphische Vielfalt verzichten, sollte er zwei Versionen der entsprechenden WWW-Seite anbieten, um den Benutzer die Wahl der jeweiligen Seite zu überlassen. Die erste Version sollte reine Textinformationen enthalten, wobei die zweite aufwendige Grafiken enthalten kann.

- Verwendung von Graphiken im JPEG-Format

  Sollen aufwendige Bilder oder Grafiken in einer WWW-Seite verwendet werden, könnten sie im JPEG-Format in das Dokument eingebettet werden, da es sich bei JPEG im Vergleich zu GIF um ein komprimierteres Bildformat handelt.

  Dabei muß jedoch beachtet werden, daß ältere Versionen von WWW-Clients existieren, die dieses Bildformat nicht verarbeiten können und für dessen Darstellung einen External Viewer benötigen.

Werden WWW-Seiten trotzdem mit Graphiken überflutet, kann es vorkommen, daß nicht wie erwartet Benutzer dadurch angezogen werden, sondern die entgegengesetzte Wirkung eintritt. Der Benutzer wird durch die graphische Vielfalt eher abgeschreckt als angezogen, wenn er minutenlang warten muß, bis ein Bild geladen ist, das keinen oder nur geringen Informationsgehalt enthält.

5.2 Technische Aspekte

5.2.1 Informationsspeicherung in URLs

Das es sich bei HTTP um ein zustandsloses Protokoll handelt, werden auch Anwendungen im WWW zustandslos sein. Dies bedeutet, daß alle Informationen, die zu einem späteren Zeitpunkt benötigt werden, vom WWW-Client gespeichert werden müssen. Die zu speichernden Informationen werden vom CGI-Skript zurück an den Client geschickt, damit sie beim nächsten Aufruf des Skripts wieder empfangen und ausgewertet werden können. Die Zustandsinformationen werden dabei in der URL abgelegt und als Hyperlink in das vom CGI-Skript generierte Dokument eingebettet. Durch Aktivieren des Links wird das Skript referenziert und mit den im URL kodierten Informationen versorgt. Die Paramterversorgung wird dabei über das CGI abgewickelt. So können beispielsweise Objekte, die aus der Datenbank extrahiert wurden, in Hypertextlinks konvertiert werden und dadurch weitere Objekte in der Datenbank adressieren.

5.2.2 Navigationshilfen

Jede WWW-Seite sollte Navigationshilfen enthalten, um den Benutzer die Möglichkeit zu geben, sich zwischen mehreren Seiten komfortabel bewegen zu können, ohne dabei die Funktionalität des jeweiligen WWW-Clients nutzen zu müssen. Die Navigationshilfen sind als Hyperlinks in die Dokumente integriert. Enthält beispielsweise eine Anwendung mehrere Ergebnisseiten, kann sich der Benutzer dieser Anwendung zwischen den einzelnen Seiten wie in einem Programm bewegen. Dies wird dadurch erzielt, daß die den Navigationshilfen zugrundeliegenden URLs Informationen speichern, die notwendig sind, um den vorherigen oder den nächsten aktiven Zustand wieder herzustellen. Zudem sollte jede Ergebnisseite dem Benutzer die Möglichkeit bieten, eine neue Anfrage starten zu können, falls er mit dem Ergebnis der ersten Anfrage nicht zufriedengestellt werden konnte.

5.2.3 Kontextsensitive Hilfe

Jede Anwendung sollte über eine Online-Hilfe verfügen, auf die der Benutzer jederzeit zurückgreifen kann. Die Beschreibungen der Eingabefelder und Auswahlmenüs eines Formulars könnten beispielsweise als Hypertextlinks in das HTML-Formular eingebettet sein. Bei Anwählen des Links wird dann auf die entsprechende Stelle in der Online-Hilfe verzweigt, die eine ausführlichere Erläuterung zu den einzelnen Eingabefeldern und Auswahlmenüs liefert. Im Folgenden wird die prinzipielle Realisierung einer kontextsensitiven Hilfe beschrieben.

Es wird angenommen, daß die Online-Hilfe auf dem WWW-Server unter dem Pfad `/<Pfadname> /Hilfe.html` erreichbar ist. Um mit Hilfe eines Links von einer Textstelle eines

Dokuments zu einer Textstelle in einem anderen Dokument gelangen zu können, müssen sowohl Quelldatei als auch Zieldatei einen entsprechenden Verweis, einen sog. Anker enthalten. Die Hilfe-Datei, die zugleich die Zieldatei eines Links darstellt, muß folgende Textstelle enthalten:

```
<A NAME=‘‘<Indexname>‘‘>Zieltext</A>
```

Die Textstelle `Zieltext` in der Datei `Hilfe.html` wird als Anker verwendet und stellt das Ziel eines Hypertextlinks dar. Sie kann über den datei-eindeutigen Index `<Indexname>` von jedem anderen Dokument aus erreicht werden. Das Quelldokument hingegen muß folgende Textstelle enthalten:

```
<A HREF=‘‘http://<server>/<Pfadname> /Hilfe.html#<Indexname>‘‘>
Quelltext</A>
```

Durch Angabe des Dateinamens `Hilfe.html` und durch Hinzufügen des datei-eindeutigen Index `#<Indexname>` wird die Textstelle `Quelltext` in der Quelldatei zu einem Link auf die entsprechende Stelle in der Hilfe-Datei. Auf diese Art läßt sich beispielsweise für jedes Suchkriterium in einem HTML-Formular eine kontextsensitive Hilfe realisieren. Die technische Realisierung der kontextsensitiven Hilfe in der WWW-Anwendung „Jahreswagenbörse“ wird beispielhaft in Kapitel 6.5.5 beschrieben.

5.2.4 Fehlertoleranz

Die zu entwickelnde WWW-Anwendung sollte ein hohes Maß an Fehlertoleranz bezüglich der Benutzereingabe aufweisen. Dies impliziert, daß eine Anwendung, die eine Suche auf einer Datenbank realisiert, eine ergebnislose Suche so weit wie möglich ausschließt. Um dies zu realisieren, muß die Anwendung verschiedene Randbedingungen erfüllen.

Konsistenzprüfung der Benutzereingabe

Realisert die Anwendung eine Suche auf einer Datenbank, ist es sinnvoll die vom Benutzer eingegebenen Suchkriterien auf ihre Konsistenz hin überprüfen. Wird keine Konsistenzprüfung vollzogen, weiß der Benutzer im Falle einer ergebnislosen Suche nicht, ob das von ihm gesuchte Objekt momentan nicht in der Datenbank enthalten ist oder überhaupt nicht existiert. Eine Möglichkeit der Konsistenzprüfung besteht in der Bereitstellung einer eigenen Tabelle in der Datenbank, die alle unzulässigen Kombinationen von Suchkriterien enthällt. Der Anbieter des Informationsdienstes kann durch einfaches Hinzufügen bzw. Entfernen von Spalten aus der Tabelle die unzulässigen Kombinationen flexibel konfigurieren.

Dynamische Generierung der Auswahlfelder

Werden Auswahlmenüs und Auswahlfelder des Eingabeformular aufgrund der in der Datenbank enthaltenen Informationen dynamisch generiert, stehen dem Benutzer beim Erstellen einer Anfrage nur solche Kriterien zur Auswahl, die auch tatsächlich in der Datenbank enthalten sind. Der Benutzer kann somit keine Kriterien verwenden, die eine ergebnislose Suche auf der Datenbank verursachen würden. Die für das Eingabeformular zu verwendenden Informationen können dabei zum Zeitpunkt der Generierung des Eingabeformulars aus der Datenbank extrahiert werden oder in einer Konfigurationsdatei abgelegt sein, aus der sie durch die Anwendung bei Bedarf ausgelesen werden können. Die letzte Alternative ist in Bezug auf die Performance wesentlich besser, da die Informationen nicht bei jedem Start der Anwendung aus der Datenbank ausgelesen werden müssen. Damit wird die Antwortzeit für den Benutzer verkürzt und die Belastung der Datenbank reduziert.

Verwendung von Auswahlmenüs und -feldern

Werden in HTML-Formularen Auswahlmenüs und -felder anstelle von Texteingabefeldern verwendet, wird die Wahrscheinlichkeit für einen Benutzerfehler und somit die Anzahl der ergebnislosen Suchen, die aus Benutzerfehlern resultieren, reduziert. Die manuelle Eingabe der Kriterien wird durch die Auswahl vorhandener Kriterien ersetzt. Der Nachteil der Verwendung von Auswahlmenüs ist, daß der Benutzer in der Eingabe der Suchkriterien eingeschränkt ist. Er kann nur solche auswählen, die ihm in den Menüs angeboten werden. Bei Verwendung von Texteingabefeldern ist die Eingabe der Suchkriterien flexibler. Der Benutzer kann dort beliebige Kriterien eingeben, nach denen in der Datenbank gesucht wird.

5.2.5 Erweiterbarkeit

Die WWW-Anwendung sollte die Möglichkeit bieten, möglichst schnell an veränderte Umgebungen angepaßt werden zu können. Zusätzlich sollte die Oberfläche der Anwendung leicht veränderbar sein und neue zusätzliche Seiten sollen erstellt werden können ohne dabei die gesamte Anwendung grundsätzlich verändern zu müssen. Eine Realisiereungmöglichkeit besteht in der Verwendung von Konfigurationsdateien, die das Aussehen sowohl der Eingabeformulare als auch der Ausgabeseiten definieren. Möchte der Informationsanbieter Änderungen an den Seiten der Anwendungen vornehmen, geschieht dies ausschließlich durch Ändern einer oder mehrerer Konfigurationsdateien. Der Source-Code der Anwendung bleibt davon unberührt. Es wäre sogar vorstellbar, duch die Verwendung verschiedener Konfigurationsdateien unterschiedliche Informationsdienste mit der gleichen Anwendung zu realiesieren. Beim Start der Anwendung wird ihr der Name einer Konfigurationsdatei übergeben. Die Konfigurationsdatei beschreibt dann beispielsweise den Informationsdienst, das Aussehen der WWW-Seiten, die Daten, auf die zugegriffen wird sowie die Lokation der

Datenbank, in der die Daten gespeichert sind.

5.3 Organisatorische Aspekte

5.3.1 Konfiguration des verwendeten WWW-Servers und WWW-Clients

Soll ein WWW-Server einen Informationsdienst anbieten, dem ein CGI-Skript zugrundeliegt, muß der Server so konfiguriert werden, daß er auch in der Lage ist, diese Skripten auszuführen. In der Regel muß dazu in der dem Server zugrundeliegenden Konfigurationsdatei ein Eintrag vorgenommen werden, der ein bestimmtes Verzeichnis als das Verzeichnis kennzeichnet, im dem derartige CGI-Skripten abgelegt sind. Da der WWW-Client beim Zugriff auf das CGI-Skript die Lokation des Skripts auf dem WWW-Server nicht kennt, verwendet er einen virtuellen Pfadnamen, um das CGI-Skript zu referenzieren. Dies setzt natürlich voraus, daß in der Konfigurationsdatei des WWW-Servers eine Abbildungsregel eingetragen ist, die den virtuellen Pfad auf den tatsächlichen Pfad auf dem WWW-Server abbildet. Ansonsten ist der Zugriff für den WWW-Client nicht möglich.

Soll der Informationsdienst nur einer bestimmten Benutzergruppe zugänglich gemacht werden, muß das dem Informationsdienst zugrundeliegende CGI-Skript über den Zugriffschutzmechanismus des jeweiligen Servers geschützt werden. Die setzt voraus, daß der Administrator der Anwendung eine Benutzergruppe einrichtet und ihr den Zugriff auf das Skript ermöglicht. Allen Personen außerhalb dieser Gruppe wird der Zugriff auf den Informationsdienst verweigert.

Der Zugriff auf einen WWW-Informationsdienst erfordert für einen WWW-Client keine besondere Konfiguration. Er muß lediglich die dem Informationsdienst zugrundeliegende URL kennen, um ihn nutzen zu können. Wird der Zugriff auf den WWW-Dienst über einen Proxy-Server abgewickelt, muß der WWW-Client hierzu entsprechend konfiguriert werden. (Kapitel 7.2.3).

5.3.2 Wartung

Die Wartung des Systems sollte einfach gehalten sein, so daß auch Personen damit betraut werden können, die nicht die technischen Einzelheiten des Systems kennen, da die Entwickler des Informationsdienstes in der Regel nicht identisch sind mit den Betreibern des Informationsdienstes. Die mit der Wartung des Systems beauftragten Personen müssen die dem Informationsdienst zugrundeliegende Datenbank mit Daten füllen bzw. bei Bedarf aktualisieren. Es sollte dabei berücksichtigt werden, daß die hierzu notwendigen Aktionen möglichst einfach zu handhaben sind.

5.3.3 Datenhaltung

Bei der Bereitstellung der Daten in der Datenbank ist sicherzustellen, daß die dort gespeicherten Daten korrektes Format besitzen, um die WWW-Anwendung nicht zusätzlich mit Konvertierungsarbeiten belasten zu müssen. Werden beispielsweise Daten aus einer Datenbank von einem Großrechner extrahiert und in eine Unix-basierte Datenbank eingespielt, ist darauf zu achten, daß die Umlaute zuvor in ein Format konvertiert werden, das von der Anwednung korrekt verarbeitet werden kann. Außerdem sollte der Datenbestand stets korrekt, vollständig und aktuell sein.

5.4 Sicherheitsaspekte

5.4.1 Validierung der Benutzereingabe

Führt ein CGI-Skript die Benutzereingabe oder Teile davon direkt aus oder werden sie an Kommandos übergeben, können dadurch Sicherheitslücken entstehen, die durch potentielle Angreifer ausgenutzt werden könnten. Angenommen, das CGI-Skript greift auf entfernte Server zu und führt dort bestimmte Aktionen aus. Damit das Skript weiß, auf welchen Rechner es zugreifen soll, gibt der Bentuzer der Anwendung den Rechnernamen an. Nehmen wir weiter an, daß der vom Benutzer eingegebene Rechnername beispielsweise an das Kommando

```
ping rechnername
```

übergeben wird, was eigentlich nicht weiter von Bedeutung ist. Gibt nun ein Benutzer in das dafür vorgesehene Feld im Eingabeformular mehr als nur den Rechnernamen ein, wie beispielsweise

```
eigener.rechnername.domäne.de ; echo >> /.rhosts ‘‘+ +‘‘,
```

und wird die Benutzereingabe ungeprüft an das Kommando `ping` übergeben, ist dadurch eine Sicherheitslücke im System entstanden, die es jedem Angreifer ermöglicht, in das eigene System einzudringen.

Um sich davor zu schützen, sollten CGI-Skripten niemals unter der Kennung `root` laufen. Sie könnten beispielsweise unter der Kennung `nobody` oder unter einer eigenen, eigens für die CGI-Umgebung eingerichteten Kennung laufen, mit der dem System kein Schaden zugefügt werden kann. Außerdem sollte es vermieden werden, Benutzereingaben ungeprüft an Kommandos wie beispielsweise `system()` oder `popen()` in C zu übergeben.

5.4.2 Benutzerauthentifizierung über POST-Methode

Wird die Benutzerauthentifizierung über ein HTML-Formular abgewickelt, d.h., werden Benutzerkennung und Paßwort über ein Formular eingegeben, sollte das HTML-Formular hierzu die HTTP-Methode `POST` verwenden. Die Daten werden auf diese Weise über die Standardeingabe an das Skript übergeben. Wird hingegen für die Benutzerauthentifizierung die `GET`-Methode verwendet, erscheinen sowohl Benutzerkennung als auch das Paßwort unverschlüsselt im `Location`-Feld des WWW-Clients und sind auf diese Weise für jedermann sichtbar.

Kapitel 6

Der Prototyp der „Jahreswagenbörse“

6.1 Einführung

Basierend auf dem in Kapitel 4 entwickelten Design eines Datenbank-Gateways und dem in Kapitel 5 erstellten Style Guide wurde der Prototyp der WWW-Anwendung „BMW Exklusivbörse für Jahreswagen , kurz „Jahreswagenbörse“ oder „JAWA“, realisiert, dessen Implementierung in diesem Kapitel beschrieben wird.

Die JAWA ist ein Jahreswagenangebot der BMW AG für den öffentlichen und somit auch privaten WWW-Anwender. Der Benutzer der JAWA hat die Möglichkeit einen seinen Wünschen entsprechenden Jahreswagen in der zugrundeliegenden Datenbank zu suchen. Er kann Auswahlkriterien wie beispielsweise Modell, Antriebsart, Karosserie, Farbe usw. angeben und erhält dazu eine Liste von Fahrzeugen, die aufgrund der vorgegebenen Kriterien in der Datenbank ermittelt wurden. Zu jedem Fahrzeug kann eine detaillierte Fahrzeugbeschreibung abgerufen werden.

In diesem Kapitel werden die Vorraussetzungen für die Implementierung, die Schnittstellen zur Datenbank, die Benutzerschnittstellen, die Architektur sowie Umgebung der JAWA und mögliche Optimierungsmöglichkeiten beschrieben.

6.2 Systemvorraussetzung

Sowohl für das Datenbank-Gateway als auch für die WWW-Client- und WWW-Server-Software gelten bestimmte Systemvorraussetzungen. Die WWW-Anwendung JAWA wurde für folgende Umgebung entwickelt:

- Betriebssystem HP-UX 9.01
- Datenbanksystem Oracle 7.0
- Datenbank-Kommunikationssoftware SQL*Net Version 2.1
- Compiler cc
- Oracle Precompiler pcc Version 1.5
- WWW-Server CERN httpd 3.0

Der Prototyp der JAWA wurde am CIP-Pool der Ludwigs-Maximilians-Universität entwickelt und ist derzeit dort im Einsatz. Er ist unter `http://neptun.cip.informatik.uni-muenchen.de:8888/cgi-bin/jawa` erreichbar.

Die JAWA setzt WWW-Clients mit Unterstützung von HTML-Level 3 voraus, da HTML-Formulare für die Eingaben der Benutzer sowie HTML-Tabellen für die Darstellung der Listen von Fahrzeugen, die aufgrund der Auswahlkriterien ermittelt wurden, verwendet werden. Von den in Kapitel 3.2 untersuchten Browsern, sind derzeit nur die Netscape-Browser in der Lage, die aus der Anwendung generierten Seiten korrekt darzustellen. Bei dem verwendeten WWW-Server wird vorrausgesetzt, daß er die Zugriffsmethoden `GET` und `POST` sowie das Ausführen serverseitiger Programme unterstützt. Dies bedeutet, daß der Server das HTTP-Protokoll in der Version 1.0 implementieren sowie CGI-kompatibel sein muß. Diese Vorraussetzungen werden alle von den in Kapitel 3.1 untersuchten Servern erfüllt.

Neben der oben beschriebenen Entwicklungsumgebung des Prototypen wurde der Prototyp auf weiteren Systemen getestet. Teile der Anwendung wurden bereits auf den Betriebssystemen Sun-OS 4.1.3 sowie IBM AIX 3.2.5 übersetzt und getestet. Die gesamte Anwendung konnte nicht auf anderen Systemumgebungen getestet werden, weil das Datenbanksystem Oracle aus technischen und lizenzrechtlichen Gründen nur auf einer Maschine verfügbar gewesen ist und die für die gesamte Übersetzung der Anwendung nötigen Bibliotheken auf den anderen Maschinen somit nicht installiert werden konnten.

6.3 Die zugrundeliegende Datenbank

Zentraler Bestandteil der JAWA ist eine Datenbank, in der die Daten gespeichert und gepflegt werden sollen. Die Komponenten der JAWA sollen über Anfragen Informationen aus der Datenbank abrufen können.

Als Datenbanksystem wurde das relationale Datenbanksystem Oracle verwendet, das am CIP-Pool bereits installiert war und für die Entwicklung der JAWA verwendet werden konnte. Ein weiterer Grund für die Verwendung von Oracle als Datenbasis für den Prototypen, war die Aussage der BMW AG, ebenfalls Oracle als Datenbank für die kommerzielle Jahreswagenbörse einsetzen zu wollen.

Der Datenbank-Server und die WWW-Anwendung JAWA laufen auf unterschiedlichen Rechnern. Die Kommunikation zwischen der JAWA und dem Datenbank-Server wird daher über SQL*Net realisiert. SQL*Net ist ein Oracle-spezifisches Produkt, daß für die Kommunikation zwischen Datenbank-Client und entfernter Datenbank eingesetzt wird.

6.3.1 Import der Anwendungsdaten

Die Daten der Jahreswagen werden in einer hierarchischen IMS-Datenbank zusammen mit den Gebrauchtwagendaten auf einem Großrechner gespeichert und gepflegt. Die Gebrauchtwagendaten enthalten sowohl Modelle der BMW AG als auch anderer Hersteller. Um die Jahreswagendaten als Datenbasis für den JAWA-Prototypen verwenden zu können, müssen sie dazu in die Unix-basierte relationale Oracle-Datenbank importiert werden. Sie werden zuerst aus dem gesamten Datenbestand der Gebrauchtwagen extrahiert und in einer sequentiellen Datei auf dem Großrechner abgelegt. Die Daten werden daraufhin auf das Unix-basierte System übertragen und stehen dort als sequentielle Datei aus einzelnen Datensätzen zur Verfügung. Für die Übertragung der Daten in die Datenbank wurden die für die JAWA relevanten Daten eines jeden Datensatzes aus der sequentiellen Datei extrahiert und in die Datenbank importiert. Es wurde dazu ein eigens hierfür entwickeltes C-Programm verwendet. Eine andere Möglichkeit die Daten in die Datenbank zu importieren, hätte darin bestanden, das Oracle-spezifische Programm „SQL Loader“ zu verwenden. Da die Jahreswagendaten auf einem Großrechner gespeichert werden, sind beispielsweise Umlaute nicht im korrekten Format dargestellt. Aus diesem Grund müssen vor dem Import der Daten alle Umlaute entsprechend angepaßt bzw. konvertiert werden. Damit wird die WWW-Anwendung nicht zusätzlich mit Konvertierungsaufgaben belastet und es müssen hierzu keine Konvertierungsroutinen implementiert werden.

6.3.2 Struktur der Datenbank

Da ein sequentieller Datensatz mehr Informationen enthält als für die JAWA eigentlich benötigt wird, wurde für den Prototypen der JAWA eine eigene Datenbankstruktur entwickelt. Die JAWA-Datenbank besteht aus einer Oracle-Tabelle namens `FAHRZEUG`, deren Aufbau in der Tabelle 6.1 beschrieben ist.

Die relationale Tabelle wird durch die sequentiellen Datenbestände versorgt. Das Feld `ANGEBOTS_NR` ist ein eindeutiger Schlüssel der Tabelle `FAHRZEUG`, über den ein einzelnes Fahrzeug angesprochen und ausgelesen werden kann. Damit ist ein WWW-Client in der Lage, direkt auf ein einzelne Fahrzeuge in der Datenbank zuzugreifen.

Tabelle 6.1: Aufbau der Tabelle `FAHRZEUG`

Feldname	Null ?	Datentyp	Inhalt
ANGEBOTS_NR	NOT NULL	CHAR(7)	Eindeutige Angebotsnummer des Fahrzeugs
MOD_TYP		CHAR(3)	Modelltyp
ANTRIEB		CHAR(3)	Antriebsart
KARO_VAR		CHAR(10)	Karosserievariante
ANZ_TUER		CHAR(1)	Anzahl der Türen
KW		CHAR(3)	Leistung in Kilowatt
PS		CHAR(3)	Leistung in PS
HUBRAUM		CHAR(4)	Hubraum
ERSTZUL		CHAR(6)	Datum der Erstzulassung (JJMMTT)
FREIDT		CHAR(4)	Freigabedatum (JJMM)
TUEV		CHAR(4)	nächster TÜV-Termin (JJMM)
PREIS		CHAR(6)	Angebotspreis
KM_STAND		CHAR(3)	KM-Stand
GRUND_FARBE		CHAR(15)	Grundfarbe des Fahrzeugs
FARBE		CHAR(30)	Farbe des Fahrzeugs
METALLIC		CHAR(1)	Metallic-Kennzeichen (J/N)
SSD		CHAR(1)	Sonnenschiebedach (J/N)
KLIMA		CHAR(1)	Klimaanlage (J/N)
B_AIRBAG		CHAR(1)	Beifahrerairbag (J/N)
ALU		CHAR(1)	Leichtmetallräder (J/N)
AHK		CHAR(1)	Anhängerkupplung(J/N)
LEDER		CHAR(1)	Lederausstattung (J/N)
AUTOMATIK		CHAR(1)	Automatik (J/N)
SA_BESCHR1		CHAR(40)	Beschreibung der Sonderausstattung (1)
SA_BESCHR2		CHAR(40)	Beschreibung der Sonderausstattung (2)
SA_BESCHR3		CHAR(40)	Beschreibung der Sonderausstattung (3)
SA_BESCHR4		CHAR(40)	Beschreibung der Sonderausstattung (4)
SA_BESCHR5		CHAR(40)	Beschreibung der Sonderausstattung (5)
SA_BESCHR6		CHAR(40)	Beschreibung der Sonderausstattung (6)
PRIO_KZ		CHAR(1)	Prioritätskennzeichen
POLSTER		CHAR(15)	Beschreibung der Polsterung

6.4 Systemarchitektur

6.4.1 Beschreibung der Komponenten

Der Prototyp der JAWA ist ein statisches CGI-Skript (Kapitel 4.7.1), das als Standalone-Gateway (Kapitel 4.5) realisiert ist und direkt mit der Datenbank kommuniziert. Die Kommunikation zwischen CGI-Skript und Datenbank erfolgt über die Datenbankzugriffschicht. Die Anwendung verwendet „Embedded“ SQL-Anweisungen (ESQL), um die Daten aus der relationalen Datenbank auszulesen und zur Verarbeitung an das Programm weiterzuleiten.

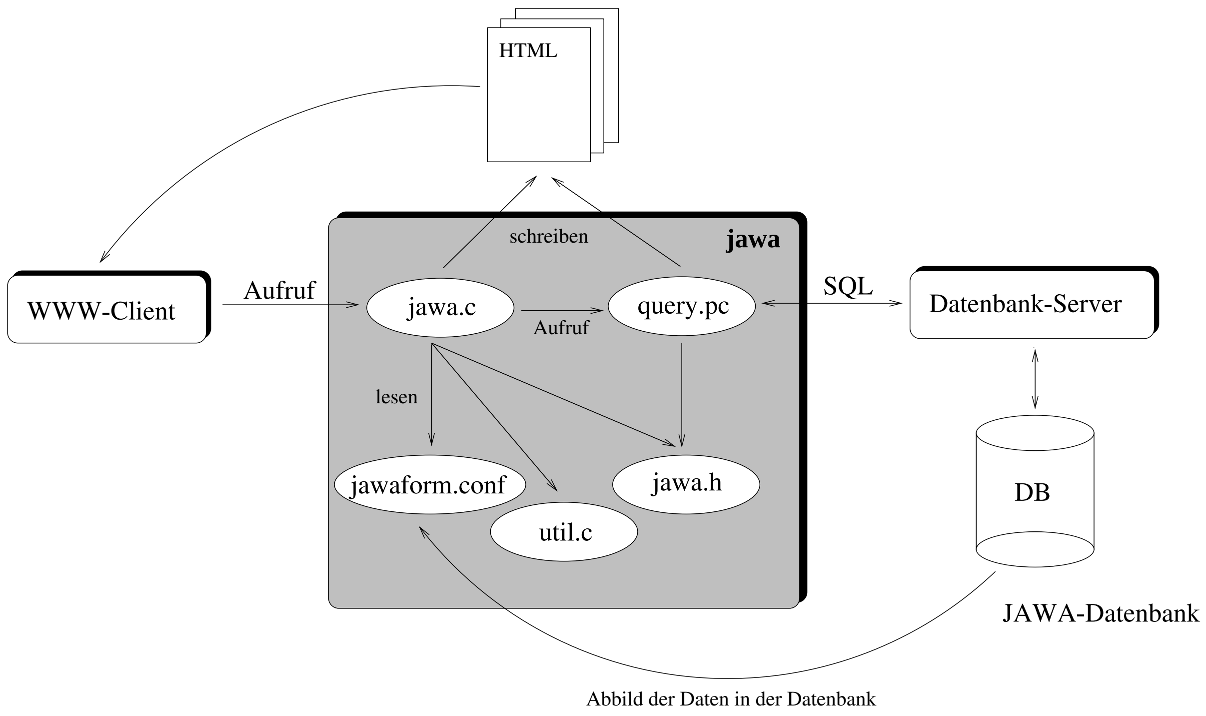

Abbildung 6.1: Die JAWA und ihre Komponenten

Die Anwendung besteht wie in Abbildung 6.1 dargestellt, aus den Modulen `jawa.c`, `query.pc` und `util.c` sowie der Headerdatei `jawa.h` und der Konfigurationsdatei für das Eingabeformular `jawaform.conf`. `jawa.c` ist ein C-Programm und stellt das „Front End“ der Anwendung dar. Es ist zuständig für die Generierung des HTML-Eingabeformulars, für die Verarbeitung der Benutzereingabe sowie die dynamische Generierung der Datenbankanfrage aus den vom Benutzer eingegebenen Kriterien. Nach Aufruf der JAWA durch den WWW-Server werden dem CGI-Skript die vom Client generierte Anfrage über das CGI übergeben. Die Anwendung prüft, ob die Benutzereingabe korrekt ist und auch korrekt an das Skript übergeben wurde. Nach erfolgreicher und fehlerfreier Verarbeitung der Benutzereingabe wird aus `jawa.c` das Modul `query.pc` aufgerufen und die dynamisch generierte

Anfrage übergeben.

`query.pc` ist ein C-Programm, in das ESQL-Anweisungen eingebettet sind und stellt somit das Datenbank-„Back End“ der Anwendung dar. Es ist zuständig für die Anmeldung bei der JAWA-Datenbank, es führt Konsistenzprüfungen der Benutzereingabe durch und schickt die vom Modul `jawa.c` erhaltenen SQL-Anfragen an die Datenbank. Nachdem die Daten aus der Datenbank ausgelesen wurden, werden Formatierungen an den Daten vorgenommen und auf die Standardausgabe geschrieben, von wo aus die Daten durch den WWW-Server zurück an den Browser geschickt werden.

`query.pc` verwendet ausschließlich Standard-SQL-Anweisungen anstelle Oracle-spezifischer SQL-Erweiterungen. Durch die Unabhängigkeit von den SQL-Dialekten, die von den verschieden Datenbanksystemen verwendet werden und durch die Aufteilung der Anwendung in zwei Komponenten ist eine Portierung der Anwendung auf ein anderes relationales Datenbanksystem ohne große Änderung des Source-Codes möglich.

Das Modul `util.c` enthält Hilfsfunktionen, die in `jawa.c` verwendet werden, um die vom WWW-Server übergebenen, url-codierten Daten zu dekodieren.

Zur Generierung des HTML-Formulars werden von der JAWA Informationen verwendet, die in der Konfigurationsdatei `jawaform.conf` (Anhang B) gespeichert sind. Diese Datei bildet die Basis für das Eingabeformular und ist ein Abbild der Daten, die in der Datenbank enthalten sind. Dies bedeutet, das in der Konfigurationsdatei nur die Auswahlkriterien enthalten sind, die auch tatsächlich in der Datenbank gespeichert sind, und daß nur diese Kriterien beim Aufbau des HTML-Formulars verwendet werden. Dem Benutzer stehen damit nur solche Kriterien zur Auswahl, die auch in der Datenbank enthalten sind. Damit wird die Wahrscheinlichkeit der „ergebnislosen Suche“ reduziert und zugleich die Toleranz gegenüber Eingabefehlern der Benutzer erhöht. Die Datei kann entweder manuell editiert oder mit Hilfe eines geeigneten Skripts erstellt werden. Das Skript muß dazu die notwendigen Informationen aus der Datenbank auslesen und daraus die Datei `jawaform.conf` generieren. Bei Änderung des Datenbestandes in der JAWA-Datenbank muß diese Datei entsprechend angepaßt werden. Werden beispielsweise alle „roten“ Fahrzeuge aus der Datenbank entfernt, muß sichergestellt werden, daß diese Farbe nicht mehr unter den Grundfarben in der Datei `jawaform.conf` enthalten ist. Dies gilt ebenfalls für alle anderen Auswahlkriterien.

6.4.2 Programmablauf

Die Suche auf der Datenbank läuft in folgenden Schritten ab:

- Der Benutzer aktiviert eine URL, welche die JAWA adressiert.
- Der Server erhält die URL, startet das Skript, belegt die CGI-Umgebungsvariablen mit den entsprechenden Informationen. Es werden keine Parameter an das Skript übergeben.

- Die JAWA wertet die Umgebungsvariablen aus, liest die Einträge aus der Konfigurationsdatei und generiert daraus ein HTML-Formular, die sog. Startseite der Anwendung. Das Formular wird an den Benutzer zurückgeschickt und die Anwendung wird beendet. Die JAWA erkennt anhand der HTTP-Zugriffsmethode sowie an der leeren Anzahl von Parametern, daß das HTML-Eingabeformular zu generieren ist.
- Der Benutzer füllt das Formular mit seinen Auswahlkriterien aus und betätigt den Knopf `Suche starten`. Die URL wird samt dem url-kodierten Formularinhalt an den WWW-Server bzw. an die JAWA geschickt.
- Der Server erhält die URL, startet die JAWA und übergibt ihr den url-kodierten Formularinhalt.
- Die JAWA erkennt anhand der verwendeten HTTP-Zugriffsmethode, daß sie eine Suche auf der Datenbank zu initiieren hat. Sie generiert aus dem übergebenen, url-kodierten Formularinhalt eine SQL-Anfrage.
- Die SQL-Anfrage wird an das Datenbanksystem geschickt.
- Der Datenbank-Server verarbeitet die Anfrage und liefert die gefundenen Fahrzeuge zurück.
- Die JAWA erhält die extrahierten Daten, formatiert sie für die Ausgabe und konstruiert aus dem Ergebnis eine HTML-Seite. Die Daten werden in Form einer Tabelle angeordnet. Jede einzelne Zeile der Tabelle ist als Hypertextlink in die HTML-Seite eingebettet, der es dem Benutzer ermöglicht, weitere Informationen zu einem Fahrzeug anzufordern.
- Die HTML-Seite wird an den Benutzer zurückgeschickt und die JAWA beendet sich.
- Der Benutzer aktiviert einen Hypertextlink aus der zuvor erhaltenen HTML-Seite und adressiert damit erneut die JAWA. Der Link enthält die URL der JAWA sowie den eindeutigen Schlüssel des Fahrzeugs, auf das zugegriffen werden soll.
- Der WWW-Server erhält die URL, startet die JAWA und übergibt ihr den Schlüssel.
- Die JAWA erkennt anhand der HTTP-Zugriffsmethode und des übergebenen Schlüssels, daß erneut eine SQL-Anfrage zu generieren ist. Dieses Mal wird der Schlüssel des gesuchten Fahrzeugs anstelle der Formulardaten verwendet, um eine SQL-Anfrage zu generieren.
- Die Datenbank verarbeitet die Anfrage und liefert das Ergebnis zurück.
- Die JAWA erhält die dem Schlüssel zugrundeliegenden Daten des gesuchten Fahrzeugs, formatiert sie und generiert aus dem Ergebnis eine HTML-Seite, die alle Informationen zu dem Fahrzeug enthält. Im Gegensatz zur vorherigen Anfrage wird dieses Mal nur ein Objekt aus der Datenbank extrahiert. Die HTML-Seite wird zurück an den Benutzer geschickt und die JAWA beendet sich.

6.5 Benutzerschnittstelle

6.5.1 Generierte HTML-Seiten

Die Benutzerschnittstelle der JAWA setzt sich aus drei WWW-Seiten zusammen: dem HTML-Eingabeformular, der ersten und zweiten Ergebnisseite. Jede einzelne Seite besteht aus dem Kopf, dem Rumpf sowie dem Trailer. Der Kopf setzt sich aus dem JAWA-Logo und dem Schritfzug *„BMW-Exklusiv-Börse für Jahreswagen“* zusammen. Der Rumpf der WWW-Seite enthält die eigentlichen Informationen für den Benutzer. Den Abschluß einer durch die JAWA generierten HTML-Seite bildet der Trailer. Er enthält Navigationshilfen, die Version der Anwendung, eine Online-Hilfe, einen Verweis auf die aktuelle Modellübersicht der BMW AG und bietet die Möglichkeit Nachrichten in Form von Electronic Mail an den JAWA-Administrator zu schicken. Jede der drei Seiten enthält den gleichen Kopf, wohingegen die Trailer der einzelnen Seiten unterschiedliche Kombinationen der oben aufgeführten Elemente enthalten.

Das HTML-Eingabeformular (Abbildung 6.2) ist die zentrale Seite der JAWA, die zugleich die „Home Page“ bzw. Startseite der WWW-Anwendung darstellt. Innerhalb dieser Seite können die Kriterien zur Suche eines Fahrzeugs in der Datenbank ausgewählt werden, die Anfragen abgeschickt und zurückgesetzt werden. Sie bietet eine Online-Hilfe an, die den Benutzer beim Ausfüllen des Formulars unterstützen soll, und die Möglichkeit, auf die aktuelle Modellübersicht zu verweisen, falls sich der Benutzer mit der Produktpallette der BMW AG nicht auskennt. Tretten während der Bedienung der JAWA Probleme auf, oder möchte der Benutzer aus irgendeinem Grund mit dem Administrator der Anwendung in Kontakt treten, kann er dies durch Aktivieren des Hypertextlinks „`Mail an JAWA-Administrator`“ tun. Durch Betätigen des Knopfes „`Suche starten`“ wird der Eingabetext des Benutzers an die JAWA geschickt und eine Suche auf der Datenbank gestartet. Es erscheint daraufhin die erste Ergebnisseite.

Zu Beginn dieser Seite werden die vom Benutzer zuvor eingegebenen Suchkriterien nochmals aufgeführt. Im Falle einer ergebnislosen Suche wird der Hinweis hinzugefügt, das kein Fahrzeug gefunden wurde, das den vom Benutzer eingegebenen Kriterien entspricht. Der Benutzer hat dann die Möglichkeit durch Betätigen des Knopfes „`Neue Anfrage`“ zum Eingabeformular zurückzukehren und eine neue Anfrage zu starten. Ansonsten werden die in der Datenbank gefundenen Fahrzeuge in Form einer Tabelle dargestellt (Abbildung 6.3). Für die Darstellung der Tabelle wird das HTML 3.0-Sprachelement `<TABLE>` verwendet. In einer ersten Designphase der Benutzeroberfläche wurde noch mit dem HTML 2.0-Element `<PRE>` experimentiert, um die gefundenen Fahrzeuge tabellarisch darzustellen. Die korrekte Formatierung der einzelnen Spalten erwies sich mit diesem Sprachelement als schwierig und die Darstellungsweise war unschön. Da bei Verwendung des Elements `<TABLE>` die Spalten der Tabelle dynamisch angepaßt werden und weil sich die Darstellung der Tabelle hiermit als kompakter und schöner erwieß, wurde für den Prototypen das HTML 3.0-Element verwendet. Der Nachteil dabei ist jedoch, daß dieses Sprachelement derzeit

Abbildung 6.2: Das Eingabeformular der JAWA

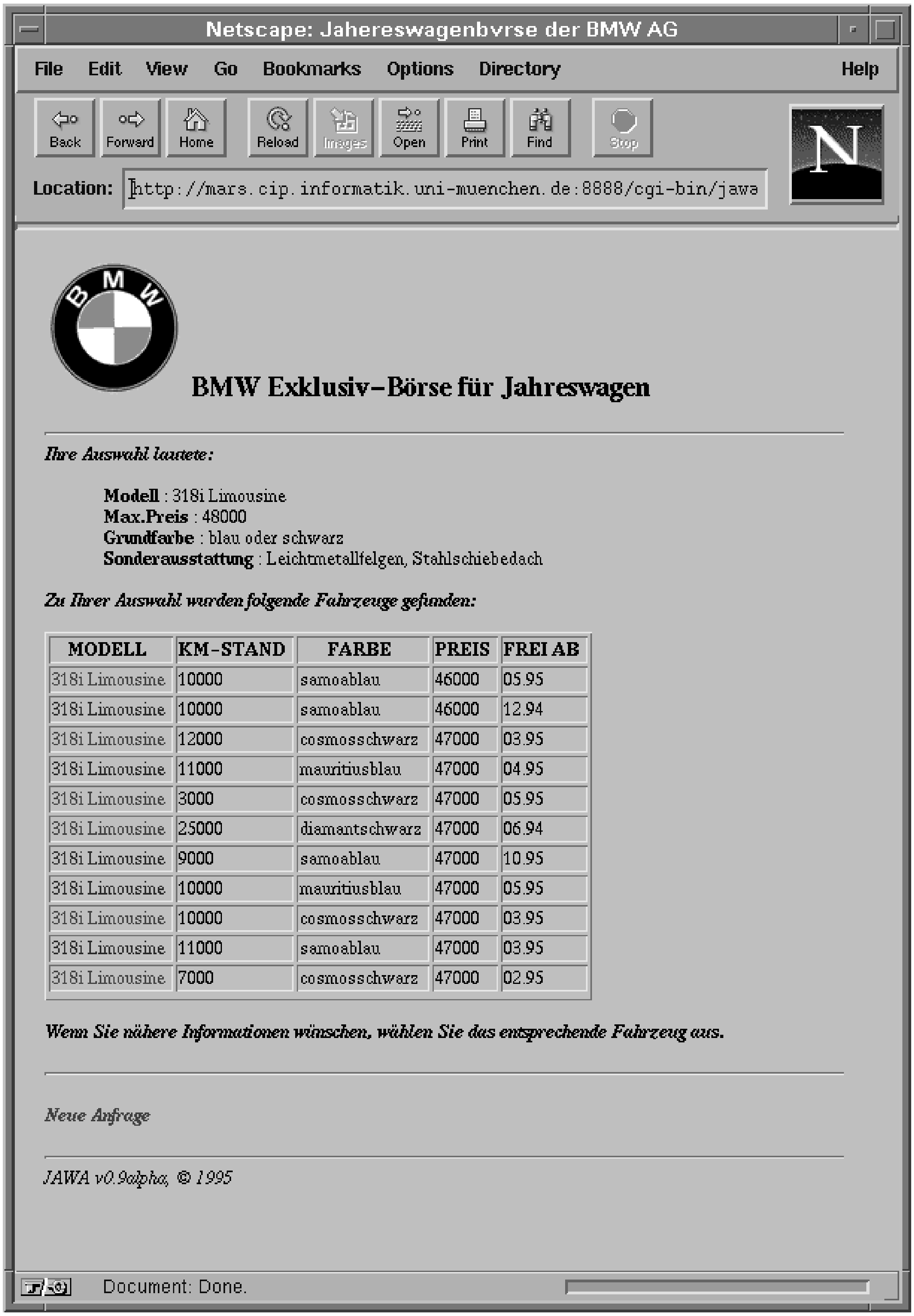

Abbildung 6.3: Erste Ergebnisseite der JAWA

von einigen WWW-Clients nicht implementiert wird (Tabelle 3.2) und somit Probleme bei der Darstellungsweise der Tabelle auftreten.

Die angezeigte Liste der gefundenen Fahrzeuge ist auf 40 Treffer beschränkt, um Benutzer daran zu hindern, durch Formulieren allgemeiner Anfragen, mit nur einer Anfrage große Mengen an Daten aus der Datenbank abzuziehen. Die Anzahl der angezeigten Suchergebnisse kann durch Änderung des Wertes der entsprechenden Variablen in der Header-Datei `jawa.h` entsprechend angepaßt werden. Innerhalb der Tabelle der gefundenen Fahrzeuge stellen die Modellbezeichnungen in der Spalte „`Modell`“ Hypertextinks auf die jeweiligen in der Datenbank gespeicherten Fahrzeuge dar. Durch Aktivieren eines Links erscheint die zweite Ergebnisseite.

Auf dieser Seite werden alle notwendigen Informationen für die Auwahl eines Jahreswagens dargestellt (Abbildung 6.4). Wünscht der Benutzer weitere Informationen, hat er die Möglichkeit durch den Link „`Weitere Informationen`“ auf eine andere HTML-Seite des WWW-Servers der BMW AG zu gelangen. Dort könnten beispielsweise allgemeine Fahrzeuginformationen der jeweiligen Modellreihe enthalten sein. Zusätzlich enthält diese Seite einen Link auf ein Photo eines Fahrzeugs dieser Modellreihe. Entspricht das angezeigte Fahrzeug nicht den Wünschen des Benutzers kann er zur vorherigen Seite zurückkehren oder durch Betätigen des Links „`Neue Anfrage`“ das Eingabeformular aktivieren und eine neue Anfrage starten.

6.5.2 Suchkriterien

Um die Suche nach einem Fahrzeug einzuschränken, werden dem Benutzer im Eingabeformular der JAWA zahlreiche Suchkriterien zur Verfügung gestellt. Folgende Kriterien stehen dem Benutzer dabei zur Auswahl:

- `Modell`
- `Antriebsart`
- `Karosserie`
- `Maximaler Preis`
- `Grundfarbe`
- `Metallic-Lackierung`
- `Sonderausstattung`

Die Kriterien `Modell`, `Antriebsart`, `Karosserie` und `Sonderausstattung` sind als „Pulldownmenüs“ organisiert. Für die Eingabe des Preises wurde ein Texteingabefeld verwendet, die Eingabe der Grundfarbe erfolgt über sog. „HTML-Check-Boxes“. Die Auswahlfelder

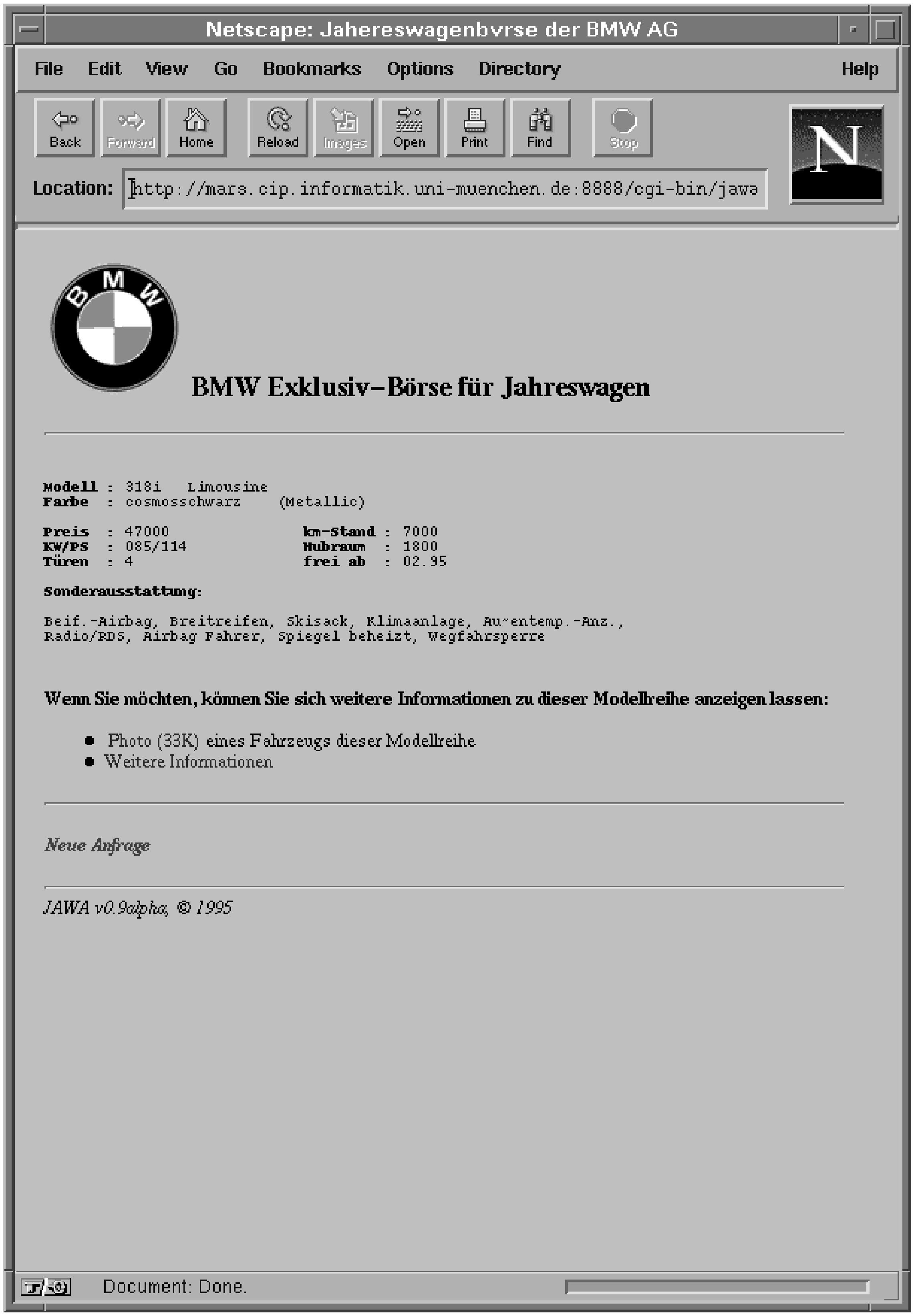

Abbildung 6.4: Zweite Ergebnisseite der JAWA

`Metallic-Lackierung` sind als „HTML-Radio-Buttons“ in das Formular eingebettet. Bei allen Feldern bis auf `Grundfarbe` und `Sonderausstattung` kann nur ein einziges Kriterium ausgewählt werden. Bei den anderen beiden Feldern ist die gleichzeitige Auswahl mehrerer Kriterien möglich.

Da das Eingabeformular dynamisch durch die JAWA generiert wird, stehen als Suchkriterien nur solche zur Auswahl, die auch tatsächlich in der Datenbank enthalten sind. Ist beispielsweise in der Datenbank kein Fahrzeug der Modellreihe 525, kein Cabrio und kein rotes Fahrzeug enthalten, werden die Kriterien 325 (bei `Modell`), Cabrio (bei `Karosserie`) sowie rot (bei `Grundfarbe`) nicht zur Auswahl angeboten. Dies gilt ebenfalls für alle anderen Suchkriterien.

Bei einem ersten Design des Eingabeformulars wurden die Kriterien `Modell`, `Antriebsart` und `Karosserie` zu einem Kriterium `Modell` zusammengefaßt und als Texteingabefeld realisiert. Das Texteingabefeld hatte den Vorteil, daß es in Bezug auf die Benutzereingabe flexibler war als die jetzt verwendeten Pulldownmenüs. Der Benutzer hatte die Möglichkeit durch Eingabe variabler Suchkriterien nach beliebigen Fahrzeugen zu suchen. Beispielsweise konnte über die Eingabe des Kriteriums `,,5*''` bzw. `,,32*''` nach allen 5er-Modellen bzw. nach allen 320er-, 325er- sowie 328-er Modellen gesucht werden. Diese Flexibilität ist zwar durch die Verwendung von Pulldownmenüs verlorengegangen, mit dem Vorteil, daß die Benutzerfehler, die aus nicht korrekter Eingabe der Kriterien resultieren, unterbunden werden konnten. Der Benutzer stellt sich nun das gewünschte Fahrzeug zusammen, indem er sich die zur Verfügung gestellten Kriterien aus den drei Pulldownmenüs auswählt. Er kann außerdem nur noch nach einzelnen Modellen und nicht mehr nach allen Fahrzeugen einer Modellreihe suchen. Durch die Verwendung von Pulldownmenüs anstelle von Texteingabefeldern konnte die Fehlertoleranz gegenüber Benutzerfehlern sowie die Anwenderfreundlichkeit der Anwendung erhöht werden.

6.5.3 Sortierung der Ausgabe

Die JAWA bietet dem Benutzer die Möglichkeit, die in der Datenbank gefundenen Fahrzeuge nach bestimmeten Kriterien zu sortieren. Im Eingabeformular wird dem Benutzer die Möglichkeit geboten, die Ausgabe der Fahrzeuge entweder nach dem Kriterium `Preis`, `Verfügbarkeit` oder `Modell` zu sortieren. Die Sortierung erfolgt dabei in aufsteigender Form.

Bei der Realiserung der Sortierung ist die Frage aufgekommen, ob die Sortierung der Ausgabe explizit durch den Benutzer oder implizit durch die Anwendung erfolgen sollte. Außerdem war es möglich, die Sortierung der Fahrzeuge im Eingabeformular oder auf der ersten Ergebnisseite anzubieten. Wird die Sortierung der Fahrzeuge auf der ersten Ergebnisseite angestoßen, hätte dies eine erneute, zweite Anfrage an die Datenbank zur Folge und würde das Datenbanksystem zusätzlich belasten. Aus Gründen der Benutzerfreundlichkeit und um die Anzahl der Datenbankzugriffe zu reduzieren, wird eine explizite Sortierung der Ausgabe realisert, die im Eingabeformular angeboten wird. Dies hat den Vorteil, daß die

Auswahl und Sortierung der Fahrzeuge mit einem Datenbankzugriff erledigt werden kann.

6.5.4 Navigationshilfen

Um dem Benutzer Navigationshilfen zwischen den einzelen HTML-Seiten der JAWA zu bieten, sind das Feld `Suche starten` und der Hypertextlink `Neue Anfrage` in die einzelnen HTML-Seiten eingebettet. `Suche starten` ist Bestandteil des Eingabeformulars und startet eine Suche auf der Datenbank, die als Ergebnis eine Liste aller Fahrzeuge liefert, die in der Datenbank enthalten sind. `Neue Anfrage` ist in beiden Ergebnisseiten enthalten und verweist auf das Eingabeformular. Der Benutzer der JAWA ist somit jederzeit in der Lage eine neue Anfrage zu starten, ohne lange durch die einzelnen Seiten der Anwendung navigieren zu müssen. Möchte der Benutzer zwischen den beiden Ergebnisseiten navigieren, sind dazu die `BACK`- bzw. `FORWARD`-Buttons der jeweiligen WWW-Clients zu verwenden.

6.5.5 Kontextsensitive Online-Hilfe

Die im Eingabeformular enthaltenen Online-Hilfe ist ein Link auf ein WWW-Dokument, das den Benutzer bei der Bedienung der JAWA unterstützen soll. Sie beschreibt den Ablauf einer Suche, die einzelnen Suchkriterien, die Möglichkeiten die gefundenen Fahrzeuge zu sortieren sowie die bestehenden Navigationsmöglichkeiten innerhalb der JAWA. Die einzelnen Suchkriterien sind außerdem als Hypertextlink in das Formular eingebettet. Durch Anklicken eines Kriteriums gelangt man an die entsprechende Stelle in der Online-Hilfe, an der die Suchkriterien beschrieben werden.

Um die Kontextsensitivität zu erläutern, wird deren Realisierung in der JAWA am Beispiel des Kriteriums `Modell` dargestellt. Bei Anklicken des Kriteriums wird an die Stelle mit der Überschrift `Sortierkriterien` in der Hilfe-Datei verzweigt. Es wird angenommen, daß das HTML-Dokument, das die Online-Hilfe enthält, unter dem Pfad `jawa/html/help.html` auf dem WWW-Server erreichbar ist. Damit ein Verweis von einer Textstelle eines Dokuments auf eine Textstelle in einem anderen Dokument realisierbar ist, müssen sowohl Quell- und Zieldokument einen entsprechenden Verweis, einen sog. Anker, enthalten. In der Hilfedatei sieht die entsprechende Stelle wie folgt aus:

```
<A NAME=``kriterien``>Sortierkriterien</A>
```

Das Wort `Sortierkriterien` wird als Anker in der Datei `help.html` verwendet. Es kann über den eindeutigen Index `kriterien` als Ziel eines Links von jedem anderen Dokument aus erreicht werden. Das Kriterium `Modell` ist als folgender Hypertextlink in das Eingabeformular der JAWA eingebettet:

```
<A HREF=``http://<server>/jawa/html/help.html#kriterien``>Modell
</A>
```

Durch Angabe des Dateinamens `help.html` und durch Hinzufügen des Index `#kriterien` wird das Kriterium `Modell` zu einem Link auf die entsprechende Stelle in der Online-Hilfe. Alle weiteren Auswahlkriterien des Eingabeformulars sind auf die gleiche Weise als Links in die Startseite der JAWA eingebettet.

6.5.6 Aktuelle Produktübersicht

Die Startseite der JAWA enthält einen Link auf die aktuelle Produktübersicht der BMW AG. Dort sind alle Bezeichnungen der Fahrzeuge enthalten, die in der Datenbank gespeichert sein können. Die Bezeichnung eines Fahrzeugs ist eine Kombination aus `Modell`, `Antriebsart` und `Karosserie`. Die Produktübersicht bietet dem Benutzer die Möglichkeit während der Bedienung der JAWA darauf zurückzugreifen, wenn er nicht weiß, ob die von ihm gewählte Kombination aus `Modell`, `Antriebsart` und `Karosserie` zulässig ist.

6.6 Die JAWA und ihre Umgebung

6.6.1 Die Umgebung für die JAWA

Der WWW-Server stellt der JAWA eine Umgebung in Form von CGI-Umgebungsvariablen zur Verfügung, die es ihr erlauben, festzustellen was zu tun ist. Die JAWA erkennt anhand der Belegung der Variablen `REQUEST_METHOD`, welche Zugriffsmethode bei der Adressierung der Anwendung verwendet wurde und auf welche Art die Parameter übergeben werden. Ist die Variable `REQUEST_METHOD` mit dem Wert `POST` belegt, weiß die Anwendung, daß sie über die HTTP-Methode `POST` adressiert wurde, daß etwaige Daten über die Standardeingabe übergeben werden und daß daraus eine SQL-Anfrage zu generieren ist. Wird als Zugriffsmethode `GET` verwendet, prüft die JAWA zusätzlich noch die Umgebungsvariable `QUERY_STRING`. Ist die Variable nicht belegt, erkennt die JAWA daran, daß die Startseite der Anwendung bzw. das HTML-Eingabeformular zu generieren ist. Ansonsten enthält diese Variable den Schlüssel zu einem Fahrzeug in der Datenbank. Die JAWA generiert daraus eine SQL-Anfrage, startet die Suche auf der Datenbank und liefert als Ergebnis die Daten zu dem entsprechenden Fahrzeug.

Eine wesentliche Aufgabe der JAWA ist die Übernahme von Daten, die ein Benutzer in das HTML-Formular eingibt. Die Art und Weise, wie der WWW-Server diese Daten für die JAWA verfügbar macht, wird durch die Besetzung des `METHOD`-Attributs des HTML-Elements `<FORM>` gegeben. Die JAWA verwendet hierzu die HTTP-Methode `POST`, bei der die Formulardaten über die Standardeingabe und nicht wie bei der alternativen `GET`-Methode über die Umgebungsvariablen übernommen werden. Der Vorteil bei der Verwendung der `POST`-Methode liegt in der Möglichkeit, zu überprüfen, ob die Daten korrekt übergeben bzw. korrekt übernommen wurden. Vom WWW-Server wird die Umgebungsvariable `CONTENT_LENGTH` bereitgestellt, welche die Anzahl der auf der Standardeingabe übergebenen Zeichen enthält. Durch einen Test, ob diese Anzahl von Zeichen tatsächlich gelesen werden konnte, ergibt sich die erwähnte Verifizierung.

6.6.2 Informationsspeicherung in URLs

Informationen, die zu einem späteren Zeitpunkt von der JAWA benötigt werden, werden in Form von URLs gespeichert und sind als solche in die von der JAWA generierten HTML-Seiten eingebettet. Im Folgenden wird angenommen, daß sich die ausführbare Datei `jawa` im Verzeichnis `cgi-bin` des Rechners befindet, auf dem der HTTP-Deamon installiert ist. Alle URLs, die in den durch die JAWA generierten HTML-Seiten auftreten, referenzieren diese Datei in folgender Weise:

`http://<server>/cgi-bin/jawa` oder
`http://<server>/cgi-bin/jawa?Query-Information`

Der erste URL ist als Navigationshilfe `Neue Anfrage` in die von der JAWA generierten Ergebnisseiten eingebettet. Bei Aktivieren dieses Links wird das Eingabeformular der JAWA geladen. Die zweite URL wird bei der tabellarischen Darstellung der gefundenen Fahrzeuge verwendet. Jede Zeile der Tabelle ist dabei ein Link auf das entsprechende Fahrzeugs in der Datenbank. Die Zeichenkette `Query-Information` wird in der Umgebungsvariablen `QUERY_STRING` abgelegt und enthält den Schlüssel des Fahrzeugs, auf das zugegriffen wird. Beide URLs verwenden bei der Referzierung der JAWA die HTTP-Methode `GET`.

6.6.3 Adressierung der JAWA

Zur Adressierung der JAWA im `cgi-bin`-Verzeichnis gibt es drei verschiedene Varianten:

- Durch direktes Öffnen der URL am WWW-Client
(`http://<server>/cgi-bin/jawa`)

  Als Zugriffsmethode wird dabei `GET` verwendet. Die Umgebungsvariable `QUERY_STRING` ist nicht belegt. Als Ergebnis der Adressierung wird das HTML-Formular generiert. Die URL kann auch als normaler Link in einem beliebigen HTML-Dokument enthalten sein, das auf die JAWA verweist.

- Durch Folgen eines Links, der in der Tabelle der gefundenen Fahrzeuge enthalten ist
(`http://<server>/cgi-bin/jawa?<Objektschlüssel>`)

  Jede einzelne Zeile der Tabelle ist ein Link auf das entsprechende Fahrzeug in der Datenbank. Als Zugriffsmethode wird dabei `GET` verwendet. Die Umgebungsvariable `QUERY_STRING` ist mit dem Wert `<Objektschlüssel>` belegt. Sie enthält den Schlüssel zu dem jeweiligen Fahrzeug. Als Ergebnis der Adressierung der JAWA werden die Informationen zu dem gesuchten Fahrzeug angezeigt.

- Als Ziel des HTML-Formulars, d.h. hinter dem ACTION-Attribut eines Formular-Anfangs-Elements
(<FORM METHOD=POST ACTION=‘‘http://<server>/cgi-bin/jawa/‘‘>)

  Durch Betätigen des Feldes Suche starten im HTML-Formular wird die JAWA referenziert. Als Zugriffsmethode wird dabei POST verwendet. Die Formulardaten werden über die Standardeingabe an die JAWA übergeben. Die Variable CONTENT_LENGTH enthält die Länge der von der Standardeingabe zu übernehmenden Zeichen. Die Umgebungsvariable QUERY_STRING ist nicht belegt. Als Ergebnis der Adressierung wird eine HTML-Seite generiert, die alle Fahrzeuge enthält, die den vom Benutzer eingegebenen Kriterien entsprechen.

6.7 Logging

Die Zugriffe auf die JAWA-Datenbank werden durch die Anwendung protokolliert. Durch jeden Zugriff auf die JAWA wird die Protokolldatei jawa.log um einen Eintrag erweitert. Jeder Eintrag setzt sich aus dem Rechnernamen des Benutzers, dem aktuellen Datum sowie aus dem von Benutzer ausgewählten Kriterien zusammen. Bei Adressierung der JAWA über das HTML-Formular werden die eingegebenen Benutzerdaten in die Protokolldatei eingefügt. Erfolgt der Aufruf der JAWA über einen Link, der in der ersten Ergebnisseite eingebettet ist, wird neben dem Rechnernamen und dem aktuellen Datum die Angebotsnummer des jeweiligen Fahrzeugs in der Protokolldatei eingetragen. Zusätzlich wird die Anzahl der ausgelesenen Fahrzeuge protokolliert. Ein Beispiel hierfür sind die Einträge

```
sunhegering7.nm.informatik.uni-muenchen.de [19/June/1995:11:41:18]
‘‘angebots_nr=97564ST‘‘ 1

pchegering4.nm.informatik.uni-muenchen.de [19/June/1995:14:34:04]
‘‘mod_type=525&antrieb=tds&karo_var=touring&preis=65000&grund_farbe=
blau&grund_farbe=schwarz&sa_beschr=Leichtmetallfelgen‘‘ 0
```

Der erste Eintrag protokolliert den Zugriff auf ein einzelnes Fahrzeug, das aus der Datenbank ausgelesen wurde. Der zweite Eintrag protokolliert einen Zugriff, der über das Eingabeformular erfolgte. Die Null am Ende des Eintrags zeigt an, daß zu den vom Benutzer eingegebenen Kriterien kein Fahrzeug in der Datenbank gefunden wurde.

Treten während dem Betrieb der JAWA Fehler auf, ist es hilfreich zu wissen, welcher Fehler den Betrieb der Anwendung unterbrochen bzw. gestört hat. Aus diesem Grund werden alle auftretenden Fehler, sowohl Benutzer- als auch Systemfehler, in der Datei jawa.err protokolliert. Jeder Eintrag setzt sich aus dem aktuellen Datum, dem Rechnernamen des Benutzers, der die JAWA gestartet hat, und einem Fehlertext zusammen. Ein Beispiel hierfür ist der folgende Eintrag:

```
[12/June/1995:14:06:00] [hpsp1.nm.informatik.uni-muenchen.de]
``ORACLE error detected: ORA-12537: TNS:connection closed``
```

Die Lokation der beiden Protokolldateien wird in der Header-Datei `jawa.h` festgelegt, und kann jederzeit beliebig geändert werden.

6.8 Fehlerbehandlung

6.8.1 Benutzerfehler

Es wurden spezielle Mechanismen in die JAWA implementiert, um die Toleranz gegenüber Benutzerfehlern bei der Bedienung der Anwendung zu erhöhen.

Leeres Eingabeformular

Schickt ein Benutzer ein HTML-Formular an die JAWA, ohne dabei ein Suchkriterium ausgewählt zu haben, hätte das eine Auflistung aller in der Datenbank enthaltenen Fahrzeuge zur Folge. Da ein Abzug der Daten aus Sicht des Betreibers der Anwendung verhindert werden soll, wurde die Ausgabe auf 40 Fahrzeuge beschränkt. Die bedeutet, daß bei Adressierung der JAWA über ein leeres Eingabeformular als Ergebnis immer die ersten 40 Datensätze der Datenbank angezeigt werden. Aus diesem Grund wird im Falle eines leeren Eingabeformulars die Anwendung verlassen und eine entsprechende Fehlermeldung an den Benuzter zurückgeschickt, mit dem Hinweis, daß eine Suche ohne Eingabe eines Suchkriteriums unzulässig ist.

Konsistenzprüfung der Benutzereingaben

Ein Beutzer, der eine ergebnislose Suche initiiert hat, weiß nicht, ob das durch ihn spezifizierte Fahrzeug momentan nicht in der Datenbank enthalten ist oder gar nicht existiert. Dies bedeutet, daß durch entsprechende Kombination der Suchkriterien `Modell`, `Antriebsart` und `Karosserie` die Suche nach dem entsprechenden Fahrzeug immer fehlschlägt, weil dieses Fahrzeug gar nicht existiert. Um diese Art der ergebnislosen Suche zu unterbinden, werden die Benutzereingaben einer Konsistenzprüfung unterzogen, bevor eine Suche auf der Datenbank gestartet wird. Es wurde dazu in der JAWA-Datenbank die Tabelle `MODELLE` angelegt, die alle gültigen Kombinationen aus `Modell`, `Antriebsart` und `Karosserie` enthält. Das Feld `MOD_VAR` ist eine Kombination der Felder `MOD_TYP` und `KARO_VAR` aus der Tabelle `FAHRZEUG`. `KARO_VAR` ist ein Feld, das alle zu einer Modellvariante zulässigen Karosserievarianten enthält. Ist das vom Benutzer im Eingabeformular spezifizierte Fahrzeug nicht in der Tabelle `MODELLE` enthalten, wird eine Fehlermeldung generiert, die den Benutzer darauf hinweist, daß die von ihm eingegebene Kombination aus `Modell`,

Tabelle 6.2: Aufbau der Tabelle `MODELLE`

Feldname	Null ?	Datentyp	Inhalt
MOD_VAR		CHAR(6)	Modellvariante, die sich aus Modelltyp und Antriebsvariante zusammensetzt
KARO_VAR		CHAR(10)	Zeichenkette, die alle zulässigen Karosserievarianten enthält

`Antriebsart` und `Karosserie` unzulässig ist. Die Anwendung beendet sich daraufhin. Der Benutzer hat auf der Startseite der JAWA die Möglichkeit, die aktuelle Modellübersicht einzusehen, die ihn bei der Auswahl eines Fahrzeugs unterstützen soll. Die Struktur der Datenbanktabelle `MODELLE` ist in Tabelle 6.2 dargestellt. Ist die Benutzereingabe hingegen gültig, werden die in der Datenbank gefundenen Fahrzeuge ausgelesen. Ändert sich die Produktpalette der BMW AG, das heißt, werden bestehende Modell- bzw. Karosserievarianten entfernt oder neue hinzugefügt, ist die Tabelle `MODELLE` in der JAWA-Datenbank entsprechend anzupassen.

Eine weitere Möglichkeit eine Konsistenzprüfung der Benutzereingabe zu realisieren, hätte darin bestanden, eine Datei zu erstellen, die ebenfalls wie die Tabelle `MODELLE` alle gültigen Kombinationen aus `Modell`, `Antriebsart` und `Karosserie` enthält. Vor Zugriff auf die Datenbank überprüft das CGI-Skript, ob die vom Benutzer eingegebene Kombination aus `Modell`, `Antriebsart` und `Karosserie` in der Datei enthalten und somit gültig ist. Erst nach erfolgreicher Konsistenzprüfung würde der Zugriff auf die Datenbank erfolgen. Dieser Ansatz hat gegenüber der implementierten Methode den Vorteil, daß bei ungültiger Benutzereingabe kein Datenbankzugriff erfolgt. Dies würde zur Reduzierung der Datenbankzugriffe sowie zur Reduzierung der Antwortzeit bei ungültigen Benutzereingaben führen.

Durch das Einführen der Konsistenzprüfung der Benutzereingaben konnte die Anzahl der ergebnislosen Suchen und somit die Anzahl der „ineffizienten“ Datenbankzugriffe reduziert werden.

6.8.2 Systemfehler

Neben den oben aufgeführten Benutzerfehlern werden auch auftretende Systemfehler durch die Anwendung abgefangen. Werden beispielsweise die für die Kommunikation zwischen WWW-Server und JAWA notwendigen CGI-Umgebungsvariablen nicht belegt, ist die JAWA nicht in der Lage, die Benutzereingabe korrekt zu verarbeiten. Bei dieser Art von Fehler wird eine Fehlermeldung generiert und die Anwendung beendet sich.

Ein in der Uni-Umgebung häufiger Fehler ist das Auftreten eines Timeouts, der durch die Oracle-spezifische Datenbank-Kommunikationssoftware *SQL*Net* verursacht wird. Der Rechner, auf dem der Datenbank-Server läuft, wird zugleich als File-Server eingesetzt und ist zu Zeiten hoher Auslastung nicht in der Lage, die Anfragen entsprechend schnell zu bearbeiten, was zu einem Timeout führt. Die JAWA generiert eine Fehlermeldung und beendet sich. Dieser Fehler kann in der BMW-Umgebung hingegen beseitigt werden, wenn der Wert für den Timout heraufgesetzt wird oder wenn WWW-Server bzw. JAWA und Datenbank-Server auf dem gleichen Rechner laufen und zur Kommunikation *SQL*Net* nicht mehr verwendet werden muß.

6.9 Optimierungsmöglichkeiten

6.9.1 Schutz vor Datenabzug

Der derzeitige Prototyp der JAWA ist nicht in der Lage, den Abzug der Daten aus der Datenbank zu unterbinden. Die Anzeige der gefundenen Fahrzeuge ist zwar derzeit auf 40 beschränkt, doch die Anzahl der Zugriffe je Zeitraum ist nicht beschränkt. Um die JAWA vor einem möglichen Datenabzug zu schützen, könnten die in Kapitel 4.8.3 beschriebenen Maßnahmen implementiert werden.

6.9.2 Die JAWA als dynamisches Datenbank-Gateway

Die JAWA ist als statisches Datenbank-Gateway implementiert. Zur Konfiguration der Auswahlkriterien im Eingabeformular der JAWA wird derzeit nur die Konfigurationsdatei `jawaform.conf` (Anhang B) verwendet. Sollen Änderungen an dem Eingabeformular bzw. an den Ausgabeseiten vorgenommen werden, ist dazu der Source-Code zu modifizieren. Um dies zu umgehen und um die Wartungsfreundlichkeit des Systems zu verbessern, könnten nächste Versionen der JAWA als dynamische Gateways (Kapitel 4.7.2) implementiert werden. Anstatt die für den Datenbankzugriff notwendigen Informationen fest im CGI-Skript zu verankern, könnten sie ausgelagert und in einer oder mehreren Konfigurationsdateien gespeichert werden. Eine Konfigurationsdatei könnte Informationen enthalten, die für die Generierung einer Datenbankanfrage wichtig sind. Relevante Informationen sind dazu beispielsweise die Benutzerkennung und das Paßwort für den Datenbankzugriff, der Name der Datenbank, die Art des Zugriffs (lesend oder schreibend), die Namen der Tabellen innerhalb der Datenbank, eventuelle „Join“-Beziehungen zwischen den einzelnen Tabellen, falls auf mehrere Tabellen gleichzeitig zugegriffen wird, sowie die Spaltennamen der einzelnen Tabellen, die aus der Datenbank extrahiert werden sollen. Daneben könnte diese Datei den Datentyp der einzelnen Tabellenspalten, die Länge der extrahierten Daten sowie eventuelle Formatierungsanweisungen für die Daten enthalten, sowie mögliche Optionen, nach denen die extrahierten Daten sortiert werden, bevor sie an den Benutzer zurückgeschickt werden. Werden mehrere solcher Konfigurationsdateien verwendet, wäre man sogar in der Lage mit einem Datenbank-Gateway unterschiedliche Datenbankzugriffe zu realisieren.

6.10 Implementierungsalternativen

Neben der Programmiersprache *C* eignet sich *perl* hervorragend für die Entwicklung von CGI-Skripten. *perl* ist eine Programmiersprache, die sehr effiziente Möglichkeiten bietet, auf Textdateien zuzugreifen, Daten zu modifizieren sowie Daten aus einer Datei zu extrahieren. *perl* bietet außerdem Erweiterungen für alle gängigen Datenbanksysteme, wie beispielsweise *oraperl* für Oracle, *sybperl* für Sybase, *isqlperl* für Informix usw., mit denen sich Zugriffe auf die jeweilige Datenbank realisieren lassen. Mit den entsprechenden Erweiterungen kann *perl* somit auch zur Implementierung von CGI-Datenbank-Gateways herangezogen werden. Da die relationalen Datenbankoperationen typischerweise textorientiert sind, eignet sich *perl* sehr gut dazu, den aus der Datenbank extrahierten Datenstrom zu verarbeiten. Nach der Implementierung des Datenbank-Gateways ist man allerdings an ein bestimmtes Datenbanksystem gebunden, weil die Erweiterungen zu *perl* datenbankspezifisch sind. Eine Änderung des zugrundeliegenden Datenbanksystems würde ein Änderung des Gateways implizieren.

Verwendet man für die Datenbankzugriffe hingegen Standard SQL-Anweisungen, die in den C-Source-Code eingebettet sind, ist man nicht an das zugrundeliegende Datenbanksystem gebunden. Dies bedeutet, daß die Anwendung auf unterschiedlichen Datenbanksystemen ablauffähig ist, ohne daß dazu der Source-Code der Anwendung geändert werden muß. Um die Unabhängigkeit vom verwendeten Datenbanksystem zu wahren, wurde die JAWA in der Programmiersprache *C* unter Verwendung von ESQL-Anweisungen implementiert.

Eine weitere Alternative der Implementierung stellt die Verwendung von *SQL*Report* dar. Es handelt sich dabei um eine Oracle-spezifische Anwendung, mit der sog. „Reports“ erstellt werden können. Ein „Report“ ist dabei eine Textdatei, die Daten enthält, die mit Hilfe von *SQL*Report* aus der Datenbank extrahiert wurden. Das Format eines solchen „Reports“ kann durch den Benuzter beliebig spezifiziert werden. *SQL*Report* setzt sich aus zwei Komponeneten zusammen. Der „Report-Generator“ ermöglicht das Auslesen der Daten aus der Datenbank sowie das Einbetten der Daten in die „Reports“. Mit Hilfe des „Report-Formatierers“ wird das Format der „Reports“ festgelegt. Beide Anwendungen besitzen eine Kommandozeilen-Schnittstelle, über die sie aufgerufen werden können. Dies ermöglicht das Ausführen der beiden Anwendungen von einer Programmiersprache wie beispielsweise *C* oder *perl* heraus.

Der Entwicklung eines solchen CGI-Datenbank-Gateways kann beispielsweise folgende Architektur zugrunde liegen: Ein Benutzer startet über ein HTML-Formular eine Anfrage an die Datenbank. Das referenzierte CGI-Skript generiert entsprechend den übergebenen Parametern eine Datei, die Anweisungen für den „Report-Generator“ enthält und zur Laufzeit von diesem ausgewertet wird. Nach Aufruf des „Report-Generators“ interpretiert dieser die beim Aufruf übergebene Datei und extrahiert entsprechend den dort enthaltenen Anweisungen die Daten aus der Datenbank. Nach dem Auslesen der Daten wird mit Hilfe des „Report-Formatierers“ ein „Report“ generiert, der HTML-Anweisungen enthält und einem normalen WWW-Dokument entspricht. Die aus der Datenbank extrahierten Daten werden in das HTML-Dokument eingebettet. Als Ergebnis der Anfrage wird das so generierte HTML-Dokument an den WWW-Client zurückgeschickt.

Kapitel 7

Auswirkungen einer Firewall auf das WWW

7.1 Einführung

Da die Anbindung eines lokalen Netzes an das Internet viele Risiken mit sich bringt, wird in der Regel nur einem einzigem Rechner im lokalen Netz der Zugang zum Internet gewährt. Diese Machine, die auch als Firewall bezeichnet wird, ist mit spezieller Software ausgestattet, um das lokale Netz vor unerlaubten Zugriffen und Attacken von außen zu schützen. Daneben wird der Firewall auch eingesetzt, um den unbemerkten und unerlaubten Export von proprietären Informationen nach außen zu unterbinden [Ran93].

Da sich parallel zu dieser Arbeit eine eigene Diplomarbeit mit dem Thema „Firewalls" beschäftigt [Hau95], wird im Rahmen dieser Diplomarbeit nicht näher auf die verschiedenen Konzepte und Archtitekturen von Firewalls eingegangen.

7.2 Proxy-Server

7.2.1 Funktionsweise eines Proxy-Servers

Ein Proxy-Server, oder auch kurz als Proxy bezeichnet, wird dazu verwendet, Benutzern innerhalb einer Firewall den Zugriff auf das WWW zu ermöglichen [LA94]. Ein Proxy-Server ist ein spezieller HTTP-Server, der entweder auf der Firewall-Maschine oder auf einem anderen Internet-brechtigten Rechner innerhalb der Firewall läuft. Möchte ein Client innerhalb der Firewall ein Dokument vom einem WWW-Server im Internet abrufen, wendet er sich dazu an den Proxy-Server. Dieser nimmt die Anfrage entgegen, leitet sie an den Server außerhalb der Firewall weiter, liest die Antwort des entfernten Servers und schickt diese dann an den anfragenden Client zurück (Abbildung 7.1).

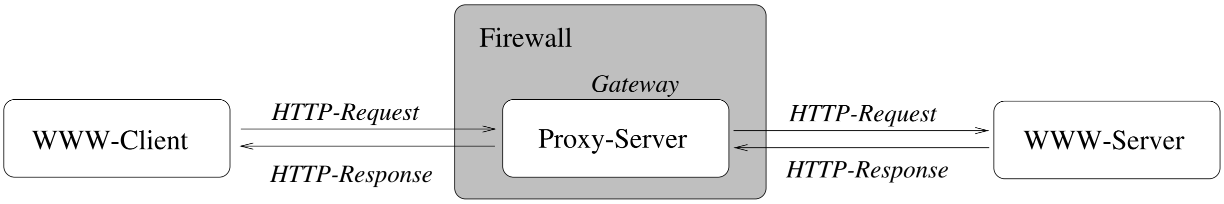

Abbildung 7.1: HTTP-Zugriff über Proxy-Server

Der Proxy-Server agiert bei dieser Art der Kommunikation sowohl als Client als auch als Server. Gegenüber dem anfragenden WWW-Client verhält er sich als WWW-Server, gengenüber dem entfernten Server als Client. Der Header der Client-Anfrage wird vom Proxy unverändert für seine Anfrage an den entfernten Server verwendet, wodurch der WWW-Client keinen Verlust an Funktionalität erleidet, wenn er einen Proxy-Server verwendet. Da das WWW weitere Internet-Dienste unterstützt ist der Proxy-Server in der Lage, als Gateway zu anderen Information-Servern wie FTP-, Gopher-,WAIS oder News-Servern zu agieren. Abbildung 7.2 zeigt, daß die Kommunikation zwischen Client und Proxy stets über das HTTP-Protokoll abgewickelt wird. Der Proxy kommuniziert mit dem entfernten Server über das jeweilige Protokoll, in diesem Fall über das FTP-Protokoll. Zudem muß der Proxy-Server die Objekte, die er von den entfernten Server erhält, in HTTP-Objekte umwandeln, bevor er sie an den WWW-Client zurückschickt.

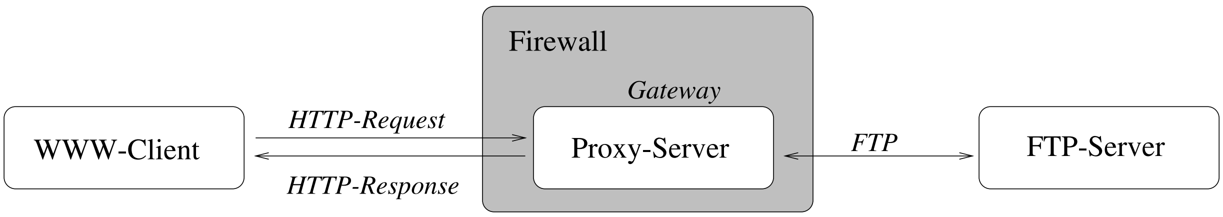

Abbildung 7.2: FTP-Zugriff über Proxy-Server

Durch die Verwendung eines Proxy-Servers lassen sich die Zugriffe auf Internet-Ressourcen kontrollieren und einschränken. Die Zugriffskontrolle kann auf der Zugriffmethode, der IP-Adresse oder auf Domain-Namen basieren. Das bedeutet beispielsweise, daß der Proxy-Server nur einigen speziellen Rechnern im lokalen Netz den Zugang zum Internet erlaubt, um zu verhindern, daß durch mißbräuchliche oder unbeabsichtigte Nutzung des Internets unnötige Kosten enstehen. Daneben läßt sich mit Hilfe eines Proxy-Servers am Internet-Übergang das WWW-Zugriffsverhaltens der lokalen Benutzer analysieren. Der Proxy zeichnet dazu Client-Transaktionen auf. Ein Eintrag in einer Log-Datei enthält beispielsweise die IP-Adresse des anfragenden Clients, Datum und Uhrzeit, die URL des angeforderten Objekts, die Größe der übertragenden Daten in Bytes sowie den Statuscode der HTTP-Transaktion.

Neben diesen Vorteilen enthält der Einsatz eines Proxy-Servers auf der Firewall-Maschine auch einen entscheidenden Nachteil. Da die Firewall den einzigen Angriffspunkt des lokalen Netzes gegenüber dem Internet darstellt, sollten aus Gründen der Sicherheit so wenig wie möglich Applikationen auf der Firewall laufen, um die eventuell vorhandenen Sicherheitslücken auf ein Minimum zu reduzieren. Denn je umfangreicher eine Software ist, desto wahrscheinlicher ist es, daß sie Fehler und somit Sicherheitslücken enthält. Wird beispielsweise der CERN Server als Proxy-Server auf der Firewall-Maschine eingesetzt, stellt er Angreifern von außen ein potentielles Angriffsziel dar.

7.2.2 Proxy-Server mit Cache

Ein Proxy-Server kann in der Regel auch als Cache für Dokumente dienen, die von entfernten Informations-Servern geholt wurden [Klu95b]. Die vom Proxy-Server angeforderten Dokumente werden nicht nur an den Client zurückgeschickt, sondern auch in einem Cache auf der Festplatte zwischengespeichert (Abbildung 7.3). Dies setzt voraus, daß der Proxy einen ausreichend großen Cache zur Verfügung stellt.

Stellt ein Client eine Dokumentenanfage, die vor ihm bereits ein anderer Client gestellt hat, muß der Proxy-Server keine Verbindung zu einem entfernten Server aufbauen, sondern schickt dem Client das Dokument zurück, das im Cache zwischengespeichert wurde. Der Einsatz eines Proxy-Servers mit Cache hat gegenüber einem normalen Proxy-Server viele Vorteile:

- Verkürzung der Antwortzeit für den Benutzer

  Der Zugriff auf WWW-Seiten wird beschleunigt, da der Client lediglich eine lokale Verbindung zum Proxy aufbauen muß. Ist das Dokument im Cache enthalten, entfällt der Verbindungsaufbau zu einem entfernten Server im Internet.

- Reduzierung des Internet-Verkehrs

  Die Anfragen an Proxy-Server belasten die Internet-Verbindung nicht. Die Leitungsbandbreite steht somit anderen Diensten zur Verfügung.

- Verringerung der Übertragungskosten

  Für den Transport vom Client zum Proxy fallen keine Gebühren an. Die Kosten, die durch das WWW enstehen, werden dadurch reduziert.

- Nutzung von WWW auch bei Fehlen einer Verbindung zum externen Server

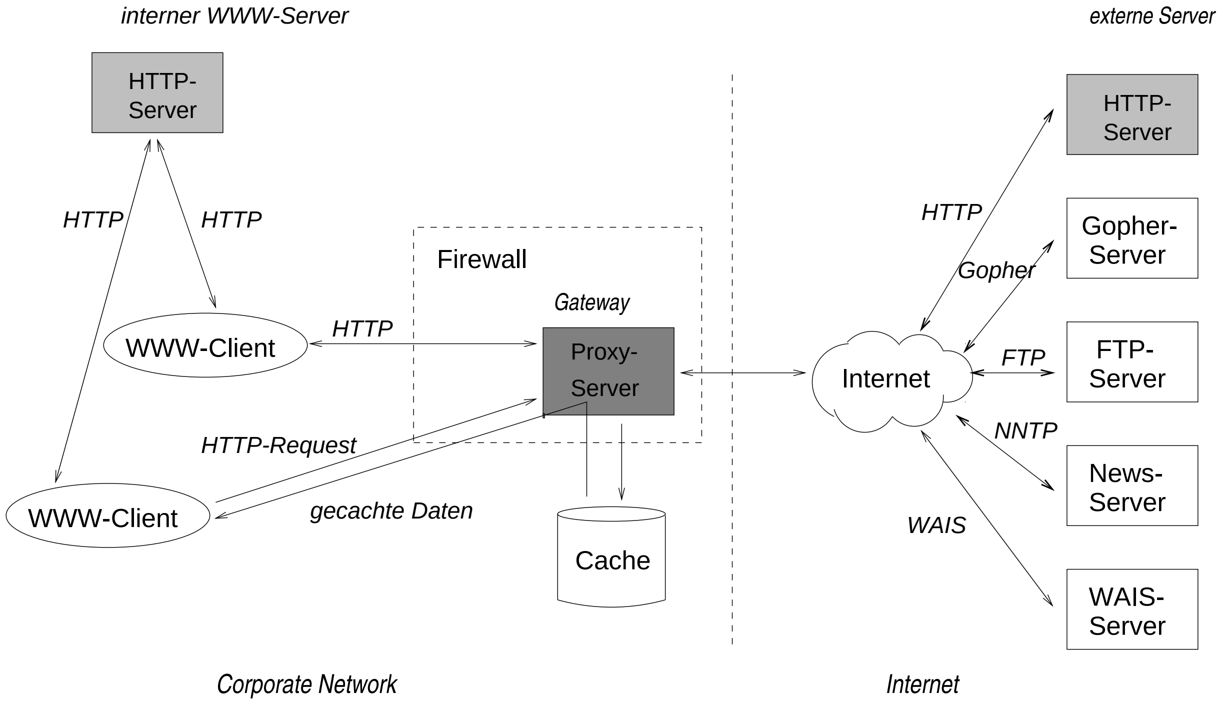

Abbildung 7.3: Zugriff auf entfernte und lokale Server unter Verwendung eines Proxy-Servers

> Ist die Verbindung zu einem entfernten Server unterbrochen, weil der Server momentan nicht verfügbar ist oder weil die externe Anbindung unterbrochen ist, können die Dokumente auf diesem Server trotzdem genutzt werden, falls sie im Cache des Proxy-Servers zwischengespeichert sind.

Aufgrund der oben aufgeführten Vorteile, kann ein Proxy-Server mit Cache auch für solche Institutionen vorteilhaft sein, die keinen Firewall am Internet-Zugang verwenden, sondern lediglich die Cache-Funktionalität des Proxy-Servers nutzen möchten. Dadurch können nicht nur die Zugriffe auf die WWW-Seiten beschleunigt sondern auch die durch WWW erzeugte Netzlast und somit auch die Übertragungskosten reduziert werden.

Da ein Proxy beide Seiten der Kommunikation versteht, können zwei oder mehr Proxies kaskadenartig hintereinander gehängt werden. Dieser Vorgang wird auch als „Proxy-Chaining“ bezeichnet. Man könnte sich vorstellen, daß beispielsweise ein größeres Unternehmen jeder Abteilung oder jedem Unternehmensbereich einen eigenen Proxy-Server zur Verfügung stellt, der die WWW-Dokumente mit hoher Geschwindigkeit ausliefert. Die Proxy-Server der Abteilungen werden von einem zentralen Proxy-Server, der die Verbindungen zu externen Servern unterhält, bedient. Um eine Überlastung des zentralen Proxy-Servers auszuschließen, sollten lediglich die Proxy-Server der einzelnen Abteilungen und keine normalen

Benutzer Zugang zum zentralen Proxy-Server erhalten.

Den erwähnten Vorteilen eines Proxy-Servers mit Cache stehen auch einige Nachteile gegenüber:

- Veraltete Daten im Cache

  Der Cache enthält möglicherweise eine veraltete Version des angeforderten Dokuments und liefert diese an den Client aus.

- Zusätzliche Ressourcen werden benötigt

  Um einen Proxy-Server betreiben zu können werden zusätzliche Resourcen wie Rechner mit ausreichend Plattenplatz, Software und Personal benötigt.

- Wenig Zugriffe auf gleiche Dokumente

  Greifen viele Benutzer auf viele verschiedene Dokumente zu, werden der Cache und die dadurch gebundenen Ressourcen nutzlos.

- Verfälschung der Zugriffsstatistiken bei entfernten Servern

  Je mehr Anwender über einen Proxy-Cache auf WWW-Seiten zugreifen, desto weniger stimmen die beim Originalserver registrierten Zugriffszahlen mit der tatsächlichen Anzahl von Zugriffen überein.

Um das Problem von veralteten Dokumenten beim Einsatz eines Proxy-Cache zu umgehen, versucht der Proxy-Server die Dokumente im Cache so häufig wie nötig zu aktualisieren und gleichzeitig so wenig wie möglich externe Zugriffe zu machen. Es werden hierzu die HTTP-Header `Last-Modified`, `If-Modified-Since` sowie `Expires` verwendet. `Expires` gibt an, wann ein Dokument als veraltet zu betrachten ist. Damit könnte man das Problem der veralteten Dokumente umgehen. Da es sich bei `Expires`, wie bei den anderen beiden Headern, um optionale Informationen handelt, ist ein WWW-Server nicht gezwungen, diese bei Auslieferung eines Dokuments zu verwenden, was in der Praxis auch nur sehr selten vorkommt. Mit Hilfe des Headers `If-Modified-Since` und der Angabe eines Datums kann der Client eine bedingte GET-Anfrage stellen. Das bedeutet, daß der Proxy-Server bzw. der entfernte Server das angeforderte Dokument nur dann schickt, wenn es nach dem in der Anfrage spezifizierten Datum geändert wurde.

Der CERN-Server besitzt, wie in Kapitel 3.1 gesehen, die Funktionalität als Proxy-Server zu agieren. Er verwendet den Header `Last-Modified`, falls `Expires` nicht verwendet wird und berechnet daraus mit Hilfe eines heuristischen Verfahrens ein künstliches Verfallsdatum eines zwischengespeicherten Dokuments.

Dynamisch generierte Dokumente stellen ein weiteres Problem für den Proxy- Server dar. Sie enthalten weder `Expires-` noch `Last-Modified-`Header, falls diese von den Entwicklern der CGI-Skripten nicht explizit angegeben werden. Dynamische Dokumente verlieren in der Regel sofort nach dem Aufruf ihre Gültigkeit und dürfen in den Cache nicht aufgenommen werden. Um dem Proxy-Server mit Cache Probleme und unnötigen Datenverkehr zu ersparen, wird den Entwicklern solcher externer Programme geraten, immer einen gültigen `Expires`-Header anzugeben.

7.2.3 Client-seitige Anforderungen

Damit eine Client die Dienste eines Proxy-Servers in Anspruch nehmen kann, müssen die Umgebungsvariablen `http_proxy`, `gopher_proxy`, `wais_proxy`, `ftp_proxy`, `news_proxy` und `no_proxy` gesetzt werden. Diese enthalten den Namen und den Port des Proxy-Servers, der für den jeweiligen Dienst zu verwenden ist, in Form einer URL. Die Umgebungsvariable

```
http_proxy=http://wwwcache.bmw.de:8009/
```

teilt einem WWW-Client beispielswiese mit, daß er für HTTP-Objekte den Proxy- Server `wwwcache.bmw.de` auf Port `8009` verwenden soll.

Die Variable `no_proxy` hat eine eine besondere Bedeutung. Sie enthält eine Liste von Domain- und Rechnernamen, für die der Client nicht den Umweg über einen Proxy-Server wählen soll. Das bedeutet, wenn ein Client ein Dokument von einem dieser Rechner abruft, wendet er sich direkt an den jeweiligen Rechner. In Abbildung 7.3 ist dargestellt, daß sich WWW-Clients innerhalb des lokalen Netzes direkt an einen internen WWW- Server wenden, ohne dabei Kontakt zum Proxy-Server aufnehmen zu müssen.

7.3 SOCKS-Server

7.3.1 Funktionsweise eines SOCKS-Servers

Ist der Zugang zum Internet durch einen Firewall geschützt, kann neben dem Proxy-Server auch ein SOCKS-Server („Socket Service Server") verwendet werden, um den Benutzern innerhalb eines lokalen Netzes Zugang zum Internet zu ermöglichen. Der SOCKS-Server [KK94] ist ein Prozess, der auf der Firewall-Maschine läuft und den Zugang zum bzw. vom Internet überwacht.

Fordert ein Client innerhalb der Firewall ein Objekt von einem Server im Internet an, wendet er sich dazu an den SOCKS-Server, der daraufhin eine Verbindung zum jeweiligen Rechner aufbaut. Client und SOCKS-Server kommunizieren über ein proprietäres Protokoll, das sog. SOCKS-Protokoll. Für den Benutzer erfolgt der Zugriff auf entfernte Server völlig transparent. Diese Transparenz wird durch die Verwendung von SOCKS-Bibliotheksfunktionen erreicht, die anstelle der normalen Socket-Funktionen von der jeweiligen Anwendung zum Verbindungsaufbau verwendet werden. Die SOCKS-Bibliotheksfunk-

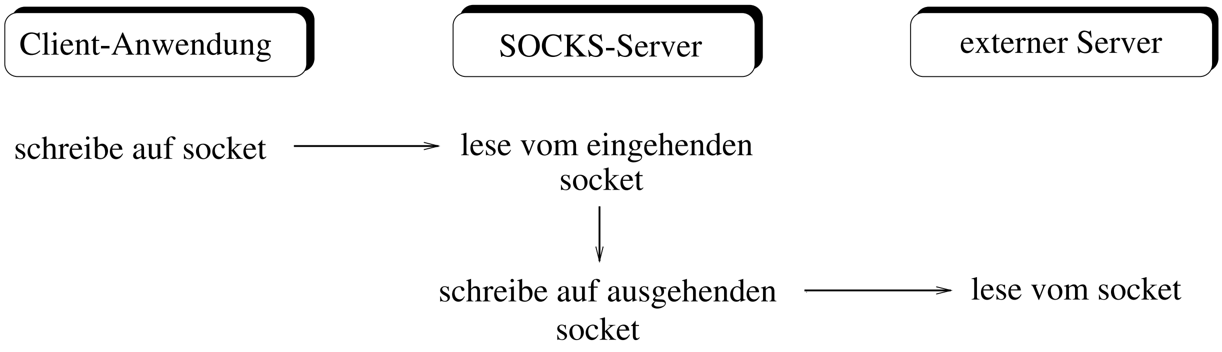

Abbildung 7.4: SOCKS-Server als transienter Socket-Server

tionen sind so ausgelegt, daß alle Netzverbindungen ausschließlich über den SOCKS-Server abgewickelt werden.

Möchte eine Anwendung eine Verbindung zu einem entfernten Server aufbauen, ruft sie die SOCKS-Bibliotheksfunktionen auf, die eine Verbindung zum SOCKS-Server auf dem Firewall initiieren. Der SOCKS-Server prüft, ob der anfragende Rechner berechtigt ist, den jeweiligen Internet-Dienst zu nutzen und baut im positiven Fall eine Verbindung zum entfernten Server auf. Aus Sicht des entfernten Servers erscheint der SOCKS-Server als Originator der Anfrage. Alle Daten die der SOCKS-Server von der externen Verbindung erhält werden von diesem unverändert an den internen Rechner weitergeleitet. Der SOCKS-Server fungiert wie in Abbildung 7.4 dargestellt, als transienter Socket-Server zwischen der internen Client-Anwendung und dem externen Rechner. Seine Aufgabe beschränkt sich auf das „Routen“ der Informationen zwischen internen und externen Socket-Verbindungen.

7.3.2 Zugriffsschutz und Logging

Der Zugriff auf externe Ressourcen wird über eine Konfigurationsdatei gesteuert, die vom SOCKS-Server beim Start ausgewertet wird. Der Zugriffsschutz basiert auf IP-Adressen. Der SOCKS-Server ist weiterhin in der Lage, Zugriffe über die Unix-Schnittstelle `syslog` zu protokollieren. Es werden sowohl abgelehnter sowie erfolgreicher Verbindungsaufbau und aufgetretene Fehler aufgezeichnet. Die Protokollinformation enthält neben dem Rechner- und Benutzernamen auch die Art der Anfrage.

7.3.3 Client-seitige Anforderungen

Um mit einem SOCKS-Server kommunizieren zu können, müssen zu der jeweiligen Client-Anwendung die SOCKS-Bibliotheksfunktionen hinzugebunden werden. Diese Art von Clients werden als „socksified“ Clients bezeichnet. Die Client-Anwendung verwendet für die

Kommunikation mit erfernten Servern auschließlich die SOCKS-Bibliotheksfunktionen anstelle der normalen Socket-Funktionen (Abbildung 7.5). Das SOCKS-Paket, das aus SOCKS-Server und SOCKS-Bibliotheksfunktionen besteht, ist frei verfügbar.

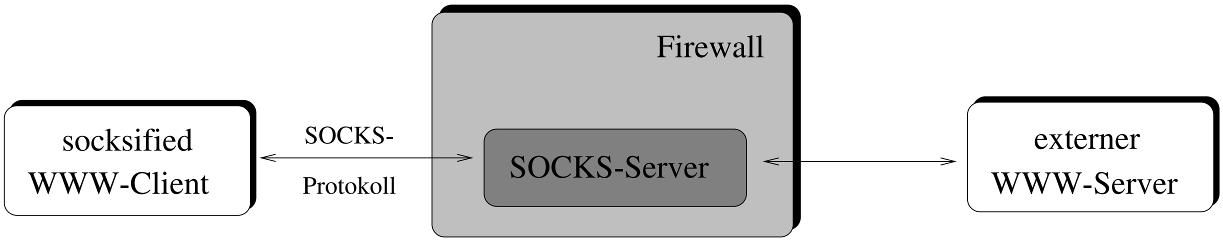

Abbildung 7.5: SOCKS-Server und socksified WWW-Client

7.4 Vergleich von SOCKS- und Proxy-Servern

Durch den Einsatz eines SOCKS-Servers haben die Benutzer transparenten Zugriff auf Dienste, die von externen Servern angeboten werden. Dies setzt aber voraus, daß hierfür zuerst die ursprünglichen Clients in socksified Clients modifiziert werden. Ein Problem ergibt sich dabei aus der Tatsache, daß PC-Software größtenteils nicht als Source-Code verfügbar ist, was es dem Benutzer unmöglich macht, seine Clients selbst zu modifizieren. Er ist somit auf Anbieter von socksified Client-Software angewiesen. Beim Einsatz eines Proxy-Servers ist dieses Problem nicht gegeben, weil die Clients hierfür lediglich bestimmte Umgebungsvariablen setzen müssen.

Mit Hilfe des SOCKS-Servers wird der Zugang zum Internet überwacht und kann gegebenenfalls für bestimmte Benutzer eingeschränkt werden. Da die Zugriffe auf Ressourcen im Internet protokolliert werden, kann damit das Zugriffsverhalten der lokalen Benutzer analysiert werden. Die Analysemöglichkeiten sind dabei nicht so vielfältig wie beim Proxy-Server, der nicht nur erfolgreichen und fehlerhaften Zugriff protokolliert, sondern auch Datum und Uhrzeit, sowie die Art des Zugriffs und die Größe der übertragenen Daten.

Im Vergleich zu einem Proxy-Server, der auf der Firewall-Maschine läuft, stellt der SOCKS-Server ein geringeres Sicherheitsrisiko dar, weil er weniger Code enthält und damit die Wahrscheinlichkeit, eine Sicherheitslücke zu enthalten, geringer ist . der SOCKS-Server ist aber im Gegensatz zum Proxy-Server mit Cache nicht in der Lage, Client-Anfragen zwischenzuspeichern und die daraus resultierenden Vorteile, wie Reduzierung der Antwortzeiten und Übertragungskosten, zu nutzen.

7.5 Socksified Proxy-Server mit Cache

Möchte man aus Gründen der Sicherheit anstelle eines Proxy-Servers einen SOCKS-Server auf der Firewall-Maschine einsetzen, um Benutzern innerhalb des lokalen Netzes transparenten Zugriff auf das Internet zu ermöglichen, aber zugleich nicht auf das Zwischenspeichern von Dokumenten sowie den daraus resultierenden Vorteilen verzichten, kann man SOCKS- sowie Proxy-Server mit Cache, wie in Abbbildung 7.6 dargestellt, einsetzen.

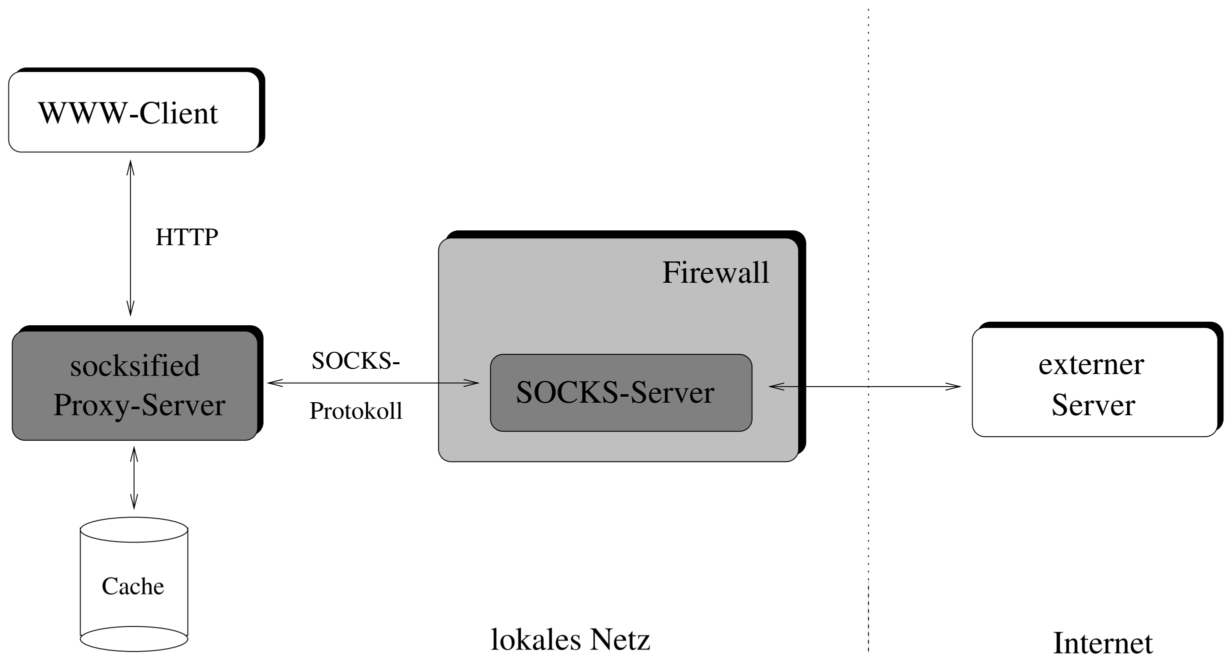

Abbildung 7.6: Socksified Proxy-Server mit Cache

Auf der Firewall läuft ein SOCKS-Server, der den Zugang zum bzw. vom Internet überwacht. Innerhalb des lokalen Netzes existiert ein Proxy-Server mit Cache, der um die SOCKS-Bibliotheksfunktionen erweitert wurde, ein sog. socksified Proxy-Server. SOCKS- und Proxy-Server kommunizieren über das SOCKS-Protokoll, Proxy-Server und Clients kommunizieren über HTTP. Damit ein Client die Dienste des Proxy-Servers in Anspruch nehmen kann, müssen die entsprechenden Umgebungsvariablen gesetzt sein.

Stellt ein Client eine Anfrage an einen externen Server, wendet er sich zuerst an seinen Proxy-Server. Dieser wiederum stellt einen Anfrage an den SOCKS-Server, der seinerseits eine Verbindung zum jeweiligen Rechner initiiert.

7.6 Mögliche Standorte für WWW-Server

Wenn ein WWW-Server seine Dienste anbietet, sollen sowohl externe Benutzer im Internet als auch lokale Benutzer in der Lage sein, den Dienst zu nutzen. Dabei ist nicht nur die Lage des Servers entscheidend, sondern auch wie er erreicht werden kann und welche Sicherheitsrisiken daraus für das lokale Netz und die darin gebundenen Ressourcen resultieren. Für die Positionierung eines WWW-Servers kommen prinzipiell drei Standorte in Frage: hinter der Firewall, auf der Firewall und vor der Firewall. Bei der Betrachtung der einzelnen Standorte werden dabei die aus der Positionierung des WWW-Servers resultierenden Gefahren für das lokale Netz sowie die möglichen Angriffspunkte für externe Angreifer betrachtet.

7.6.1 WWW-Server hinter der Firewall

Wie in Abbildung 7.7 dargestellt, befindet sich bei dieser Konfiguration der WWW-Server hinter der Firewall im lokalen Netz. Der Firewall muß dabei so konfiguriert sein, daß externe Zugriffe auf den WWW-Server möglich sind. Dies bedeutet, daß bis auf HTTP-Transaktionen der gesamte Verkehr von außen blockiert werden muß. Der WWW-Server ist sowohl vom Internet als auch vom lokalen Netz aus direkt erreichbar. Sowohl der Firewall als auch der WWW-Server selbst stellen bei dieser Konfiguration einen möglichen Angriffspunkt dar, da der WWW-Server von außen direkt erreichbar ist. Die Maschine, auf dem der WWW-Server läuft, muß daher genau so sicher sein wie der Firewall, um unerlaubte Zugriffe von außen zu unterbinden. Wird die Maschine angegriffen und eine Sicherheitslücke entdeckt, stehen dem Angreifer alle lokalen Ressourcen zur Verfügung. Die Sicherheit des lokalen Netzes hängt somit von der Sicherheit dieser Maschine ab. Die Positionierung des WWW-Servers hinter der Firewall ist zwar möglich, aber aus Gründen der Sicherheit nicht ratsam.

7.6.2 WWW-Server auf der Firewall

Läuft der WWW-Server auf der Firewall-Maschine (Abb. 7.8) ist dies vorteilhaft, da sowohl externe als auch interne Benutzer direkten Zugriff auf den angebotenen Dienst haben. Weder die verwendeten Clients noch der Firewall müssen hierzu modifiziert werden. Auf der Firewall sollten so wenig wie möglich Anwendungen laufen, um das Sicherheitsrisiko bei einem möglichen Angriff von außen zu minimieren. Der WWW-Server stellt aber in Bezug auf die Sicherheit ein Risiko dar, da nicht vollkommen ausgeschlossen werden kann, daß große Software-Produkte wie WWW-Server nicht mit Fehlern behaftet sind. Für einen potentiellen Angreifer stellen nur die Firewall selbst und die darauf laufenden Applikationen einen Angriffspunkt dar. Wird der WWW-Server von außen angegriffen und eine Sicherheitslücke entdeckt, steht dem Angreifer ein Loch in der Firewall zur Verfügung, über das er Zugriff auf das lokale Netz hat. Aus diesem Grund ist die Positionierung des WWW-Servers auf der Firewall nicht ratsam.

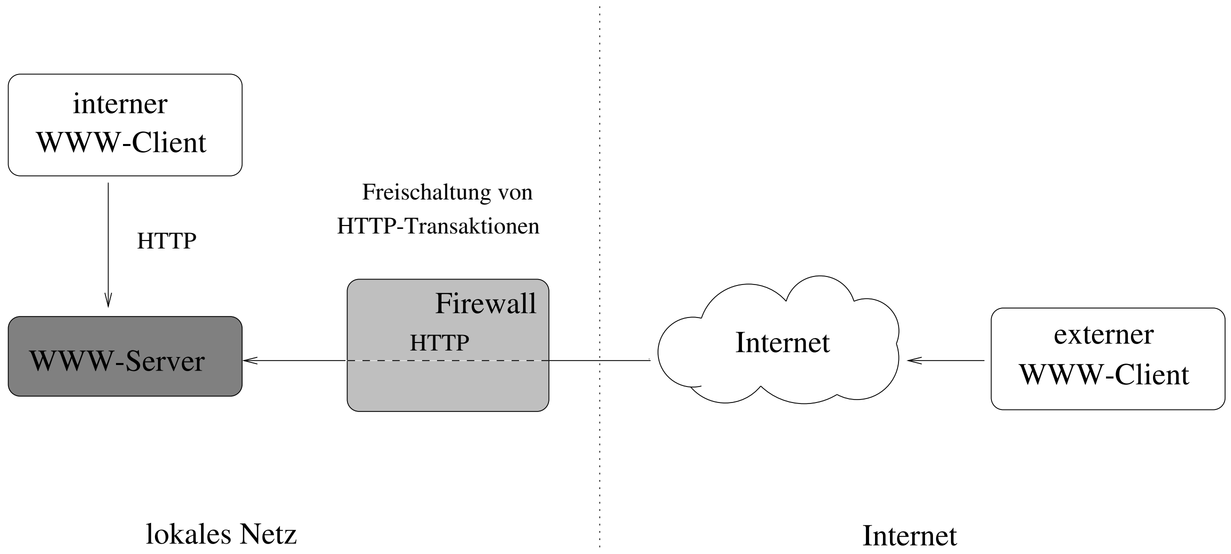

Abbildung 7.7: WWW-Server hinter der Firewall

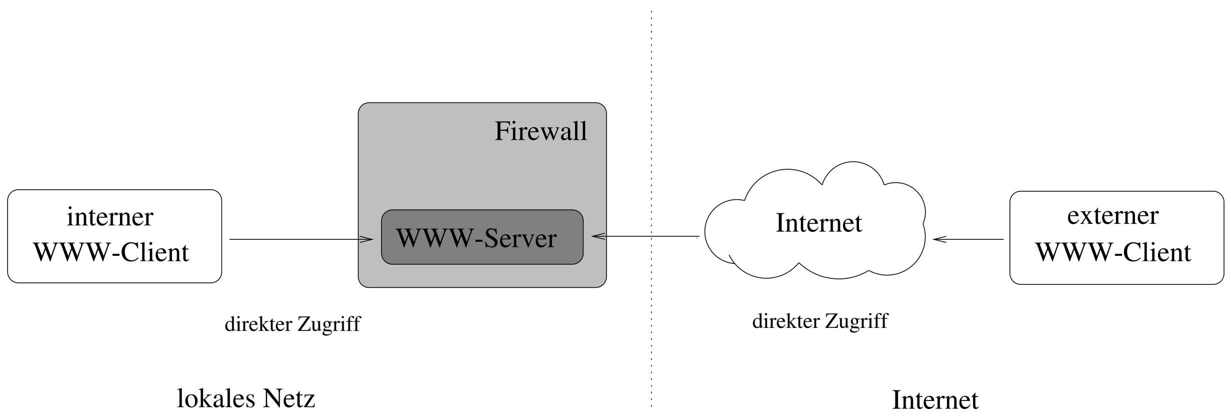

Abbildung 7.8: WWW-Server auf der Firewall

7.6.3 WWW-Server vor der Firewall

Die Positionierung des WWW-Servers vor der Firewall, wie in Abbildung 7.9 dargestellt, stellt im Vergleich zu den anderen beiden Konfigurationen den sichersten Standort des WWW-Servers dar. Der Standort des Server als auch der WWW-Server selbst stellen kein Sicherheitsrisiko für das lokale Netz dar. Sollte der WWW-Server angegriffen und eine Sicherheitslücke entdeckt werden, hat der Angreifer keine Möglichkeit auf lokale Ressourcen zuzugreifen, da unerlaubte Zugriffe von außen weiterhin durch den Firewall unterbunden werden. Der Firewall ist weiterhin der einzige Angriffspunkt für einen potentiellen Angreifer. Der Zugriff auf den WWW-Server ist sowohl für externe als auch interne Benutzer transparent möglich, wobei interne Clients hierzu einen Proxy- oder SOCKS-Server verwenden müssen.

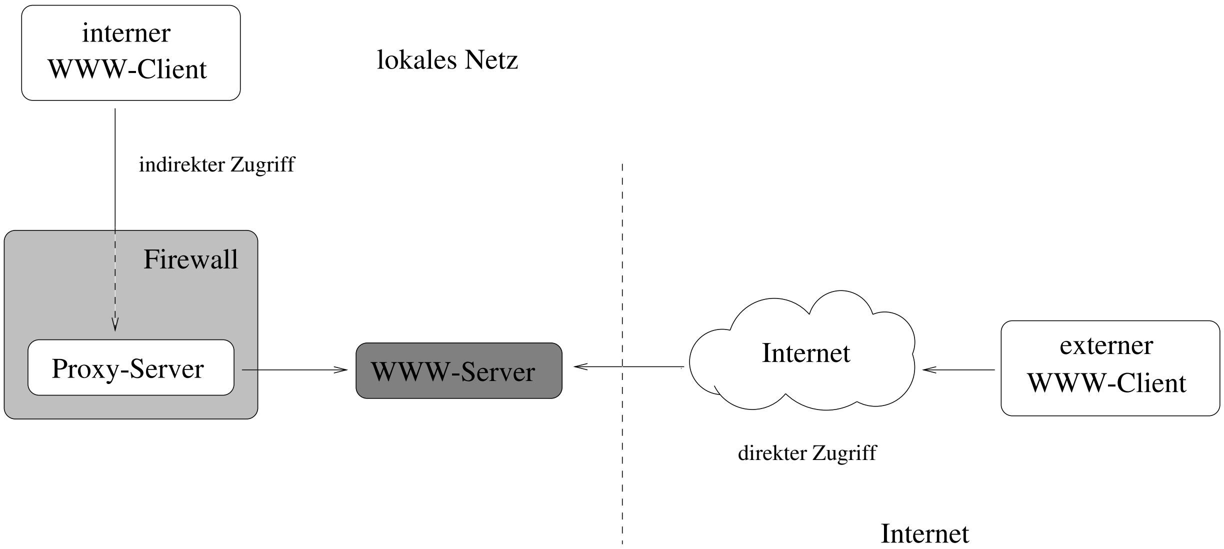

Abbildung 7.9: WWW-Server vor der Firewall

Wie bei den anderen beiden Konfigurationen enthält der WWW-Server vor der Firewall Daten, die sowohl für alle Internet-Benutzer zugänglich sind, als auch interne Unternehmensdaten, die nur lokalen Benutzern zugänglich sein dürfen. Diese Daten können beispielsweise über den Zugriffsschutzmechanismus des WWW-Servers vor unerlaubten Zugriffen geschützt werden. Gelingt es externen Benutzern diese Schutzmechanismen zu umgehen, haben sie ungehinderten Zugriff auf die auf diesem Server abgelegten Unternehmensdaten. Um dieses Risiko abzusichern, bleibt die Möglichkeit interne Unternehmensdaten auf einem separaten WWW-Server abzulegen, wo sie vor unerlaubten Zugriffen sicher sind.

7.6.4 Interner und externer WWW-Server

Wie in Abbildung 7.10 dargestellt, bieten bei dieser Konfiguration sowohl ein interner als auch ein externer WWW-Server seine Dienste an. Der externe Server liegt vor der Firewall und ist sowohl für externe als auch interne Benutzer erreichbar. Er enthält nur allgemeine Daten, die für externe und lokale Benutzer zugänglich sein sollen. Interne Benutzer haben über einen Proxy- bzw. SOCKS-Server indirekten Zugriff auf den externen WWW-Server und die dort zur Verfügung gestellten Informationen. Benutzer aus dem Internet greifen direkt darauf zu. Sollte es einem potentiellen Angreifer gelingen, eine Sicherheitslücke im externen Server zu finden, ist das lokale Netz weiterhin durch den Firewall geschützt. Der interne WWW-Server liegt hinter der Firewall im lokalen Netz. Er ist nur für lokale Benutzer zugänglich und enthält sensible Unternehmensdaten, die vor unerlaubten Zugriffen geschützt sind. Externe Benutzer, die den WWW-Server hinter der Firewall erreichen wollen, werden durch diesen daran gehindert. Die möglichen Angriffspunkte potentieller Angreifer reduzieren sich somit auf den Firewall.

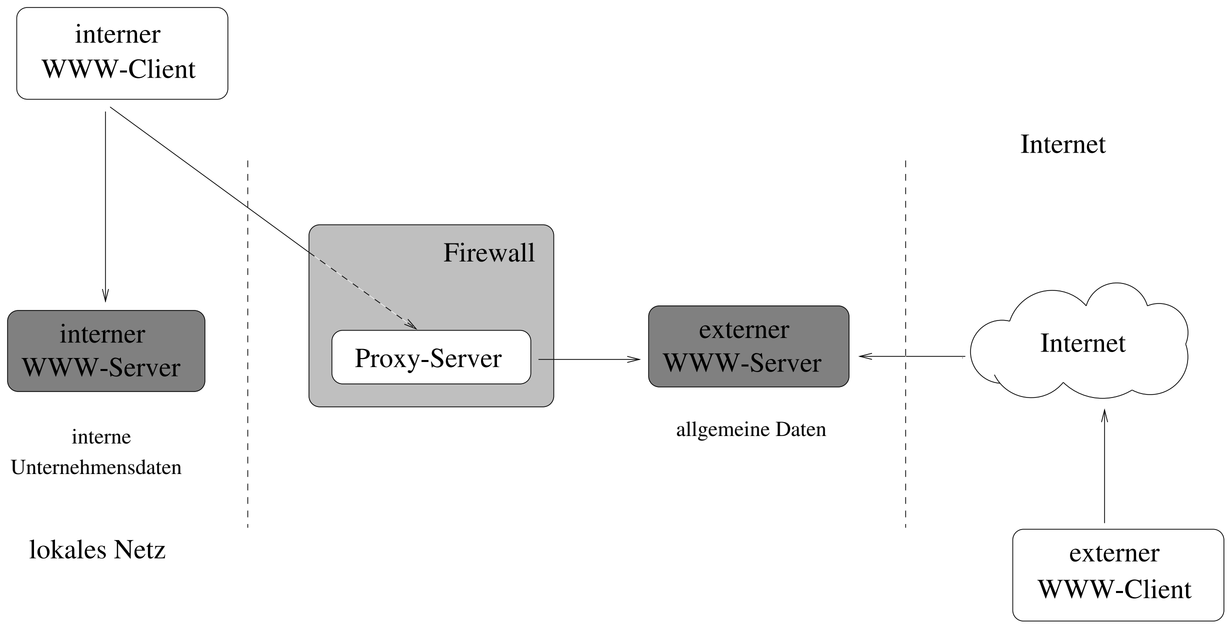

Abbildung 7.10: Interner und externer WWW-Server

7.6.5 Auswirkungen auf das BMW-Umfeld

Basierend auf der Analyse möglicher Standorte für WWW-Server wird der BMW AG empfohlen, einen internen und externen WWW-Server, wie in Kapitel 7.6.4 beschrieben, zu betreiben. Der interne Server sollte nur BMW-interne Unternehmensdaten enthalten, die ausschließlich lokalen Benutzern für den internen Gebrauch zur Verfügung stehen. Der externe WWW-Server sollte allgemeine Daten sowie von der BMW AG angebotene Informationsdienste enthalten, die öffentlich über das Internet zugänglich sein sollen. Für die „Jahreswagenbörse“ bedeutet dies, daß sowohl die WWW-Anwendung JAWA als auch die der JAWA zugrundeliegende Datenbank auf dieser Maschine installiert werden müssen. Der Betrieb des externen WWW-Servers und darauf laufender Anwendungen stellt für das Corporate Network der BMW AG kein Sicherheitsrisiko dar. Gelingt es einem potentiellen Angreifer eine Sicherheitslücke im WWW-Server oder in einer laufenden Anwendung zu entdecken, ist das lokale Netz weiterhin durch den Firewall geschützt. Werden die der JAWA zugrundeliegenden Daten bzw. die Anwendung selbst sabotiert, können sie jederzeit wierderhergestellt bzw. in Betrieb genommen werden. Das Original der Jahreswagendaten liegt zusammen mit den restlichen Gebrauchtwagendaten auf einem Großrechner im lokalen Netz, der vor unerlaubten Zugriffen geschützt ist. Da in der JAWA-Datenbank nur ein Duplikat der Daten gespeichert wird, können diese im Falle einer Zerstörung durch potentielle Angreifer ohne großen Aufwand wieder hergestellt werden. Dazu werden die vom Großrechner in einer Datei extrahierten Jahreswagendaten auf den externen Rechner übertragen und dort wieder in die JAWA-Datenbank eingespielt.

Kapitel 8

Zusammenfassung

Im Rahmen dieser Diplomarbeit wurde die WWW-Anwendung „BWW Exklisivbörse für Jahreswagen“, kurz „JAWA“ genannt, entwickelt und prototypisch implementiert. Grundlage der Entwicklung bildeten die Untersuchung und Bewertung verschiedener Architekturen für WWW-Datenbank-Gateways und deren Betriebsaspekte sowie die Erstellung eines Style-Guides für WWW-Anwendungen. Basierend auf den in dieser Arbeit durchgeführten Untersuchungen wurde dann der Prototyp der JAWA implementiert. Bei dem der JAWA zugrundeliegenden CGI-Skript handelt es sich um ein statisches Standalone-Gateway zwischen einer Oracle-Datenbank und dem WWW. Der Protoyp der JAWA kann als Grundlage für weitere Entwicklungen, z.B für die Entwicklung einer WWW-Anwendung „BMW Exklusivbörse für Gebrauchtwagen“, verwendet werden.

Weiterhin wurden in dieser Diplomarbeit die Auswirkungen der Verwendung einer Firewall auf das WWW untersucht sowie die möglichen Standorte eines WWW-Servers evaluiert und bewertet. Proxy-Server als auch SOCKS-Server können auf der Firewall eingesetzt werden, um WWW-Clients innerhalb der Firewall Zugang zum Internet zu verschaffen, wobei die Verwendung eines SOCKS-Servers die sicherere Lösung darstellt, mit dem Nachteil, daß SOCKS-Server nicht in der Lage sind, bereits angeforderte Dokumente in einem Cache zwischenzuspeichern. Soll ein WWW-Server seine Dienste im Internet anbieten, hat sich das Bereitstellen eines internen WWW-Servers hinter der Firewall sowie eines externen Servers vor der Firewall als die sicherste Lösung für den Informationsanbieter dargestellt. Für die JAWA bedeutet dies, daß sie als WWW-Dienst auf dem externen WWW-Server angeboten wird.

Anhang A

CGI-Umgebungsvariablen

Der WWW-Server vermittelt dem CGI-Skript zahlreiche Informationen über sich selbst und die vom WWW-Client empfangene Anfrage, die in Umgebungsvariablen abgelegt werden. Die Umgebungsvariablen werden beim Aufruf des Skripts durch den Server gesetzt. Bei den Umgebungsvariablen unterscheidet man Variablen, die server-spezifische Informationen enthalten, und Variablen die mit anfragespezifischen Informationen belegt sind.

Folgende Variablen besitzen für einen bestimmten Server stets einen festen Wert (server-spezifische Informationen):

- `SERVER_SOFTWARE`

  Name und Version des WWW-Servers, der die Client-Anfrage bearbeitet und das CGI-Skript startete.
  Format: *name/version*
  Beispiel: `CERN/3.0`

- `SERVER_NAME`

  Name des Rechners, auf dem der WWW-Server läuft. Die Bezeichnung erfolgt in Form eines Host-Namens, eines DNS-Alias-Namens oder einer IP-Adresse.
  Beispiel: `www.informatik.uni-muenchen.de`

- `GATEWAY_INTERFACE`

  Version der CGI-Spezifikation, zu der der Server kompatibel ist.
  Form: `CGI/`*Version*
  Beispiel: `CGI/1.1`

Die Werte der folgenden Variablen hängen jeweils von der Art des Zugriffs durch den Client ab (anfragespezifische Informationen):

- `SERVER_PROTOCOL`

  Name und Version des HTTP-Protkolls, das die Anfrage auslöste.
  Format: `HTTP/`*Version*
  Beispiel: `HTTP/1.0`

- `SERVER_PORT`

  Portnummer des Servers, auf der die Client-Anfrage eingegangen ist.
  Beispiel: `80`

- `REQUEST_METHOD`

  Die Zugriffsmethode, die im Header der Client-Anfrage enthalten ist.
  Beispiel: `GET`

- `PATH_INFO`

  Enthält Extra-Pfadinformationen, die durch den Client angegeben wurden. Möchte ein Client das aufzurufende Programm mit Parametern oder zusätzlicher Information versorgen, kann er dies tun, in dem er diese, durch einen Schrägstrich getrennt, an die URL des CGI-Skripts hängt. Diese Extra-Information, die auch als „Path-Information" bezeichnet wird, wird in der Variablen `PATH_INFO` abgelegt.

  Beispiel: Wird die Path-Information „`Weitere+Informationen`" an die URL des Skripts gehängt (`http://Servername/CGI/Test-Skript/Weitere+Informationen`), enthält die Variable `PATH_INFO` den Wert „`Weitere+Informationen`".

- `PATH_TRANSLATED`

  Der Server legt den vollständigen Pfad des aufgerufenen CGI-Skripts in dieser Variablen ab. Er verwendet dazu den Wert von `PATH_INFO` und bildet diesen virtuellen Namen gemäß den Abbildungsregeln, die in seiner Konfigurationsdatei aufgeführt sind, auf den vollen physischen Pfad ab.

  Beispiel: Sei in der Konfigurationsdatei des Servers eine Abbildungsregel von „`/*`" auf „`/usr/local/pub/WWW/*`" enthalten, dann sind die Variablen wie folgt belegt:

  ```
  PATH_INFO=/CGI/Test-Skript
  PATH_TRANSLATED=/usr/local/pub/WWW/CGI/Test-Skript
  ```

- `QUERY_STRING`

  Diese Variable ist in folgenden Fällen gesetzt:

  (a) Der Aufruf des CGI-Skripts erfolgt von einem Dokument aus, das dem Benutzer erlaubt, einen Suchbegriff einzugeben (`ISINDEX`-Element in HTML).

  (b) Der Zugriff auf das externe Programm erfolgt von einem „Inline Clickable Image" aus (`ISMAP`-Element in HTML+).

(c) Das CGI-Skript bearbeitet ein HTML-Formular, das der WWW-Client mit der Zugriffsmethode `GET` an den Server geschickt hat.

In diesen Fällen hängt der Client ein Fragezeichen, gefolgt von den Parametern der jeweiligen Aktion, an die URL an. Die Parameter sind dabei der vom Benutzer eingegebene Suchbegriff bei `ISINDEX`-Dokumenten, die Mauskordinaten bei anklickbaren Bildern oder die vom Benutzer gemachten Eingaben bei HTML-Formularen.

- `REMOTE_HOST`

  Der Name der Maschine, von dem die Client-Anfrage kam. Ist der Client für die Verwendung eines Proxy-Servers konfiguriert (Kapitel 7.2.3), erscheint hier der Name des Proxy-Servers.
  Beispeil: `wwwcache.leo.org`

- `REMOTE_ADDR`

  Die IP-Adresse des Rechners, von dem die Client-Anfrage stammt. Hat der Client einen Proxy-Server spezifiziert (s. Kap. 7.2), erscheint hier die IP-Adresse des Proxy-Servers.
  Beispiel: `129.187.214.1`

- `AUTH_TYPE`

  Unterstützt der Server Benutzer-Authentisierung oder handelt es sich bei dem aufgerufenen Skript um ein geschütztes Dokument, enthält diese Variable die zu verwendende Authentisierungsmethode.
  Bespiel: `Basic`

- `REMOTE_USER`

  Handelt es sich bei dem aufgerufenen Skript um ein geschütztes Dokument, enthält diese Variable den Namen des Benutzers, der auf das CGI-Skript zugegriffen hat. Der Benutzername muß dabei nicht identisch sein mit der Benutzerkennung des Anwenders, die er auf der Maschine besitzt.

- `REMOTE_IDENT`

  Läuft auf dem Client-Rechner ein Authentisierungs-Server gemäs RFC 931, kann der WWW-Server dem CGI-Skript die Bentuzerkennung des Anwenders in dieser Variablen mitteilen. Aber aus Gründen der Glaubwürdigkeit solcher Authentisierungs-Server, sollte das CGI-Skript diese Variable ausschließlich für Logging-Zwecke benutzen.

- `CONTENT_TYPE`

  Diese Variable enthält bei Client-Anfragen, bei denen die HTTP-Methoden `POST` oder

PUT verwendet werden, Angaben über den MIME-Typ der Daten. Bei der HTTP-Methode GET ist diese Variable nicht belegt.
Beispiel: `application/x-www-form-urlencoded`

- CONTENT_LENGTH

  Wurde bei der Client-Anfrage die HTTP-Methode POST oder PUT verwendet, enthält diese Variable die Länge der Daten in Bytes, die auf der Standardeingabe verfügbar sind. Bei der HTTP-Methode GET ist diese Variable nicht belegt.

Zusätzlich zu den oben aufgeführten Umgebungsvariablen teilt ein WWW-Server, der zu CGI/1.1 kompatibel ist, dem Skript die Header-Information der Client-Anfrage über weitere Umgebungsvariablen mit. Dazu wird der Header- Information das Prefix „HTTP_“ vorangestellt und alle Kleinbuchstaben in Großbuchstaben umgewandelt. Zusätzlich werden alle „-“ durch „_“ ersetzt.
Beispiel: Die Header-Information `Accept:` wird in der Umgebungsvariablen HTTP_ACCEPT abgelegt.

Anhang B

Konfigurationsdatei für das JAWA-Eingabeformular

```
#  Datei: jawaform.conf
#
#  Konfigurationsdatei fuer das Eingabeformular der Jahreswagenboerse
#
#  Beim Aufruf der JAWA ohne Parameter wird diese Datei ausgewertet und
#  das entsprechende Eingabeformular daraus generiert.
#
#  Die Reihefolge der Eintraege ist festgelegt und lautet:
#       Modellvarianten,
#       Antiebsvarianten,
#       Karosserievarianten,
#       Grundfarben,
#       Sonderausstattungen
#
#  Zeilen die mit '#' beginnen, werden als Kommentare gewertet und
#  ignoriert
#
#
#  Modellvarianten
Modell: 316, 318, 320, 325, 328, 518, 520, 525, 730, M 3
#
#  Antriebsvarianten
Antrieb: i, td, tds
#
#  Karosserievarianten
```

```
Karosserie: Limousine, Coupe, Cabrio, compact, touring
#
#  Grundfarben
Grundfarben: beige, blau, gelb, grau, rot, schwarz, silber, violett,
weiss, gr&uuml;n
#
#  Sonderausstattungen
Sonderausttatung: Anh&auml;ngerkupplung, Automatik, Beifahrerairbag,
Klimaanlage, Leichtmetallfelgen, Lederausstattung, Stahlschiebedach
```

Anhang C

Probleme bei der Implementierung der JAWA

C.1 Expires-Header

Da es sich bei den von der JAWA generierten HTML-Seiten um dynamische WWW-Dokumente handelt, sollten diese sowohl vom WWW-Client als auch von eventuell eingesetzten Proxy-Servern nicht zwischengespeichert werden. Um dies zu realisieren, muß das der JAWA zugrundeliegende CGI-Skript einen gültigen Expires-Header generieren, der angibt, daß die Gültigkeit der dynamischen HTML-Seiten sofort nach deren Generierung abläuft.

Nachdem die Generierung des HTTP-Headers implementiert war, trat jedoch folgendes Problem auf: Zur Laufzeit der JAWA werden drei verschiedene HTML-Seiten generiert. Um zwischen den einzelnen WWW-Dokumenten zu navigieren, sind die `BACK`- sowie `FORWARD`-Schalter des jeweiligen WWW-Clients zu verwenden. Der Browser greift dazu auf die im Cache zwischengespeicherten HTML-Dokumente zu und lädt sie bei Bedarf. Da die Generierung des Expires-Headers eine Zwischenspeicherung der dynamisch generierten HTML-Seiten im Cache verhinderte, konnte somit nicht mehr die Funktionalität des WWW-Clients ausgenutzt werden, um zwischen den einzelnen WWW-Seiten der JAWA zu navigieren. Wollte man beispielsweise durch Betätigen des `BACK`-Schalters am WWW-Client von der zweiten Ergebnisseite zur ersten Ergebnisseite zurückkehren, war dies nicht mehr möglich, da die erste Ergebnisseite nicht mehr im Cache des WWW-Clients zwischengespeichert wurde. Der Client zeigte dies durch eine entsprechende Meldung an. Um diese Seite erneut zu laden, mußte der `RELOAD`-Schalter des Browsers betätigt werden. Aufgrund dieses Sachverhalts wurde die Generierung des Expires-Headers wieder ausgesetzt. Die von der JAWA-generierten HTML-Seiten werden somit durch den WWW-Client als auch durch einen beim Zugriff auf die JAWA verwendeten Proxy-Server zwischengespeichert.

C.2 Debugging

Während der Implementierungsphase der JAWA hat sich die Fehlersuche („Debugging") als relativ schwierig erwiesen. Trat zur Laufzeit der Anwendung ein Fehler auf und führte dies zu einem Absturz derselben, wurde dies beispielsweise vom Netscape-Client durch die Meldung `,,Document contains no data''` angezeigt. Man wußte aber zu diesem Zeitpunkt nicht, an welcher Stelle der Fehler aufgetreten ist. Das Einfügen von Debug-Meldung zur Lokalisierung des Fehlers war zwecklos, da diese vom WWW-Client nicht angezeigt wurden. Der Browser zeigte bei jedem Absturz der Anwendung ausschließlich obige Meldung an. Daher mußten die JAWA bzw. Teile der JAWA zuerst auf der Kommandozeile getestet werden, bevor sie in der eigentlichen Umgebung mit einem WWW-Server gestartet wurden. Das Modul `query.pc` wurde als eigenständiges Programm übersetzt, mit Hilfe ausgewählter Beispielanfragen an die JAWA-Datenbank getestet und überprüft, ob es die gewünschte Ausgabe liefert. Ebenso verhielt es sich mit dem Modul `jawa.c`. Es wurde ebenfalls als eingenständiges Programm übersetzt und mit Hilfe eines Shell-Skripts, das die notwendigen CGI-Umgebungsvariablen setzt, getestet. Erst als beide Module auf der Kommandozeile fehlerfrei liefen, konnten sie gemeinsam übersetzt und in der eigentlichen Umgebung mit einem WWW-Server gestartet werden.

C.3 WWW-Client-Software

Die WWW-Anwendung JAWA wurde mit allen in Kapitel 3.2 beschriebenen WWW-Clients getestet. Es traten dabei verschiedene Probleme auf, die im Folgenden beschrieben werden.

C.3.1 Layout der Seiten

Da HTML keine Layout-Informationen enthält, ist das konkrete Layout eines HTML-Dokuments und damit die visuelle Darstellung für den Betrachter den einzelnen WWW-Clients überlassen. Dies bedeutet, das Layout-Informationen, wie beispielsweise das Seitenformat, die Seitengröße, der Abstand zwischen den Absätzen sowie die Zeichensätze für Text und Überschriften je nach WWW-Client unterschiedlich dargestellt werden. Dies wurde vor allem bei der Gestaltung des Eingabeformulars der JAWA deutlich. Konnte das HTML-Formular beispielsweise bei dem Unix-basierten Netscape-Client auf einer Bildschirmseite dargestellt werden, war damit nicht sichergestellt, daß dies auch bei den anderen WWW-Clients der Fall ist. Bereits während der Entwicklung der Bedienoberfläche der JAWA war es nötig, diese auf unterschiedlichen WWW-Clients zu testen. Aufgrund dieser Problematik mußten die Auswahlkriterien im Eingabeformular so gestaucht werden, daß sie auch bei dem Browser mit der „ungünstigsten" Darstellungsweise möglichst auf einer Bildschirmseite dargestellt werden konnten.

C.3.2 Darstellung von Tabellen

Wie bereits in Kapitel 6.5.1 erwähnt, werden für die Darstellung der in der Datenbank gefundenen Fahrzeuge auf der ersten Ergebnisseite HTML-Tabellen (`<TABLE>`) anstelle des HTML-Elements `<PRE>` verwendet. Da es sich bei `<TABLE>` um ein HTML 3.0-Sprachelement handelt, wird es derzeit noch nicht von allen WWW-Clients implementiert. Unter den in Kapitel 3.2 betrachteten WWW-Clients ist der Unix-basierte Client „Lynx“ nicht in der Lage, HTML-Tabellen darzustellen. Die untersuchte Version dieses Browsers ist damit für den Betrieb der JAWA nicht geeignet. Es ist aber zu erwarten, daß zukünftige Versionen dieses Browsers dieses Sprachelement implentieren und daß er damit als WWW-Client für die JAWA eingesetzt werden kann.

C.3.3 Vorformatierte Pulldownmenüs

Aus Gründen des Layouts wurden im Eingabeformular der JAWA vorformatierte Pulldownmenüs implementiert. Es wurde dazu das HTML-Element `<PRE>` verwendet. Bis auf den WWW-Client „NCSA Mosaic for Microsoft Windows“ wurde das HTML-Formular von allen anderen untersuchten WWW-Clients korrekt dargestellt. Da in der HTML-Spezifikation die Verwendung von vorformatierten Pulldownmenüs nicht ausgeschlossen wird und da alle anderen Browser das HTML-Formular korrekt darstellen, ist davon auszugehen, daß es bei der nicht korrekten Darstellungsweise bei „NCSA Mosaic for Microsoft Windows“ um einen Fehler im Anwendungsprogramm handelt. Damit kann dieser Browser als WWW-Client für die JAWA nicht eingesetzt werden.

Anhang D

WWW-Entwicklungsumgebungen

D.1 Existierende Datenbank-Gateways

D.1.1 Oracle World Wide Web Interface Kit

Der Oracle WWW Interface Kit [Ora95], der im Folgenden kurz als „Interface-Kit“ bezeichnet wird, ist ein Softwarepaket, daß von der Firma Oracle zur Verfügung gestellt wird. Der Interface-Kit ist eine Sammlung von Public Domain Datenbank-Gateways für das relationale Datenbanksystem Oracle v7. Einige der Anwendungen wurden von Oracle selbst sowie von anderen Entwicklern zur Verfügung gestellt. Der Interface-Kit ist kein offizielles Produkt der Firma Oracle und ist daher kostenlos im Internet verfügbar. Oracle bietet zwar Unterstützung bei Problemen, doch ist dieser Support sehr begrenzt. Jedes einzelne Tool des Softwarepakets kann als Gateway für Oracle-Datenbanken eingesetzt oder als Grundlage für eigene Entwicklungen verwendet werden. Der Interface-Kit setzt sich aus folgenden Anwendungen zusammen:

- WOW

  WOW ist eine Entwicklungsumgebung, die es erlaubt, eigene CGI-Gateways für WWW-Server zu entwickeln. Es handelt sich um ein in der Programmiersprache *C* entwickeltes Tool. Das Datenbank-Backend ist in der Oracle-spezifischen Programmiersprache *PL/SQL* verfaßt.

- Decoux

  Decoux ist ein Datenbank-Gateway, das sowohl in den Programmiersprachen *perl* und *C* implementiert ist. Für die Implementierung der Datenbank-Backends wurden *oraperl* bzw. die Oracle-spezifische Programmiersprache *Pro*C* verwendet. Das Tool ermöglicht nur lesenden Zugriff auf die Datenbank.

- oraywww

  Oraywww ist ein Gateway, daß in *perl* bzw. *oraperl* implementiert ist. Es ermöglicht die dynamische Generierung von HTML-Formularen, die zu einem späteren Zeitpunkt vom Benutzer verwendet werden können, sowie die dynamische Änderung von bereits erstellten HTML-Formularen. Bei der Erstellung der Formulare wird auf die Systemtabellen des Datenbankssystems zugegriffen, die Informationen darüber enthalten, welcher Benutzer auf die in der Datenbank gespeicherten Tabellen und Views Zugriff hat.

- WORA

  Das Datenbank-Gateway WORA ist ein *C*-Programm, dessen Datenbank-Backend in der Oracle-spezifischen Programmiersprache *Pro*C* implementiert ist. Es ermöglicht lesenden Zugriff auf die in der Datenbank gespeicherten Tabellen und Views. Laut Aussage von Oracle kann der schreibende Datenbankzugriff auf einfache Weise zusätzlich implementiert werden.

- TSS

  Bei TSS (Text Search System) handelt es sich um eine Entwicklungsumgebung, mit der eine freie Textsuche auf beliebigen Datenbasen implementiert werden kann. Als Datenbasis für TSS können sowohl statische Dateien als auch Informationen, die in Oracle-Datenbanken gespeichert sind, verwendet werden. Das TSS-Gateway ist in *C* implementiert und bietet eine Schnittstelle zur Oracle-spezifischen Programmiersprache *PL/SQL*.

Bei den einzelnen Tools handelt es sich um dynamische CGI-Skripten (Kapitel 4.7.2), die als Standalone-Gateways (Kapitel 4.5.1) implementiert sind.

D.1.2 GSQL

GSQL [Ng93] ist ein dynamisches CGI-Datenbank-Gateway für SQL-basierte relationale Datenbanksysteme, das in der Programmiersprache *C* implementiert ist. GSQL ist Public Domain und frei verfügbar. Augrund der modularen Architektur des Gateways kann es unabhängig von dem zugrundeliegenden Datenbanksystem eingesetzt werden. Für das Datenbank-Backend existieren derzeit Implementierungen für die Datenbanksysteme von Sybase und Oracle. Weitere Entwicklungen eigener Backends sind jedem Anwender freigestellt. Das Backend-Programm kann in jeder beliebigen Programmiersprache wie beispielsweise *C* oder *oraperl* entwickelt werden. Für den Zugriff auf die zugrundeliegende Datenbank verwendet das CGI-Skript eine Konfigurationsdatei. Sie enthält die Benutzerkennung, das Paßwort, die Namen der Tabellen und Tabellenspalten, auf die zugegriffen werden soll, die Joins zwischen den einzelnen Tabellen sowie weitere Informationen, die zur dynamischen Generierung der Datenbankanfrage verwendet werden. Die Erstellung einer solchen Konfigurationsdatei ist dokumentiert, wodurch jeder Anwender in der Lage ist, seine eigene Datei zu erstellen. Eine hinzugefügte Beipieldatei soll das Erstellen solcher Konfigurationsdateien erleichtern.

D.1.3 WDB

WDB [Ras95] ist ein Public Domain-Softwarepaket, das ein dynamisches CGI-Datenbank-Gateway für das relationale Datenbanksystem Sybase enthält. Das Gateway ist in *perl* bzw. *sybperl* implementiert. Für den Zugriff auf die zugrundeliegende Datenbank werden eine oder mehrere Konfigurationsdateien verwendet, die bei WDB auch als „form definition files“ bezeichnet werden. Jede einzelne Konfigurationsdatei kann dabei eine unterschiedliche Sicht auf die Datenbank beschreiben. WDB generiert anhand dieser Dateien die für den Datenbankzugriff vom Benutzer verwendeten HTML-Formulare. Mit Hilfe einer Konfigurationsdatei können außerdem Kovertierungen der aus der Datenbank extrahierten Daten spezifiziert werden, die durch die Anwendung zur Laufzeit ausgeführt werden. Das Softwarepaket enthält zusätzlich ein Tool, das Informationen über die Tabellen aus der Datenbank ausliest und daraus automatisch eine Konfigurationsdatei erstellt.

Da der datenbankspezische Code von WDB in einem eigenen Modul isoliert ist, kann das Gateway relativ einfach auf andere Datenbanksysteme, wie beispielsweise Oracle, Informix oder Ingres, portiert werden, die Standard-SQL unterstützen und eine Perl4-Schnittstelle bieten. Derzeit existieren neben Sybase auch Implementierungen für die Datenbanksysteme Informix und mSQL. Die einzelnen Backends sind dabei in *isqlperl* sowie *MsqlPerl* implementiert.

D.2 Software zur Unterstützung der CGI-Skript-Entwicklung

Über die im CGI-Standard spezifizierten Verfahren können CGI-Skripten mit Parametern versorgt werden. Je nach verwendeter HTTP-Methode werden die Parameter entweder auf der Kommandozeile, auf der Standardeingabe oder über CGI-Umgebungsvariablen an das externe Programm übergeben. Die zu übergebenen Parameter sind dabei in URL-spezifischer Syntax kodiert. Möchte ein CGI-Skript die übergebenen Parameter verarbeiten, muß es die Parameter abhängig von der verwendeten HTTP-Methode vom jeweiligen Übergabeort einlesen und dekodieren. Um die Entwicklung solcher CGI-Skripten zu vereinfachen, wurden Tools, sog. „CGI-Bibliothken“, von verschiedenen Personen und Institutionen im Internet entwickelt, die das Einlesen und Dekodieren der an das CGI-Skript übergebenen Parameter übernehmen und damit die CGI-Skript-Entwicklung erleichtern. Die Parameter werden nach der Verarbeitung durch das jeweilige Tools entweder in Datenstrukturen oder Umgebungsvariablen abgelegt, wo sie durch das CGI-Skript einfach weiterverwendet werden können.

Es existieren derzeit CGI-Bibliotheken für die Programmiersprachen *C* und *perl*. Die CGI-Bibliothek von der Firma „Enterprise Integration Technologies“ [Ent] enthält C-Funktionen, die für die Entwicklung von CGI-Skripten in der Programmiersprache *C* verwendet werden können. Das Tool *Un-CGI* [Gri] ist ein *C*-Programm, das die dekodierten Parameter in Umgebungsvariablen bereitstellt. Aus dem Programm heraus können dann weitere CGI-Skripten aufgerufen werden, die in *C*, *perl* oder einer Shell-Kommandosprache implementiert sein können. Bei *cgi-lib.pl* [Bre95] und *CGI.pm* [Ste95] handelt es sich um perl4- und perl5-Bibliotheken, die zur Implementierung von CGI-Skripten in *perl* verwendet werden können.

Daneben werden vom *National Center for Supercomputing Applications (NCSA)* bei seinem WWW-Server *NCSA http* [McC94] zusätzlich drei C-Module mitgeliefert, die die Entwicklung von CGI-Skripten in der Programmiersprache *C* vereinfachen sollen. Die Datei `util.c` enthält Funktionen zum Einlesen und Dekodieren der an das CGI-Skript übergebenen Parameter. In den Dateien `query.c` und `post-query.c` wird die Verwendung dieser Funktionen abhängig von den HTTP-Zugriffsmethoden `GET` und `POST` beispielhaft dargestellt. Die Funktionen aus `util.c` werden auch in der JAWA zur Verarbeitung der an die Anwendung übergebenen Parameter verwendet.

Vom *CERN* werden die beiden Tools `cgiparse` und `cgiutils` zur Verfügung gestellt. Sie werden mit dem frei verfügbaren WWW-Server CERN httpd [LBLF94] mitgelifert. Es handelt sich dabei um ausführbare Programme, die die Entwicklung von CGI-Skripten in einer Shell-Kommandosprache erleichtern. Mit Hilfe des Tools `cgiparse` kann ein Shell-Skript die übergebenen Parameter einlesen, dekodieren und herausfiltern. Das Tool `cgiutils` läßt sich zur einfachen Generierung von HTTP-Headern nutzen. Die Verwendung der beiden Tools wird in [Klu94b] ausführlich beschrieben.

Bei allen hier vorgestellten Tools und Funktionen handelt es sich um Public Domain Software, die im Internet frei verfügbar ist.

Anhang E

WEB-Crawler

Da das WWW eine enorme Anzahl von Ressourcen beherbergt und ihre Zahl ständig steigt, erweist sich das Auffinden gesuchter Informationen im WWW zunehmend schwieriger. Zu diesem Zweck existieren eine Vielzahl von Anwendungen bzw. Suchdiensten, die das Lokalisieren gesuchter Objekte und Informationen im WWW erleichtern sollen. Die den WWW-Suchdiensten zugrundeliegenden Programme werden auch als „Web-Crawler" bezeichnet. Die Web-Crawler sind Suchmaschinen [Jun94b], die zur Suche nach Informationen zu bestimmeten Themengebieten verwendet werden können. Man kann dabei zwei Arten von Suchmaschinen unterscheiden.

Es existieren Suchmaschinen bei denen man ein Dokument oder einen Dienst selbst registrien muß oder der jeweilige Web-Crawler durchsucht ausgehend von einem Dokument das WWW und speichert Informationen zu weiteren Dokumenten, die während der Suche gefunden wurden. Die registrierten Dokumente sowie die Informationen zu gefundenen WWW-Dokumenten werden von dem Web-Crawler in einer lokalen Datenbank gespeichert, wo sie für Suchanfragen zur Verfügung stehen. Der Benutzer einer solchen Suchmaschine kann über eine Benutzerschnittstelle eine Anfrage an die Suchmaschine stellen und erhält als Ergebnis eine Liste aller Informationen, die zu dem von ihm eingegebenen Suchbegriffen in der Datenbank gefunden wurden. Beispiele für derartige Suchmaschine sind:

- Lycos

  Lycos ist ein Web-Crawler der Carnegie Mellon Universität, der zur Zeit ca. 5.6 Millionen WWW-Dokumente registriert hat. Er ist unter `http://lycos.cs.cmu.edu/` erreichbar. *Lycos* durchsucht das WWW regelmäßig nach neuen Dokumenten und nimmt sie in seine Datenbank, den Lycos-Katalog, auf. Weiterhin bietet er eine Schnittstelle, die über die „Home Page" erreichbar ist, über die der Benutzer manuell seine eigenen URLs registrieren lassen kann, ohne daß Lycos sie selbst gefunden und registriert hat. Neue Dokumente werden ein Mal wöchentlich in den Lycos-Katalog aufgenommen.

- WebCrawler

  WebCrawler ist ein WWW-Suchdienst, der von America Online betrieben wird. Er ist unter `http://webcrawler.cs.washington.edu/WebCrawler/Home.html` erreichbar. Er kann zur Lokalisierung von Ressourcen im WWW verwendet werden. Er durchsucht das WWW und generiert aus dem Ergebnis einen Index aus den Dokumenten, die er gefunden hat. Der Benutzer kann diesen Suchdienst verwenden, um nach bestimmen Informationen im WWW zu suchen. Damit eine gesuchte Ressource bei einer Benutzeranfrage gefunden wird, muß sie nicht unbedingt vom *WebCrawler* selbst lokalisiert worden sein. *WebCrawler* bietet zusätzlich eine Schnittstelle, über die der Benutzer seine eigenen URLs registrieren lassen kann. Es dauert ungefähr eine Woche bis aus den neu registrierten Dokumente ein neuer Index aufgebaut wird.

- World Wide Web Worm

  World Wide Web Worm, kurz *WWWW* genannt, ist ein Web-Crawler, bei dem zur Zeit drei Millionen URLs registriert sind. Er ist unter `http://www.cs.colorado.edu/home/mcbryan/WWWW.html` erreichbar. Er durchsucht wie die beiden anderen Web-Crawler das WWW und speichert die zu einem WWW-Dokument gefundenen Informationen, wie beispielsweise URL oder Titel des Dokuments, in der WWWW-Datenbank. Der Benutzer kann in dieser Datenbank nach den gewünschten Informationen suchen. Zusätzlich bietet der *WWWW* einem Benutzer die Möglichkeit seine eigenen URLs direkt registrieren zu lassen, ohne daß sie explizit vom Web-Crawler lokalisiert werden müssen.

- Yahoo

  Bei *Yahoo* handelt es sich um eine virtuelle Bibliothek bzw. ein Online-Adreßbuch für Internet-Dienste. Im Gegensatz zu den anderen Web-Crawlern ist *Yahoo* keine Suchmaschine, die aktiv nach Dokumenten im WWW sucht. Soll ein neuer Dienst oder ein neues Dokument in die *Yahoo*-Datenbank aufgenommen werden, muß der Dienst bzw. das Dokument explizit über ein spezielles Online-Formular registriert werden. *Yahoo* ist unter `http://www.yahoo.com/` erreichbar.

Neben den oben beschriebenen Web-Crawlern existieren auch Suchmaschinen, bei denen man Indexinformationen zu angebotenen WWW-Dokumenten und Diensten zur Verfügung stellen muß, die dann von der jeweiligen Suchmaschine gesammelt werden. Aus diesen Indexinformationen wird dann ein sog. „Master-Index“ aufgebaut, der als Grundlage für Suchanfragen von Benutzern verwendet wird. Im Gegensatz zu den oben beschriebenen Web-Crawlern durchsuchen diese Suchmaschinen das WWW nicht nach neuen Dokumenten. Ein Beispiel für eine derartige Suchmaschine ist *ALIWEB*, der unter `http://web.nexor.co.uk/public/aliweb/aliweb.html` erreichbar ist. Um einen angebotenen Dienst bzw. ein Dokument bei *ALIWEB* registrieren zu lassen, muß eine Index-Datei erstellt werden, die den Dienst bzw. das Dokument beschreibt. Für jeden Dienst und jedes Dokument ist ein eigener Eintrag in dieser Datei zu erstellen. Die Index-Datei kann dabei manuell oder automatisch durch ein entsprechendens Skript erstellt werden und muß einem speziellen Format [NEX94] entsprechen. Die Lokation der Index-Datei wird dann *ALIWEB* mit Hilfe eines Registrierungsformulars mitgeteilt. *ALIWEB* holt daraufhin regelmäßig diese Index-Datei und generiert daraus den sog. „Master Index" aus den entsprechenden Indexinformationen. Die Aufnahme neuer Informationen in die *ALIWEB*-Datenbank erfolgt innerhalb eines Tages.

Soll die Indexdatei automatisch erstellt werden, kann dazu das Skript `site-index.pl` [Jun94a] verwendet werden. Damit dieses Skript die Datei mit den Index-Informationen erstellen kann, müssen die Indexinformationen zu jedem Dokument im Header des jeweiligen Dokuments aufgeführt werden. Es werden dazu die HTML-META-Sprachelemente verwendet. Soll beispielsweise die JAWA als WWW-Informationsdienst in die Indexdatei aufgenommen werden, muß der HTML-Header des JAWA-Eingabeformulars folgende META-Sprachelemente enthalten:

```
<META name="description" content="Jahreswagenb&ouml;rse der BMW AG">
<META name="keywords" content="Jahreswagen Jahreswagenboerse BMW">
<META name="resource-type" content="service"> \\
<META name="Admin-Handle" content="jawamaster@bmw.de">
```

Die verwendeten META-Informationen müssen zwischen den HTML-Elementen <HEAD> und </HEAD> eingefügt werden. Bei der Erstellung dieser META-Informationen ist zu beachten, daß bei dem META-Element `keywords` eventuelle Umlaute nicht in HTML-Schreibweise, sondern als „ae", „oe", usw., angegeben werden.

Um nun sicherzustellen, daß beispielsweise die JAWA als neuer WWW-Dienst von den einzelnen Web-Crawlern gefunden wird, muß die Anwendung bei den jeweiligen WWW-Suchdmaschinen registriert werden. Soll die JAWA außerdem im ALIWEB-Index aufgeführt werden, muß dazu, wie oben beschrieben, die entsprechende Index-Datei erstellt werden.

Literaturverzeichnis

[BF93] N. S. Borenstein and N. Freed. MIME (multipurpose internet mail extensions) part one: Mechanisms for specifying and describing the format of internet message bodies. RFC 1521, Network Working Group, September 1993. ftp://ftp.internic.net/rfc/rfc1521.txt.

[BH95] Michael Björn and Ryosuke Hotaka. A WWW Gateway for Interactive Relational Database Management. In *Proceedings of the First Australian World Wide Web Conference*, Mai 1995. http://www.scu.edu.au/ausWeb95/papers/integrating/bjorn/.

[BL94] Tim Berners-Lee. Uniform Ressource Locators. Internet Draft, IETF URI Working Group, März 1994. http://www.w3.org/hypertext/www/Addressing/URL/url-spec.txt.

[BLC95] Tim Berners-Lee and Daniel Conolly. Hypertext Markup Language (Version 2.0). Internet Draft, HTML Working Group, Mai 1995. ftp://ds.internic.net/internet-drafts/draft-ietf-html-spec-03.txt.

[BLCGP93] Tim Berners-Lee, Robert Cailliau, Jean Francois Groff, and Bernd Pollermann. *World Wide Web: The information universe*. CERN, Genf, 1993. http://www.w3.org/pub/doc/Article_9202.ps.Z.

[BLFN95] T. Berners-Lee, R.T. Fielding, and H.F. Nielsen. Hypertext Transfer Protocol - HTTP/1.0. Internet Draft, HTTP Working Group, März 1995. http://www.ics.uci.edu/pub/ietf/http/draft-ietf-http-v10-spec-00.txt.

[BLMM94] Tim Berners-Lee, L. Masinter, and M. McCahill. Uniform Ressource Locators. RFC 1738, Network Working Group, Dezember 1994. ftp://ds.internic.net/rfc/rfc1738.txt.

[BMGW] Garreth Blythe, Lou Montulli, Michael Grobe, and Stephen Ware. *Lynx Users Guide Version 2.3*. Academic Computing Services, University of Kansas. Online-Dokumentation, http://www.cc.ukansas.edu/lynx_help/Lynx_users_guide.html.

[BMW92] BMW AG. *Exklusivbörse für gebrauchte Automobile: Benutzerhandbuch für Systembetreuer (deutsche Ausgabe)*. Abteilung FI-53, München, November 1992. Dokumentnummer I.

[Bou95] Thomas Boutell. *World Wide Web Frequently Asked Questions*, 1995. Online-Dokumentation, `http://sunsite.unc.edu/boutell/faq/www_faq.html`.

[Bre95] Steven E. Brenner. *CGI Form Handling in Perl*, Juni 1995. Online-Dokumentation, `http://www.bio.cam.ac.uk/web/form.html`.

[CB94] William R. Cheswick and Steven M. Bellovin. *Firewalls and Internet Security*. Addison Wesley, Reading Massachusetts, 1994. ISBN 0-201-63357-4.

[Cla94] Alexander Clausnitzer. *Realisierung einer WWW- und einer ASCII-Version der OMNIS-Recherche*. Diplomarbeit am Institut für Informatik der Technischen Universität München, November 1994.

[CVW95] Alexander Clausnitzer, Pavel Vogel, and Stephan Wiesener. A WWW Interface to the OMNIS/Myriad Literature Retrieval Engine. In *Proceedings of the Third International WWW Conference*, April 1995. `http://www.igd.fhg.de/www/www95/proceedings/papers/65/omnis-www95/omnis-www.html`.

[DE94] R. Doelz and T. Etzold. The Use of WWW in Biological Research. In *Proceedings of the First International WWW Conference*, 1994. `http://www.cern.ch/PapersWWW94/doelz.ps`.

[DR94] John December and Neil Randall. *The World Wide Web Unleashed*. SAMS Publishing, Indianapolis, 1994. ISBN 0-672-30671-4.

[EMD94] David Eichmann, Terry McGregor, and Dan Danley. Integrating Structured Databases into the Web: The MORE System. In *Proceedings of the Second International WWW Conference*, 1994. `http://rbse.jsc.nasa.gov/eichmann/WWW94/MORE.ps`.

[Ent] Enterprise Integration Technologies. *CGI-Library*. Online-Dokumentation, `http://wsk.eit.com/wsk/dist/doc/libcgi/libcgi.html`.

[Gri] Steven Grimm. *Un-CGI Version 1.5*. Online-Dokumentation, `http://www.hyperion.com/ koreth/uncgi.html`.

[Hau95] Rainer Hauck. *Sicherheitskonzept für den BMW-Internet-Zugang*. Diplomarbeit am Institut für Informatik der Technischen Universität München, August 1995.

[HD95] Paul E. Hoffman and Ron Daniel. Generic URN Syntax. Internet Draft, IETF URI Working Group, April 1995. `ftp://ftp.internic.net/internet-drafts/draft-ietf-uri-urn-syntax-00.txt`.

[HH89] Jack L. Hursch and Carolyn J. Hursch. *Working with Oracle Version 6.0*. Windcrest Books, 1989. ISBN 0-8306-3246-8.

[Hud] R. L. Hudson. *UMass Information Navigator*. Office of Information Technology, University of Massachusetts. Online-Dokumentation, `http://info.oit.umass.edu/navigator.html`.

[Jun94a] Achim Jung. *Einrichten einer ALIWEB Datenbank über WWW-Server*. Fakultät für Informatik, Technische Universität München, Juni 1994. `http://www.leo.org/admin/www_ind/aliweb.html`.

[Jun94b] Achim Jung. *Suchmaschinen*. LEO, Technische Universität München, Oktober 1994. `http://www.leo.org/infosys/meta-ind/search-eng/`.

[KK94] Davis Koblas and Michelle Koblas. *SOCKS*, 1994. `ftp://ftp.nec.com/pub/security/socks.cstc/socks_usenix_paper.ps.gz`.

[Klu94a] Rainer Klute. Gegen Sprachbarrieren - HTML und HTTP: Hypertext Markup Language und Transfer Protokoll. *iX Magazin*, (3), 1994. Verlag Heinz Heise GmbH & Co. KG.

[Klu94b] Rainer Klute. Generischer Generator - Dynamische Dokumente mit dem CERN-WWW-Server. *iX Magazin*, (9), 1994. Verlag Heinz Heise GmbH & Co. KG.

[Klu94c] Rainer Klute. Handgestrickt - Mosaic im World Wide Web: Installation und Konfiguration. *iX Magazin*, (2), 1994. Verlag Heinz Heise GmbH & Co. KG.

[Klu94d] Rainer Klute. Zusammengewebt - Internet-Informationsdienste mit Mosaic. *iX Magazin*, (2), 1994. Verlag Heinz Heise GmbH & Co. KG.

[Klu94e] Rainer Klute. Zweiter Gang - Dynamische Dokumente mit dem CERN-WWW-Server. *iX Magazin*, (8), 1994. Verlag Heinz Heise GmbH & Co. KG.

[Klu95a] Rainer Klute. *Das WWW-Kompendium*. Addison Wesley (Deutschland) GmbH, Bonn, 1995.

[Klu95b] Rainer Klute. Zwischenstation - Mit dem Proxy-Server Zeit und Geld sparen. *iX Magazin*, (2), 1995. Verlag Heinz Heise GmbH & Co. KG.

[LA94] Ari Luotonen and Kevin Altis. *World Wide Web Proxies*, April 1994. `http://www.w3.org/hypertext/WWW/Proxies/Proxies.ps`.

[LBLF94] Ari Luotonen, Tim Berners-Lee, and Henrik Frystyk. *CERN httpd*. CERN, Genf, September 1994. Online-Dokumentation, `http://www.w3.org/hypertext/WWW/Daemon/Status.html`.

[Let94] Stanley Letovsky. *Web/Genera - A Web-to-Sybase Gateway*, 1994. Online-Dokumentation, `http://gdbdoc.gdb.org/letovsky/genera/genera.html`.

[LPBN94] Cricket Liu, Jerry Peek, Jones Bryan, and Adrian Nye. *Managing Internet Informations Services*. O'Reilly & Associates Inc., Sebastopol, Californien, 1994. ISBN 1-56592-051-1.

[LS95] Daniel LaLiberte and Michael Shapiro. The Path URN Specifikation. Internet Draft, IETF URI Working Group, July 1995. `ftp://ftp.internic.net/internet-drafts/draft-ietf-uri-urn-path-01.txt`.

[Lyn95] Patrick Lynch. *Yale C/AIM WWW Style Manual*. Yale Center for Advanced Instructional Media, Mai 1995. Online-Dokumentation, `http://info.med.yale.edu/caim/StyleManual_Top.html`.

[McC93] Rob McCool. *The Common Gateway Interface*. National Center for Supercomputing Applications, University of Illinois at Urbana-Champaign, 1993. Online-Dokumentation, `http://hoohoo.ncsa.uiuc.edu/cgi/overview.html`.

[McC94] Rob McCool. *NCSA httpd Overview*. National Center for Supercomputing Applications, University of Illinois at Urbana-Champaign, April 1994. Online-Dokumentation, `http://hoohoo.ncsa.uiuc.edu/docs/Overview.html`.

[Moo93] K. Moore. MIME (multipurpose internet mail extensions) part two: Message header extensions for non-ascii text. RFC 1522, Network Working Group, September 1993. `ftp://ftp.internic.net/rfc/rfc1522.txt`.

[Nat95a] National Center for Supercomputing Applications. *NCSA Mosaic for Microsoft Windows*. University of Illinois at Urbana-Champaign, 1995. Online-Dokumentation, `http://www.ncsa.uiuc.edu/SDG/Software/WinMosaic/HomePage.html`.

[Nat95b] National Center for Supercomputing Applications. *NCSA Mosaic for the X Windows System*. University of Illinois at Urbana-Champaign, 1995. Online-Dokumentation, `http://www.ncsa.uiuc.edu/SDG/Software/XMosaic`.

[Net95a] Netscape Communications Inc. *Netscape Commerce Server*, 1995. Online-Dokumentation, `http://home.mcom.com/MCOM/products_docs/server.html`.

[Net95b] Netscape Communications Inc. *Netscape Navigator*, 1995. Online-Dokumentation, `http://home.mcom.com/MCOM/products_docs/client.html`.

[Net95c] Netscape Communications Inc. *The Netscape Server API*, 1995. `http://home.mcom.com/newsref/std/server_api.html`.

[NEX94] NEXOR. *ALIWEB Format*, 1994. `http://www.nexor.co.uk/public/aliweb/doc/format.html`.

[Ng93] Jason Ng. *GSQL - A Mosaic-SQL Gateway.* National Center for Supercomputing Applications, University of Illinois at Urbana-Champaign, Dezember 1993. Online-Dokumentation, `http://www.ncsa.uiuc.edu/SDG/jason/pub/gsql/starthere.html`.

[Obe95] Karl Obermeier. Aufgeblasen - Grafikflut im Internet. *iX Magazin*, (6), 1995. Verlag Heinz Heise GmbH & Co. KG.

[Ocr94] Constantin F. Ocrainets. *WORA: WWW-Oracle Gateway*, 1994. Online-Dokumentation, `http://weirdb.jinr.dubna.su/wora/`.

[OP95] Rocquefort O'Leary and Anthony Pollard. OraForm: Generically FORM'ing a multi-table query for the Oracle RDBMS. In *Proceedings of the First Australian World Wide Web Conference*, Mai 1995. `http://www.scu.edu.au/ausWeb95/papers/tools/oleary/`.

[Ora92a] Oracle Corporation. *Oracle7 Server Concepts Manual.* Redwood City, Californien, Dezember 1992. Online-Dokumentation.

[Ora92b] Oracle Corporation. *Programmer's Guide to the ORACLE Precompilers Release 1.5.* Redwood City, Californien, Dezember 1992. Online-Dokumentation.

[Ora93] Oracle Corporation. *Oracle Book version 1.0*, 1993. Online-Dokumentation für das relationale Datenbanksystem Oracle.

[Ora95] Oracle Corporation. *Oracle World Wide Web Interface Kit*, 1995. `http://dozer.us.oracle.com:8080/`.

[PF94] Mike Pingleton and Tom Fischer. Utilizing Mosaic and the WWW in an Operations Environment. In *Proceedings of the Second International WWW Conference*, 1994. `http://www.ncsa.uiuc.edu/SDG/IT94/Proceedings/CorInfSys/fischer/fischer.html`.

[PR94] James Pitkow and Mimi Recker. *Results from the Second WWW User Survey.* GVU Center, College of Computing, Georgia Institute of Technologie, Dezember 1994. `http://www.gatech.edu/gvu/user_surveys/User_Survey_Home.html`.

[RA94] Marcus J. Ranum and Frederick M. Avolio. *A Toolkit and Methods for Internet Firewalls.* Trusted Information Systems Inc., 1994. `ftp://ftp.tis.com/pub/firewalls/usenix_paper.ps`.

[Rag95] Dave Ragget. Hypertext Markup Language Specification Version 3.0. Internet Draft, HTML Working Group, März 1995. `ftp://ds.internic.net/internet-drafts/draft-ietf-html-specv3-00.txt`.

[Ran93] Marcus J. Ranum. *Thinking about Firewalls.* Trusted Information Systems Inc., April 1993. `ftp://ftp.tis.com/pub/firewalls/firewalls.ps`.

[Ras95] Bo Freese Rasmussen. *WDB - A Web Interface to Sybase*. European Southern Observatory (ESO), 1995. Online-Dokumentation, `http://arch-http.hq.eso.org/bfrasmus/wdb/wdb.html`.

[Ric95a] Alan Richmond. *A Basic HTML Style Guide*. High Energy Astrophysics Science Archive Research Center (HEASARC) at NASA Goddard Space Flight Center, 1995. `http://guinan.gsfc.nasa.gov/Style.html`.

[Ric95b] Alan Richmond. *CyberWeb's Top Tips for Web Authoring*. CyberWeb Software, 1995. `http://WWW.Stars.com/Tutorial/Style/`.

[SBGK94] Martin Scheller, Klaus-Peter Boden, Andreas Geenen, and Joachim Kampermann. *Internet: Werkzeuge und Dienste*. Springer Verlag, Berlin Heidelberg, 1994. ISBN 3-540-57968-0.

[Sch94] H. Schäfer. *Konzeptüberarbetiung - BMW Exklusivbörse (Btxi) für Gebrauchtwagen: Erweiterung für Dienst- und Jahreswagen*. BMW AG (Abteilung FI-20), München, July 1994.

[Sec94] Arthur Secret. *Oracle Server*. CERN, Genf, 1994. `http://www.w3.org/hypertext/WWW/RDBGate/Implementation.html`.

[Sjo94] Martin Sjolin. A WWW Front End to an OODBMS. In *Proceedings of the Second International WWW Conference*, 1994. `http://www.ncsa.uiuc.edu/SDG/IT94/Proceedings/Databases/sjolin/sjolin.html`.

[SM94] K. Sollins and L. Masinter. Functional Requirements for Uniform Resource Names. RFC 1737, Network Working Group, Dezember 1994. `ftp://ftp.internic.net/rfc/rfc1737.txt`.

[SMTW95] Keith E. Shafer, Eric J. Miller, Vincent M. Tkac, and Stuart L. Weibel. URN Services. Internet Draft, IETF URI Working Group, July 1995. `ftp://ftp.internic.net/internet-drafts/draft-ietf-uri-urn-resolution-01.txt`.

[Spe94] Simon Spero. *Analysis of HTTP Performance Problems*. UNC Sunsite / EIT, 1994. `http://sunsite.unc.edu/mdma-release/http-prob.html`.

[Spe95a] Simon Spero. *HTTP-NG Architectural Overview*. UNC Sunsite / EIT, 1995. `http://www.w3.org/hypertext/WWW/Protocols/HTTP-NG/http-ng-arch.html`.

[Spe95b] Simon Spero. *Next Generation Hypertext Transfer Protocol*. UNC Sunsite / EIT, März 1995. `http://sunsite.unc.edu/ses/ng-notes.txt`.

[Spe95c] Simon Spero. *Progress on HTTP-NG*. UNC Sunsite / EIT, 1995. `http://www.w3.org/hypertext/WWW/Protocols/HTTP-NG/http-ng-status.html`.

[St"93] Günther Stürmer. *Oracle 7, Die verteilte semantische Datenbank*. dbms publishing, 1993. ISBN 3-930124-00-9.

[Ste95] Lincoln D. Stein. *CGI.pm - a Perl5 CGI Library*. Whitehead Institute, MIT Center for Genome Research, Mai 1995. Online-Dokumentation, `http://www-genome.wi.mit.edu/ftp/pub/software/WWW/`.

[Tec94] Technische Universität München. *TIS TU - Informationssystem (Dokumentation)*. Institut für Informatik, September 1994.

[VH94] Carlos A. Varela and Caroline C. Hayes. Zelig: Schema-Based Generation of Soft WWW Database Applications. In *Proceedings of the First International WWW Conference*, 1994. `http://www.cern.ch/PapersWWW94/cvarel.ps`.

[Wei93] Erich Weichselgartner. Unendliche Tiefen - WWW: Hypertext-basiertes Werkzeug im Internet. *iX Magazin*, (12), 1993. Verlag Heinz Heise GmbH & Co. KG.

[Wei94] Erich Weichselgartner. Im Rausch - Vom Wissenschaftsnetz zum elektronischen Einkaufszentrum. *iX Magazin*, (9), 1994. Verlag Heinz Heise GmbH & Co. KG.

[WS91] Larry Wall and Randal L. Schwartz. *Programming perl*. O'Reilly & Associates Inc., Sebastopol, California, 1991. ISBN 0-937175-64-1.