Freie Universität Berlin, Fachbereich Mathematik und Informatik

DIPLOMARBEIT
Entwurf und Entwicklung einer erweiterbaren Softwarearchitektur zur Verbreitung kontextsensitiver Informationen

Betreuer:
Prof. Dr.-Ing. habil. Jochen H. Schiller
Prof. Dr. habil. Agnès Voisard

Autor:
Philipp Ottlinger

philipp@ottlinger.de
2004-06-14

Selbständigkeitserklärung

Ich erkläre hiermit, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

Unterschrift: .....

Berlin, den 2004-06-14

Abstract

This diploma thesis describes the process of developing an open and well structured modular software architecture to distribute context-sensitive information in the area of location-based services.

The software's name is TIP – tourist information provider. TIP works in the area of location-based services. A user – identified by her/his current position and a defined profile of interests – contacts the system with a mobile device, which needs to have network communication abilities and certain ressources for displaying graphical information and navigating through it.

The programming language Java is used to implement a client-server-based system, working in a browser to reach as many different types of mobile devices as possible. The approach of creating an additional peer-to-peer-based application has been abandoned mainly due to Java's mobile edition's (J2ME) poor functional range in the context of complex distributed applications. Instead open standards and open-source frameworks like HTML, CSS and Struts are used to provide a modular client-server implementation, thus leveraging the application's abilities to easily cooperate and integrate with future hardware and software extensions. The main target has been the implementation of an open, well-documented and well-structured solution. TIP consists of a powerful database and application server to deliver information to its clients. This makes TIP a cluster-capable application.

The work's result needs further testing and evaluation under real circumstances, but features a complete multi-layered architecture.

keywords:
Apache Jakarta Tomcat, client-server, distributed information system, Eclipse, HTML, J2ME, Java, JSP, location-based service, P2P, PostgreSQL, Struts

Inhaltsverzeichnis

Abstract **i**

Danksagung **v**

1 Einleitung **1**
1.1 Anwendungsszenario 2
1.2 Aufgabenstellung 3
1.3 Gliederung der Diplomarbeit 4

2 Fachliches Umfeld **5**
2.1 Ortsabhängige Dienste (location-based services) 5
2.2 Verteilte Informationssysteme 9
2.3 Freie Software (open source software) 12
2.4 Zusammenfassung 14

3 Abgrenzung zu existierenden Informationssystemen **15**
3.1 Ziel-Netzarchitektur 15
3.1.1 Client-Server-Architektur 15
3.1.2 Einsatz mobiler Geräte 18
3.2 Verwendung grafischer Oberflächen 19
3.2.1 Native Oberflächengestaltung 19
3.2.2 Browserbasierte Zugänge 20
3.3 Zusammenfassung 25

4 Konzeptioneller Entwurf **27**
4.1 Rahmenbedingungen 27
4.2 Datenbankmodell 33
4.3 Anforderungsprofil 35
4.4 Entwurfsmuster und MVC-Architektur 35

4.4.1 UML-Darstellung 40
4.4.2 Logische Struktur der Anwendung 42
4.5 Zusammenfassung 46

5 Implementierung **47**
5.1 Datenbankserver 48
5.2 Anwendungsserver 50
5.3 Programmiersprache 52
5.3.1 Verwendete Rahmenwerke 53
5.3.2 Eingesetzte Programmierwerkzeuge 63
5.4 Umsetzung 70
5.4.1 Datenbank und geografische Erweiterung 70
5.4.2 MVC-Entwurfsmuster 72
5.4.3 Paketstruktur der Anwendung 79
5.5 Zusammenfassung 82

6 Diskussion **87**
6.1 Probleme bei der Realisierung 87
6.1.1 Technische Probleme 87
6.1.2 Inhaltliche Probleme 93
6.2 Erfahrungsgewinne 95
6.3 Zusammenfassung 96

7 Zusammenfassung und Ausblick **99**
7.1 Systemimmanente Verbesserungen 100
7.2 Test und Vergleich mit anderen Ansätzen 102

Verzeichnisse **105**
Abkürzungsverzeichnis 105
Abbildungsverzeichnis 109
Tabellenverzeichnis 110
Literaturverzeichnis 111
Linkverzeichnis 119

Anhang **121**
.1 Inhalt der beiliegenden CD 121
.2 Grafiken und Visualisierungen des TIP-Systems 123

Danksagung

Ich bedanke mich bei meinen Betreuern für die tatkräftige Unterstützung und die offenen Gedanken meinem Thema gegenüber.

Die Zusammenarbeit mit der FU Berlin war hervorragend, insbesondere die Hilfe der Technik bei der Datenbankumsetzung sowie die vielen Gespräche mit den Doktoranden der Arbeitsgruppe Technische Informatik. Des weiteren bedanke ich mich bei Katja Löffler für die Zeit der Diskussionen und Erklärungen zu Ihrer Diplomarbeit, dem entworfenen Datenmodell und Möglichkeiten, dieses in meine Arbeit einzubauen.

Besonderer Dank gilt meiner Familie und meiner Freundin Heike, die mich stets ermutigt haben, diese Arbeit zu schreiben und bei der Korrektur eine große Hilfe waren.

Ebenso bedanke ich mich für die wertvolle Arbeit aller Open-Source-Programmierer, ohne deren Rahmenwerke und Software ich diese Arbeit nicht hätte schreiben können.

Meinem Kommilitonen Markus Hindorf gilt besonderer Dank für die jahrelange erfolgreiche Zusammenarbeit während des Studiums; die unzähligen gemeinsam durchgestandenen Übungszettel und Projektnächte sowie die klaren Gedanken, wenn es um Probleme der Implementierung ging oder um Entwicklerphilosophie im Allgemeinen.

Philipp Ottlinger

Berlin, den 2004-06-14

Kapitel 1

Einleitung

Diese Diplomarbeit beschreibt das Vorgehen beim Entwurf und der Entwicklung eines Softwaresystems zur Verbreitung kontextsensitiver Informationen. Hintergrund der Arbeit ist die Idee eines Touristeninformationssystems – tourist information provider (TIP), das an der Freien Universität Berlin entwickelt wurde [1]. In einer anderen Diplomarbeit an der FU Berlin [33] wurde ein Datenmodell vorgestellt, das als Grundlage für die hier entwickelte Software benutzt und angepasst wurde.

Hauptziel der Entwicklung war der Aufbau eines gut dokumentierten modularen Systems, damit eventuell durchzuführende Änderungen nicht zu einem kompletten Neuentwurf führen müssen. Mögliche Änderungen sind sowohl technischer als auch inhaltlicher Natur. Technische Änderungen resultieren aus der Einführung neuer Geräte und veränderter Netzzugänge. Bei mobilen Geräten hat die Verbreitung von Funknetzadaptern neben bestehender Funktelefonanbindung als Netzzugang stark zugenommen. Derzeit handelt es sich um kabellose Netzwerkkarten (WLAN) und Bluetooth. WLAN-Zugänge stellen eine kostengünstige breitbandige Art des Internetzugangs dar, während der in der Mobiltelefonwelt verbreitete GPRS-Standard (General Packet Radio Service) einen schmalbandigen teureren Dienstzugang bietet.

Ein anderer Aspekt bei der Softwareentwicklung war die Benutzung von offenen Standards, Rahmenwerken und freier Software. Das entstandene Programmpaket stellt ein Informationssystem für Touristen dar, das sich mit Hilfe eines Internetbrowsers bedienen lässt. Die Verwendung offener Standards sichert dabei die einfache Anpassung sowohl auf Server- als auch auf Klientenseite.

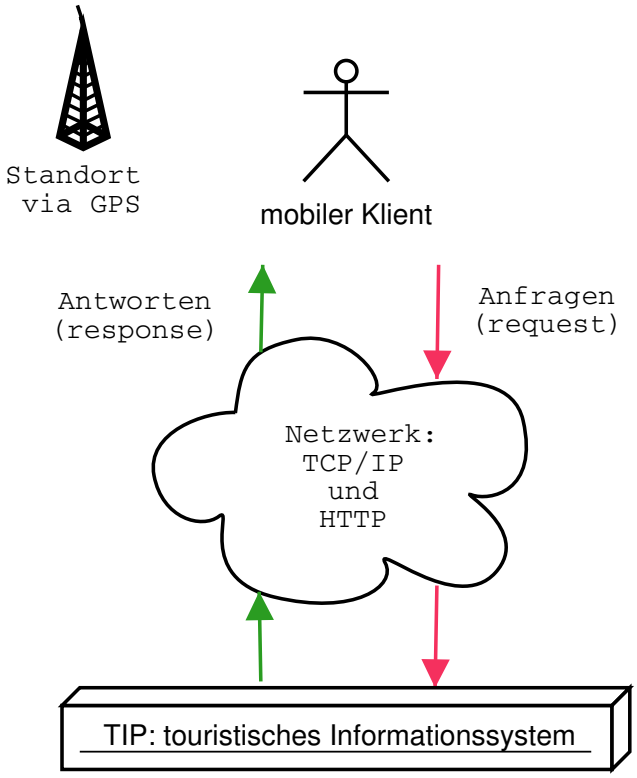

Abbildung 1.1: TIP-Anwendungsszenario

Bereits am Markt existierende Systeme beschränken sich häufig auf wenige kaum veränderbare Geschäftsvorfälle und stellen somit eher statische als dynamische Systeme dar. Ein wirtschaftlicher Hintergrund dafür ist die Suche nach marktfähigen Anwendungen für die neue Mobilfunkgeneration Universal Mobile Telecommunications System (UMTS) [22]. Aus diesem Grund hat man sich vorwiegend darauf beschränkt, Funktionsprototypen mit einfacher Bedienung zu entwickeln und diese schnell in den Massenmarkt einzuführen.

1.1 Anwendungsszenario

Abbildung 1.1 zeigt den schematischen Aufbau des TIP-Systems, dessen Komponenten nach dem Client-Server-Paradigma entwickelt sind; ein leistungsstarker Server versorgt einen ressourcenschwachen mobilen Klienten. Dieser kontaktiert das Informationssystem über ein beliebiges Netz, das über verbindungsorientierte gesicherte Protokolle wie TCP/IP Verbindungen

aufbauen kann. Diese Verbindungen werden mit Hilfe von Hypertext Transfer Protocol (HTTP) zum Datenaustausch zwischen Dienstnehmer (Client) und Dienstanbieter (Server) benutzt.

Zusätzlich zu Anmeldeinformationen, mit denen das TIP-System dem Benutzer ein Profil zuweisen kann, sendet der Klient seine aktuelle Position. Diese Eingaben werden benutzt, um dem Benutzer einen aktuellen Anwendungskontext zuordnen zu können, der sich aus seiner aktuellen Position und dem gespeicherten Interessenprofil ergibt. Dieses Profil kann beispielsweise beinhalten, dass der Anwender an Denkmälern interessiert ist. Daraus folgt, dass das TIP-System beim Vorhandensein eines Turmes in der näheren Umgebung auf diesen hinweist. Entfernt sich der Benutzer von seiner gemeldeten Position, kann er vom TIP-Server aktuelle Standortinformationen einholen, die an sein Profil angepasst ausgeliefert werden. Somit kann beispielsweise ein Tourist an jedem beliebigen Ort in einem unbekannten Stadtteil seine Besichtigungstour starten. Die Daten des Stadtteils müssen im TIP-System erfasst sein und können dann nach Einstellung eines Benutzerprofils durch den Touristen abgefragt werden. Diese TIP-Grobübersicht wird im Lauf der nächsten Kapitel zu einer verfeinerten Übersicht (Abbildung 4.8, S. 45) erweitert, die Aufschluss über das zu Grunde liegende Softwaresystem gibt.

1.2 Aufgabenstellung

Aufgabe der Diplomarbeit ist die Erstellung eines modularen Softwaresystems unter Verwendung offener Standards. Der anfangs verfolgte Weg, zwei Anwendungteile für verschiedene Netztopologien zu entwickeln, wurde verworfen, da die direkte Benutzer-Benutzer-Kommunikation (P2P) derzeit auf Grund von technischen Schwierigkeiten nicht für mobile Geräte umsetzbar ist. Stattdessen wurde eine Client-Server-Infrastruktur geschaffen, die es ermöglicht, ressourcenschwache mobile Geräte in das Informationssystem einzubinden. Dazu wird ein browserbasiertes Anwendungssystem benutzt, das durch seine Modularität genügend Schnittstellen bietet, um zukünftige Hardware- und Softwareerweiterungen zu integrieren. Im Vergleich zu existierenden Informationssystemen bietet die hier entwickelte TIP-Anwendung die Möglichkeit, neue Funktionalität einfach zu integrieren. Dies wird durch eine gut dokumentierte und strukturierte Anwendungsarchitektur begünstigt.

Während des Anwendungsentwurfs und der eigentlichen Implementierung konnten viele positive und negative Erfahrungen mit den entsprechenden Ser-

verkomponenten und Softwareentwurfsentscheidungen gesammelt werden. Im Gegensatz zu naiven Entwurfsansätzen, die sich in kurzer Zeit implementieren lassen, kostete der Entwurf hier viel Mühe, wird aber dadurch langfristig einfacher zu warten und anzupassen sein. Im Rahmen der Diplomarbeit wurden verschiedene verfügbare Rahmenwerke miteinander verglichen und eine optimale Auswahl zur Implementierung des TIP-Systems eingesetzt.

1.3 Gliederung der Diplomarbeit

Die Diplomarbeit gibt im folgenden Kapitel 2 einen fachlichen Einstieg in die Thematik verteilter ortsabhängiger Systeme. Dabei werden verschiedene für Informationssysteme relevante Techniken zur Positionsbestimmung vorgestellt. Das darauf folgende Kapitel 3 stellt andere Informationssysteme vor und beschreibt die den jeweiligen Implementierungen zu Grunde liegende Anzeigetechnik und Kommunikationsstruktur. Alle Systeme sind nach dem Client-Server-Ansatz entworfen. Die Umsetzung anderer Topologieparadigma – wie Peer-To-Peer-Kommunikation (P2P) – erscheint derzeit als schwierig realisierbar und ist Kernelement aktueller Forschungen. Die hierbei auftretenden Probleme lassen sich technisch auf die eingeschränkten Fähigkeiten mobiler Geräte zur Kommunikation in Netzen und deren reduzierte Möglichkeiten zur Anzeige grafischer Oberflächen und Navigation zurückführen. Daraus werden im Kapitel 4 Rahmenbedingungen zum Entwurf des TIP-Systems abgeleitet. Diese betreffen die verwendeten Serverkomponenten sowie Elemente der Softwarearchitektur. Dabei wird auf Entwurfsmuster der objektorientierten Programmentwicklung zurückgegriffen, die im Kapitel 5 mit der eigentlichen Implementierung in Verbindung gebracht werden. Außerdem werden die benutzten Softwarewerkzeuge zur Programmentwicklung und Konfiguration vorgestellt, die erst eine strukturierte Softwareentwicklung ermöglichen und somit maßgeblich zum Entstehen des TIP-Systems beigetragen haben. Die bei der Umsetzung gemachten Erfahrungen sowohl technischer als auch inhaltlicher Natur werden im Kapitel 6 diskutiert und führen im Kapitel 7 zu einer Zusammenfassung und einem Ausblick, was mit dem geschaffenen System zukünftig gemacht werden kann. Im Anhang der Arbeit (ab S. 121) befinden sich ein Inhaltsverzeichnis der beigelegten CD sowie Bildschirmfotos der Anwendung im Betrieb und der eingesetzten Visualisierungswerkzeuge, auf die im laufenden Text verwiesen wird.

Kapitel 2

Fachliches Umfeld

In diesem Kapitel wird das fachliche Umfeld der Arbeit erläutert. Dazu zählen Entscheidungen zur Modellierung der auftretenden Daten sowie daraus resultierende Annahmen über die spätere Leistungsfähigkeit und Ausbaufähigkeit des Systems. Die am Ende gezogenen Konsequenzen haben Einfluss auf die Umsetzung des Systems, worauf in Kapitel 4 und 5 näher eingegangen wird, nachdem im Kapitel 3 bestehende Systeme eingeführt wurden.

2.1 Ortsabhängige Dienste (location-based services)

In den letzten Jahren hat die Vernetzung unterschiedlicher technischer Geräte und Dienste stark zugenommen. Ortsabhängige Dienste haben daran einen wichtigen Anteil. Technisch werden dazu aktuelle Positionsdaten der Benutzer berücksichtigt, um spezielle an die Situation angepasste Dienste zur Verfügung stellen zu können [24].

Diese Dienste verbreiten kontextsensitive Informationen, wenn ihre Ausgabe flexibel an den aktuellen Anwendungszustand angepasst werden kann. Im Fall eines Touristeninformationssystems besteht der Anwendungskontext aus der aktuellen Position des Benutzers und seinem Interessenprofil. Diese Daten werden vom System bei der Auslieferung von Informationen an den Benutzer berücksichtigt, wodurch das System kontextsensitive Informationen zur Verfügung stellt. Kontextsensitive Anwendungssysteme generieren ihre Ausgaben stets dynamisch. Statische Systeme liefern ihre Informationen nur in Abhängigkeit vom internen Zustand der Anwendung aus. Et-

waige standortbezogene Einschränkungen werden nicht berücksichtigt. Ein kontextsensitives Informationssystem kann beispielsweise die aktuelle Netzanbindung des Benutzers berücksichtigen. Ist dieser über ein schmalbandiges teures Netz – wie bei Mobiltelefonen über GPRS – angebunden, werden nur textuelle Informationen übertragen, während breitbandig angebundenen Klienten Multimediainhalte zugesandt werden.

Im Umfeld des elektronischen (e-commerce) und mobilen Handels (m-commerce) stellen sich verschiedene zusätzliche Herausforderungen an ortsabhängige Informationssysteme, auf die im Folgenden kurz eingegangen wird. Die verwendeten mobilen Geräte zeichnen sich durch eine geringe CPU-Leistung und stark eingeschränkte Eingabehilfen aus. Die Benutzung eines browserbasierten Ansatzes mit einer vereinfachten Navigation durch Links erscheint hier angemessen. Diese Technik ermöglicht ein hierarchisches Bewegen innerhalb der Anwendung [48] und der Benutzer wird pro Hierarchiestufe mit möglichst vielen Auswahlmöglichkeiten konfrontiert. Diese Herangehensweise soll die Gesamtanzahl aller Hierarchieebenen möglichst gering halten und dem Benutzer eine einfache Bedienung ermöglichen.
Die mit dem elektronischen Handel gemachten Erfahrungen, die sich durch die folgenden sieben **C** [5, 61] charakterisieren lassen, können auf ortsabhängige Systeme übertragen werden:

1. **C**ontext – Kontext
2. **C**ontent – Inhalte
3. **C**ommunication – Nachrichtenverbindung
4. **C**ommerce – Handel
5. **C**ommunity – Gemeinschaft und Gemeinsamkeit
6. **C**ustomization – Anpassungsfähigkeit und Flexibilität der Anwendung
7. **C**onnection – Netzanbindung und Kommunikationsfähigkeit

Unter Kontext wird in dieser Arbeit die Position und das Profil des Benutzers verstanden. Dieser Kontext korrespondiert mit den ausgelieferten Inhalten und deren Anpassungsfähigkeit. Das bedeutet, dass sowohl das Profil als auch spezielle Benutzerwünsche zur Darstellung beachtet werden. Das können beispielsweise multimediale Inhalte oder reine Texte sein. Die Auswahl der übertragenen Inhalte hängt von der aktuellen Netzanbindung ab

und wird auch dadurch bestimmt, wieviel der Dienstnehmer bereit ist an den Dienstgeber zu bezahlen. Je schmalbandiger der Netzzugang ist, desto höher sind gewöhnlich die für den Benutzer auftretenden Leitungskosten. Der Dienstnehmer sollte in der Lage sein, zwischen kostenfreien und kostenpflichtigen Inhalten auswählen zu können. Diese Fähigkeit zur Auswahl ermöglicht Dienstanbietern, den kommerziellen Handel in Informationssysteme zu integrieren. Dadurch können mit geeigneten Geschäftsmodellen und kostenpflichtigen Zusatzdiensten Umsätze generiert werden. Durch eine höhere Auswahl von Diensten kann das Informationssystem für eine größere Anzahl von Benutzern attraktiv sein. Nebenher ist es möglich, dass eine Gemeinschaft von Benutzern entsteht, die ähnliche Profile und Interessen haben und soziale mit technischen Komponenten verknüpft.
Außerdem lässt sich das durch Client-Server-Architekturen geprägte Bild des elektronischen Handels (e-commerce) durch Einarbeitung ortsabhängiger Dienste basierend auf Peer-To-Peer-Techniken zum mobilen Handel (m-commerce) erweitern. Dadurch können die Interessen der Benutzer stärker beachtet werden.

Da sich die Klienten und ihre Geräte räumlich bewegen, bedarf es Techniken zur Bestimmung der aktuellen Position des Benutzers. In Abhängigkeit von dieser kann das Informationssystem dem Benutzer relevante Informationen zusenden. Die Positionsbestimmung kann dabei sowohl vom Benutzer selbst als auch unter Zuhilfenahme eines Sensornetzes erfolgen. Besitzt der Benutzer ein Mobiltelefon, so kann dieses zur automatischen Positionsbestimmung genutzt werden. Dabei dient das Telefon als Markierung, die durch die Sendemasten des Mobilfunkanbieters lokalisierbar ist und einer logischen Position innerhalb des Netzes zugeordnet werden kann. Bestimmt der Benutzer seine Position selbst, so kann er je nach eingesetzter Technik Infrarot-, Funk- oder Ultraschallsignale aussenden. Das TIP-System arbeitet derzeit nach dieser Art der Positionsbestimmung, da es keine eigene Netzinfrastruktur besitzt, die den Klient logisch positionieren könnte. Die Anwendungsentwicklung wird dadurch vereinfacht, da als Positionsangabe ein Koordinatenpaar benutzt werden kann. Nach Roth [21] lassen sich Techniken der Lokalisierung wie folgt klassifizieren:

- Satellitennavigationssystem – GPS
- Lokalisierung innerhalb von Gebäuden
- Netzwerkgestützte Positionsbestimmung – GSM/ WLAN

Die Existenz von Satellitennavigationssystemen – wie beispielsweise Global Positioning System (GPS) – ist militärisch motiviert, um bei einem eventuellen Ausfall konventioneller Nachrichten- und Navigationssysteme kommunikationsfähig zu bleiben. Durch die veränderte politische Lage nach dem Ende des Kalten Krieges werden diese Systeme zunehmend zivil genutzt. Sie liefern Navigationsdaten für Flug- und Fahrzeuge und steigern dadurch die Effizienz logistischer Operationen.

Hauptnachteil bei der Ortung mit GPS ist der nötige direkte Sichtkontakt zum korrekten Signalempfang. Zur Verbreitung von GPS-Signalen in Gebäuden oder Orten, wo der direkte Sichtkontakt unmöglich ist, können zusätzliche Zwischenverstärker installiert werden, die eine fast flächendeckende Nutzung ermöglichen. Diese müssen vom Dienstanbieter installiert werden. Zusätzlich zum bestehenden GPS existiert eine Verbesserung aGPS [5], die eine schnellere Lokalisierung ermöglicht, wobei kontinuierlich Positionsdaten mit dem Zentralsystem ausgetauscht werden.

Die Lokalisierung innerhalb von Gebäuden ist für den Einsatz in touristischen Informationssystemen eher uninteressant, da diese eine eigene Infrastruktur aus Signalsendern benötigen. Hier erscheint die Nutzung von GPS sinnvoller, da der Aktionsradius der Klienten größer als ein Haus oder eine mit Signalsendern ausgestattete Gegend ist.

Zur netzwerkgestützten Positionsbestimmung können bestehende Funknetze benutzt werden. Die fast flächendeckend vorhandene Infrastruktur von Mobilfunkanbietern (GSM) ermöglicht ebenso wie kabellose lokale Netze (WLAN) eine relative Bestimmung der Position des Benutzers. Da beide Systeme eine zellenartige Grundstruktur aufweisen und räumlich begrenzt zu empfangen sind, lässt sich ein Benutzer bis in eine Zelle zurückverfolgen. Die Zellgröße bestimmt hier also die Genauigkeit der Positionsbestimmung. Eine metergenaue Positionsbestimmung mit GPS ist militärischen Zwecken vorbehalten und entfällt für zivile Anwendungen. Die Standardgenauigkeit von GPS beträgt dabei ca. 100 m [21], ist somit also für den Einsatz in touristischen Informationssystemen geeignet. Auf Grund dieser technischen Eigenschaften werden im TIP-System Positionsabfragen stets mit einer Unschärfe von $\pm$ 5% betrachtet.

Die Benutzung von GPS zur Positionsbestimmung erscheint für touristische Anwendungen geeignet, da in mobilen Geräten zunehmend mehr GPS-Adapter eingesetzt werden, die entweder als Zusatzsteckmodule oder als eingebaute Chips (embedded chips) [18, 42] existieren. Damit können

diese Geräte für ortsabhängige Dienste benutzt werden und ermöglichen den Teilnehmern durch eine einfache Lokalisierungstechnik eine hohe Mobilität.

Die Hardwareausstattung und das verwendete Betriebssystem der mobilen Geräte bestimmen deren Fähigkeit, mit komplexen Informationssystemen zu kommunizieren. Hauptsächlich verbreitet sind folgende mobile Betriebssysteme:

1. Symbian OS
2. Palm OS
3. Microsoft Windows CE / Microsoft Windows Mobile 2003
4. Linux

Symbian OS gilt als der Marktführer bei mobilen Geräten [16, 55]. Funktional bietet Symbian OS vorwiegend Routinen, die Telefonfunktionen der mobilen Geräte ansprechen. Zur Programmierung existieren verschiedene Rahmenwerke (UIQ, Nokia Series 60) zur Oberflächengestaltung. Palm OS hat eine reduzierte grafische Oberfläche, deren Funktionsumfang mehr zur Verwaltung von Terminen und Kalendereinträgen dient und sich durch eine einfache Bedienung auszeichnet [8]. Microsoft stellt mit Windows CE eine reduzierte Windows-Betriebssystem-Umgebung für mobile Geräte zur Verfügung. Dabei ist eine Auswahl von Multimedia-Anwendungen bereits vorinstalliert. Die Benutzung der Geräte lehnt sich stark an die Windows-Umgebungen von Desktop-PCs an. Linux wird bisher nur im experimentellen Stadium eingesetzt und ist gering verbreitet [17].

Zusätzlich zur Softwareausstattung der mobilen Geräte stellt die Standardisierung von Hardware hohe Anforderungen an die Flexibilität von Anwendungssystemen. Diese sollten so modular wie möglich aufgebaut sein, um zukünftige Hardwareschnittstellen zu unterstützen und neue Softwarelösungen integrieren zu können. Auf der Serverseite kommen häufig verteilte Systeme zum Einsatz, die im Folgenden charakterisiert werden.

2.2 Verteilte Informationssysteme

Eigentlich kann jedes derzeit bestehende komplexe Computersystem als verteiltes Informationssystem betrachtet werden sobald große Mengen von Daten verwaltet und dargestellt werden [45]. Um dies auch unter hoher Anfragelast zu gewährleisten, werden einzelne Anwendungskomponenten auf verteilt

liegende Rechnerknoten ausgelagert und laufen dort ab. Für den Benutzer verteilter Systeme ist diese Tatsache oftmals nicht direkt erkennbar.

In Touristeninformationssystemen fallen verschiedene Arten von Daten an, die vom System abgearbeitet werden müssen und zu diesem Zweck auf verteilt liegende Rechnerknoten ausgelagert werden können. Das sind:

- statische Daten
- dynamische Daten

Um statische Daten, wie Bilder und Texte, handelt es sich bei den im TIP-System hinterlegten Informationen zu Sehenswürdigkeiten und Standorten. Unter dynamischen Daten versteht man hier die vom Benutzer erzeugten Laufzeitdaten. Dies sind sowohl Profileinstellungen als auch vom System erfasste Kommunikationsdaten (Positionsdaten und ausgewählte Menübefehle). Diese können auf verteilt agierende Rechnersystemen gespeichert und bearbeitet werden – zur Koordination der einzelnen Systemkomponenten bedarf es netzwerkfähigen Betriebssystemen oder so genannter Middleware. Diese stellt Schnittstellen zur Benutzung der virtualisierten Komponenten zur Verfügung, ohne dass der Benutzer die Details der Benutzung steuert oder davon Kenntnis hat. Die Middleware stellt dabei eine Art spanische Wand zur darunter liegenden Infrastruktur dar, die sicherstellt, dass auf Middleware aufbauende Anwendungssysteme nicht nur in einer speziellen Hardwareumgebung funktionieren.

Für verteilte Informationssysteme ergeben sich dadurch vielfältige Möglichkeiten zur Datenmodellierung, -speicherung und -verwaltung. In einer Diplomarbeit an der Freien Universität Berlin [33] wurden folgende näher untersucht:

- RDF und RDF Schema
- Topic Maps
- relationale und objektrelationale Datenbanken

Die relationale Modellierung bot dabei inhaltlich die größte Aussagekraft und überzeugte durch den implementierten Funktionsumfang.

Hauptvorteil von Datenbanksystemen ist die einfache Möglichkeit, durch Verwendung einer strukturierten Abfragesprache (SQL) gezielt Daten auszuwählen. Dies geschieht bei üblichen Datenbanksystemen mit vergleichs-

weise geringem Aufwand, da viele optimierte Algorithmen zur Datenverwaltung und Abfragebearbeitung existieren. Zusätzlich gibt es für Datenbanksysteme geografische Erweiterungen, die den Einsatz im Bereich ortsabhängiger Systeme ermöglichen. Dadurch entfällt der aufwändige Vorgang der Umrechnung oder Berechnung der Position des Benutzers. Stattdessen können die Daten direkt an etwaige Klienten ausgeliefert werden.

Datenbanksysteme eignen sich zur Implementierung verteilter Anwendungs- und Informationssysteme, da sie in der Lage sind, auf hohe Anforderungen durch angepasste Systemstrukturen zu reagieren. Dazu zählen Clustering-Technologien, die es ermöglichen, auf verteilt im Netzwerk liegenden Rechner-Knoten (host nodes) Aufgaben abzuarbeiten. Somit ist es möglich, ressourcenaufwändige Operationen wie Sortierungen oder das Zusammenfügen von Tabellen (join) zu optimieren. Mit Hilfe dieses Ansatzes zur Lastverteilung wird eine bessere Auslastung der vorhandenen Netzressourcen ermöglicht. Ebenfalls vermieden wird der zusätzliche Bedarf von Hardware; bestehende Komponenten können nahtlos in das Unternehmensnetz integriert werden und stellen ihre Ressourcen zur Verfügung. Beispielsweise verfolgt der Datenbankanbieter Oracle mit seiner aktuellen Datenbankversion 10g diesen Ansatz und verspricht eine höhere Effizienz bei der Ausnutzung bestehender IT-Ressourcen und eine gesteigerte Leistungsfähigkeit [40].

Um Anwendungen auf solchen Clustern ablaufen lassen zu können, müssen diese einen strukturierten Entwurf haben und ihre Dienste in verschiedene Schichten aufteilen. Kernkonzept ist dabei die Virtualisierung von Ressourcen, die es für Dienstbenutzer undurchschaubar macht, welcher Rechner oder welche Netzkomponente genau den Dienst zur Verfügung stellt. Die hier implementierte Anwendung baut auf dieser Idee auf, worauf im Abschnitt 4.3 näher eingegangen wird.

Datenbankmanagementsysteme stellen eine einfache Möglichkeit zur Verfügung, große Datenmengen zu speichern und zu verwalten; sie bilden damit das Rückgrat von verteilten Informationssystemen. Zusätzlich bieten sie vielfältige Schnittstellen zur Softwareentwicklung an. Diese Vorteile erleichtern den Softwareentwurf und vereinfachen zukünftige Migrationen. Dabei ist es auch möglich, die Netzinfrastruktur und insbesondere die Art der Kommunikation im Netz zu verändern. Datenbanken benötigen zur Kommunikation keine eigenen Protokollstapel. Sie lassen sich somit einfach in bestehende Systeme integrieren und an veränderte technische Standards anpassen.

Ebenso können zu einer bestehenden Datenbank beliebig neue Tabellen mit gewünschten Datenstrukturen hinzugefügt werden. Dies sichert die Ausbaufähigkeit des Systems. Datenbankmanagementsysteme bieten von sich aus ein einfaches Hinzufügen von Konsistenzprüfungen (trigger, constraint) und Sichten auf Daten (temporäre Tabellen oder Views). Dadurch kann der aktuelle Datenbestand der Anwendung vor Inkonsistenzen geschützt werden. Ein kompletter Neuentwurf wird bei veränderten Anforderungen somit umgangen.

Konzeptionell ähnlich integriert sich die Java-Unternehmensplattform (J2EE) in das Gebiet verteilter Informationssysteme. Diese von der Firma Sun standardisierte Softwareentwicklungsumgebung bietet kleinen und großen Firmen auf verschiedenen Schichten Möglichkeiten, Anwendungen zu entwickeln und auszuliefern.
Hauptaugenmerk von Sun ist die Plattformunabhängigkeit der eingesetzten Softwarelösung. Durch die Verwendung von Java als zu Grunde liegende Programmiersprache wird ein hohes Maß an Unabhängigkeit von Herstellererweiterungen oder eingesetzten Betriebssystemen ermöglicht, das bisher nahezu unmöglich war. Der Trend zu solchen interoperablen Systemen lässt sich auch aus der verstärkten Nutzung von Web-Services und XML (Extensible Markup Language) ableiten. Deren Nutzen ist die transparente Bereitstellung von Diensten über allgemeine Protokoll- und Beschreibungssprachen. Dabei wird XML zur Übertragung von Inhalten benutzt. XML ist eine textuelle Markierungssprache (markup language), die vom W3C (World Wide Web Consortium) standardisiert wurde. Markierungssprache bedeutet hierbei, dass Inhalte einer gewissen Ordnung genügen müssen und in Beschreibungstexte verpackt sind.

2.3 Freie Software (open source software)

Ähnlich wie die Nutzung offener Standards erscheint die Nutzung kostenfrei verfügbarer Software bei der Umsetzung verteilter Informationssysteme sinnvoll, da diese einfach miteinander gekoppelt werden können. Des Weiteren kann mit diesem Ansatz ein hoher Grad von Interoperabilität mit Systemen anderer Hardware- und Softwarehersteller erreicht werden. Dieser Aspekt gewinnt sowohl für kommerzielle als auch nichtkommerzielle Anbieter zunehmend an Bedeutung.

Erfolgreiche Open-Source-Projekte, wie das Apache-Webserver-Projekt [26], zeigen, dass sie das traditionelle Modell des Softwarevertriebs – Software entwickeln und dann als fertiges Programmpaket verkaufen – sinnvoll ergänzen und vielfältige Möglichkeiten zur Generierung von Umsätzen für kommerzielle Firmen bieten. Zusätzlich zur reinen Softwarelösung bildet sich oftmals eine Open-Source-Community um das Projekt. Deren Mitglieder arbeiten an der weiteren Reifung des Softwareproduktes und geben durch ihre Detailkenntnisse wertvolle Hilfe bei Problemen. Diese Tätigkeiten übernimmt sonst ein kommerzieller Supportdienst oder die Service-Abteilung des Herstellers.

Im Rahmen der Entwicklung von TIP bietet sich durch die Offenheit der eingesetzten Softwarelösungen die Möglichkeit, ein modulares System zu bauen, dass nach außen viele Schnittstellen für Erweiterungen bietet.

Durch die Möglichkeit, Einsicht in den Quellcode freier Software zu erhalten, wird im umgekehrten Sinne von den Entwicklern eine nötige Disziplin des Dokumentierens und Formatierens eingefordert. Die genauen Regeln werden in so genannten Coding Guidelines [54] beschrieben und sollen eine sinnvolle Gruppenarbeit von teilweise weltweit agierenden Entwicklern ermöglichen. Ebenso ist es möglich, aus dem veröffentlichten Quellcode Ideen und Lösungsansätze für eigene Entwurfs- und Entwicklungsentscheidungen abzuleiten.

Ein weiterer Vorteil freier Software ist die hohe Verfügbarkeit von Anschluss- und Folgeprodukten. Dies sind häufig zusätzliche Werkzeuge (tools), die die Arbeit mit dem eigentlichen Produkt oder dessen Konfiguration vereinfachen. Im Bereich der Java-Programmierung finden sich unzählige Rahmenwerke und Open-Source-Projekte, die zusätzlich zur von der Firma Sun zur Verfügung gestellten Programmiersprache die Entwicklung neuer Software erleichtern und die Benutzung bestehender Systeme vereinfachen sollen. Einige dieser Projekte wurden hier benutzt, worauf in Kapitel 5 näher eingegangen wird.

Viele der Open-Source-Projekte stehen in direktem Wettbewerb mit kommerziellen Firmen. Aus dieser Sicht erscheint es zusätzlich interessant zu prüfen, inwieweit die Nutzung frei erhältlicher Werkzeuge und Entwicklungsumgebungen für die Umsetzung von Projekten mit komplexen Strukturen geeignet ist.

2.4 Zusammenfassung

Im letzten Kapitel wurde das fachliche Umfeld dieser Arbeit dargelegt. Das zu entwickelnde Softwaresystem ist ein verteiltes Informationssystem, welches ortsabhängige Dienste anbietet und datenbankzentrierte Ansätze zur Datenspeicherung verfolgt. Das im Kapitel 1 gezeigte Anwendungsszenario (Abbildung 1.1, S. 2) zeigt ein Beispiel eines ortsabhängigen Informationssystems. Der Benutzer sendet seine aktuelle Position an den TIP-Server, der ihm an seinen aktuellen Kontext angepasste Informationen zurücksendet.

Zur Auslieferung der kontextsensitiven Inhalte benutzt die TIP-Anwendung freie Standards und kostenfrei verfügbare Softwareumgebungen. Dies ermöglicht – durch ein mehrschichtiges Zusammenspiel von Serversystemen und Softwarerahmenwerken – die Schaffung eines modularen touristischen Informationssystems, dessen genaue Struktur in den folgenden Kapiteln ausgearbeitet wird.

Kapitel 3

Abgrenzung zu existierenden Informationssystemen

Dieses Kapitel charakterisiert einige bestehende Systeme zur Verbreitung kontextsensitiver Informationen. Aus den dadurch gewonnenen Erfahrungen werden Anforderungen für das implementierte System abgeleitet, die dann im folgenden Kapitel 4 den Entwurf der Anwendung bestimmen.

3.1 Ziel-Netzarchitektur

Die bestehenden Systeme sind alle für eine bestimmte Netzarchitektur entworfen und umgesetzt. Dabei dominiert die Client-Server-Struktur stark, während der Einsatz auf mobilen Geräten nur mit bestimmten Einschränkungen seitens Mobilität und Datenabfragekapazität realisiert werden kann.

3.1.1 Client-Server-Architektur

Die Client-Server-Architektur zeichnet sich durch einen ressourcenmächtigen Server aus. Die Klienten fragen vorwiegend nur Daten vom Server ab, die dort aufbereitet und verarbeitet wurden. Deshalb verfügen sie meist über weniger Rechenkraft und Ressourcen wie Festplatten- oder Arbeitsspeicher. Da die Klienten im Gegensatz zum Server mobil sind, können sie nicht gleichwertig im Sinne ihrer Ausstattung sein. Zusätzlich braucht man zum Erhalt eines mobilen Netzes mehr Zusatzinformationen, um die Weiterleitung von Nachrichten und damit die Auslieferung von Diensten ermöglichen zu können,

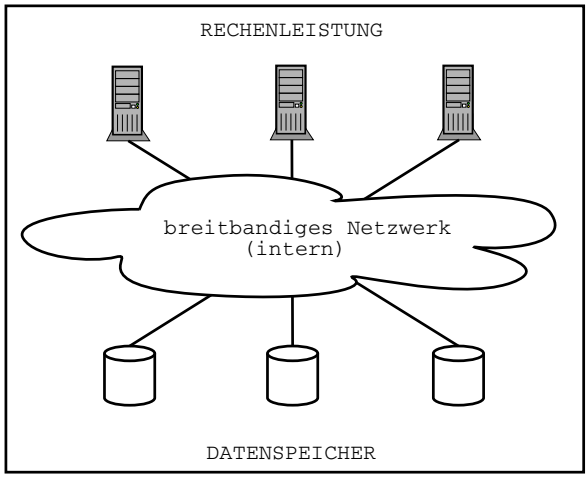

Abbildung 3.1: Serverseitige Struktur verteilter Informationssysteme

als dies beispielsweise in ortsgebundenen kabelbasierten Netzen der Fall sein kann. Ebenso müssen die mobilen Geräte und der Server ständig in Kontakt bleiben, damit der Server seine Dienste an die Bedürfnisse des mobilen Gerätes beziehungsweise seines Nutzers anpassen kann. Die Client-Server-Struktur eignet sich zum schnellen Testen von Anwendungen. Diese müssen nicht auf jedem mobilen Gerät einzeln installiert und konfiguriert werden, sondern lassen sich oft ohne hohen Aufwand mit normalen Standrechnern (z.B. der PC des Softwareentwicklers) einsetzen. Ebenso günstig wirkt sich aus, dass für die Einrichtung eines Servers bestehende Hardware benutzt werden kann, während je nach Anwendungsbereich mobile Geräte zusätzlich erworben werden müssen.

Auf der Serverseite besteht das verteilte Informationssystem aus einem breitbandigen internen Netzzugang, der die Komponenten des Clusters miteinander verbindet (siehe Abbildung 3.1). Dieser Ressourcen-Cluster virtualisiert für seine Benutzer alle in ihm vorhandenen Komponenten und optimiert damit den Zugriff für Dienstnehmer. Dabei werden große Datenspeicher – z.B. Datenbanken oder Dateisysteme – und eine hohe Rechenleistung benötigt. Unter Virtualisierung versteht man, dass für den Dienstbenutzer nicht erkennbar ist, welche Komponente des Servers seine Abfrage bearbeitet. Diese Arbeitsteilung steuert der Server autonom von den Dienstnehmern. Abgesehen von der internen Verarbeitung muss der Server auch den Kontakt nach außen zu den mobilen Geräten zur Verfügung stellen. Je nach An-

Abbildung 3.2: Klassen möglicher mobiler Geräte

bindung kann dies ein kabelgebundener (LAN, WAN) oder ein kabelloser Zugang (WLAN, GPRS) sein. Die vom Dienstnehmer gewählte Zugangsart entscheidet, ob verbindungsorientierte (TCP) oder verbindungslose (UDP) Protokolle benutzt werden [23]. Da es sich in den meisten Fällen um eine interaktive Beziehung zwischen Diensterbringer und Dienstnehmer handelt, werden vorwiegend verbindungsorientierte zuverlässige Protokolle eingesetzt. In Anwendungsszenarien, die keinen kontinuierlichen Kontakt zwischen Server und Klient benötigen, können verbindungslose ungesicherte Protokolle eingesetzt werden.

Der Dienstnehmer besitzt in dieser Arbeit ein mobiles Gerät. Der Grad der Mobilität bestimmt die Leistungsfähigkeit des Gerätes. Je mobiler das Gerät desto geringer seine Leistungsfähigkeit in Bezug auf Darstellung grafischer Inhalte und Netzkommunikation. Dieser Zusammenhang ergibt sich aus der Stromversorgung der mobilen Geräte durch Batterien. Abbildung 3.2 zeigt die Klassen mobiler Geräte, die von bisher existierenden Systemen abgedeckt werden. Das sind: persönliche digitale Assistenten (PDA) – als Mixgerät aus Terminverwaltung, Kalender und Telefon –, reine Mobilfunktelefone sowie mobile Computer wie Laptops oder Tablet-PCs.

Die Größe der Geräte bestimmt auch deren Fähigkeit zur Darstellung grafischer Inhalte und schränkt die Möglichkeiten der Benutzerinteraktion und Navigation ein. Im Falle eines Mobiltelefons besitzt man nur den Tastenblock, während bei einem PDA oftmals die Möglichkeit besteht, durch

Berührung des Bildschirms Navigationsbefehle auszuführen. Ein Tablet-PC oder ein Laptop verfügen meist über Tastatur und Maus oder mausähnliche Geräte (touchpad), die eine vereinfachte Navigation bieten und nicht nur Eingaben über die Tastatur gestatten.

3.1.2 Einsatz mobiler Geräte

Mobile Geräte werden im Rahmen verteilter Informationssysteme vorwiegend von Dienstnehmern eingesetzt. Dies liegt an der vorherrschenden Client-Server-Architektur. Wenn mobile Geräte auch als Dienstgeber benutzt werden, handelt es sich um eine Peer-To-Peer-Infrastruktur. Bei dieser wird jeder Netzteilnehmer als gleichwertig betrachtet und kann sowohl als Dienstgeber als auch Dienstnehmer fungieren. Diese Aspekte werden von keinem der bisher existierenden Systeme realisiert, was teilweise an den bisher ungelösten Problemen im Bereich der Peer-To-Peer-Netze liegt. Diese lassen sich folgenden Kategorien zuordnen:

- Routing von Nachrichten in hochmobilen Netzen
- Informationsklassifikation und Verteilung in hochmobilen Netzen

Das Routing von Informationen betrifft die technische Ebene der Kommunikation. Ein Netz, dessen Teilnehmer mehrheitlich unregelmäßig miteinander verbunden sind, kann schwerlich zur gesicherten Weiterleitung von Informationen von Teilnehmer zu Teilnehmer benutzt werden.
Die andere Problemklasse ist mehr semantischer Natur. In einem Netz, dessen Teilnehmer sich durch Batterien mit Strom versorgen, bestehen verschiedene Interessen in Bezug auf Weiterleitung von Nachrichten. Nicht jeder Benutzer möchte seine vorhandenen Ressourcen komplett zum Erhalt des Netzes einsetzen, sondern strebt eher eine lange Laufzeit seines mobilen Gerätes an. Eine mögliche Lösung dieses Problems ist die Einbeziehung des Benutzers und die manuelle Auswahl eines Kompromisses zwischen eigener Gerätelaufzeit und Einbringung ins Netz.

Das an der Universität Lancaster entwickelte GUIDE-Projekt [31, 32] bezieht Informationen aus der Umgebung zur Speicherung und effizienteren Auslieferung von Informationen an Klienten mit ein. Technisch handelt es sich um eine WLAN-Umgebung. Die einzelnen Zugangspunkte werden mit Tablet-PCs abgefragt, die über einen großen Bildschirm verfügen. Die Zugangspunkte sind so an Datenspeicher angeschlossen, dass die Informationen

zu geografisch um den Zugangspunkt nahe liegenden Sehenswürdigkeiten lokal gespeichert sind. Somit wird die Gesamtnetzlast verringert und die Leistungsfähigkeit des Systems erhöht. Diese Technik eignet sich für ein Netz, das aus einer minimalen Anzahl von festen Teilnehmern oder Netzknoten besteht. Im Fall von WLAN sind das die Netzzugangspunkte.
Unter Einbeziehung von Peer-To-Peer-Elementen wäre es zusätzlich möglich, ein hybrides Informationssystem aufzubauen, das sowohl Informationen vom Server abfragt als auch Informationen durch direkten Kontakt mit in der Umgebung befindlichen mobilen Klienten austauschen kann. Dieser Ansatz wurde auf Grund von technischen Problemen (siehe Abschnitt 4.1) im Laufe der Arbeit verworfen, sollte aber nach Erscheinen der nächsten Java-Version für mobile Geräte wieder aufgenommen werden.

Für die Softwareentwicklung von großer Bedeutung ist die Art und Weise wie mobile Geräte grafische Inhalte darstellen, womit sich der nächste Absatz beschäftigt.

3.2 Verwendung grafischer Oberflächen

Grundsätzlich bieten sich zwei Möglichkeiten, grafische Inhalte auf mobilen Geräten darzustellen:

- Verwendung nativer, hardwarenaher grafischer Oberflächen
- Oberflächendarstellung mit Hilfe anderer Programme (z.B. Browser)

3.2.1 Native Oberflächengestaltung

Um native grafische Oberflächen handelt es sich, wenn auf dem mobilen Gerät ein Programm läuft, das separat programmierte Routinen zur Grafikanzeige und Navigation enthält. Dies können beispielsweise Oberflächen sein, die in C/C++ programmiert oder mittels J2ME umgesetzt sind. Oftmals lassen sich diese Oberflächen nur durch Verwendung spezieller hardwarenaher Rahmenwerke benutzen. Im Falle des Betriebssystems Symbian OS geht dies nur mit C++, womit die Implementierung nur auf mobilen Geräten lauffähig ist, die mit Symbian OS betrieben werden. Jede zusätzlich unterstützte Geräteklasse erfordert einen kompletten Neuentwurf der Software.

Die Umsetzung mittels J2ME [53] gestaltet sich nicht weniger problematisch, da man entweder im hardwarenahen Modus jeden Pixel einzeln ansteuern kann oder das Java-Konzept von mehrschichtigen Oberflächen benutzbar ist [37]. Dieses Konzept ist verwandt zu den in der Java-Welt benutzten Swing- und AWT-Bibliotheken zur Erzeugung grafischer Oberflächen. Dabei wird der Bildschirm in Schichten unterteilt, an die man grafische Elemente wie Boxen oder Knöpfe anheften kann. Um eine vereinfachte Darstellung zu ermöglichen, existieren so genannte Layout-Manager-Klassen, die die Inhalte automatisch an die Gerätemerkmale anpassen. Ein hybrider Modus aus teilweiser Verwendung der vorgefertigten Grafikelemente und der nativen pixelorientierten Ansteuerung einzelner Bildbereiche ist nicht möglich.

Die Verwendung der jeweiligen grafischen Rahmenwerke vereinfacht die Benutzung etwas, erscheint jedoch immer noch unzureichend für die Entwicklung kommerzieller Anwendungen oder verteilter Informationssysteme. Derzeit werden mit Hilfe der Rahmenwerke vorwiegend Spiele implementiert und an Benutzer verkauft. Diese installiert man einmalig auf seinem mobilen Gerät und erhält eine speziell an die grafischen Fähigkeiten des Gerätes angepasste Softwareversion. Die für verteilte Informationssysteme zusätzlich erwünschte Fähigkeit zur Kommunikation über verschiedene Netze wird dafür nicht gebraucht, da es sich meistens um Einzelspieler-Anwendungen handelt.

3.2.2 Browserbasierte Zugänge

Anwendungen, bei denen grafische Oberflächen nicht durch native Programmierung entstehen, laufen häufig in Internetbrowsern und bieten den Vorteil, dass einfache Protokolle und Datenformate zum Datenaustausch benutzt werden können. Dadurch erfordert die Benutzung einer browserbasierten Anwendung keine komplexe Installation auf Dienstnehmerseite. Es genügt der Aufruf einer Internetadresse (URL) in einem entsprechend konfigurierten Browser. Die Inhalte werden häufig mit dem Protokoll HTTP vom Server zum Klienten übertragen. Der Browser baut aus den gelieferten HTML-Seiten autonom die grafische Oberfläche auf. HTML ist eine vom World Wide Web Consortium (W3C) standardisierte Sprache zur besonderen Auszeichnung von Inhalten [9, 60]. Dies geschieht durch Verpackung des Inhalts in so genannte Bezeichner (tags), die immer paarweise auftreten und somit den Anfang und das Ende der Daten genau markieren.

Ein Vorteil dieses Standards ist seine hohe Verbreitung durch das Internet. Technisch bietet die Weiterentwicklung von HTML zu XHTML – Extensible Hypertext Markup Language – auch andere Möglichkeiten, die Inhalte standardkonform darzustellen. Dies wird durch eine Validierung der Dokumente gegen so genannte Dokumenttypbeschreibungen (DTD) erreicht, die von Softwarewerkzeugen (parser) automatisch durchgeführt werden kann.
Als nachteilig erweist sich die mangelhafte Unterstützung für individuelle Eingabeelemente. Derzeit sind nur folgende Formen von Interaktion mit dem Benutzer möglich:

- Texteingabe- und Passwortfelder
- Textboxen
- Mehrfach- und Einfachauswahllisten
- Bilder
- Knöpfe (buttons) sowie
- Verlinkung (URL) zu anderen Ressourcen

Die folgenden Absätze beschreiben verschiedene Technologien zur flexiblen Anpassung von Inhalten an Benutzerpräferenzen.

3.2.2.1 CSS und HTML

Die Kopplung von Inhalten und Darstellungsbefehlen gilt im Allgemeinen als schlechter Stil, da die Gefahr besteht, dass bei Änderungen wichtige Quellcodesegemente in ihrer Funktion gestört werden. Des weiteren ist es nicht möglich, die Art der Darstellung zentral und einfach zu verändern.
Aus diesem Grund sind Verfahren zur Trennung von Inhalt und Darstellung beliebt. Dabei ist es möglich, für jeden HTML-Befehl festzulegen, wie der Browser Inhalte hinter diesem Befehl darstellen soll. Diese Information wird in getrennten Dateien hinterlegt, die man Darstellungsschemata oder Cascading Stylesheet (CSS) nennt. CSS-Dateien sind Textdateien und können daher mit jedem Programm geöffnet und angepasst werden.

Die nötigen CSS-Dateien liegen auf den Servern des Dienstanbieters. Fordert der Dienstnehmer ein Dokument an, so wird dieses an den Browser zusammen mit der CSS-Datei geschickt, der die Inhalte entsprechend

formatiert und für den Benutzer anzeigt. Diese Technik wird auch im hier entworfenen Programm eingesetzt, wobei eine CSS-Datei für die komplette Anwendung existiert. Durch Anpassung dieser Datei kann das gesamte Aussehen der Anwendung verändert und schnell angepasst werden. Die Einbindung der CSS-Datei in den HTML-Quellcode erfolgt mittels einer URL-Angabe, wodurch für die gesamte Anwendung eine CSS-Datei verwendet werden kann, die der Dienstnehmer auch nur einmal vom Server anfordern muss. Die meisten Browser verfügen über einen Datencache, in dem sämtliche Inhalte von Internetseiten abgelegt werden können, um ein wiederholtes Herunterladen gleicher Inhalte vom Server zu vermeiden.

Das vorher erwähnte GUIDE-Projekt (siehe 3.1.2), das touristische Informationen auf Tablet-PCs anzeigt, benutzt eine Java-Anwendung, in die HTML-Inhalte unter Verwendung der CSS-Technologie eingebettet sind. Der Benutzer kann in der Java-Anwendung seinen aktuellen Status verändern oder neue Eingaben machen. HTML und CSS werden benutzt, um die jeweiligen Inhalte (Bilder, Texte) anzuzeigen. Da diese Anzeigekomponenten Teil der nativen Anwendung sind, kann man sie nur bedingt mit einem reinen Browserzugang vergleichen. Zur besonderen Darstellung auf Mobiltelefonen existiert eine spezielle Markierungssprache, auf die im nächsten Abschnitt eingegangen wird.

3.2.2.2 WML

Mobiltelefone zeichnen sich im Vergleich zu mobilen Computern durch eine geringere Leistungsfähigkeit, kleinere Anzeigegrößen, eingeschränkte Navigation und Netzzugänge mit geringer Bandbreite aus. Diese Einschränkungen wurden beim Entwurf einer eigenen Markup-Sprache für mobile Geräte (WML [22]) berücksichtigt. Der Datenaustausch bei Verwendung von WML ist durch eine Karten-Metapher gekennzeichnet. Alle Inhalte werden bestimmten Kartenstapeln zugeordnet, zwischen denen ein Benutzer navigieren kann. Ein Kartenstapel besteht dabei aus logisch zusammenhängenden einzelnen Karten, die über eine Navigation durch Links ähnlich wie bei HTML miteinander verbunden sind. Da die Karten alle eine bestimmte Größe haben, lassen sie sich schrittweise auf mobile Geräte übertragen, wobei jeweils immer nur eine feste Menge von Karten übertragen wird und die Verbindung danach theoretisch beendet werden kann. Zur Anzeige wird ein spezieller

WML-Browser benötigt, der die komprimiert übertragenen Inhalte in Kartenform erhält und diese auf das Gerät angepasst darstellt. Somit erreicht WML ebenfalls eine Trennung der Darstellung und des Inhalts, ist jedoch heutzutage kaum verbreitet. Unter dem Gesichtspunkt der Einführung der nächsten Mobilfunkgeneration und neuer breitbandiger Funktechnologien erscheint die Karten-Metapher als nicht mehr zeitgemäß.

Das MOPAS-Projekt aus Norwegen [41] hat ein Schiffsinspektionssystem unter Einsatz mobiler Geräte im kommerziellen Umfeld entworfen. Dabei sollten die durch Einführung von Wireless Application Protocol (WAP) und WML veränderten Technologien benutzt werden. Das System stellt eine Hilfe bei der Abarbeitung von Prüflisten bei der Schiffsinspektion dar. Dabei werden Informationen in gerätespezifischen Formaten ausgeliefert; für mobile Geräte geschieht dies mittels WML. Bei der Durchführung des Projekts wurden Untersuchungen zur Oberflächengestaltung vorgenommen, die gezeigt haben, dass der Einsatz der Kartenmetapher nur für bestimmte Umgebungen sinnvoll ist. Soll beispielsweise eine Art von Prüfliste abgearbeitet werden, so kann man die Einzelprüfungen auf Karten unterbringen. Für kompliziertere Anwendungsfälle erscheint dies wiederum eher hinderlich. Mit der Einführung von WAP 2.0 können Inhalte statt nur mit WML auch mit XHTML an entsprechende Mobiltelefone ausgeliefert werden, was die umständliche Benutzung von WML vermeidet.

3.2.2.3 XSL und XML

Ähnlich wie CSS und HTML (siehe Abschnitt 3.2.2.1) bieten XML Style Language (XSL) und Extensible Markup Language (XML) die Möglichkeit, Inhalte von der Art der Darstellung zu trennen. Die eigentlichen Inhalte, also Daten, werden dabei textuell in XML-Dateien abgelegt. Zur Darstellung auf einem beliebigen Gerät werden diese mit XSL-Darstellungsbeschreibungen verbunden. Jedes XSL-Dokument beinhaltet genauere Informationen zur Darstellung von Inhalten auf einer bestimmten Geräteklasse. Es kann somit XSL-Dateien für:

- mobile Geräte – Mobiltelefon, PDA, Pocket-PC –
- mobile Computer – wie Laptop oder Tablet-PC – oder
- Standgeräte (PC) mit verschiedenen Bildschirmgrößen geben.

Somit besteht eine große Flexibilität für den Dienstanbieter. Technisch entfällt das Zusammenfügen von Inhalt und Darstellung auf Dienstnehmerseite, wie dies bei CSS/HTML der Fall ist. Stattdessen fügt der Server das angeforderte XML-XSL-Dateipaar zu einer entsprechenden HTML-Datei zusammen, die dann an den Klienten ausgeliefert wird. Diese Datei kann mit einem normalen Browser angezeigt werden, da sie standardkonform ist und keine nativen Elemente zur Oberflächensteuerung mehr enthält. Der Vorgang des Zusammenfügens von XML-Daten mit XSL-Stilbeschreibungen heißt XSL-Transformation (XSLT).

Die Bedeutung von XML und XML-benutzenden Technologien hat in den vergangenen Jahren stark zugenommen, da Firmen mit unterschiedlicher Hardware- und Softwareinfrastruktur zunehmend miteinander kooperieren und Daten austauschen oder gemeinsam bearbeiten müssen.
Als vorteilhaft erweist sich dabei eine besondere Eigenschaft von standardisierten XML-Dokumenten: sie müssen einem Dokumenttyp (DTD) entsprechend formatiert sein. Durch diese Einschränkung wird vermieden, dass syntaktisch inkorrekte Daten beim Datenaustausch inkonsistente Systeme hervorrufen und somit den Gesamtbetrieb negativ beeinflussen. Das automatische Prüfen der Daten kann dadurch unabhängig vom eigentlichen Datenladen und -importieren geschehen. Die dafür nötigen Dokumentenanalysewerkzeuge wurden innerhalb der letzten Jahre verfeinert.
Im Zusammenhang mit ortsabhängigen Systemen besteht unter Verwendung von XML-Standards die Möglichkeit, zwischen verschiedenen Anbietern von Touristeninformationssystemen über vorher definierte Schnittstellen Daten auszutauschen. Dies wird durch die meisten relationalen Datenbankmanagementsysteme unterstützt, die auch für verschiedene Programmiersprachen den XML-Export und -Import steuern können. In dieser Hinsicht erweist sich die Verwendung von Datenbanksystemen auch hier als geeignet.

Wenn die mobilen Geräte statt HTML direkt XML-Daten austauschen, verursacht dies eine sehr hohe Datenlast. Da die eigentlichen Nutzdaten immer in XML verpackt und dann geprüft werden müssen, bedarf es hohem Rechenaufwand und Speicherbedarf zur Pufferung der Daten beim Ein- und Auslesen. In manchen Szenarien kann man diese Probleme dadurch abmildern, dass man den XML-Datenstrom komprimiert. Im mobilen Kontext erweist sich dies jedoch als wenig tragfähig, da die Batterielaufzeit der mobilen Geräte durch verstärktes Komprimieren und Dekomprimieren der Datenströme abnimmt.

Tian et.al. [38] haben gezeigt, dass in bestimmten Szenarien, in denen mobile Geräte mit geringer Bandbreite angeschlossen sind, durch Komprimierung transportierter Inhalte Leistungszuwächse erzielt werden können.

In Echtzeitsystemen verursacht die XML-Technologie durch Konvertierungs- und Formatierungsarbeiten zusätzlich hohe Latenzzeiten und scheint damit für den Einsatz in ortsabhängigen Informationssystemen bei Verwendung leistungsschwacher batteriebetriebener mobiler Endgeräte derzeit nicht geeignet. Unter Verwendung von XSLT wird jedoch an die mobilen Geräte nur HTML ausgeliefert, was von den geräteeigenen Browsern dargestellt wird und in Bezug auf die Bandbreite einen Kompromiss zwischen komprimiertem WML und unkomprimiertem textreichen XML darstellt.

Das CATIS-System [2] der Northwestern University, USA, bietet ein auf dem .NET-Framework der Firma Microsoft basierendes Touristeninformationssystem, das sich durch die Verwendung von XML-Technologien auszeichnet. Dabei werden Web-Services zur Kommunikation der internen Dienste des Informationssystems benutzt. Das bedeutet, dass die Abfrage von verfügbaren Informationen von einem Anwendungsserver an ein UDDI-Verzeichnis (Universal Description, Discovery, and Integration) weitergeleitet wird, das ein zentrales Repositorium für touristische Informationen darstellt. Der UDDI-Server bietet Zugriff auf Beschreibungen von Web-Services, die in WSDL (Web Services Description Language) spezifiert sind. Der UDDI-Server wird dabei über das SOAP-Protokoll (Simple Object Access Protocol) abgefragt und bietet somit eine standardisierte Schnittstelle zum Datenaustausch, die von beliebigen Software- und Hardwareherstellern benutzbar ist. CATIS existiert derzeit nur als Prototyp und hat noch keine kommerzielle Anwendung gefunden.

3.3 Zusammenfassung

Dieses Kapitel hat aufgezeigt, welche Probleme bei der Gestaltung verteilter Informationssysteme für mobile Geräte derzeit auftreten. Diese Probleme hängen eng mit der eingeschränkten Leistungsfähigkeit mobiler Geräte zusammen. Vorwiegend zum Einsatz kommen Client-Server-basierte Systeme, bei denen ein ressourcenstarker Verbund mehrschichtiger Server Dienste und Daten zur Verfügung stellt.

Die eigentliche Darstellung der Inhalte ist je nach Ansatz auf das mobile Gerät verlagert oder wird vom Server in Abhängigkeit der Geräteklasse des Dienstnehmers durchgeführt. Beide Ansätze wurden skizziert und mit einem bisher bestehenden System in Verbindung gebracht. Die Darstellung hat gezeigt, dass ein offenes erweiterbares System nötig ist, um verteilte Informationssysteme mit verschiedenen mobilen Teilnehmern zu realisieren. Die Anforderungen dazu werden im nächsten Kapitel entwickelt, woraus dann die Architektur der entwickelten Software abgeleitet wird.

Kapitel 4

Konzeptioneller Entwurf

Die letzten Kapitel haben gezeigt, dass beim Entwurf und der Implementierung ortsabhängiger verteilter Informationssysteme diverse Einschränkungen zu beachten sind. Diese sind sowohl inhaltlicher als auch technischer Natur. Im Folgenden wird dargelegt, welche Rahmenbedingungen und Anforderungen an die zu entwickelnde Software gestellt wurden.

4.1 Rahmenbedingungen

Das zu schaffende Softwaresystem muss technische und inhaltliche Einschränkungen beachten. Die technischen Rahmenbedingungen sind durch die Eigenschaften der verwendeten mobilen Geräte bestimmt. Dies sind, wie im vorigen Kapitel beschrieben, Fähigkeiten zur grafischen Anzeige und Navigation sowie benutzte Netzschnittstellen. Die Kommunikationsadapter beeinflussen den Datendurchsatz der Client-Server-Verbindung, während die grafischen Gerätefähigkeiten die Art und Weise der Darstellung bestimmen. Im Rahmen dieser Arbeit wird vom verwendeten Netzzugang dahingehend abstrahiert, dass ein TCP/IP-fähiges Kommunikationssystem angenommen wird, welches über HTTP Verbindungen aufbauen kann [22].

Durch die seit einigen Jahren geplante Einführung von IPv6 [56] kann jedes mobile Gerät eine eigene öffentliche IP-Adresse zugeordnet bekommen. Aus Sicht der Dienstanbieter ergibt sich dadurch die Möglichkeit, benutzte Dienste und Anwendungen über verschiedene Dienstanbieter hinweg abrechnen und nachvollziehen zu können. Derzeit besitzt jeder Teilnehmer eines mobilen Netzes eine private IP-Adresse, die durch Firewall- oder Routing-

Server in eine öffentliche Adresse umgewandelt wird. Dieser Vorgang wird auch als IP-Masquerading oder Network Address Translation (NAT) bezeichnet. Ein Vorteil dieser Technologie aus Sicht des jeweiligen Netzbetreibers ist, dass der mobile Klient durch Vorhandensein einer privaten IP-Adresse keine Dienste anbieten kann und andererseits vor Angriffen aus anderen Netzen geschützt bleibt, da diese nur die öffentlich gültige IP-Adresse des Routers kennen. Dieses mögliche Sicherheitsrisiko hat bisher eine weite Verbreitung von IPv6 im Mobilfunkbereich verhindert.

Für das hier zu entwickelnde Touristeninformationssystem wird das Vorhandensein eines Internet-Browsers vorausgesetzt, der in der Lage ist, standardkonforme HTML-Seiten zu lesen und Grafiken im Format JPEG und PNG anzuzeigen. Des Weiteren sollte der Browser Javascript verarbeiten und ausführen können, da diese Routinen zur Prüfung von Eingaben im Browser benutzt werden. Die Anwendung ist für eine Auflösung von 640 x 480 Pixel optimiert. Zum Testen an nicht mobilen Computern wird dabei um die eigentliche Seite ein Rand aufgebaut, so dass die Inhalte in einer Breite von 640 Pixeln angezeigt werden. Diese Bildschirmauflösung erscheint als Mindestgröße geeignet, da diverse Anbieter ihre Geräte darauf einstellen. Zusätzlich existieren mit dem *Opera Web-Browser* und dem *Internet Explorer für Pocket PC* Anwendungen, die in der Lage sind, Webseiten auf die jeweiligen Gerätegrößen anzupassen und Bilder entsprechend zu skalieren. Die Verwendung von Browsern erspart die Entwicklung einer schwergewichtigen Anwendung, die hohe Anforderungen an das mobile Gerät stellt. Zusätzlich lässt sich die Anwendung bei Benutzung eines Internetbrowsers auch von anderen Computern aus testen; es bedarf dazu nicht unbedingt eines mobilen Gerätes. Ein weiterer Vorteil der Browsertechnologie im Gegensatz zu nativen Lösungen ist die Fähigkeit, HTML-konforme Inhalte durch Konverter für sehbehinderte Menschen aufzubereiten, womit eine zusätzliche Barrierefreiheit des Informationsdienstes erreicht wird. Die Anwendung kann schnell auf verschiedenen Gerätetypen getestet werden, da diese nur über einen Internetzugang verfügen müssen. Dies erleichtert die spätere Anpassung an neue Geräte- und Anzeigetypen.

Durch die Verwendung offener Standards und die Schaffung einer flexiblen Architektur lässt sich ein touristisches Informationssystem realisieren. Im Gegensatz zu existierenden Ansätzen wird hier der Schwerpunkt auf eine gut strukturierte und dokumentierte Architektur und die Verwendung von Entwurfsmustern gelegt. Bisher existierende Systeme beschränken sich häufig

auf die Entwicklung funktionaler Prototypen, über deren Architektur nichts bekannt ist. Das hier entwickelte verteilte System arbeitet nach dem Client-Server-Prinzip. Dabei liegen sowohl die gesamte Logik als auch die Daten der Anwendung auf der Serverseite, während der mobile Klient diese nur anzeigt und durch Navigationslogik den Anwendungskontext verändert.

Anfangs bestand die Idee, zwei getrennte Anwendungen unter verschiedenen Paradigmen zu entwerfen - eine P2P-basierte Anwendung auf mobilen Geräten und eine Client-Server-basierte mehrschichtige Anwendung. Der mobile Teil der Anwendung sollte unabhängig vom Client-Server-System den direkten Kontakt zu Klienten mit ähnlichen Interessen ermöglichen und einen alternativen Datenaustausch bieten. Dieser Ansatz wurde verworfen, da die Programmierumgebungen für mobile Geräte derzeit vorwiegend die Entwicklung von Spielen unterstützen. Dies erschwert die Implementierung von Kommunikationsbeziehungen bei Benutzung verschiedener Kommunikationswege. Ebenso ungünstig ist die geräteunabhängige Oberflächenprogrammierung. Jeweils verfügbare Rahmenwerke wie UIQ oder J2ME eignen sich nur bedingt, da entweder nur spezielle Geräte unterstützt werden oder nur ein geringer Funktionsumfang zur Verfügung steht. Die Problematik der Oberflächenprogrammierung auf mobilen Geräten wurde in Abschnitt 3.2 erläutert.

Inhaltlich kann die Einbeziehung von P2P-Ideen interessant sein, da damit eine Unabhängigkeit vom Dienstanbieter ermöglicht wird, die das System für Benutzer attraktiver machen kann. Ebenso verändert dieser Ansatz die Interaktion mit dem System. Während sonst vorwiegend ein Benutzer über ein Gerät mit einem Server kommuniziert, besteht im P2P-Kontext auch die Möglichkeit, direkt mit anderen Benutzern in Kontakt zu treten. Diese direkte Benutzer-Benutzer-Kommunikation kann, bei vertrauensvollem Umgang aller Beteiligten, eine höhere Qualität der Inhalte aus Sicht der Benutzer hervorrufen, da möglicherweise kritischere Bewertungen durch Dienstnehmer als durch den Dienstanbieter selbst veröffentlicht werden.

Der ursprünglich in dieser Diplomarbeit verfolgte Ansatz sollte eine separat installierbare Software schaffen, die die Benutzer-Benutzer-Kommunikation auf Dienstnehmerseite fördert. Dabei sollte die Java-Umgebung J2ME der Firma Sun [53] zum Einsatz kommen, da diese mobile Ausgabe der Programmiersprache Java einen speziell auf ressourcenschwache mobile Geräte reduzierten Funktionsumfang im Vergleich zur Standardausgabe (J2SE) zur Verfügung stellt.

Name	**Monitorauflösung und Farbanzahl**
Farb-Telefon mit Mobiltelefontastatur und zwei Steuerungsknöpfen	180 x 208 Pixel bei 256 Farben
Graustufen-Telefon mit Mobiltelefontastatur und zwei Steuerungsknöpfen	180 x 208 Pixel bei 256 Grautönen
Multimedia-Telefon mit Mobiltelefontastatur, zwei Steuerungsknöpfen und Audio/Video-Funktionen	180 x 208 Pixel bei 256 Farben
Telefon mit kompletter Tastatur (QWERTZ/QWERTY) sowie zwei Steuerungsknöpfen	640 x 240 Pixel bei 256 Farben

Tabelle 4.1: Geräteeigenschaften im WTK-Simulator 2.1 [52]

Zur Entwicklung mobiler Anwendungen, die auf Java-Technologie basieren, wird von der Firma Sun das Wireless Toolkit (WTK) in der Version 2.1 für alle gängigen Betriebssysteme angeboten. Dieses Programm ist kostenfrei erhältlich und soll die Anwendungsentwicklung für mobile Geräte vereinfachen. Jede Java-Anwendung (Midlet) wird als Paket (JAD) an Klienten ausgeliefert und aus Sicherheitsgründen signiert. Zur Vereinfachung der Erstellung und Signierung einer J2ME-Anwendung stellt das WTK spezielle Routinen zur Verfügung und ermöglicht zusätzlich eine einfache Simulation der Programme an einem Gerätesimulator. Dabei werden die gerätespezifischen Eigenschaften wie Monitorgröße und Art der Tastatur berücksichtigt. Im Simulator existieren die in der Tabelle 4.1 aufgeführten Gerätemodelle als Standard. Es lassen sich zusätzlich eigene Geräteklassen hinzufügen, deren Geräteeigenschaften in textuellen Konfigurationsdateien spezifiziert werden müssen.

Für mobile Touristeninformationssysteme ist dieser Ansatz nur begrenzt brauchbar, da durch Simulation auf einem Entwickler-PC kein wirklicher Vergleich mit realen Geräten möglich ist. Einziger Vorteil bleibt die Möglichkeit, Anwendungspakete schnell auf ihre Funktionalität zu testen.

Als problematisch erweist sich bei J2ME-basierten Lösungen derzeit auch die mangelnde Unterstützung durch Hardwarehersteller. Am Markt existieren nur wenige Geräte, die den aktuellen J2ME-Spezifikationen genügen. Daher kann man die selbst entwickelte Software zwar am Simulator testen, aber schwerlich auf real existierende Geräte bringen. J2ME basiert auf zwei Spezifikationen für mobile Geräte. Dies sind:

1. Connected Limited Device Configuration – CLDC 1.1 (JSR 30, 139) – `http://java.sun.com/products/cldc/`
2. Mobile Information Device Profile – MIDP 2.0 (JSR 37, 118) – `http://java.sun.com/products/midp/`

CLDC beinhaltet Routinen zum Datenaustausch (I/O) sowie Unterstützung für einfache Java-Datentypen und zielt auf Geräte ab, die über eine sehr geringe Leistung verfügen. Dies betrifft sowohl Prozessor, Speicher als auch die grafische Anzeige. Erst ab Version 1.1 unterstützt CLDC die Verarbeitung von Gleitkommazahlen. Als Weiterentwicklung und funktionale Erweiterung gilt MIDP, das in der Version 2.0 besonders für die Entwicklung von Spielen eine neue grafische Umgebung zur Verfügung stellt und die Multimediafähigkeiten der Geräte erweitert. Es ist in der aktuellen Version möglich, Audio- und Videosequenzen abzuspielen. Zusätzlich verfügen Geräte, die der MIDP-Spezifikation genügen, auch über die Fähigkeit, Verbindungen über HTTP aufzubauen.

Die Weiterentwicklung und Erweiterung von Java geschieht in einem Standardisierungsgremium, dem Hersteller, Entwickler und Interessierte angehören und nennt sich Java Community Process (JCP). Zur Standardisierung vorgeschlagene Entwürfe heißen Java Specification Requests (JSR) und sind derzeit noch nicht im Funktionsumfang der Sprache Java enthalten.
Im Standardisierungsprozeß befinden sich dort Erweiterungen, die eine Programmierung ortsabhängiger Informationssysteme stark vereinfachen, da sie bessere Unterstützung für die Oberflächen- und Netzprogrammierung bieten:

- **JSR 46: Foundation Profile** – hiermit sollen Netzeigenschaften mobiler Geräte in einer API näher beschrieben werden.
- **JSR 62: Personal Profile Specification** – diese Spezifikation bietet in mobilen Umgebungen die Speicherung von Informationen zu Internetverbindungen und deren Eigenschaften und eignet sich somit für die Entwicklung verteilter Informationssysteme.

- **JSR 75: PDA Optional Packages for the J2ME-Platform**– mit Hilfe dieses Paketes soll die erweiterte Nutzung von PDA-Fähigkeiten ermöglicht werden. Dazu gehören Dateiaustausch und -zugriff sowie die Verwaltung persönlicher Daten, womit eine bessere P2P-Kommunikation ermöglicht werden könnte.

- **JSR 120: Wireless Messaging API** – diese Spezifikation soll bestehende mobile Profile von J2ME-Umgebungen um die Fähigkeit zur Kommunikation mit Funknetzen erweitern. Damit ist die Schaffung von Anwendungen möglich, die mehrere Netzverbindungen aufbauen und während ihrer Laufzeit benutzen.

- **JSR 129: Personal Basis Profile Specification** – die Verwendung leichtgewichtiger grafischer Komponenten (AWT) soll mit dieser Spezifikation auf mobilen Geräten mit Netzanbindung ermöglicht werden. Der Standardisierungsvorschlag *JSR 217: Personal Basis Profile Version 1.1 for the J2ME Platform* passt dieses Paket an die neu verfügbare Java 1.4 API an und bietet somit Entwicklern die Möglichkeit, mit einer aktuellen Java-Version mobile Anwendungen zu entwickeln.

- **JSR 135: Mobile Media API** – dieses Paket soll für mobile Geräte die erweiterte Nutzung von Multimedia (Audio und Video) ermöglichen. Dabei wird auf die besonderen Eigenschaften der mobilen Geräte eingegangen, indem spezielle Methoden zur Steuerung von Multimedia-Datenströmen zur Verfügung gestellt werden.

- **JSR 169: JDBC Optional Package for CDC/Foundation Profile** – durch dieses Paket werden mobile Geräte in der Lage sein, speziell auf ihr Verbindungsprofil angepasst, mit Datenbanken zu kommunizieren. Dazu wird eine Erweiterung der bestehenden Bibliotheken zum Zugriff auf Datenbanken mit Java (JDBC) zur Verfügung gestellt.

- **JSR 172: J2ME Web Services Specification** – der Zugriff auf Web-Services von mobilen Geräten aus wird mit dieser Spezifikation ermöglicht. Dadurch können die mobilen Geräte komplexe Infrastrukturen und Dienste benutzen und eignen sich für die Umsetzung verteilter Informationssysteme.

- **JSR 179: Location API for J2ME** – diese Spezifikation unterstützt die Verwendung von ortsabhängigen Diensten dahingehend, dass das mobile Gerät Befehle zur Lokalisierung ausführen kann und über entsprechende Adapter Positionsdaten abliefern kann. Damit existiert die Möglichkeit, GPS-Adapter mit Java anzusprechen und auf mobilen Geräten zu benutzen [47].

- **JSR 190: Event Tracking API for J2ME** – dieser Standardisierungsvorschlag erleichtert die Entwicklung von Anwendungen auf mobilen Geräten, da die vom Benutzer ausgelösten Ereignisse über ein definiertes Protokoll an einen Server geliefert werden können, der die Anwendungslogik entsprechend steuert.

Da sich der derzeit verfügbare Funktionsumfang von J2ME eher zur Entwicklung von einfachen Anwendungen und Spielen eignet, wurde der Ansatz eines getrennten Programms zur P2P-Kommunikation verworfen. J2ME stellt in der aktuellen Version zu wenig Funktionen im Bereich der Netz- und Oberflächenprogrammierung zur Verfügung, um ein verteiltes Informationssystem darüber realisieren zu können. Bedingt durch Javas Ansatz einer virtuellen Maschine – genannt K Virtual Machine (KVM) – ist es zusätzlich nicht möglich, niedrigere Schichten der mobilen Geräte anzusprechen. Dazu zählen bei Mobiltelefonen beispielsweise Funktionen zum Annehmen von Telefongesprächen oder Aufbau von Wählverbindungen.

Sind die erwähnten Standardisierungsvorschläge in den Funktionsumfang von J2ME integriert, sollte der P2P-Ansatz wieder aufgegriffen werden.

4.2 Datenbankmodell

Das hier als Grundlage verwendete Datenmodell wurde von Katja Löffler in ihrer Diplomarbeit [33] entwickelt. Es verfügt unter anderem über folgende Eigenschaften:

- Sehenswürdigkeiten sind in Klassen strukturiert

- Informationen sind Themenbereichen zugeordnet

- Informationen sind hierarchisch gespeichert

Informationen sind die textuellen Inhalte, denen Bilder zugeordnet sind, wobei jedes einzelne Bild zu einer Sehenswürdigkeit gehört. Um die einzelnen Informationen zu strukturieren, sind diese Themenbereichen zugeordnet. Die Klassifikation von Sehenswürdigkeiten und Informationen hat Auswirkungen auf die Definition eines Benutzerprofils. Dieses besteht aus einer Menge von Themen und Sehenswürdigkeitsgruppen, wobei die Themen die Art der Informationen eingrenzen, an denen ein Benutzer interessiert ist. In Sehenswürdigkeitsgruppen sind alle Sehenswürdigkeiten zusammengefasst, an denen der Benutzer interessiert ist. Ein Vorteil dieser Trennung ist, dass der Benutzer sein Profil auf einfache Art und Weise ändern kann. Beispielsweise ist es möglich, sich nicht für den Themenbereich *barocke Architektur* zu interessieren, aber dennoch Informationen zu der Sehenswürdigkeit *barockes Schloss* zu erhalten.

Da das Profil somit nicht aus einer langen Liste besteht, sondern thematisch untergliedert ist, besteht eine geringere Gefahr, den Benutzer bei der Auswahl zu überfordern. Stattdessen wird ein stufenartiges Bearbeiten des eigenen Profils ermöglicht.

Die hierarchische Speicherung von Informationen bedeutet, dass zu einer Sehenswürdigkeit verschiedene Informationen in der Datenbank existieren. Besucht der Benutzer eine Sehenswürdigkeit mehrmals, wird ihm jeweils eine neue Information angezeigt, die sich in einer anderen Hierarchiestufe befindet. Dieser Vorgang wiederholt sich, bis alle Informationen zu einer Sehenswürdigkeit angezeigt wurden.

Die Tabellenstruktur wurde auf das hier entwickelte TIP-System angepasst und erweitert. Im Bereich der Benutzerverwaltung wurden neue Tabellen hinzugefügt. Die zu Sehenswürdigkeiten vorliegenden Bilder wurden in der Ursprungsversion [33] durch Programmlogik ausgewählt. Jetzt werden verkleinerte Vorschaubilder und die Originale unter Angabe ihrer Position (URL) und ihrer Eigenschaften (Höhe, Breite) in der Datenbank gespeichert. Somit können sie unter korrekter Angabe ihrer Größe in die Anwendung eingebunden werden. Bisher wurde die Bildgröße nicht mit in der HTML-Datei gespeichert, wodurch sich der Seitenaufbau verändert, wenn der Browser das Bild in den vorher dargestellten Text einfügt. Durch Angabe der Bildgröße wird vermieden, dass sich die Darstellung beim Laden der Seite mehrfach verändert und unter Umständen den Benutzer irritiert.

4.3 Anforderungsprofil

Die in dieser Diplomarbeit entwickelte Software basiert auf folgenden Annahmen:

- Verwendung offener Standards
- Benutzung frei verfügbarer Rahmenwerke
- Klientengeräte mit Internetbrowser
- Darstellung für eine Auflösung von 640 x 480 Pixeln ausgelegt
- serverzentrierte Architektur mit leichtgewichtigen Klienten

In den folgenden Abschnitten werden die Grundlagen für die in Kapitel 5 beschriebene Umsetzung gelegt. Dabei wird hier die Architektur der Software näher betrachtet. Diese resultiert im folgenden Kapitel in dem konkreten Entwurf der TIP-Anwendung.

4.4 Entwurfsmuster und MVC-Architektur

Die Benutzung von Entwurfsmustern in der objektorientierten Programmierung wurde durch das Buch „Entwurfsmuster – Elemente wiederverwendbarer objektorientierter Software“ von Gamma et.al. [11] geprägt, dessen Autoren oftmals auch als „Gang of Four“ (GoF [59]) bezeichnet werden.

Die Idee, bestehende Lösungsansätze als Entwurfsmuster allgemein zu spezifizieren, um sie wiederverwenden zu können, ist ein wissenschaftliches Prinzip, das sich, aus der Architektur kommend, in der Softwareentwicklung langsam durchsetzt. Vorher war die Implementierung einer Architektur stets eng an die eingesetzte Hardware und Software gebunden und beschränkte sich in ihrer Anwendbarkeit auf diesen Kontext. Die mit objektorientierten Sprachen gewonnene Fähigkeit, Programmzustände und -logik in Klassen zu kapseln, vereinfacht die Umsetzung allgemeiner Lösungen und erhöht deren Fähigkeit zur Wiederverwendung in anderen Programmsystemen.

Ein Entwurfsmuster besteht aus einer Problembeschreibung in einem bestimmten Kontext des Programmentwurfs und seiner Lösungsbeschreibung. Diese Lösung lässt sich in allgemeinen Modellierungssprachen wie Unified

Modeling Language (UML) darstellen oder beispielhaft an Quellcodeausschnitten darlegen. Für Softwareentwickler sollen somit einfach zu verwendende Implementierungsansätze geschaffen werden, die es ermöglichen, dass beim Entwurf einer komplexen Anwendung eine bessere Architektur ein klareres Design zur Folge hat. Das Design eines Softwaresystems ist die Implementierung einer Architektur, die auf einer abstrakten Ebene stattfindet.

Das hier entwickelte Softwaresystem arbeitet mit folgenden Entwurfsmustern (design pattern), die kurz erläutert werden:

- Singleton-Pattern
- Factory-Pattern
- Model-View-Controller-Pattern (MVC)
- Iterator-Pattern

Das Singleton-Pattern ist ein Hilfsmittel bei der Erzeugung von Objektinstanzen, mit dem man sicherstellt, dass von einem Objekt jeweils nur ein Exemplar während der Laufzeit existiert. Auf dieses Exemplar kann man von der gesamten Anwendung aus zugreifen. Dies ist beispielsweise sinnvoll, wenn die Klasse häufig benutzte Funktionen kapselt oder eine aufwändige Initialisierung vornimmt, um dann eine gewisse Geschäftslogik zur Verfügung zu stellen. Um dies technisch umzusetzen, wird der Konstruktor der Klasse nach außen unsichtbar, während eine statische Methode zur Verfügung steht, die beim ersten Aufruf ein Objekt der eigenen Klasse erzeugt und eine Referenz darauf bei wiederholten Aufrufen zurückgibt. Somit ist sichergestellt, dass nur ein Objekt erzeugt wurde, welches von der gesamten Anwendung - durch Aufruf der statischen Methode - benutzt werden kann (siehe Abbildung 4.1). Das Factory-Pattern wird bei der Instantiierung einer Klasse benutzt. Normalerweise wird eine Instanz einer Klasse durch Benutzung des Konstruktors erzeugt und der Typ der Klasse ist festgelegt. Will man im Programmablauf ein Objekt erzeugen, das eine abstrakte Schnittstelle implementiert, so kann durch Benutzung des Factory-Entwurfsmusters dieses Objekt an einer unbekannten Stelle erzeugt werden, während man als Benutzer dieses Objekt einfach nur benutzt.

Im wirklichen Leben will man beispielsweise nach einer sportlichen Aktivität, weil man durstig ist, etwas trinken. Dieser Ablauf beschreibt eine abstrakte Schnittstelle. Das nach dem Sport erhaltene Getränk stellt die erhaltene Objektinstanz dar. Dabei ist es egal, ob es sich um Wasser, Tee

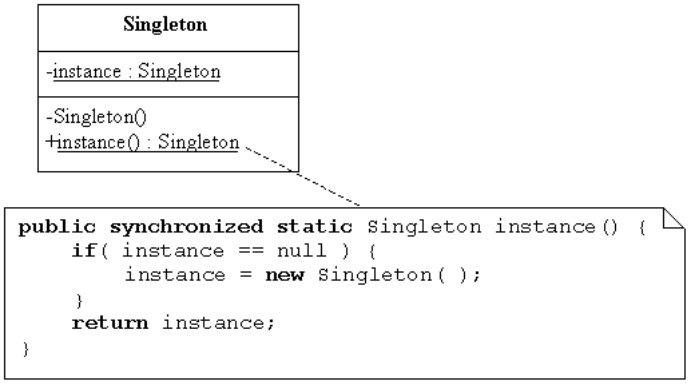

Abbildung 4.1: Singleton-Pattern als UML-Diagramm

oder ein Fruchtsaftgetränk handelt. Alle diese speziellen Getränke genügen der abstrakten Schnittstelle nach-dem-Sport-trinkbar. Der Kühlschrank fungiert dabei als Fabrik. Häufig ist es sinnvoll, von einer Fabrik nur ein Exemplar während der Laufzeit der Anwendung zu haben, da man sonst bildlich gesprochen für jeden Getränkewunsch einen neuen Kühlschrank anschaffen würde. Diese Arbeitsersparnis wird durch Verwendung des Singleton-Musters erreicht. Damit wird zusätzlich die durch Objektorientierung erwünschte hohe Wiederverwendbarkeit erreicht.

Das Produkt der Fabrik gehorcht einer abstrakten Schnittstelle. Diese wird vom nutzenden Klient als Typ angegeben und ermöglicht somit einen Zugriff auf das entsprechende Objekt. Zur Laufzeit wird dabei eine konkrete Objektinstanz von der Fabrik erzeugt, die der abstrakten Schnittstelle genügt. Die Erzeugung der konkreten Objektinstanz kann dabei auf verschiedenen Wegen geschehen – als Fernaufruf (RMI) oder lokal. Für den Benutzer bleibt dieser Vorgang durch die Benutzung der Fabrik transparent. Abbildung 4.2 zeigt die theoretische Struktur des Factory-Musters. Dabei benutzt der Client sowohl die Fabrik als auch das Produkt in einer abstrakten Art und Weise, das heißt, ohne zu wissen, welchen Typ die konkreten Implementierungen der Klassen haben. Der einzige Zugriff auf die Fabrik geschieht durch Aufruf der *createProduct()*-Methode, die als Schnittstelle zur Objekterzeugung definiert ist. Ein weiteres Entwurfsmuster organisiert die Elemente einer grafischen Oberfläche so, dass diese sich in drei Komponenten aufteilen lassen:

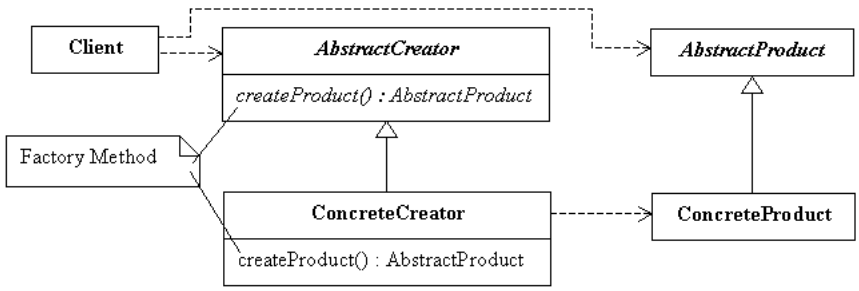

Abbildung 4.2: Factory-Pattern in der UML-Darstellung

1. **Model – Modell:** beinhaltet den Zustand der Anwendung
2. **View – Sicht:** stellt das Modell visuell dar
3. **Controller – Steuerung:** verarbeitet Benutzereingaben

Die Model-View-Controller-Komponenten (MVC) der Anwendung sind voneinander unabhängig implementiert. Das Modell der Anwendung kapselt die Daten der Anwendung, in die das Verhalten und der Zustand der Anwendung kodiert sind. Beispielsweise kann die Anwendung eine Anmeldung an einem Softwaresystem nur durchführen, wenn die nötigen Anmeldedaten – oft Benutzername und Passwort – eingegeben sind. Diese stellen das Modell der Anwendung dar und steuern indirekt deren Zustand, da ohne die Eingabe der Werte der Anmeldevorgang keinen Sinn macht. Die Sicht der Anwendung kann unter Einbeziehung verschiedener Techniken umgesetzt werden. Denkbar ist hier eine native grafische Oberfläche oder die Verwendung von dynamisch generierten HTML-Seiten. Die Sicht besteht sowohl aus Komponenten zur reinen Anzeige von Daten als auch Komponenten zur Eingabe von Daten wie beispielsweise Formulare. Ein Vorteil des MVC-Musters ist, dass die Sicht durch die Unabhängigkeit von der Logik und dem Datenbestand der Anwendung beliebig programmiert werden kann. Dies ermöglicht eine hohe Wiederverwendbarkeit von Komponenten, wenn man eine Anwendung sowohl internetfähig als auch als einzelnes Programm ablaufbar machen möchte. Die Steuerungsschicht übernimmt die vom Benutzer gemachten Eingaben (Maus- und Tastaturereignisse) und übersetzt diese durch Änderung des Modells in einen neuen Zustand der Gesamtanwendung.

Ein Beispiel einer MVC-Konstruktion ist in Abbildung 4.3 zu sehen. Dabei dient ein Zähler als zu Grunde liegendes Modell der Anwendung.

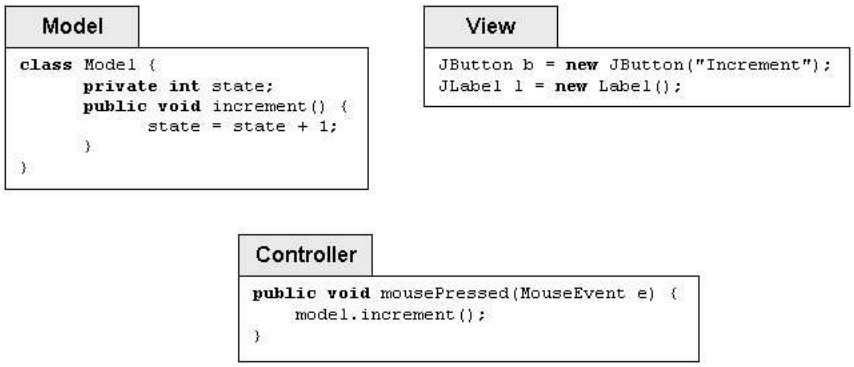

Abbildung 4.3: Komponenten einer MVC-Architektur in Java

Dieser wird mit Hilfe der *mousePressed()*-Methode gesteuert. Die Ansicht der Anwendung besteht hier aus einem Eingabeknopf (JButton) und einer Textzeile (JLabel), die ihren Zustand in Abhängigkeit der Benutzereingaben verändern. Diese Zustandsänderung wird durch einen Mausklick oder eine Tastatureingabe erreicht und veranlasst die Steuerungsschicht, den Zähler / das Modell zu verändern. Man kann sich hier die Ansicht der Anwendung auch als physikalischen Knopf vorstellen, der einen elektrischen Impuls an die Anwendungssteuerung weitergibt. Dieser würde in gleicher Art und Weise unter Benutzung der Steuerungsschicht das Modell verändern und den Zähler inkrementieren. Das Struts-Rahmenwerk [27] implementiert eine Umsetzung des MVC-Entwurfsmusters. Auf die Details der Verwendung von Struts wird im Abschnitt 5.4.2 eingegangen.

Ein weiteres benutztes Entwurfsmuster vereinfacht den Zugriff auf Mengen von Objekten, die in so genannten Containern oder Kollektionen zusammengefasst werden. Dem Benutzer bieten diese dann vereinfachte Zugriffsmethoden auf die als Inhalte abgelegten Objekte. Mengen von Obst oder Gemüse kann man beispielsweise gruppieren, was der Erzeugung von Kollektionen oder Containern entspricht. Auf dem Apfel-Container kann es Methoden zum Einkaufen geben, die auf einem einzelnen Apfel mühsamer auszuführen sind. Will man Äpfel kaufen, kann man jeden Apfel einzeln auswählen oder eine Tüte mit Äpfeln kaufen. Die Tüte entspricht dabei dem Container, wobei nicht jeder Apfel einzeln gekauft werden muss. Zudem bietet die Tüte durch ihre Klarsichtfolie die Möglichkeit, die Anzahl der Äpfel zu ermitteln. Solche Operationen sind auf Containern sinnvoll, da die Be-

stimmung der Anzahl der Elemente auf einem einzelnen Objekt trivial ist. In der Programmierung bieten sich Container auch zum Transport von referentiell unabhängigen Objekten an. Diese können somit an einen Benutzer ausgeliefert werden, der dann lokal die Objektmenge bearbeiten kann. Ein Vorteil dieser Lösung ist, dass die Methoden zur Erzeugung des Objektcontainers dem Benutzer unbekannt sein können. Bei der Verwendung von Datenbanken lässt sich das Kollektionen-Konzept sehr sinnvoll einsetzen, da mit einer geöffneten Datenbankverbindung alle Elemente ausgelesen werden können und dann insgesamt an den Benutzer geschickt werden. Der naive Ansatz des Öffnens einer Datenbankverbindung, Auslesen eines Objekts und Schließen der Datenbankverbindung entfällt somit.

Möchte man alle Objekte, die sich in einem Container befinden, nacheinander benutzen, ist es sinnvoll, einen so genannten Iterator einzusetzen. Dieser gestattet über vorher festgelegte Methoden den Zugriff auf eine darunter liegende Objektmenge und der Vorteil für den Benutzer liegt darin, dass dieser nicht wissen muss, wie die Objektmenge technisch organisiert ist. Der Iterator bietet Methoden zum sequentiellen Abarbeiten von Objektmengen. Abbildung 4.4 zeigt das Zusammenspiel von Iterator und Container. Der Container wird hier als Aggregat bezeichnet. Iterator und Aggregat sind abstrakte Schnittstellen, die erst durch konkrete Implementierungen in der Programmierung benutzbar sind. Beispielsweise können Aggregate verkettete Listen, einfache Listen oder mehrdimensionale Felder sein. Auf diesen kann es verschiedene Iteratoren geben, die beispielsweise alle Elemente ungeordnet oder nach bestimmten Regeln geordnet abarbeiten. In der Abbildung 4.4 wird zur Erzeugung eines Iterators das Factory-Pattern eingesetzt und zeigt den hohen Grad der Nutzung und Verwendbarkeit der Entwurfsmuster.

Das entwickelte TIP-System benutzt alle hier vorgestellten Entwurfsmuster und zeigt deren Anwendbarkeit in verteilten Informationssystemen.

4.4.1 UML-Darstellung

Zur vereinfachten Darstellung komplexer Softwaresysteme wird häufig auf Modellierungssprachen oder Diagramme zurückgegriffen. Die Verwendung von UML ermöglicht die Darstellung von Sachverhalten mit Hilfe verschiedener Diagrammtypen. Dabei kommen funktionale, dynamische und objektbasierte Techniken zum Einsatz. Funktionale Modelle beschreiben die Funktionalität eines Programms aus Sicht des Benutzers, während dynamische

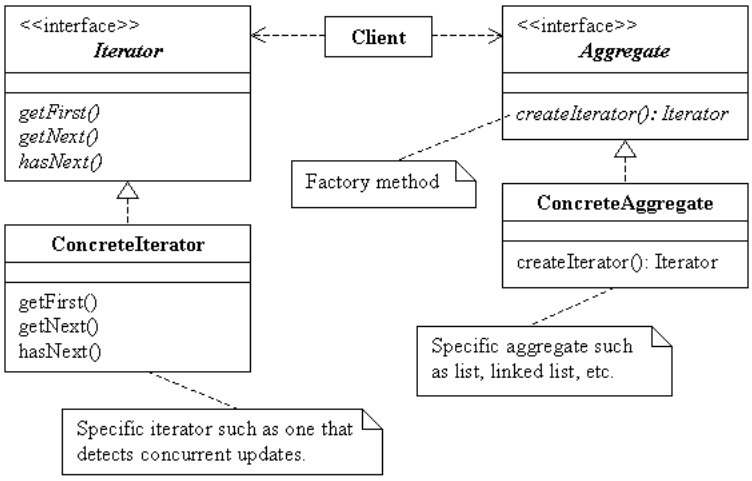

Abbildung 4.4: Zusammenspiel von Iterator und Collection zum Zugriff auf Mengen von Objekten

Modelle das interne Verhalten der Systeme aufzeigen. Objektbasierte Modellierungstechniken zeigen die Struktur des Gesamtsystems basierend auf Klassen, Schnittstellen und Vererbungsbeziehungen. Zusätzlich werden Attribute und Methoden der jeweiligen Klassen in Diagrammen visualisiert.

Zur Beschreibung der Funktionalität einer Software empfiehlt sich die Darstellung in so genannten Anwendungsfalldiagrammen (use-case diagram), die aufzeigen, was ein Benutzer (Akteur) am System machen kann, abstrahieren aber, wie dies systemintern geschieht. Die einzelnen auszuführenden Aktionen können voneinander abhängen oder Vererbungshierarchien bilden. Im folgenden werden beispielhaft zwei Anwendungsfalldiagramme des touristischen Informationssystems gezeigt und erläutert, die sich an die UML-Syntax [15] anlehnen. Das in Abbildung 4.5 gezeigte Use Case-Diagramm zur Anmeldung am touristischen Informationssystem beinhaltet die Eingabe von Logindaten und einer aktuellen Position. Diese kann entweder als ein Koordinatenpaar oder als textuelle Bezeichnung des aktuellen Standorts eingegeben werden. Diese Eingaben werden vom Informationssystem validiert und führen bei Erfolg zum Benutzermenü, in dem weitere Aktionen ausführbar sind. Abbildung 5 (S.128) zeigt die browserbasierte Oberfläche bei der Anmeldung am System. Das Bild zeigt einen Fehler in der Eingabe, der beim Validieren der einzelnen Eingabefelder gemeldet wird. Nach erfolgreicher Anmeldung kann, wie in Abbildung 4.6 gezeigt, die Be-

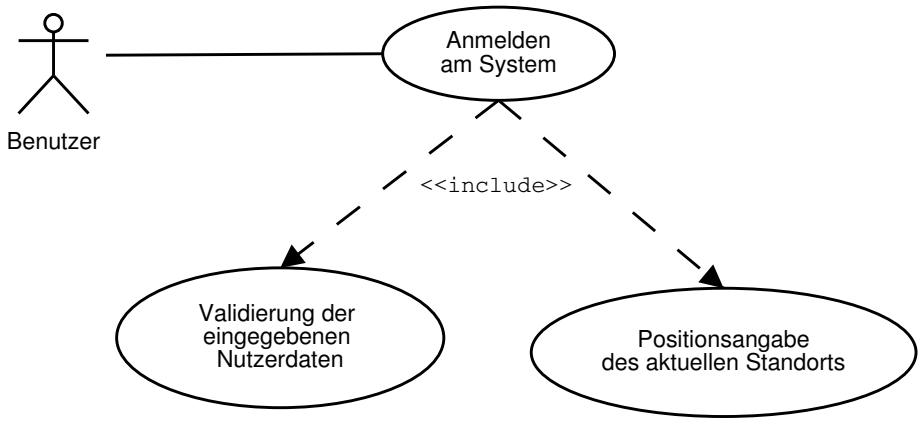

Abbildung 4.5: Use Case-Diagramm: TIP-Anmeldung

arbeitung von Profilinformationen ausgeführt werden. Da sich das Profil in zwei Teile – das Sehenswürdigkeitsgruppen- und Themenprofil – untergliedert, müssen für die Bearbeitung verschiedene Schritte ausgeführt werden. Das Diagramm zeigt auch, dass zwischen den einzelnen Aktionen Abhängigkeiten bestehen. So kann das Benutzerprofil nur nach erfolgreicher Anmeldung geändert werden und beinhaltet die Bearbeitung des Themen- und Sehenswürdigkeitsgruppenprofils. Im Anhang befinden sich Abbildungen des touristischen Informationssystems, die die Bearbeitung des Sehenswürdigkeitsgruppenprofils (Bild 7, S. 130) und die Anzeige eines Themenprofils (Bild 8, S. 131) zeigen.

4.4.2 Logische Struktur der Anwendung

Aus Abbildung 4.7 lässt sich die grobe Struktur des TIP-Systems ableiten, das sich in zwei Teile untergliedert: Komponenten zur Datenanzeige und -verwaltung sowie die eigentliche Logik des Touristeninformationssystems. Diese unterteilt sich wiederum in die Steuerungslogik und die eigentliche Geschäftslogik der Anwendung, worunter man den genauen Ablauf der einzelnen Aktionen des TIP-Systems versteht. Zur genauen Steuerung des Anwendungszustands und zur etwaigen Fehlerbehandlung müssen spezielle Fehlerkontrollmechanismen implementiert werden, die unabhängig von der Geschäftslogik sind, welche in Aktionsklassen des verwendeten Rahmenwerkes gekapselt ist. Neben der Kommunikation mit der Datenbank muss

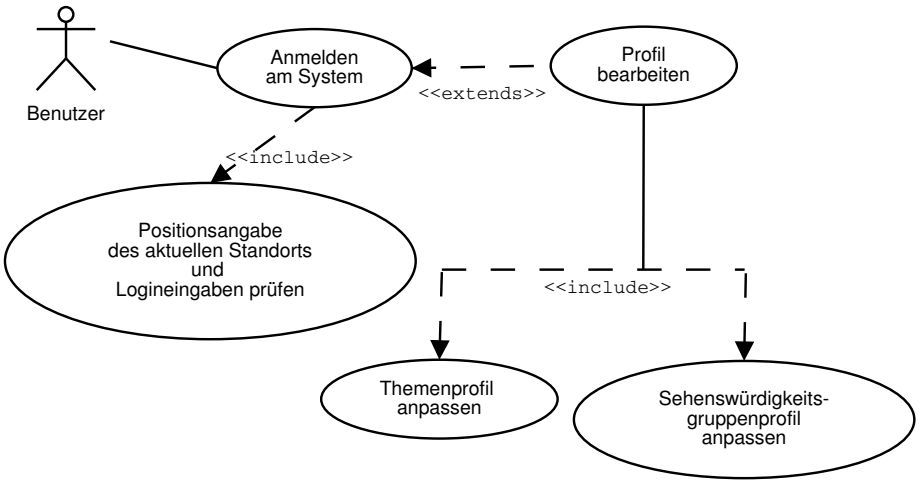

Abbildung 4.6: Use Case-Diagramm: TIP-Profilbearbeitung

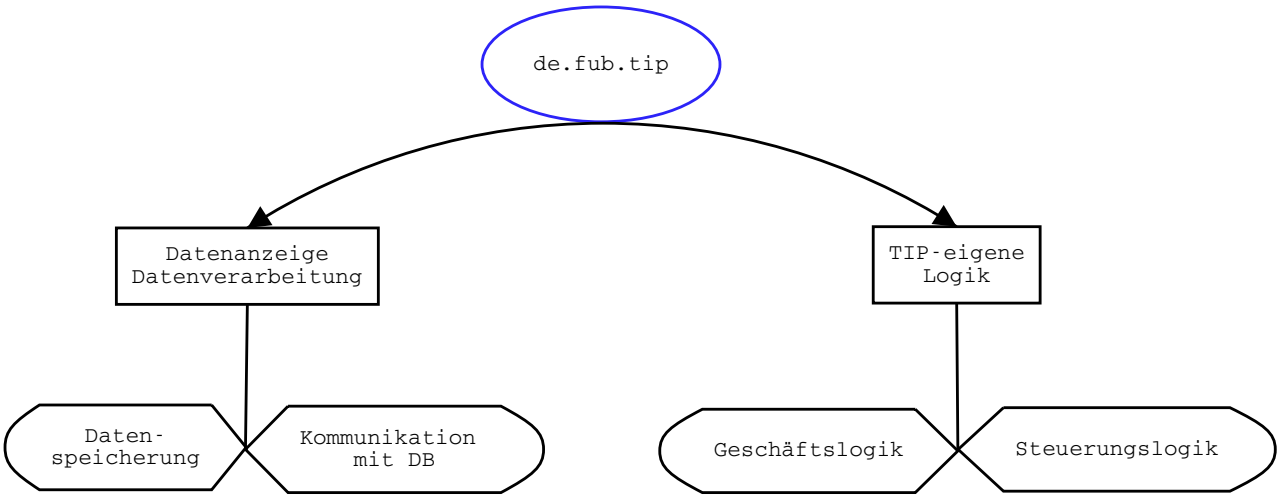

Abbildung 4.7: Logische Struktur des TIP-Systems

die Datenverarbeitungskomponente der Anwendung auch Persistenzmechanismen zur Verfügung stellen. Diese ermöglichen eine von der Datenbank unabhängige Speicherung der jeweiligen Inhalte. TIP benutzt dazu so genannte Wertobjektklassen, die im Abschnitt 5.4.3 detaillierter beschrieben werden.

Zur inhaltlich klareren Gliederung eines Softwaresystems lassen sich Anwendungsteile in so genannten Paketen (packages) ablegen. Diese Pakete korrespondieren mit dem Namen von Verzeichnissen und sollen nach den Richtlinen der Firma Sun [54] eindeutig sein. Zur besseren Unterscheidung enthält die erste Komponente den Landesnamen nach dem ISO-Standard 3166 kodiert. Das Hauptverzeichnis der hier entwickelten Anwendung liegt im Paket *de.fub.tip*, wobei nach dem Land das Projekt TIP der Freien Universität Berlin den Paketnamen eindeutig macht. Darunter liegende Pakete kapseln die verwendeten Rahmenwerke, den Datenbankzugriff und die Geschäftslogik der Anwendung. Diese Grobeinteilung passt sich den Vorgaben des MVC-Entwicklungsmusters an (siehe Abschnitt 4.4).

Die HTML-Vorlagen, die durch den Server bei der Auslieferung an die Klienten angepasst werden, liegen an einem von der Programmlogik unabhängigen Ort. Das Wurzelverzeichnis dieser Dateien heißt `pages`. Die Unterverzeichnisse sind wie die Pakete der Anwendung inhaltlich gegliedert. Es gibt Verzeichnisse, die Elemente zur Eingabe (`pages/input`) oder zur Datenanzeige (`pages/data/anzeige`) enthalten. Die Konfigurationsdateien befinden sich ebenfalls in separaten Verzeichnissen und ermöglichen so, bei etwaigen Migrationen oder strukturellen Änderungen, eine schnelle Anpassung der Anwendung. Der komplette Verzeichnisbaum der TIP-Anwendung ist in Abbildung 15 (S. 138) zu sehen.

Zum einfacheren Verständnis zeigt Abbildung 4.8 den schematischen Aufbau des TIP-Systems. Ein mobiler Benutzer meldet sich unter Angabe seiner aktuellen Position am touristischen Informationssystem an, das seine Dienste nach außen transparent zur Verfügung stellt. Der Klient muss über ein Netz verbunden sein, das seine Teilnehmer mittels TCP/IP unter Verwendung von HTTP ansprechen kann. Im Systemkern werden verschiedene Datenspeicher abgefragt. Dies geschieht durch eine vorher konfigurierte Anwendungslogik. Hier werden diese Komponenten durch eigene Java-Klassen umgesetzt und sind im Bild unten links lila umrandet. Zur eigentlichen Anzeige werden die aus den Datenspeichern gelesenen Daten dynamisch in Dateivorlagen – in der Abbildung unten rechts grün umrandet – eingebunden. Zur flexibleren Dar-

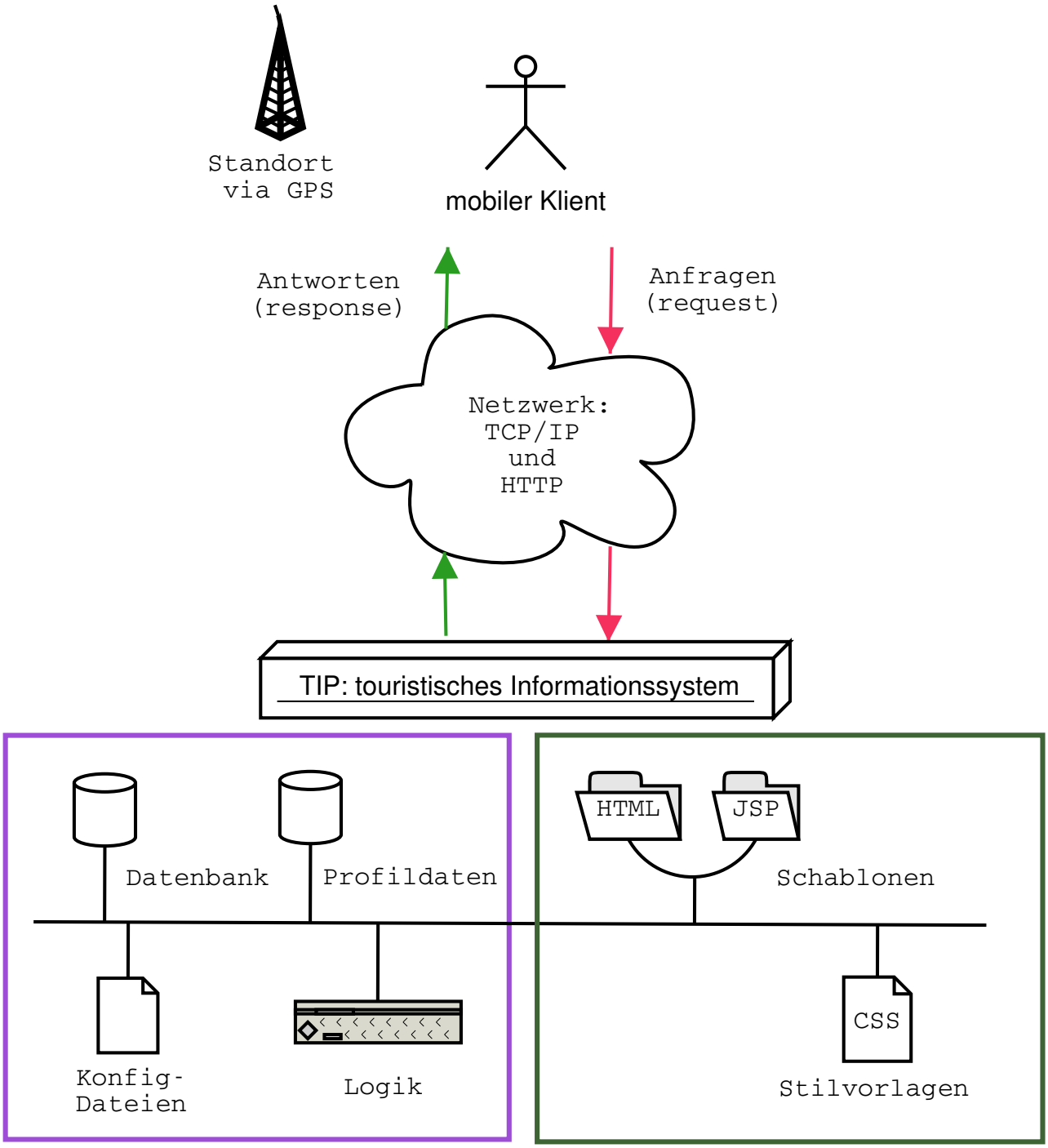

Abbildung 4.8: Schematischer Aufbau des TIP-Systems

stellung der eigentlichen Inhalte speichert der TIP-Server Stilvorlagen (CSS), die bei der Auslieferung an den Klienten zu einer HTML-Datei zusammengefügt werden. Das in Kapitel 1 vorgestellte Anwendungsszenario wird hiermit technisch präzisiert. Weitere Details zur Umsetzung der TIP-Software finden sich im folgenden Kapitel 5.

4.5 Zusammenfassung

Dieses Kapitel hat die beim Softwareentwurf aufgetretenen Schwierigkeiten erläutert, welche zu einer inhaltlichen Änderung der Diplomarbeit geführt haben. Der Ansatz, durch eine separate Anwendung direkte P2P-Kommunikation zu ermöglichen, wurde verworfen. Stattdessen wird eine Client-Server-basierte Anwendung entwickelt, deren Schwerpunkt in der Erweiterbarkeit besteht. Dazu wird auf Entwurfsmuster der objektorientierten Programmentwicklung zurückgegriffen. Die Anwendung lässt sich als verteiltes Informationssystem in Komponenten zur Datenverwaltung und -anzeige zerlegen. Diese steuern den aktuellen Anwendungszustand, bieten Zugriff auf Daten aus der Datenbank und visualisieren diese durch Erzeugung dynamisch generierter HTML-Seiten.

Im Kontext offener Standards und frei erhältlicher Rahmenwerke erläutert das folgende Kapitel die bei der Umsetzung der TIP-Anwendung verfolgten Strategien und geht auf Probleme bei der Arbeit ein.

Kapitel 5

Implementierung

Nachdem im vorigen Kapitel die konzeptionellen Grundlagen für die Entwicklung des touristischen Informationssystems gelegt wurden, beschäftigt sich dieses Kapitel mit der konkreten Umsetzung. Dabei wird vorwiegend auf die serverseitig verwendeten Komponenten eingegangen.

Bei dem implementierten Informationssystem handelt es sich um eine Mehr-Schichten-Architektur [14]. Diese unterteilt eine Client-Server-Anwendung in folgende Komponenten:

- GUI-Schicht
- Fachkonzept-Zugriffsschicht
- Fachkonzeptschicht
- Datenhaltungs-Zugriffsschicht
- Datenhaltungsschicht

Als Ergänzung zur Drei-Schichten-Architektur, die nur aus einer GUI-Schicht sowie Fachkonzept- und Datenhaltungsschicht besteht, bietet die Mehr-Schichten-Architektur zusätzliche Zwischenschichten, die Aufgaben und Abhängigkeiten kapseln. Die GUI präsentiert die Daten, die durch die Fachkonzept-Zugriffsschicht von der Fachkonzeptschicht gelesen wurden. Die Datenhaltungsschicht gibt ihre Daten an eine separate Datenhaltungs-Zugriffsschicht ab. Diese kann dann den fachlichen Zustand der Objekte aus der Fachkonzeptschicht an die Datenhaltungsschicht weitergeben.

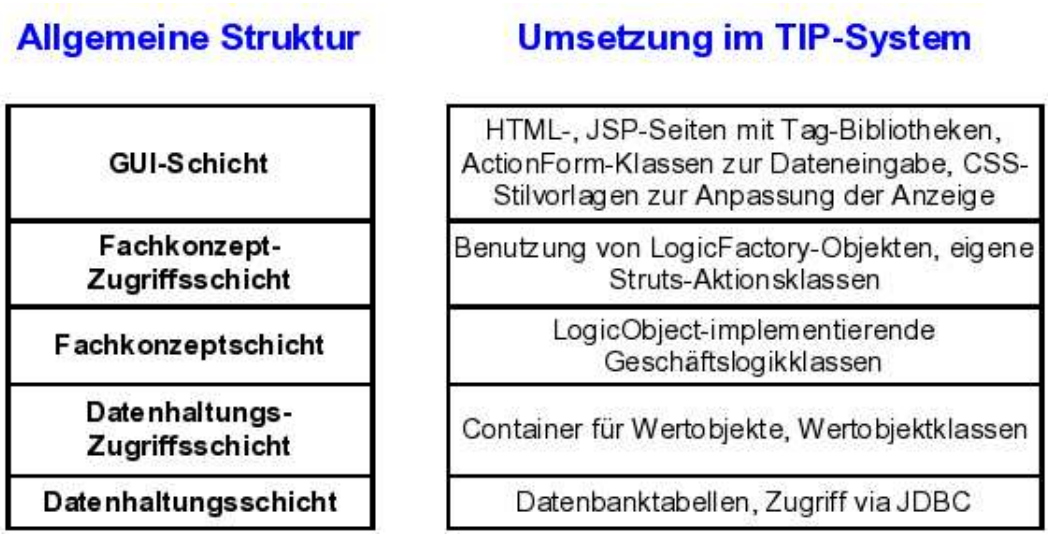

Abbildung 5.1: Mehr-Schichten-Architektur im TIP-System

Die Verwendung einer mehrschichtigen Architektur realisiert die Entwurfsziele Wiederverwendbarkeit, Portabilität und Wartbarkeit. Dies wird dadurch erreicht, dass die einzelnen Schichten über klar definierte Schnittstellen miteinander arbeiten und ihre innere Funktionsfähigkeit voreinander kapseln, die bei Bedarf ohne Verletzung der Abhängigkeiten neu implementiert werden kann.

In Abbildung 5.1 werden die Komponenten der TIP-Anwendung den einzelnen Schichten einer mehrschichtigen Architektur zugeordnet. Die Übersicht soll als Grobüberblick dienen; die folgenden Absätze gehen detailliert auf die einzelnen Elemente ein. Die Serverseite der Anwendung baut auf Datenbank- und Anwendungsserver, die die Datenhaltungs- und Fachkonzeptschicht bilden.

5.1 Datenbankserver

Beim Entwurf verteilter Informationssysteme empfiehlt sich der Einsatz relationaler Datenbanken, da diese über geografische Erweiterungen verfügen und einen hohen vorhandenen Funktionsumfang für die Anwendungsentwicklung bieten.

Open-Source-Datenbanksysteme mit hohem Verbreitungsgrad sind MySQL und PostgreSQL. MySQL [39] ist stark verbreitet, weil es mit anderen Open-Source-Technologien kombiniert werden kann. Diese Konfiguration wird als *LAMP* - Linux, Apache Webserver, MySQL-Datenbank, PHP/Perl/Python-Skriptsprache - bezeichnet.
Für den Einsatz in verteilten komplexen Informationssystemen eignet sich MySQL nur bedingt, da es über keine Transaktionssteuerung verfügt und keine kostenfreien geografischen Erweiterungen bietet. Auch auf Grund dieser inhaltlichen Schwächen wird hier als Datenbanksystem PostgreSQL eingesetzt. Das Postgres-System ist komplett kostenfrei und kann auf den meisten Betriebssystemen installiert werden; dies sind Linux, Mac OS X, Solaris, Unix und Windows. Dadurch besteht keine Einschränkung bezüglich der Hardwareauswahl. Zusätzlich zu den geografischen Erweiterungen bietet die Postgres-Datenbank als objektrelationales Datenbankmanagementsystem (ORDBMS) [10, 25] diverse Möglichkeiten der Arbeit an der Datenbank:

- **psql** – direkter Zugriff auf die Datenbank per Text-Konsole
- **PL/pgSQL** – Verwendung einer datenbanknahen Abfrage- und Programmiersprache
- **ODBC/ JDBC** – plattform- und programmiersprachenübergreifender Zugriff über festgelegte Schnittstellen (API)
- **Skript-/Programmiersprachen** – Die Unterstützung für Zugriffe mittels perl oder C kann direkt in die Datenbank kompiliert werden.

Der implementierte Funktionsumfang des Datenbanksystems macht Postgres zu einer funktional mächtigen Lösung. Der Zugriff über eine Text-Konsole sichert ab, dass im Vergleich zu anderen Systemen der Zugriff auf die Daten nicht eng an die Anwendung selbst gekoppelt ist. Des weiteren lässt sich das Datenladen sowohl durch getrennte Anwendungen als auch direkt per Konsole durchführen und sichert somit eine Unabhängigkeit von den Anwendungssoftwareanbietern. Die vielfältigen Zugriffsmöglichkeiten erleichtern auch die Wartung und Pflege der Daten, da diese von verschiedenen Ebenen aus sichtbar sind. Der Endanwender wird stets über Programme auf die Daten zugreifen, während der Dienstanbieter direkt an der Datenbank arbeiten kann. Diese Kapselung sorgt für einen gesicherten Umgang mit den Daten, da nicht jeder Benutzer über gleiche Zugriffsrechte verfügt.

Das Postgres-Datenbanksystem bietet mit der Postgis-Erweiterung [44] die Möglichkeit, geografische Daten direkt in der Datenbank zu verarbeiten. Postgis stellt besondere Routinen zur Anzeige und Bearbeitung von Daten zur Verfügung. Diese müssen manuell jeder einzelnen Datenbank hinzugefügt werden und lassen sich nicht von vornherein als Standard aktivieren. Dadurch wird eine automatische Installation erschwert.

Das hier implementierte System arbeitet auf einer Datenbank des Fachbereichs Mathematik und Informatik der Freien Universität Berlin. Die Datenbankversion heißt: *PostgreSQL 7.3.4 on i686-pc-linux-gnu, compiled by GCC 2.95.4*. Die geografische Erweiterung ist in der Version *POSTGIS 0.7* installiert. Als Betriebssystem wird Linux mit einem 2.4.19er Kernel eingesetzt, der spezielle Unterstützung für High-performance Journaling File Systems (XFS) bietet. XFS-Dateisysteme empfehlen sich zum Einsatz auf Datenbankservern, da sie große Dateien mit variablen Blockgrößen unterstützen und zur einfacheren Administration erweiterte Zugriffsrechte bieten. Das Dateisystem benutzt als Datenstruktur B-Bäume, die eine hohe Leistungsfähigkeit auch bei großen Dateimengen ermöglichen. Zusätzlich handelt es sich bei XFS um ein Journaldateisystem, welches im Fall von Systemabstürzen oder Festplattenproblemen im Vergleich zu traditionellen Dateisystemen eine schnelle Fehlerkorrektur der Festplatte ermöglicht. Dies wird erreicht, indem ein Journal, das Metadaten zum Festplatteninhalt und den ausgeführten Schreib- und Lesevorgängen enthält, zur Rekonstruktion von Fehlern benutzt wird und somit die Dateisysteme im Falle eines Systemausfalls schneller wieder für die Benutzung verfügbar sind.

Die Datenbank bildet die niedrigste Schicht des touristischen Informationssystems, da sie Daten in rein textueller Form enthält und verarbeitet. Sie bildet im Rahmen der Mehr-Schichten-Architektur somit die Datenhaltungsschicht.

5.2 Anwendungsserver

Zur Umsetzung der fachlichen Konzepte einer Anwendung werden so genannte Anwendungsserver eingesetzt. Diese bieten den Anwendungen eine Laufzeitumgebung und kontrollieren deren Ausführung. Das bedeutet auch, dass sie Aufgaben der Lastverteilung oder der Fehlerprotokollierung übernehmen. Somit können die Anwendungen die Dienste des Anwendungsservers benutzen und müssen beispielsweise keine eigenen Funktionen zur Fehlerprotokol-

lierung in ihren Komponenten implementieren. Ein Anwendungsserver mit hoher Verbreitung ist Jakarta Tomcat, der von der Apache Software Foundation [29] als Open-Source-Projekt verfügbar ist. Jakarta Tomcat stellt eine Implementierung der Java Servlet- und Java Server Pages (JSP)-Spezifikation dar, die von der Firma Sun veröffentlicht wurden. Servlets sind spezielle Java-Klassen, mit denen Anfragen an Webserver so abgearbeitet werden können, dass die zurückgelieferte HTML-Seite dynamisch mit Inhalt gefüllt werden kann und keine vorher festgelegte statische Seite ausgeliefert wird.

Servlets und JSP-Seiten benötigen eine spezielle Laufzeitumgebung, die mit einem Web-Server kommuniziert und auch Web-Container genannt wird. Jakarta Tomcat stellt solch einen Web-Container zur Verfügung. Dieser arbeitet mit einem Web-Server, der HTTP-Anfragen von Klienten bearbeiten kann, zusammen. Hier wird dazu der Apache HTTP-Server [26] eingesetzt (Linux Version 1.3.27, Windows Version 2.0.49). Der Tomcat-Web-Container hat die Version 5.0.19. Beide Systeme können sowohl auf Linux- als auch auf Windows-Betriebssystemumgebungen installiert werden und sind unabhängig von der Datenhaltungsschicht, die auch davon räumlich getrennt installiert sein kann.

Die Anwendungen - wie beispielsweise das TIP-System - werden in Archivform (WAR und EAR) installiert und lassen sich durch Konfigurationsdateien im XML-Format einfach anpassen. Die Archive enthalten dabei nur kompilierte Klassen und die benötigten Konfigurationsdateien. Die Konfiguration einer Anwendung auf dem Anwendungsserver geschieht wie folgt:

- Festlegung von Zugriffsrechten pro Anwendung
- Bekanntmachung des Anwendungsarchivs (WAR)
- Start der Anwendung durch den Anwendungsserver

Alle nötigen Einstellungen müssen per Hand in Konfigurationsdateien vorgenommen werden. Dazu gehört auch die Einbindung von benutzten Rahmenwerken. Dabei sind Anpassungen sowohl der Rahmenwerkskonfigurationen selbst als auch zum Zusammenspiel von Anwendungsserver, Anwendung und Rahmenwerk vorzunehmen. Für die TIP-Anwendung wurden dazu die Dateien `struts-config.xml` und `web.xml` angepasst.

5.3 Programmiersprache

Die zur Umsetzung benutzte Programmiersprache ist Java in Verbindung mit dem Struts-Rahmenwerk [27]. Die Firma Sun stellt Java in mehreren Ausführungen kostenfrei zur Verfügung. Dies sind derzeit:

- **Java 2 Standard Edition (J2SE)** –
  bietet Basisfunktionalität und ist Grundlage für andere Java-Versionen zur Softwareentwicklung

- **Java 2 Enterprise Edition (J2EE)** –
  erweitert J2SE um die Fähigkeit zur komponentenbasierten mehrschichtigen Softwareentwicklung in Unternehmen

- **Java 2 Micro Edition (J2ME)** –
  reduziert die J2SE zur Programmierung von mobilen Geräten

Im Gegensatz zu servletbasierten Ansätzen, die Programm- und Darstellungslogik miteinander vermischen, wird hier J2SE 1.4.2_04 zur Implementierung einer mehrschichtigen Architektur benutzt. Java eignet sich zur Anwendungsentwicklung auf verschiedenen Betriebssystemen. Die Standardausgabe (J2SE) bietet umfangreiche Bibliotheken zur I/O- und Netzprogrammierung sowie zur Erstellung grafischer Oberflächen. Zusätzlich dazu ist Java durch seinen hohen Verbreitungsgrad und der Verfügbarkeit kostenfreier Rahmenwerke und Entwicklungswerkzeuge gegenüber anderen Produkten (z.B. Microsoft .NET) für die Entwicklung verteilter Software- und Informationssysteme attraktiv. Diese Eigenschaft wurde im Rahmen der Diplomarbeit stark genutzt, wobei nur kostenfrei verfügbare Entwicklungsumgebungen und unterstützende Softwarewerkzeuge (tools) zum Einsatz kamen.

Damit soll nicht behauptet werden, dass kostenfreie Werkzeuge grundsätzlich besser oder funktional mächtiger sind als ihre kommerziellen Pendants. Die Firma Microsoft bietet mit ihrer kostenpflichtigen .NET-Umgebung Softwareentwicklern die Möglichkeit, verschieden umfangreiche Anwendungstypen zu erstellen. Dazu gehören internetfähige Mehrschichtanwendungen und leicht- und schwergewichtige Komponenten sowohl auf Klienten- als auch Serverseite.

5.3.1 Verwendete Rahmenwerke

Die Benutzung von Rahmenwerken in der Softwareentwicklung hat Vor- und Nachteile. Hauptnachteil ist die damit verbundene Abhängigkeit von Konstruktionen des Rahmenwerks, die sich in einer geringeren Flexibilität äußert. Zur Implementierung eigener Funktionen muss man sich an Regeln des Rahmenwerkes halten und kann nur an bestimmten Stellen Funktionalität hinzufügen. Aus der Nutzung von Rahmenwerken lassen sich im Allgemeinen folgende Vorteile ableiten [46]:

- Benutzung einer funktional fertigen Anwendung zur Schaffung eigener Software
- Erweiterung bestehender Funktionalität durch Vererbungs- und Schnittstellenbeziehungen
- Beschleunigte Softwareentwicklung durch Verwendung textueller Konfigurationsdateien und hohem Modularisierungsgrad

Die im Rahmen der Diplomarbeit gemachten Erfahrungen relativieren einige der oben genannten Vorteile, worauf im nächsten Kapitel detaillierter eingegangen wird. Dennoch hat die Verwendung der Rahmenwerke die Entstehung des TIP-Systems insgesamt positiv beeinflusst.

Rahmenwerke zur Implementierung von webbasierten Anwendungen lassen sich in ereignis- und aktionsgesteuerte Rahmenwerke unterteilen. Aktionsgesteuerte Rahmenwerke gleichen in ihrer Ablauflogik dem *Request-Response*-Prinzip einer Anfrage mittels HTTP [7]. Dabei schickt der Klient an einen HTTP-fähigen Webserver Anfragen. Diese nennen sich *Request* und werden vom HTTP-Server verarbeitet. Als Resultat erhält der Klient den entsprechenden Inhalt als Datei unter Angabe der genauen Zeichenanzahl zugesandt. Diese Anwort, *Response* genannt, wird beispielsweise von einem Webbrowser des Klienten als Internetseite dargestellt. Die Rahmenwerke machen sich dieses Verhalten zu Nutze und ordnen einer eingehenden Anfrage eine Aktion zu. Diese Aktion kann beispielsweise eine Java-Klasse sein, die beim Aufruf abläuft und selbst erstellt werden muss. Nach dem MVC-Muster wird eine Anfrage vom Klienten meistens durch eine Tastatur- oder Mauseingabe in der Ansichtsschicht der Anwendung ausgelöst. Die ausgelöste Aktion verarbeitet alle Parameter der Anfrage und ruft Methoden der Geschäftslogik auf, deren Ergebnis meistens in einer veränderten Version der Ansicht angezeigt wird.

Die Art und Weise wie die Ansicht dabei technisch realisiert ist, unterscheidet sich von Rahmenwerk zu Rahmenwerk. Es ist dabei möglich, mit so genannten Schablonen (templates) zu arbeiten. Diese unterteilen die Ansicht einer Webanwendung in mehrere getrennt voneinander existierende Teile, die es ermöglichen, wiederkehrende statische Ansichten getrennt von den Dateien der dynamischen Ansicht zu speichern. Der HTTP-Server muss nicht immer, die komplette Seite einer Anwendung neu berechnen, da er die statischen Elemente bereits fertig in seinem Seiten-Cache-Speicher vorhalten kann und somit eine schnellere serverseitige Abarbeitung ermöglicht wird.
Das Apache Velocity-Projekt [30] stellt eine kostenfreie Schablonenumgebung (template engine) zur Verfügung. Der Vorteil von Velocity liegt in seiner Fähigkeit, verschiedenformatige Ausgaben zu generieren. Dadurch ist es möglich, die Seite speziell für einen Internetbrowser zu optimieren oder als Postscript-Dokument zum Ausdrucken bereitzustellen. Zusätzlich dazu bietet Velocity dem Webdesigner unabhängig vom Programmierer der Geschäftslogik die Möglichkeit, in seinen HTML-Seiten spezielle Skriptbefehle zur Ablaufsteuerung (Schleifen und Bedingungen) einzufügen. Die Schnittstelle zur Geschäftslogik stellt ein so genanntes Kontextobjekt dar, in dem der Zustand der Anwendung und die von der Geschäftslogik generierten Inhalte abgespeichert sind.
Das Struts-Rahmenwerk bietet mit der Tiles-Erweiterung auch die Möglichkeit, die Webseitenerzeugung durch Verwendung von Schablonen zu optimieren. Auf Grund des nötigen Einarbeitungsaufwands wurde in der vorliegenden Implementierung auf den Einsatz der Tiles Template-Technologie verzichtet. Stattdessen wurden nur selbst erstellte HTML-Seiten unter Verwendung von Tag-Bibliotheken eingesetzt. Derzeit ist nicht absehbar, welches Darstellungsrahmenwerk sich langfristig durchsetzen wird. Daher wird durch die strikte Implementierung unter Einhaltung des MVC-Entwurfsmusters gesichert, dass die TIP-Anwendung auch mit zukünftig verfügbaren Darstellungsmechanismen arbeiten kann. Alle HTML-Vorlagen sind inhaltlich gegliedert und getrennt von der Programmlogik (siehe Abschnitt 5.4.3, S. 79) abgelegt und finden sich im Verzeichnis `pages`, dessen Struktur Abbildung 15 (S. 138) zeigt.

Als Beispiel für ein ereignisorientiertes Rahmenwerk zur Entwicklung von Anwendungen für das Internet gilt das Apache Tapestry-Projekt [28]. Im Vergleich zu aktionsorientierten Rahmenwerken bietet die Ereignisorientierung die Möglichkeit, Quellcodefragmente gezielt einzelnen Aktionsereignissen zu-

zuordnen. Dies entspricht einer objektorientierten Vorstellung vom Ablauf der Anwendung: typsichere Objektinstanzen verändern ihren Zustand durch Methodenaufrufe und agieren miteinander. Dadurch ist es möglich, eine Tapestry-Anwendung während ihrer Laufzeit nach Fehlern zu durchsuchen, da man ihren Zustand von außen abfragen kann. Dazu dient der so genannte Tapestry Inspector, der im Lieferumfang enthalten ist und in jede eigene Anwendung eingebunden werden kann.

Möchte man keine Rahmenwerke einsetzen, muss man entweder die gesamte durch Rahmenwerke verfügbare Steuerung selbst implementieren oder kann auf existierende Skripting-Sprachen zurückgreifen. Diese Sprachen – wie beispielsweise PHP [43] – sind sehr populär, da sie eine schnelle Anwendungsentwicklung ermöglichen. Ihr Einsatz in komplexen serverbasierten Anwendungssystemen erscheint derzeit als nicht empfehlenswert, da durch die Durchmischung von Quellcode, Geschäftslogik und HTML-Darstellung eine geringe Wiederverwendung ermöglicht wird. Eine funktionale Modularisierung der Anwendung nach den Vorgaben des MVC-Entwurfsmusters wird bei der Verwendung von Skriptsprachen erschwert.

Durch den Einsatz von Java als objektorientierte Hochsprache entfällt die Benutzung von serverseitigen Skriptsprachen. Dazu zählt auch die herkömmliche Verwendung von JSP, wo Java-Quellcode direkt mit HTML-Inhalten vermischt wird. Die weit verbreitete Nutzung von Java Servlets ändert diesen funktionalen Nachteil nicht. Java Servlets sind spezielle Klassen, die Geschäftslogik kapseln und beim Aufruf als Ergebnis eine HTML-Seite produzieren, die an den anfragenden Klienten ausgeliefert wird. Zur strikten Vermeidung der herkömmlichen Skriptingsprachen werden hier Datenbankinhalte, Programmlogik – in Form von Java-Klassen – und Ansichtskomponenten voneinander getrennt.

Beim Vergleich von Tapestry und Struts fällt auf, dass Tapestry durch seine objektorientierte Arbeitsweise eigentlich stärker verbreitet sein sollte. Derzeit wird aber von vielen kommerziellen und nichtkommerziellen Anbietern das Struts-Framework eingesetzt. Es ist für den hiesigen Kontext geeigneter, da bei touristischen Informationssystemen eher viele kleine Aktionen ausgeführt werden. Dies wird durch den aktionsorientierten Ansatz von Struts unterstützt. Ein objektorientierter Ansatz ist beispielsweise eher für Reisebuchungssysteme mit längeren Einzeltransaktionen geeignet. Im Fall einer Reisebuchung kann der Zustand der Aktion Reise-buchen als Objekt modelliert werden, dessen Zustand durch Setzen von Reiseinformationsobjekten

(Startzeitpunkt, Startort, Ziel, etc.) durch den Benutzer verändert wird.

Die Tatsache, dass alle verfügbaren Softwarerahmenwerke nach dem MVC-Entwurfsmuster arbeiten, erleichtert die Schaffung eines modularen Systems. Zur Implementierung wurde hier auf das Struts-Rahmenwerk zurückgegriffen, da es für den Einsatz in verteilten Informationssystemen optimiert ist. Zusätzlich existieren auf Grund der hohen Verbreitung viele Internetseiten und -foren, die bei Problemen behilflich sein können.

5.3.1.1 Struts-Rahmenwerk

Das Struts-Rahmenwerk setzt sich zum Ziel, bei der Implementierung als Verstrebung (eng. strut = Strebe) der einzelnen Komponenten zu dienen. Deshalb stellt es als zentralen Bestandteil die Steuerungsschicht einer Anwendung (controller) zur Verfügung. Diese wird zur Entwicklung webbasierter Mehrschichtanwendungen benutzt und ist als Klasse in der Programmiersprache Java geschrieben. Zusätzlich werden verwandte Technologien wie XML und JSP eingesetzt.

Struts wurde von Craig R. McClanahan im Jahr 2000 erfunden und im Mai 2000 zu den Projekten der Apache Software Foundation [58] hinzugefügt. Durch die Verwendung von Struts in der Version 1.1 konnten folgende Eigenschaften in der Implementierung des touristischen Informationssystems verwendet werden:

- Implementierung des MVC-Entwicklungsmusters
- Validierung von Eingaben in HTML-Formularen
- Streng vorgegebener Arbeitsablauf der Anwendung durch festgelegte Controller-Komponente
- Keine Vorgabe in Bezug auf Modell- und Ansichtsschicht der Anwendung
- Möglichkeiten der Lokalisierung und Internationalisierung
- Integrierte Sitzungsverwaltung (session management)

Auf die spezielle Ausprägung des MVC-Musters im Struts-Rahmenwerk wird im Abschnitt 5.4.2 eingegangen.

Die Validierung von Benutzereingaben macht es möglich, den Quellcode zur Prüfung der Eingaben von der eigentlichen Programmlogik zu trennen. Das Rahmenwerk kann die Prüfung sowohl implizit als auch explizit vornehmen. Die explizite Prüfung erfordert, dass eine eigene Java-Klasse von der Klasse *ActionForm* des Rahmenwerks abgeleitet wird. In dieser müssen alle Eingabeelemente als Objekte hinterlegt sein. Zusätzlich existiert eine *validate()*-Methode, die beim Senden der Daten an den HTTP-Server ausgeführt wird und die Prüfungen vornimmt. Möchte man nicht für jedes Eingabeformular eine eigene Klasse erzeugen, kann man auf das *DynaActionForm*-Konzept von Struts zurückgreifen. Dabei erzeugt das Rahmenwerk während der Laufzeit eine generische *ActionForm*-Objektinstanz. In separaten XML-Konfigurationsdateien wird dabei festgelegt, welchen Regeln eine Eingabe zu genügen hat. Zur Beschreibung der Regeln können reguläre Ausdrücke benutzt werden, die beispielsweise die Validierung einer eingegebenen E-Mail-Adresse stark vereinfachen.

In der vorliegenden TIP-Anwendung wurden stets eigene *ActionForm*-Klassen geschrieben, da die Einarbeitung in die Konfigurierung der Validierungserweiterung von Struts zu viel Zeit in Anspruch genommen hätte und die Anwendung nur aus einfachen Eingabeformularen besteht. Der Vorteil eigener Klassen besteht in der hohen Flexibilität der Anwendung. Außerdem wird durch eigene Validierklassen klarer, auf welche Art und Weise Struts den Ablauf einer Anwendung steuert.

Das Rahmenwerk gibt durch Bereitstellung der Steuerungskomponente den Ablauf einer Aktion vor und erleichtert somit die Programmierung. Da Struts keine Einschränkungen bezüglich des Modells und der Ansichtsschicht der Anwendung vorgibt, wurde hier auf JSP in Verbindung mit so genannten Tag-Bibliotheken (JSTL) zurückgegriffen. Diese ermöglichen es, Datenbankobjekte anzuzeigen oder Kollektionen mit Schleifen abzuarbeiten, wodurch das direkte Einfügen von Java-Befehlen in die JSP-Seite entfällt.

Die Tag-Bibliotheken stellen erweiterte Tags zur Verfügung, mit denen der HTML-Code standardkonform und durch Parser validierbar bleibt. Ein Vorteil dieser Fähigkeit ist, dass man die Tags extern mit Java-Code verbinden kann und so anwendungsspezifische eigene Tag-Bibliotheken erzeugen kann. Dadurch bleibt der Quellcode der HTML-Seite lesbarer, da er keine Programmlogik enthält, wie folgende Beispiele (Abbildung 5.2 und 5.3) kurz verdeutlichen.

```
1 <body>
2  <p>
3   Herzlich Willkommen <%= session.getAttribute("username") %>
4  </p>
5 </body>
```

Abbildung 5.2: traditioneller JSP-Skripting-Ansatz

```
1 <body>
2  <p>
3   Herzlich Willkommen <user:firstName/>
4  </p>
5 </body>
```

Abbildung 5.3: Anwendung von JSTL/Tag-Bibliotheken

Um die Benutzung zu vereinfachen, existiert in der JSTL zusätzlich eine so genannte Expression Language (EL), die Variablenzugriffe und logische Abfragen auf Daten vereinfacht. Im Gegensatz zum naiven JSP-Ansatz ist es hier möglich, den eigentlichen Inhalt lesend zu verstehen. In Abbildung 5.3 wird dazu ein Tag mit dem Namen *user* zur Anzeige des Vornamens benutzt (Zeile 3). Das TIP-System benutzt die Funktionalität von JSTL, EL und strutseigenen Tag-Bibliotheken.

Struts unterstützt die Lokalisierung und Internationalisierung von Anwendungen. Dies bedeutet, dass die Anwendung ihre Sprache und die Darstellung ihrer Inhalte an länderabhängige Standards anpassen kann. Die Anzeige eines Datums variiert beispielsweise zwischen 2004-05-01 und 05/01/04 je nachdem, ob die Anwendung in deutscher oder englischer Sprache ausgeführt wird.

Eine Anwendung heißt internationalisiert, wenn ihre textuellen Inhalte getrennt von der eigentlichen Anwendung gespeichert werden, also nicht in die Anwendung kodiert sind. In Java werden dazu so genannte *Resource-Bundles* [20] genutzt. Diese speichern nach dem Schlüssel-Wert-Prinzip zu einem festgelegten Schlüssel in einer bestimmten Sprache einen Inhalt ab.
Das touristische Informationssystem besitzt dazu eine Textdatei `application.properties`, in der alle textuellen Inhalte der Anwendung gespeichert sind.

Beim Start der Anwendung sendet der Webbrowser Länderinformationen an den Anwendungsserver, der prüft, ob für das angegebene Länderkürzel (z.B. de_DE für Deutschland oder ja_JP für Japan) eine spezielle Ressourcendatei vorhanden ist. Kann keine Sprachdatei gefunden werden, wird immer die Standardsprache geladen. Im TIP-System ist die Standardsprache deutsch. Die Anwendung kann beliebig internationalisiert werden, da ohne erneutes Kompilieren der Anwendung sprachenspezifische Informationen hinzugefügt werden können. Dazu muss dem Namen der Ressourcendatei das entsprechende Länderkürzel hinzugefügt werden. Eine Unterscheidung innerhalb der ausgewählten Sprache kann ebenfalls getroffen werden und hat andere Inhalte für den britisch-englischen Sprachraum (en_UK) im Vergleich zum amerikanisch-englischen Sprachraum (en_US) zur Folge. Abbildung 5.4 zeigt einen Ausschnitt zweier Ressourcendateien, deren Inhalte für den deutschen und englischen Sprachraum sind. Zeile 2 zeigt, dass den Textzeilen dynamisch Inhalte durch Parameter hinzugefügt werden können. Hier kann der aktuelle Benutzername während der Laufzeit des Programms eingefügt werden. Ein Vorteil dieser Technik ist, dass es keiner Änderung am Programm bedarf, wenn man den Benutzernamen beispielsweise an erster Stelle ausgeben möchte; dazu muss nur die Ausgabereihenfolge in der Ressourcendatei angepasst werden. Die Verwendung von Ressourcendateien beschleunigt die Anwendungsentwicklung, da beim Hinzufügen neuer Komponenten textuelle Inhalte stets zentral hinzugefügt werden können.

Unter Lokalisierung versteht man den Prozess der Anpassung der Anwendung an andere Sprachen- und Kulturkreise. Dabei werden nicht nur Texte übersetzt, sondern auch Multimediadateien (Klänge und Grafiken) und Darstellungsformate verändert. Das touristische Informationssystem bietet im jetzigen Stadium nur eine Anpassung der textuellen Inhalte. Die Bilder der Sehenswürdigkeiten werden in allen Sprachen gleich angezeigt. Das aktuelle Datenmodell unterstützt für die Inhalte in der Datenbank keine Mehrsprachigkeit, was in einem zukünftigen Entwicklungsschritt behoben werden sollte. Daher sind alle Inhalte aus der Datenbank in englischer Sprache, während die eigentliche TIP-Anwendung in deutscher und englischer Sprache geladen werden kann. Abbildung 3 (S. 126) zeigt das touristische Informationssystem in deutscher Sprache. Ändert man die Spracheinstellung im benutzten Webbrowser auf englisch, zeigt sich der TIP-Startbildschirm wie in Abbildung 4 (S. 127) zu sehen. Zur Anzeige in einer anderen Sprache bedarf es keiner Änderungen am Quellcode oder an der Konfiguration des TIP-Systems.

```
1 # application_de_DE.properties
2 menue.title = TIP-Benutzerhauptmenue
3 menue.message = Sie sind eingeloggt als {0}.
4 menue.auswahl.1 = Anzeigen und Bearbeiten Ihres Profils
5 # application_en_UK.properties
6 menue.title = TIP main menu
7 menue.message = Your current login is {0}.
8 menue.auswahl.1 = Show and edit your profile
```

Abbildung 5.4: Verwendung von Ressourcendateien zur Internationalisierung

Das Rahmenwerk verhindert durch ein integriertes Sitzungsmanagement eine Vermischung aus Darstellungs- und Geschäftslogik im Rahmen der Anwendungsentwicklung. Bei der Verwendung von herkömmlichen Skriptsprachen (PHP, reines JSP) wird Anwendungslogik häufig in Form von zusätzlichen Parametern von einer Skriptseite zur nächsten transportiert. Dieser Zustand wird innerhalb der HTML-Seiten in einem besonderen Sichtbarkeitsbereich Session gespeichert. Abbildung 5.5 zeigt die Verwendung von URL-Parametern zum Transport von Zustandsinformationen. Dabei wird der aufzurufenden Seite mit Hilfe diverser Parameter der aktuelle Anwendungszustand mitgeteilt. Dies hat bei Änderungen an der Software zur Folge, dass alle HTML-Seiten nach dem Auftreten dieses URL-Aufrufes durchsucht werden müssen, um Parameteranpassungen vorzunehmen. Ebenso wird eine einfache Lesbarkeit des HTML-Codes verhindert und eine Arbeitsteilung zwischen Webdesigner und Programmentwickler erschwert.

```
out.println("
 <a href='de.fub.tip.xyz.protoEvents?detail=1&obj="
   +hilf[0]+"&X="+X+"&Y="+Y+"&mehr=1&alt=1'>
  (mehr Details anzeigen)
 </a>
");
```

Abbildung 5.5: Negativbeispiel zur Benutzung von URL-Parametern (Ausschnitt aus einem Java Servlet)

```
<a href="\moredetails.do">
(mehr Details anzeigen)
</a>
```

Abbildung 5.6: Sitzungsverwaltung mit Struts - Vermeidung von URL-Parametern

In Abbildung 5.6 wird das Struts-Rahmenwerk zur Verwaltung des Anwendungszustands benutzt. Sprechende Link-Namen erhöhen die Lesbarkeit des HTML-Codes. Bei Softwareänderungen entfällt die Bearbeitung des HTML-Codes, da die Logik getrennt von der Darstellung kodiert ist. Der Zustand der Anwendung wird in Java-Objektinstanzen gespeichert und von der Steuerungsschicht der Anwendung verwaltet.
Ein häufiger Ansatz, zu Zwecken der Fehlersuche zusätzliche Parameter einzuführen, wird damit ebenfalls umgangen. Oftmals werden diese im laufenden Betrieb der Anwendung nicht entfernt und rufen Sicherheitsprobleme hervor, da Rückschlüsse auf die zu Grunde liegende Datenverwaltungsstruktur leichter möglich sind.

Die Verwendung des Struts-Rahmenwerkes vermeidet diese potentiellen Sicherheitslücken, da die Steuerungsschicht der Anwendung nur auf bekannte Aktionen abbildet und diese nur sinnvoll in einem vorher definierten Kontext laufen. Dieses Verhalten der einzelnen Aktionen wird durch strikte Trennung der Komponenten nach dem MVC-Muster erreicht. Während der Anwendungsentwicklung konnte die Fehlersuche vereinfacht werden, da der Zustand der Java-Geschäftslogik- und -Zustandsobjekte ohne Veränderung der HTML-Ansichtskomponenten ausgegeben werden konnte.

5.3.1.2 Datenbankverbindungspool

Im Gegensatz zum naiven Ansatz, die Verbindung zu Datenbanken in jeder Anwendungskomponente selbst zu verwalten, bietet die Verwendung eines Datenbankverbindungspools eine zentrale Verwaltung mehrerer Datenbankverbindungen und bei der Benutzung eine einheitliche Schnittstelle zur Anwendung.
Ein Vorteil der Benutzung des Verbindungspools ist, dass die Geschäftslogikschicht beim Zugriff auf die Datenbank nicht die genauen Datenbankverbindungsparameter wissen muss. Dies erleichtert das Austauschen der Datenbank oder das Hinzufügen neuer Datenbankverbindungstypen, da diese nur im Datenbankverbindungspool konfiguriert werden müssen und der Anwendung dann transparent zur Verfügung stehen.

Datenbankpooling-Mechanismen stehen sowohl auf Anwendungsserverseite als auch durch Benutzung des Struts-Rahmenwerkes zur Verfügung. Da hier mit Struts die gesamte Anwendungssteuerung geschieht, wird der Struts-Datenbankpool benutzt, der, wie in Abbildung 5.7 zu sehen ist, konfiguriert werden kann. Bei der Initialisierung der Struts-Anwendung werden bis zu zehn Datenbankverbindungen vorgehalten (Zeile 16). Jede Datenbank, die Element des Verbindungspools ist, wird durch einen eindeutigen Schlüssel *tipdb* gekennzeichnet (Zeile 2). Über diesen kann die Anwendung eine Datenbankverbindung des jeweiligen Typs zentral anfordern. Damit entfällt die mit JDBC sonst nötige mehrstufige Registrierung der Datenbankeigenschaften zur Benutzung einer Datenbankverbindung. Im touristischen Informationssystem vereinfacht der konfigurierte Datenbankverbindungspool die Benutzung externer Datenquellen und vermeidet eine umständliche Benutzung in einzelnen Quellcodesegmenten.

In Abhängigkeit von der Anbindung und dem Standort der Datenbank ist der Verbindungsaufbau zeit- und ressourcenaufwändig. Wenn mehrere Komponenten jeweils eigene Datenbankverbindungen aufbauen, die sie nur kurze Zeit benutzen, muss die Datenbank einen hohen Verwaltungsaufwand betreiben. Dieser wird durch den Datenbankverbindungspool übernommen, der zur Datenbank die vorher festgelegte Anzahl von Verbindungen öffnet und diese nach Bedarf an die Anwendung weiterreicht.
Bei Problemen mit der Verfügbarkeit der Datenbanksysteme muss man bei der naiven Herangehensweise jede Komponente mit Datenbankzugriff separat anpassen, während man bei Verwendung eines Verbindungspools diesen nur

```
<data-sources>
 <data-source key="tipdb"
        type="org.apache.commons.dbcp.BasicDataSource">
    <set-property property="driverClassName"
        value="org.postgresql.Driver" />
    <set-property property="url"
        value="jdbc:postgresql://db.inf.fu-berlin.de/pg" />
    <set-property property="validationQuery"
        value="SELECT␣version();" />
    <set-property property="defaultAutoCommit"
        value="false" />
    <set-property property="defaultReadOnly"
        value="false" />
    <set-property property="username" value="user" />
    <set-property property="password" value="password" />
    <set-property property="maxActive" value="10" />
    <set-property property="maxWait" value="5000" />
 </data-source>
</data-sources>
```

Abbildung 5.7: Datenbankpoolverwaltung mit Struts - Ausschnitt aus `struts-config.xml`

zentral umkonfigurieren muss. Die Benutzung des Datenbankverbindungspools erhöht die Fähigkeit der Anwendung, in Netzclustern abzulaufen und senkt die Fehleranfälligkeit der Gesamtanwendung, da mehrere Datenbanken für den Benutzer transparent eingesetzt werden können. Im Fall von TIP kann somit durch Manipulation der Konfigurationsdatei (Abbildung 5.7) ein Datenbankwechsel durchgeführt werden.

5.3.2 Eingesetzte Programmierwerkzeuge

5.3.2.1 Versionierungssystem

Zur einfacheren Softwareentwicklung wurde im Rahmen der Diplomarbeit ein Versionierungssystem eingesetzt. Dieses System unterstützt die Arbeit von Softwareentwicklern, da es eine zu entwickelnde Anwendung als Modul in einem zentralen Repositorium speichert. Diese zentrale Verwaltung sichert einen korrekten Zustand der Anwendung, wenn mehrere Entwickler gleichzeitig oder ein Entwickler von verschiedenen Standorten aus Anwendungsentwicklung betreiben.

Das hier verwendete System CVS – Concurrent Versioning System [6, 13] – ist weit verbreitet. Es kommt auf vielen linuxbasierten Betriebssystemen zum Einsatz und wird an der FU für die Projektarbeit genutzt.
CVS gliedert sich nach dem Client-Server-Schema. Ein zentraler CVS-Server verwaltet die einzelnen Projekte als Module und ermöglicht so den zugreifenden Klienten den konkurrierenden Zugriff. Ein Rechner kann gleichzeitig CVS-Server und -Client sein. Vorwiegend kommen als Server unixbasierte Systeme zum Einsatz, während auf Klientenseite beliebige Betriebssysteme benutzt werden können. Typischerweise bietet CVS über eine Textkonsole Zugang zum Repositorium und es existieren für die meisten Betriebssysteme zusätzlich grafische Umgebungen, die die Arbeit vereinfachen.
Eine Arbeit am Projekt gliedert sich aus Klientensicht in folgende Schritte:

1. Herunterladen des aktuellen Projektstands (checkout)

2. Lokales Arbeiten am Projekt

3. Abgleich mit dem Server (update)

4. Festschreiben der Änderungen und Versionswechsel (commit)

Der Vorteil des Systems besteht darin, dass gleichzeitig mehrere Benutzer am Projekt arbeiten können. Beim Zurückschreiben der Daten in das Projektrepositorium muss jeder Benutzer eine textuelle Beschreibung seiner Änderungen geben. Somit entsteht eine Art Logbuch der Arbeit am Projekt. Der CVS-Server führt die Änderungen aller Benutzer im Repositorium zusammen. Damit dieser Vorgang maschinell erledigt werden kann, bedarf es der genauen Absprache im Entwicklerteam, wer welche Quellcodeteile bearbeitet. Kann der Server die Inhalte nicht automatisch zusammenführen, muss der Benutzer per Hand die Änderungen einpflegen.

Durch die Fähigkeit von CVS einzelne Versionen als getrennte Anwendungszweige abzuspalten, kann die Softwareentwicklung in größeren Projekten vereinfacht und besser strukturiert werden. Hier kam diese Fähigkeit von CVS nicht zum Einsatz, da nur ein Entwickler am Projekt gearbeitet hat. Dennoch ist der Einsatz von CVS sinnvoll, da durch entsprechende Visualisierungen eine Übersicht über die geleistete Arbeit möglich ist. Dazu wurde auf ein an der FU Berlin entwickeltes Visualisierungswerkzeug für CVS-Repositorien zurückgegriffen. StatCvs [49] ist in Java geschrieben

und bietet verschiedene automatisch generierte Sichtweisen auf das CVS-Repositorium. Diese sind im Webbrowser als normale HTML-Seiten anzeigbar und ermöglichen so, unabhängig von etwaiger CVS-Client-Software, den Zugriff auf das Projekt. Für neue Projektmitglieder bietet sich damit die Möglichkeit, den bisherigen Projektverlauf nachzuvollziehen. In Ergänzung zu einem gut dokumentierten Quellcode, kann somit eine leichtere Einarbeitung ermöglicht werden. Durch StatCvs werden folgende Sachverhalte grafisch dargestellt:

- Übersicht über den Repositoriumsinhalt:
  - Verzeichnisstruktur des Projekts
  - Größe und Verhältnis der Dateien zueinander
- Übersicht über die Veränderung am Quellcode des Projekts:
  - Änderungsansicht für einzelne Benutzer
  - Gesamtansicht der Veränderungen am Projekt

Im Anhang befinden sich ab S. 137 Beispielausgaben von StatCvs. Abbildung 14 zeigt beispielsweise die Entwicklung der Quellcodelänge pro Projekttag. Der Arbeitsrhythmus eines Entwicklers wird aus Abbildung 16 deutlich. Dabei werden die Anzahl der Schreibvorgänge im CVS-Repositorium pro Tag und pro Woche für einen Entwickler dargestellt.

5.3.2.2 Entwicklungsumgebung Eclipse

Die Auswahl einer Entwicklungsumgebung hat stets etwas mit den Vorlieben des Entwicklers zu tun und lässt sich nicht generell entscheiden. Das große Angebot an kommerziellen und nichtkommerziellen Entwicklungsumgebungen erschwert die Auswahl, da jedes Produkt in bestimmten Kontexten Vorzüge gegenüber anderen hat.

Hauptvorteil bei der Verwendung von Eclipse [12] ist die Plattformunabhängigkeit, weshalb es hier in der Version 3M08 unter Linux eingesetzt wurde. Eclipse ist eine integrierte Entwicklungsumgebung (IDE), die zusätzlich zum eigentlichen Quellcode-Editor verschiedene Werkzeuge zur Projektverwaltung und zur Fehlersuche zur Verfügung stellt. Eclipse bietet durch eine offene Architektur die Möglichkeit, verschiedene Zusatzprogramme mit in die Entwicklungsumgebung einzubinden. Ein Großteil dieser

so genannten Plugins ist kostenfrei erhältlich und erleichtert die Softwareentwicklung und die Konfiguration mehrschichtiger Anwendungen. Während der Arbeit für die Diplomarbeit wurden folgende Werkzeuge eingesetzt, die die Anwendungsentwicklung erleichtert und besser strukturiert haben.

1. **Ant** – automatisiertes Erzeugen von Programmpaketen
2. **CVS-Integration** von Eclipse-Projekten durch die IDE selbst
3. **Java2html** – Darstellung von Quellcode als HTML-Datei
4. **Struts-Console** – spezieller Editor für `struts-config.xml`
5. **Log4j** – standardisierte Ausgabe von Laufzeit- und Fehlerinformationen

Am Nützlichsten erwies sich die Integration von Ant [57] in Eclipse. Dieses Werkzeug erstellt, basierend auf einer Konfigurationsdatei `build.xml`, fertige Anwendungspakete oder Dokumentationen. Im Gegensatz zum weit verbreiteten *make* benötigt Ant auf der Zielplattform eine Java-Laufzeitumgebung und kann plattformunabhängig ablaufen.

Die Konfigurationsdatei unterteilt ein Projekt in verschiedene Teilprojekte (target). Diese können separat aufgerufen werden und unterstützen eine modulare Struktur. Im Fall der Diplomarbeit lässt sich somit der Aufbau der Dokumentation (StatCvs, Javadoc) getrennt vom eigentlichen Kompiliervorgang durchführen, was bei der Fehlersuche hilfreich ist. Ebenso nützlich sind die Funktionen zur Zusammenstellung einer JAR/WAR-Datei, um die Anwendung auf Anwendungsservern ausliefern zu können (deployment). Damit können häufig benutzte komplex zu konfigurierende Abläufe per Knopfdruck zur Verfügung gestellt werden. Die Integration von CVS in die Entwicklungsumgebung ermöglicht das einfache Verwalten von Eclipse-Projekten mit CVS. Ebenso bietet Eclipse eine grafische Oberfläche zur Repositoriumsverwaltung und verschiedene Ansichten, um Versionsvergleiche durchführen zu können. Der Einsatz von CVS hat die Fehlersuche und strikte Kommentierung aller Vorgänge am TIP-System stark vereinfacht.

Java2html [36] bietet zusätzlich zur normalen Dokumentation mittels Javadoc die Möglichkeit, den Quellcode formatiert auszugeben. Dadurch kann man ohne spezielle Editoren den Quellcode visuell aufbereitet betrachten. Die Art der Darstellung lässt sich durch CSS-Dateien beliebig anpassen. Hier wurde nur die Fähigkeit benutzt, den Quellcode als HTML darzustellen.

Java2html verfügt auch über andere Ausgabeformate, die sich durch Integration mit Ant automatisch generieren lassen.

Zur Ausgabe von Laufzeitinformationen zum Zweck der Fehlersuche wurde das Log4j-Paket des Apache Logging-Projektes [3] verwendet. Bei der Implementierung verteilter Anwendungen können oftmals keine direkten Ausgaben auf der Konsole gemacht werden, womit die Fehlersuche erschwert wird. Das Log4j-Paket stellt eine Logginghierarchie zur Verfügung, die absichert, dass die Anwendung in verschieden intensiven Ausgabemodi laufen kann. Derzeit sind dies beispielsweise *warn, info* und *debug*. Unter Verwendung dieser festgelegten Loggingtypen kann ohne Änderung am Quellcode der Umfang der Ausgabe durch Anpassung der Konfigurationsdatei `log4j.properties` bestimmt werden. Log4j kann Ausgaben in entfernte oder lokale Dateien speichern oder sich mit betriebssystemnahen Loggingdiensten (Unix syslog oder NT event logger) verbinden. Die Benutzung von log4j ersparte, dass jede TIP-Komponente proprietäre Loggingmechanismen implementieren muss. Dadurch wird die Leistungsfähigkeit der Gesamtanwendung gesteigert und das Schreiben von Statusinformationen geschieht zentral und so ressourcensparend wie möglich. Da das Struts-Rahmenwerk zur Einbindung eigener Plugins und Werkzeuge Schnittstellen zur Verfügung stellt, wurde hier ein eigenes Log4j-Plugin auf der Ebene des Rahmenwerks installiert. Dieses wird beim Start der Anwendung auf dem Anwendungsserver von Struts selbst initialisiert, womit im Programm direkt Fehlerausgaben erzeugt werden können, ohne dass eine aufwändige Initialisierung der Logging-Komponente geschehen muss.

Die Auswahl von Eclipse und zugehöriger Plugins ist ein Teil der Implementierung des TIP-Systems, da durch Verwendung dieser Werkzeuge eine gut strukturierte und dokumentierte Anwendungsentwicklung ermöglicht wurde.

5.3.2.3 Anwendungsserver

Bei den meisten Linux-Distributionen ist ein Anwendungsserver Teil der Standardinstallation. Der in der Suse 8.2-Distribution enthaltene Anwendungsserver Tomcat 4.0.17 erwies sich während der Programmentwicklung als hinderlich. Stattdessen wurde eine selbst installierte Version 5.0.19 verwendet, die einen erheblich größeren Funktionsumfang hat und sich während der Programmentwicklung häufig besser neu starten und anpassen ließ. Dies

ist darauf zurückzuführen, dass die im Lieferumfang der Distribution enthaltene Software aus Sicherheitsgründen unter einem anderen Benutzer läuft und sich nur vom Administrator (root) anpassen lässt. Dieser Umstand ist für die Phase der Fehlersuche und bei häufigem Anwendungsneustart eher hinderlich. Zusätzlich vereinfacht sich die Zusammenarbeit mit Rahmenwerken wie Struts oder Hilfsmitteln wie Ant, wenn der Anwendungsserver unter dem lokal angemeldeten Benutzer läuft.

Wenn die Anwendung in Produktionsbetrieb geht, sollte wieder auf die Standardkonfiguration gewechselt werden, da diese auf einer höheren Sicherheits- und Schutzstufe arbeitet. Zusätzlich lassen sich Sicherheitsupdates für im Lieferumfang enthaltene Softwarepakete automatisch installieren und benötigen keine manuelle Neuinstallation und Konfiguration der Software.

In Zusammenarbeit mit dem Webserver zur Auslieferung von HTML-Inhalten und Beantwortung von Anfragen unter Verwendung von HTTP stellt der Anwendungsserver eine Laufzeitumgebung für JSP-Seiten und Java Servlets zur Verfügung. Technisch wird beim Aufruf einer JSP-Seite diese vom Java-Compiler in ein Java Servlet übersetzt, das dann ausgeführt wird und als Ergebnis eine HTML-Seite liefert, die der Web-Server an den Klienten sendet. Der Vorteil bei der Verwendung von Struts ist, dass bei wiederholten Aufrufen nicht jede Seite neu kompiliert werden muss. Struts erstellt beim Erstaufruf eine Objektinstanz, die wiederholt verwendet wird und dadurch die Ausführungszeit der vom Benutzer ausgewählten Aktion verkürzt. Durch die Einhaltung des MVC-Entwurfsmusters lassen sich Komponenten mehrfach verwenden. Objekte, die keinen benutzerspezifischen Zustand kapseln, wie beispielsweise Aktionen zur Anwendungssteuerung können bei richtiger Programmierung wiederverwendet werden. Dies beschleunigt die Benutzung der Anwendung, stellt aber höhere Anforderungen an die Hardwareausstattung des Rechners mit der Anwendungsserversoftware, da dieser genügend Arbeitsspeicher haben muss, um die Objektinstanzen vorhalten zu können. Ein großer Arbeitsspeicher und eine schnelle CPU verringern die Antwortzeit beim Erstaufruf erheblich, da der Kompiliervorgang weniger Zeit in Anspruch nimmt. Während der Programmentwicklung wurden alle TIP-Aktionsklassen so entworfen, dass sie wiederverwendet werden können.

Der Tomcat-Anwendungsserver bietet vielfältige Möglichkeiten der Anwendungskonfiguration und stellt dazu webbasierte Konfigurationsumgebungen zur Verfügung. Diese sind unter Linux in der Standardinstallation aus

Sicherheitsgründen deaktiviert und erschweren den Einstieg, da man zuerst die Dokumentation lesen muss und diverse Änderungen an Konfigurationsdateien vorzunehmen sind. Installiert man den Anwendungsserver stattdessen unter Windows, so ist die Konfigurationsumgebung aktiviert und bietet einen schnellen Einstieg in die Arbeit.

Zur Feinanpassung der Konfiguration empfiehlt sich Linux eher, da es dort ohne viel Aufwand möglich ist, in einer Textkonsole die Anwendung neuzustarten oder den Anwendungsserver in ein Ant-Skript einzubinden. Aus diesem Grund erfolgte die gesamte Anwendungsentwicklung unter Linux.

5.3.2.4 Browser

Zur korrekten Darstellung des touristischen Informationssystems empfiehlt sich ein Webbrowser aus folgender Auswahl:

- Microsoft Internet Explorer – ab Version 6
- Mozilla Browser Suite – ab Version 1.4
- Mozilla Firefox (ehemals Mozilla Firebird) – ab Version 0.8
- Opera – ab Version 7.23

Die verwendeten Browser verfügen alle über die Fähigkeit, standardkonformes HTML anzuzeigen. Die Darstellung unterscheidet sich bei Verwendung verschiedener Internetbrowser durch andere Schriftarten und -abstände, da die konkrete Anzeige dem jeweiligen Darstellungsalgorithmus des Browsers unterliegt. Die Verwendung von Javascript empfiehlt sich für die Eingabevalidierung, ist aber für die Funktion der Anwendung nicht notwendig.
Im Opera-Browser kam es in den Versionen 7.11 bis 7.50 zu Darstellungsproblemen bei dynamisch generierten Seiten, die vermutlich auf Probleme im Dokumenten-Caching-Algorithmus zurückzuführen sind. Beim wiederholten Aufruf einer Seite erschienen Inhalte aus einem vorherigen Aufruf, die durch ein Neuladen (refresh) verschwanden. Vermutlich wird das Auslesen des Caches in Abhängigkeit vom URL-Typ erfolgen. Durch die Verwendung des Struts-Rahmenwerkes haben dynamisch generierte Seiten eine scheinbar statische URL. Der Aufruf der Anwendung geschieht beispielsweise über `http://touristeninfo.de/login.do`. Der Browser interpretiert den Aufruf der Webseite falsch und lädt deren vermeintlichen Inhalt aus dem Cache. Alle anderen Browser reagieren auf ein wiederholtes Aufrufen der gleichen

Anwendungsseiten korrekt und stellen diese neu dar. Im Gegensatz zu anderen Rahmenwerken, die komplett mit Javascript arbeiten, wird hier auf eine so kodierte Ansichtsschicht verzichtet. Die Verwendung von Javascript hat den Nachteil, dass der Funktionsumfang je nach verwendetem Browser stark variiert, da die Javascript-Implementierungen verschieden sind. Stattdessen übernimmt das Struts-Rahmenwerk die komplette logische Steuerung der Anwendung, während im Webbrowser nur HTML-Inhalte dargestellt werden müssen. Dies vereinfacht die Wartung der Anwendung, da man nur einzelne Komponenten bearbeiten muss und bei konsequenter Einhaltung der Unabhängigkeit der einzelnen Schichten die Gesamtanwendung ausführbar bleibt.

Die gemachten Erfahrungen zeigen, dass die Implementierung einer browserbasierten Anwendung lohnenswert ist. Die Software ist unter verschiedenen Plattformen – getestet wurden Linux und MS Windows 2000/XP – lauffähig und zeichnet sich durch die gute Integration mit dem Anwendungsserver als schnell startbar aus.

5.4 Umsetzung

Nachdem in den vorigen Abschnitten die benutzten Softwareumgebungen und -werkzeuge eingeführt wurden, wird hier auf die genauere Umsetzung im touristischen Informationssystem eingegangen. Dabei wird die am Kapitelanfang erwähnte mehrschichtige Architektur näher erläutert.

5.4.1 Datenbank und geografische Erweiterung

Aus der vorhandenen Diplomarbeit von Katja Löffler [33] wurde eine Datenbankkopie erzeugt, die nach einer Portierung auf eine andere Datenbankinstanz zur Arbeit benutzt wurde. Die einzelnen Veränderungen am Datenmodell wurden schrittweise vorgenommen und in separaten SQL-Skripten festgehalten und dokumentiert. Da die verwendete geografische Erweiterung direkt in das Datenbanksystem kompiliert wurde, lassen sich derzeit keine SQL-Skripte generieren, die die Tabellenschemata von den Daten separieren. Dieser Umstand erschwert eine schnelle Migration auf eine andere Datenbank und man muss manuell in den von PostgreSQL erzeugten SQL-Skripten eine Trennung von Metadaten und Daten vornehmen.

Die der Diplomarbeit beigefügte CD (siehe CD-Inhaltsverzeichnis, S. 121) enthält den aktuellen Inhalt der Datenbank als SQL-Skriptdatei.

Auf Grund dieser Schwierigkeiten wurde auf eine existierende Datenbankinstanz am Fachbereich Mathematik und Informatik an der FU Berlin zurückgegriffen. Diese bietet eine funktionierende PostgreSQL-Datenbank mit Postgis-Erweiterung. Der Versuch, eine aktuelle Postgis-Erweiterung in eine aktualisierte Datenbankversion zu integrieren, scheiterte, weshalb die entfernt benutzte Datenbankinstanz nicht mit der aktuellsten Version arbeitet.

Ein Nachteil bei der Nutzung der externen Datenbank ist die Abhängigkeit von funktionierenden Netzzugängen. Im Rahmen der Diplomarbeit stellte sich die Abhängigkeit von der Universitätsinfrastruktur als kritisch heraus, da diese wegen technischer Ausfälle und Wartungsarbeiten oft nicht verfügbar war.

Trotz aller Probleme mit der Verfügbarkeit hat sich die Datenhaltungsschicht bestehend aus Postgres und Postgis unter Linux als sehr zuverlässig herausgestellt, da parallel zum touristischen Informationssystem über die Textkonsole an der Datenbank gearbeitet werden konnte. Die Anwendung greift über Java-Bibliotheken zum Datenbankzugriff (JDBC) auf die Datenbank zu. Dabei wurde strikt auf die Verwendung von *PreparedStatements* geachtet. Diese bieten die Möglichkeit, eine SQL-Abfrage mit Parametern zu versehen. Bevor die Parameter mit Inhalten gefüllt werden, kann das Datenbankmanagementsystem für die Abfrage bereits einen Ausführungsplan erstellen. Nachdem die Parameterdaten gesetzt wurden, beginnt das Datenbanksystem unter Benutzung des vorher generierten Ausführungsplans unverzüglich mit der Bearbeitung der Abfrage. Wird der gleiche SQL-Befehl mehrfach aufgerufen, kann die Datenbank auf vorher im Cache gespeicherte Ausführungspläne zurückgreifen. Dies verringert die Ausführungszeit der SQL-Anfrage insgesamt. Ähnliche Geschwindigkeitsvorteile resultieren auch aus der Benutzung der Fähigkeit zum Stapelbetrieb von *PreparedStatements*. Dabei wird eine Menge von SQL-Abfragen als Stapel (batch) gesammelt an die Datenbank geschickt. Diese führt die Abfragen nacheinander aus und gibt ein Ergebnis zurück. Im Vergleich zur naiven Herangehensweise des einzelnen Abschickens von Abfragen wird weniger Netzverkehr erzeugt. Die Datenbank muss bei hoher Anfragelast nicht viele kleine einzelne Transaktionen ausführen und kann dadurch schneller Anfragen mehrerer Klienten bearbeiten.

Zur Ausnutzung dieser Geschwindigkeitsvorteile verwenden alle eine Datenbank benutzende Komponenten zur Abfragebearbeitung *Prepared-Statement*-Objektinstanzen. Damit wird der Datenbankzugriff der Anwendung zusätzlich zur Benutzung eines Datenbankverbindungspools beschleunigt.

5.4.2 MVC-Entwurfsmuster

Der folgende Absatz geht näher auf die Implementierung des MVC-Entwurfsmusters bei der Verwendung von Struts ein und es werden zusätzlich die verschiedenen anwendungsbezogenen Schichten der Mehr-Schichten-Architektur des TIP-Systems (siehe Abbildung 5.1, S. 48) näher erläutert.

Abbildung 5.8 zeigt die genaue Ausprägung des MVC-Entwurfsmusters bei der Benutzung des Struts-Rahmenwerkes, wobei durch das Rahmenwerk die zentrale Steuerung der Anwendung zur Verfügung gestellt wird. Fest vorgegeben sind dabei das *ActionServlet* und der *RequestProcessor*. Das Servlet nimmt die Anfragen des Benutzers über eine bestimmte URL-Endung entgegen – beispielsweise `http://touristeninfo.de:8080/start.do`. Nach einer syntaktischen Prüfung der Parameter wird vom *ActionServlet* der *RequestProcessor* aufgerufen. Dieser beinhaltet die Logik des Struts-Rahmenwerkes und versucht, die aus der Benutzeranfrage gelesenen Parameter auf eine Aktion abzubilden. Sowohl die einzelnen Aktionsklassen als auch die Konfiguration sind anwendungsspezifisch und müssen selbst umgesetzt werden. Die Abbildung von Aktionsaufrufen auf Aktionsklassen geschieht in der Konfigurationsdatei `struts-config.xml`, die für jede Anwendung separat erstellt werden muss.

5.4.2.1 Model - Datenhaltungsschicht

Die Datenhaltungsschicht respektive das Modell der Anwendung stellt die unterste Ebene in der Mehr-Schichten-Architektur dar. Diese speichert den Zustand der Anwendung und beinhaltet in Abhängigkeit davon die Menge von aktuell erreichbaren Zuständen.
Die Benutzung einer relationalen Datenbank erzwingt die Umwandlung der einzelnen Tupel in Java-Objekte. Zu diesem Zweck bietet die Datenhaltungs-Zugriffsschicht so genannte Wert- oder Ansichtsobjekte (value/view objects). Diese gleichen in ihrem Aufbau einer Java Bean-Klasse (EJB). Beans kapseln Dateninhalte als Objekte und stellen Zugriffsmethoden zum Auslesen

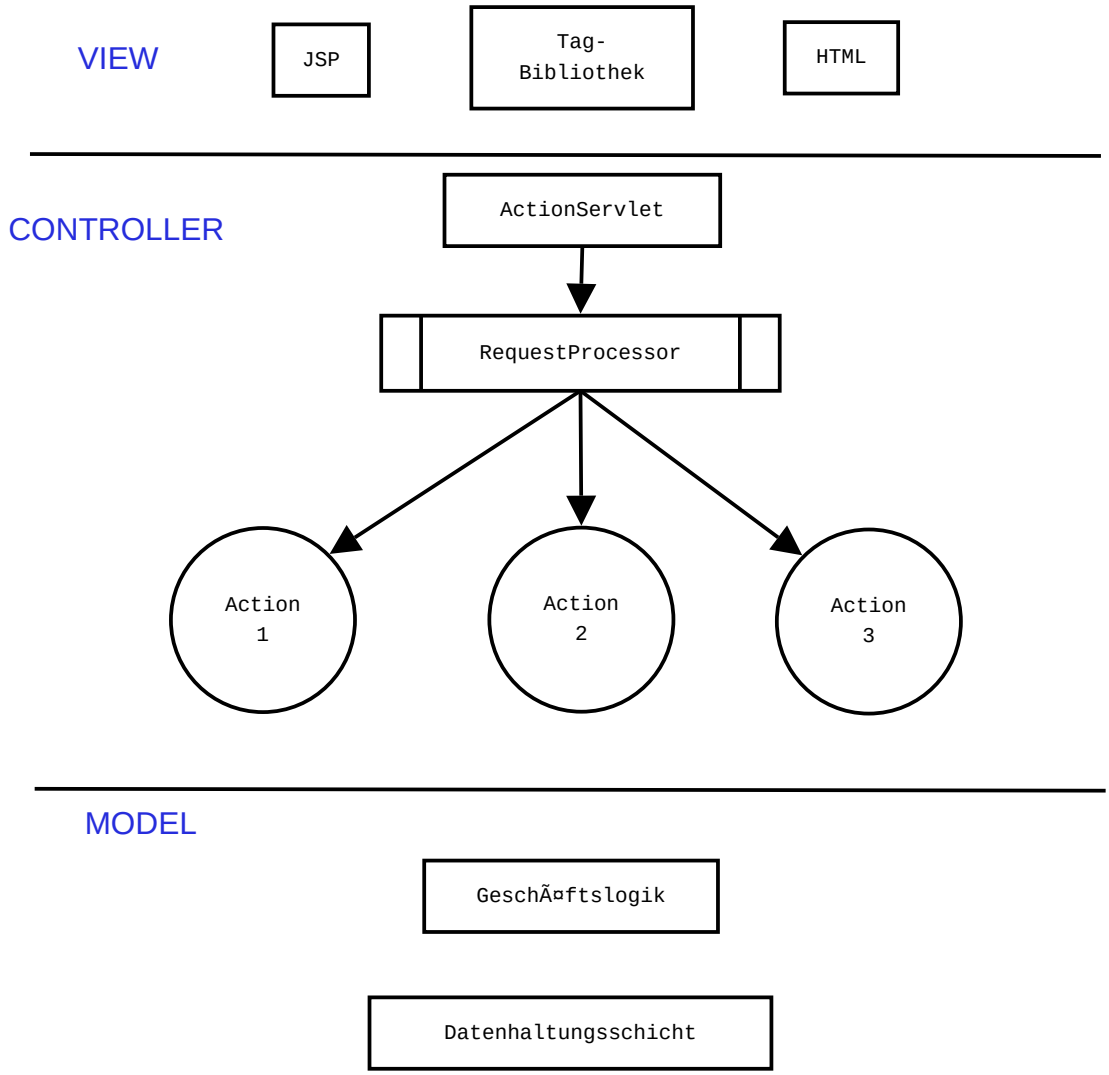

Abbildung 5.8: Komponenten des Struts-Rahmenwerkes aufgeteilt nach dem MVC-Muster

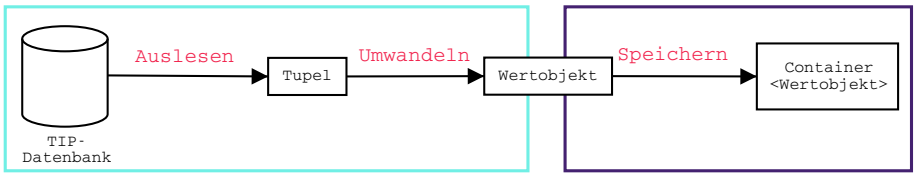

Abbildung 5.9: Zusammenhang von Datenhaltungs- und Datenhaltungszugriffsschicht

und Neusetzen (getter/setter) der Werte zur Verfügung. Um die Abbildung von relationalen Tabellenstrukturen auf Objekte vorzunehmen, kann man entweder die Bean-Klassen entsprechend manuell erstellen oder Generatoren einsetzen. Da das Datenmodell spärlich dokumentiert war, erfolgte die Einarbeitung schrittweise. Dabei wurden die einzelnen Wertobjekte per Hand definiert und als Bean-Klasse implementiert. Das Hinzufügen der entsprechenden Datenzugriffsmethoden erfolgte durch Quellcodegeneratoren der Entwicklungsumgebung. Ähnlich wie bei der manuellen Erstellung der Klassen zur Eingabevalidierung bot sich hierbei die Möglichkeit, den genauen Ablauf besser zu verstehen und die spezifische Repräsentation von Datenbankinhalten granularer festlegen zu können als dies durch automatische Konverter oder Persistenzrahmenwerke geschieht.

Zur einfacheren Verwaltung von Kollektionen von Datenbankobjekten wurden in die Datenhaltungszugriffsschicht so genannte Objekt-Container integriert, die es dem Benutzer ermöglichen, über alle Elemente zu iterieren. Diese Objekte werden aus den einzelnen Struts-Aktionen heraus durch separate gekapselte Logikobjekte mit Inhalt gefüllt und dann durch die Ansichtskomponente der Anwendung angezeigt. In Abbildung 5.9 ist der Zusammenhang zwischen Datenhaltungs- und Datenhaltungszugriffsschicht gezeigt. Die Datenhaltungsschicht – links cyan umrandet – beinhaltet die aus der Datenbank ausgelesenen Tupel, die in Wertobjekten gespeichert werden. Die im jeweiligen Schritt ausgeführten Operationen sind rot markiert. Die Wertobjekte werden in Containern gespeichert. Diese sind vom jeweiligen Typ des Wertobjekts, was durch die Benutzung der generischen Schreibweise *Container <Wertobjekt>* angezeigt wird. Die Wertobjektcontainer werden im nächsten Schritt zur Anzeige der Daten benutzt und bilden den zentralen Bestandteil der Datenhaltungszugriffsschicht, die hier rechts lila dargestellt ist.

Die beiden Schichten der Datenhaltung setzen eine Trennung der Geschäftslogik von den Aktionen um. Dies sichert, dass auf das Modell der Anwendung durch verschiedene Mechanismen zugegriffen werden kann. Das eigentliche Auslesen der Daten geschieht durch Datenbankzugriffe mittels JDBC. Die Ergebnisse der SQL-Abfragen werden in Wertobjekten abgespeichert, die wiederum Teil eines Containers zur einfacheren Verwaltung sind. Damit bei einer Weiterentwicklung der Anwendung die Wertobjekte als Java Beans (EJB) benutzt werden können, lassen sich diese serialisieren. Das bedeutet, dass die Objektinhalte in Dateien gespeichert werden können. Die Fähigkeit zur Objektserialisierung sichert, dass die Objekte über Netzverbindungen zwischen Rechnern ausgetauscht werden können, was bei verteilten Informationssystemen durch Benutzung verteilter Rechnerknoten oder entfernten Compileraufrufen (RMI) möglich ist.
Der bisher implementierte Funktionsumfang der Anwendung erforderte vorwiegend ein Auslesen der Daten aus der Datenbank. Aus diesem Grund verfügen die Objekt-Container nicht über Methoden zum Zurückschreiben der Objekte, wenn diese verändert wurden. Die Architektur des touristischen Informationssystems ist absichtlich so offen gestaltet, dass diese Funktionalität dem Container einfach hinzugefügt werden kann. Es bedarf dabei nur einer neuen Methode, die die Daten in die Datenbank zurückschreibt und in der die Wertobjekte deserialisiert und in der Datenbank abgespeichert werden.

5.4.2.2 View - Ansichtsschicht

Die Ansichtskomponente einer Struts-Anwendung wird oft mit JSP realisiert. Dabei werden statisch vorgegebene HTML-Inhalte durch Ausführung von Aktionen dynamisch mit Inhalten gefüllt. Zur vereinfachten Anzeige von Daten oder Objektmengen bietet sich der Einsatz von Tag-Bibliotheken an.

Im touristischen Informationssystem wurden dazu JSTL und EL benutzt, um den JSP-Code für Nichtanwendungsentwickler deutlich lesbarer zu machen. Das Struts-Rahmenwerk bietet zusätzlich eigene Tag-Bibliotheken. Diese beinhalten spezielle Tags zur Anzeige von internationalisierten Inhalten und zur Verwaltung solcher Ressourcendateien. Damit ist es möglich, ohne Hinzufügen von reinem Java-Quellcode die JSP-Seite funktional zu erweitern.

Der Codeausschnitt (Abbildung 5.10) zeigt, wie die Anzeige von abonnierten Themen im TIP-System umgesetzt ist. Dabei wird über eine Kollektion

von Themen-Wertobjekten iteriert (Variable: *aboTopics* - Zeile 2). Die Inhalte der einzelnen Objekte werden durch Zugriff auf die *get()*-Methoden der *TopicVO*-Wertobjekt-Klassen ausgelesen (Zeilen 17, 18, 21). Dazu wird EL benutzt, welche durch *${...}* eingeleitet wird und auf Variablen aus der aktuellen Sitzung oder der Anfrage zugreifen kann. Der Sichtbarkeitsbereich auf Variablen muss dabei nicht explizit angegeben werden. EL prüft alle verfügbaren Sichtbarkeitsbereiche der Anwendung auf Vorhandensein der angeforderten Variable und ersetzt diese bei der Umwandlung der jeweiligen JSP-Seite in eine HTML-Ausgabe. Alle im touristischen Informationssystem benutzten HTML-Vorlagen sind selbst erstellt und an die jeweilige Anwendungslogik angepasst.

Eine besondere Klasse von anzuzeigenden Inhalten bilden Fehler- (*ActionError*) und Anwendungsmitteilungen (*ActionMessage*). Diese sind in eigenen Klassen gekapselt und ermöglichen in Verbindung mit der Fähigkeit zur Darstellung internationalisierter Inhalte die flexible Anzeige von Informationen zur Laufzeit. Abbildung 5 (S. 128) im Anhang zeigt das Auftreten eines Fehlers beim Anmeldevorgang am System. Jede Fehlermeldung kann dabei an Eingabefelder des Formulars gebunden werden und erleichtert somit die Fehlersuche für den Benutzer.

5.4.2.3 Controller - Steuerungsschicht

Die Steuerungsschicht der Anwendung wird durch das Struts-Rahmenwerk beeinflusst. Eigene Aktionen müssen dazu die strutseigene Klasse *Action* erweitern und Geschäftslogik aufrufen. Je nach Rückgabewert der Geschäftslogikmethoden leiten die Aktionen an andere weiter oder kehren zur bisherigen Seite zurück und geben Fehlermeldungen aus.

Dies ist beispielsweise sinnvoll, wenn der Benutzer ungültige Eingaben gemacht hat. Die Überprüfung der Daten erfolgt dabei syntaktisch durch die zugehörige Formularklasse (*ActionForm*), deren *validate()*-Methode aufgerufen wird. Die semantische Prüfung geschieht durch Aufruf von Methoden der Geschäftslogik und ist somit von der syntaktischen gekapselt und unabhängig von Struts implementiert. Alle im touristischen Informationssystem umgesetzten Aktionen besitzen ihre zugehörige *Action*-Klasse und im Fall von Benutzereingaben ebenfalls Formularklassen. Diese wurden in der Entwurfsphase komplett manuell erstellt.

```
<c:choose>
 <c:when test="${aboTopics␣!=␣null}">
  <bean:message key="utopics.abo"/>
  <br />
  <table border="0">
    <tr class="header">
     <td class="header">
       <bean:message key="utopics.kopf.name"/>
     </td>
     <td class="header">
        <bean:message key="utopics.kopf.desc"/>
     </td>
    </tr>
      <c:forEach items="${aboTopics}" var="topic">
       <tr>
         <td>
        <c:out value="${topic.name}" />
                ( <c:out value="${topic.id}" /> ) <BR/>
         </td>
         <td>
        <c:out value="${topic.description}" /> <BR/>
         </td>
       </tr>
     </c:forEach>
  </table>
 </c:when>
 <c:otherwise>
   <bean:message key="utopics.abo.leer"/><br />
  </c:otherwise>
</c:choose>
```

Abbildung 5.10: Quellcodeausschnitt einer Struts-View-Komponente

Die Abbildung von URL-Angaben auf Aktionen geschieht in der zentralen Konfigurationsdatei von Struts. Im laufenden Betrieb werden Struts-URL-Angaben mit einer speziellen Endung versehen. Diese ist bei Standardinstallationen `.do` und ermöglicht dem Web-Server, beim Lesen einer URL gezielt an das Struts-Rahmenwerk weiterleiten zu können. Zusätzlich dazu bietet Struts auch die Möglichkeit, logische Namen für Weiterleitungsziele und Dateien zu vergeben. Dadurch ist es möglich, innerhalb der Anwendung eine bestimmte Ressource zu verwenden, ohne deren genaue Position zu wissen. Das touristische Informationssystem nutzt diese Fähigkeit, um die CSS-Datei der Anwendung korrekt in alle JSP-Seiten einzubinden und im Fehlerfall zu

speziellen Fehlerseiten weiterzuleiten. Diese Strategie vermeidet, dass bei der Definition der einzelnen Aktionen wiederholt Fehlerseiten festgelegt werden müssen, die von einer Vielzahl von Aktionen gemeinsam benutzt werden. Stattdessen werden diese einmalig global definiert und stehen im gesamten Anwendungskontext zur Verfügung. Das Auftreten von Datenbankfehlern wird auf diese Art und Weise zentral behandelt. Bevor der Benutzer die Fehlerseite mit der Möglichkeit zur Rückkehr zur vorigen Seite sieht, werden von der Anwendung umfangreiche Zustandsangaben in Log-Dateien geschrieben. Dadurch wird die Fehlersuche vereinfacht und dem Benutzer die Betrachtung javaspezifischer Fehlermeldungen erspart.

Abbildung 5.11 zeigt eine Beispielkonfiguration zur Durchführung einer Anmeldung. Dem URL-Aufruf `login` wird die Aktion *LoginAction* zugeordnet (Zeile 1). Diese verweist im Erfolgsfall auf das Benutzermenü (Zeile 4), während beim Auftreten von Fehlern wieder zur Anmeldeseite zurückgekehrt wird (Zeile 6).

```
1 <action path="/login" type="de.fub.tip.actions.LoginAction"
2   name="loginForm" validate="true"
3   input="/pages/input/anmeldung.jsp" scope="session">
4      <forward name="success"
5         path="/pages/menu/menue.jsp" />
6      <forward name="failure"
7         path="/pages/input/anmeldung.jsp" />
8 </action>
```

Abbildung 5.11: Beispielkonfiguration von Struts-Aktionen - Ausschnitt aus `struts-config.xml`

Je nach Inhalt der jeweiligen Aktion werden nach erfolgter Zuordnung zu einer Aktionsklasse die ausgelesenen Parameter semantisch geprüft. Beim Anmeldevorgang wird in dieser Phase ein Abgleich mit den in der Datenbank gespeicherten Daten gemacht. Die Parameterprüfung wird in eigenen Klassen gekapselt. Damit kann man die einmal implementierte Funktionalität auch an anderer Stelle in der Anwendung wiederverwenden.
Die Geschäftslogikobjekte werden mit Hilfe des Factory-Entwurfsmusters erzeugt, worauf im Abschnitt 4.4 näher eingegangen wurde. Zum Zugriff auf die entsprechenden Logikobjekt-Fabriken wird das Singleton-Pattern eingesetzt. Dadurch kann die eigentliche Erzeugung der Anwendungslogik auf

andere Rechner ausgelagert werden. Diese Strategie ermöglicht die Wiederverwendung der programmierten Klassen bei Anwendungsveränderungen, da die Schnittstelle zur Benutzung (die Fabrik) in ihrer Semantik unverändert bleibt. Zusätzlich lässt sich das touristische Informationssystem somit innerhalb eines Netzclusters betreiben, wobei spezielle Clusterknoten zur Objekterzeugung existieren können.

Die Logikobjekte stellen die Fachkonzeptschicht dar, da sie Methoden enthalten, die die Geschäftslogik von Aktionen ausführen. Die Fabrikobjekte stellen die Fachkonzeptzugriffsschicht dar, da sie die Logikobjekte für den Benutzer transparent erzeugen und zur Verfügung stellen.

Im Allgemeinen verarbeitet die Steuerungsschicht der Anwendung Anfragen der Benutzer und entscheidet, welche Geschäftslogikprozesse auszuführen sind. Die Abbildung von Anfragen auf Aktionen geschieht durch das strutseigene *ActionServlet*, dessen Verhalten durch die in der Konfigurationsdatei `struts-config.xml` gemachten Angaben bestimmt wird.

5.4.3 Paketstruktur der Anwendung

In diesem Absatz wird die genaue Struktur des TIP-Systems erläutert und dabei nur der Java-Teil der Anwendung berücksichtigt. HTML- und JSP-Seiten sind in einem ähnlich strukturierten Verzeichnisbaum abgelegt – siehe Abschnitt 4.4.2 (S. 42). Logische Java-Pakete werden auf Verzeichnisse abgebildet, in denen sich die jeweiligen Klassen des Pakets befinden. Dadurch sollen große Anwendungssysteme sinnvoll strukturiert werden können. Als Wurzelverzeichnis fungiert im TIP-System das Paket *de.fub.tip*. Alle weiteren sind in der Tabelle 5.4.3 (S. 80) aufgeführt.

Im Hauptverzeichnis *de.fub.tip* befindet sich die zentrale Loggingkomponente der Anwendung. Diese ist als Struts-Plugin realisiert und wird beim Start der Anwendung vom Anwendungsserver initialisiert und stellt ihre Dienste autonom zur Verfügung. Alle Komponenten des touristischen Informationssystems nutzen dieses Plugin zur Ausgabe von Fehler- und Laufzeitinformationen mittels log4j.

Die Anwendung besteht aus 13 Paketen. In den Paketen *de.fub.tip.actions* und *de.fub.tip.actionforms* befinden sich auf das Rahmenwerk Struts bezogene Klassen. Alle anderen Pakete kapseln und strukturieren die Logik des

Paketname	**Kurzbeschreibung**
de.fub.tip	stellt alle Klassen der Diplomarbeits-implementierung zur Verfügung.
de.fub.tip.actionforms	stellt alle *ActionForm*-Objekte zur Verfügung.
de.fub.tip.actions	stellt alle *Action*-Objekte zur Verfügung.
de.fub.tip.actions.admin	stellt alle *Action*-Objekte zur Verfügung, die im ADMIN-Bereich auftauchen.
de.fub.tip.actions.debug	stellt alle *Action*-Objekte zur Verfügung, die im DEBUGGING-Bereich auftauchen.
de.fub.tip.datenanzeige	stellt Objekte zur Anzeige von Daten zur Verfügung.
de.fub.tip.datenanzeige.beans	stellt Bean-Objekte zur Anzeige von Daten zur Verfügung.
de.fub.tip.datenanzeige.container	stellt *Container*-Objekte zur Anzeige von Daten zur Verfügung.
de.fub.tip.datenanzeige.ormapper	stellt objektrelationale Mapper zur Datenanzeige zur Verfügung. (OR-Mapper)
de.fub.tip.datenbank	stellt Hilfsroutinen beim Datenbank-zugriff zur Verfügung.
de.fub.tip.datenbank.factory	stellt Fabrik-Objekte zur Erzeugung von Anwendungslogik-Objektinstanzen zur Verfügung.
de.fub.tip.datenbank.logik	stellt die Implementierung der Logik-Objekte zur Verfügung.
de.fub.tip.exceptions	stellt eigene in TIP verwendete *Exception*-Unterklassen zur speziellen Behandlung von Ausnahmezuständen zur Verfügung.

Tabelle 5.1: Aufteilung der TIP-Anwendung in Java-Pakete

touristischen Informationssystems. Sie stellen ihre Dienste unabhängig von Struts zur Verfügung.

Die Steuerungsschicht der Anwendung wird durch Klassen aus dem Paket *de.fub.tip.actions* umgesetzt, die zur besseren Unterscheidung alle auf Action enden. Inhaltlich gleichen diese Klassen Ablaufdiagrammen, da sie prinzipiell folgende Struktur haben: nach der Erzeugung eines Geschäftslogikobjektes wird auf diesem eine Methode aufgerufen, die die jeweilige Anwendungslogik ausführt. In Abhängigkeit vom Rückgabewert oder dem Auftreten eines Ausnahmezustands wird auf eine neue Aktion weitergeleitet, der als Parameter das Ergebnis mitgeschickt wird. Die Unterpakete *de.fub.tip.actions.admin* und *de.fub.tip.actions.debug* passen sich der inhaltlichen Gliederung der Anwendung an. Diese hat ein Administrator- und Debugging-Menü. Die jeweils ausführbaren Aktionen sind den jeweiligen Paketen zugeordnet.

Alle Klassen des Paketes *de.fub.tip.actionforms* enden auf ActionForm und beinhalten den Inhalt eines Formulars als Datenstruktur. Die in den Klassen implementierte *validate()*-Methode überprüft die vom Benutzer gemachten Eingaben auf syntaktische Richtigkeit. Exemplare der Formularklassen werden vom Struts-Rahmenwerk verwaltet und in Verbindung mit den entsprechenden Aktionen durch den Anwendungsserver zur Verfügung gestellt.

Die Pakete unter *de.fub.tip.datenanzeige* beinhalten Klassen zur vereinfachten Datenanzeige. Die beiden Schnittstellen *Container* und *ContainerInhalt* unterscheiden Objekte nach Objektkollektionen und Inhalten. Im Paket *de.fub.tip.datenanzeige.beans* befindet sich eine Klasse, die zur Bearbeitung des Profils ein Wertobjekt aus dem Paket *de.fub.tip.datenanzeige.ormapper* in eine geeignete HTML-Darstellung umwandelt und damit den Zugriff auf die Daten aus der Datenbank vereinfacht. Würde Java in der aktuellen Version Generizität unterstützen, wäre das Paket *de.fub.tip.datenanzeige.container* fast ohne Nutzen, den es enthält derzeit spezielle Objektcontainer-Klassen, die einen lesenden Zugriff auf eine Kollektion von Wertobjekten bieten und durch einen einzigen generischen Wertobjektcontainer ersetzt werden können. Die Container werden von der Ansichtsschicht der Anwendung benutzt, um in JSP-Seiten Objektmengen anzuzeigen. In diesen Container-Klassen müssen funktionale Erweiterungen zum Zurückschreiben von Informationen in die Datenbank vorgenommen werden, wenn die Anwendung auch schreibende Zugriffe unterstützen soll.

Die eigentlichen Containerinhalte – also Wertobjekte – befinden sich im Paket *de.fub.tip.datenanzeige.ormapper*. Alle Klassen implementieren die Schnittstelle *ViewObject*. Diese bietet zur vereinfachten Fehlersuche eine standardisierte Methode zur Ausgabe des Objektinhalts mittels log4j. Alle Klassen dieses Paketes enden auf VO (view object).
Spezielle Hilfsroutinen zur Arbeit mit der Datenbank stellt das Paket *de.fub.tip.datenbank* zur Verfügung. Zur vereinfachten Benutzung von Postgis-Methoden in *PreparedStatements* besitzt die Klasse *DBFunktionen* statische Hilfsmethoden. Da die Benutzung des Datenbankpools von Struts nicht mit erweiterten Datentypen von Postgis funktioniert, wird eine separate Komponente zur Verfügung gestellt, die den Datenbankzugriff steuert. Dabei enthält die Klasse *DBKonfig* Informationen zur verwendeten Datenbank, die von den Komponenten benutzt werden, die keine Datenbankverbindung über den Verbindungspool herstellen.

Die Geschäftslogik der gesamten Anwendung ist in den Paketen *de.fub.tip.datenbank.factory* und *de.fub.tip.datenbank.logik* abgelegt. Die Schnittstellen *LogicFactory* und *LogicObject* dienen zur Unterteilung der Objekte in Fabriken und eigentliche Geschäftslogikimplementierungen. Die Fabriken sind gemäß dem Singleton-Muster entworfen und gleichen sich in ihrem Aufbau, da sie konkrete Logikobjekte erzeugen und an das TIP-System ausliefern.

Das Paket *de.fub.tip.datenbank.exceptions* stellt besondere Ausnahmeklassen zur Verfügung. Diese drücken spezielle Anwendungszustände aus, zu denen es im Java-Sprachumfang keine angepassten Fehlermeldungsklassen gibt. Die Klasse *NoUserLoggedInException* beschreibt beispielsweise den Zustand eines illegalen Aktionsaufrufs für den eine Anmeldung erforderlich ist. Diese Ausnahme wird ausgelöst, wenn ein Benutzer versucht, ohne vorherige Anmeldung eine Aktion aufzurufen, die eine Anmeldung erfordert. Sprechende Klassennamen erleichtern die Fehlersuche für Softwareentwickler. Da alle Klassen von der Java-Klasse *Exception* erben, enden sie zur einfacheren Zuordnung alle auf Exception.

5.5 Zusammenfassung

In diesem Kapitel wurden die einzelnen Elemente der Mehr-Schichten-Architektur mit konkreten Implementierungen in Verbindung gebracht. Dazu wurden eigene Erweiterungen der Basisfunktionalität des Struts-

Rahmenwerkes vorgenommen. Diese Klassen enthalten Geschäftslogik und bilden relationale Tupel auf Objekte ab. Die im Kapitel 4 gelegten konzeptionellen Grundlagen zu objektorientierten Entwurfsmustern wurden bei der Umsetzung beachtet und umgesetzt.

Als Ergebnis ist ein serverbasiertes Informationssystem entstanden, das sich durch eine gut dokumentierte Struktur auszeichnet und diverse Rahmenwerke benutzt. Die Einarbeitung in die einzelnen Rahmenwerke war sehr zeitaufwändig. Der Open-Source-Charakter der eingesetzten Werkzeuge ermöglichte, dass viele Dokumentationen und anregende Beispiele im Internet verfügbar sind. Diese haben sich positiv auf den Entwurf der Anwendung ausgewirkt, da aus ihnen viele Ideen zur Problemlösung abgeleitet wurden.

Der Wechsel von einer traditionellen Drei-Schicht-Architektur hin zu einem mehrschichtigen Anwendungslayout ist sehr zeitintensiv bei der Umsetzung, hat aber ein zukunftssichereres Design zur Folge. Durch die Einführung der einzelnen Schichten ist es möglich, von den verwendeten Softwarewerkzeugen zu abstrahieren. Dies ermöglicht beispielsweise die Verwendung eines anderen Anwendungsservers oder Datenbanksystems. Da die einzelnen Komponenten voneinander unabhängig implementiert sind und ihre Funktionalität durch Methoden der Geschäftslogik austauschen, kann die Anwendung auch einfach um neue Funktionalität ergänzt werden. Ein weiterer Vorteil ist, dass die Anwendung durch das Vorhandensein verschiedener Schichten besser für den Ablauf in Rechnerclustern geeignet ist. Die mögliche räumliche Trennung von Datenbank- und Anwendungsserver erfordert keine Veränderung am Quellcode der Software.

Abbildung 5.12 (S. 85) zeigt den Ablauf einer Kommunikationsbeziehung eines Klienten mit dem TIP-System. Zur Benutzung des Informationssystems müssen Klienten ein Menü auswählen. Diese Auswahl resultiert in der Auswertung der generierten URL-Adresse durch die Steuerung des Rahmenwerks. Kann das Rahmenwerk der übergebenen URL eine Aktionsklasse zuordnen, wird ein Exemplar dieser Klasse erzeugt und ausgeführt. Je nach Art der Aktion sind etwaige Benutzereingaben erforderlich. Diese geschehen in der zugeordneten Formularklasse. Nachdem der Benutzer alle Eingaben gemacht hat und diese korrekt validiert wurden, wird an eine JSP-Seite weitergeleitet, die vom Anwendungsserver auf dem mobilen Gerät des Benutzers als HTML-Seite angezeigt wird. Jetzt beginnt der Prozess von vorn und der Benutzer kann eine neue Aktion auslösen oder eine URL manuell in die Adresszeile des Browser eingeben.

Das nächste Kapitel geht näher auf Probleme bei der Implementierung ein. Zusätzlich wird der gemachte Entwurf kritisch bewertet und Verbesserungsmöglichkeiten aufgezeigt.

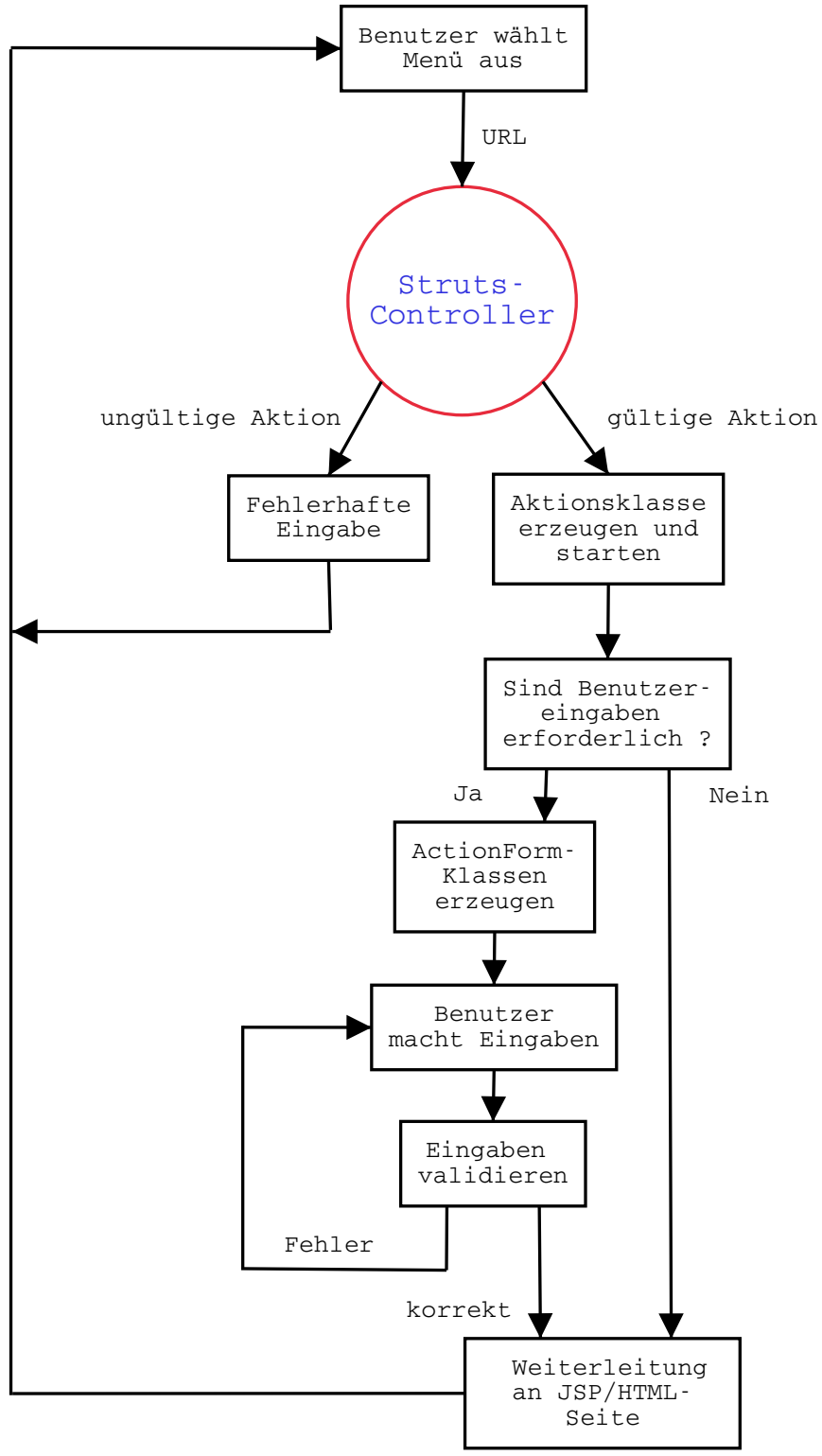

Abbildung 5.12: Ablaufschema der Kommunikation mit dem TIP-System

Kapitel 6

Diskussion

Die Diskussion der im Rahmen der Diplomarbeit entwickelten Software gliedert sich in zwei Teile. Zuerst wird detaillierter auf Probleme bei der Umsetzung eingegangen. Diese gliedern sich in technische Probleme, die durch die eingesetzte Softwareauswahl bedingt sind, und inhaltliche Schwächen, die aus den gemachten Entwurfsentscheidungen abzuleiten sind. Darauf folgt eine Bewertung der gemachten Erfahrungen.

6.1 Probleme bei der Realisierung

Den Einstieg in die Diplomarbeit bildete die Einarbeitung in die Entwurfsmuster und entsprechend verfügbare Rahmenwerke. Dazu wurden diverse Internetforen sowie Vorlesungsunterlagen [35] herangezogen.

6.1.1 Technische Probleme

Die detaillierte Einarbeitung in die Rahmenwerke erfordert ein gutes Verständnis der Zusammenarbeit aller beteiligten Komponenten. Dies sind insbesondere:

- Datenbanksystem
- Betriebssystem
- Anwendungsserver
- Programmierumgebung

Im Allgemeinen lässt sich feststellen, dass die Konfiguration der einzelnen Schichten mit viel Arbeit verbunden ist. Die oft umfangreichen Dokumentationen müssen durchgearbeitet werden, um entsprechende Konfigurationsdateien manipulieren und anpassen zu können.

Im Fall des Datenbanksystems fiel dies nicht so schwer, da eine interaktive Text-Konsole existiert, die durch SQL-Abfragen eine Sicht auf die Daten ermöglicht. Da hier das Datenmodell einer anderen Diplomarbeit [33] verwendet wurde, existierte keine formale Dokumentation, wie dies die anderen Softwarewerkzeuge und Rahmenwerke bieten. Ebenso entfiel die Möglichkeit, Hilfe durch Internetforen o.Ä. in Anspruch zu nehmen. Stattdessen diente nur die Text-Konsole als Navigator durch das Datenmodell, das zusätzlich durch die eingepflegten Beispieldaten klarer wurde.

Das als Grundlage benutzte Datenmodell bietet mit seiner hierarchischen Datenspeicherung einen interessanten Ansatz zur Modellierung im Rahmen touristischer Informationssysteme, der auf Grund von Zeitmangel hier nicht vollständig funktional implementiert wurde. Eine mehrsprachenfähige Speicherung von Daten wird derzeit nicht unterstützt. Dies würde zu umfangreichen Änderungen am Tabellenschema und den hinterlegten Beispieldaten führen.

Die Kombination aus Postgis und Postgres führte zu Problemen bei der gemeinsamen Installation. Das Datenbanksystem allein lässt sich genau nach Anleitung auf einem Linux-Betriebssystem installieren. Die Integration von Postgis gelingt nicht nach Anleitung, wodurch der Wert der Erweiterung absinkt.

Die Verwendung und Einbeziehung geografischer Daten verhindert eine problemlose Migration der Datenbank auf eine andere SQL-kompatible, da Erweiterungen anderer Hersteller nur bedingt kompatibel zu den Postgis-Erweiterungen sind. Generell gestaltet sich eine Datenbankmigration schwierig, da trotz Einhaltung des SQL-Standards jeder Hersteller eigene Erweiterungen an Datentypen und -formaten in sein Produkt integriert, um es für einen speziellen Kundenkreis attraktiver zu machen.

Geeigneter wäre hier die Verwendung eines Datenbankmanagementsystems, das geografische Erweiterungen (GIS) als Standard in seinem Funktionsumfang hat. Dies wird bisher von keinem Open-Source-Datenbanksystem umgesetzt. Als positiv erwiesen sich die vom Datenbanksystem zur Verfügung gestellten verschiedenen Zugänge zu den eigentlichen Daten. Dazu wurden hier hauptsächlich die Text-Konsole und der Zugriff mittels

JDBC verwendet. Unter Linux war dies problemlos möglich und erleichterte die Softwareentwicklung und Fehlersuche, da die mit dem TIP-System gemachten Datenbankänderungen gleich in der Text-Konsole angezeigt werden konnten.

Eine im Sinne der Verfügbarkeit noch zuverlässigere Programmentwicklung kann erreicht werden, wenn zum Zweck der Anwendungsentwicklung Datenbank- und Anwendungsserver auf dem gleichen Rechner laufen. Dadurch verliert man die Abhängigkeit von funktionierenden Internetverbindungen zur Anwendungsentwicklung. Während der Arbeit an der Diplomarbeit stellte sich die oft nicht funktionierende Verbindung zur Datenbankinstanz an der Freien Universität Berlin als problematisch und produktivitätshemmend heraus.

Die Wahl von Linux als zu Grunde liegendes Betriebssystem kann grundsätzlich empfohlen werden, da die modulare Struktur des Systems ein leichtes Hinzufügen von Softwarekomponenten erlaubt. Durch den Mehrbenutzerbetrieb ist eine rein konsolenbasierte (shell) und oberflächenbasierte grafische Arbeit (X-Windows) am System gleichzeitig möglich. Die Integration diverser frei verfügbarer Editor-Programme und Verwaltungswerkzeuge ins Betriebssystem erleichtert die Konfiguration und Verwaltung zusätzlich. Diese Tatsache ändert nichts an der anfangs komplizierten Einrichtung der einzelnen Softwarekomponenten. Deren reine Installation geht sehr einfach, da es sich um fertige Pakete mit Installationsroutinen handelt. Die Konfiguration der Anwendung erfordert einen hohen Einarbeitungsaufwand in die Prinzipien der Softwarearchitektur jedes Programmpaketes und erforderte eine daran angepasste Struktur des TIP-Systems, die im Kapitel 5 näher erläutert wurde.
Unter Windows-Betriebssystemen wird das automatische Starten des Anwendungsservers stark vereinfacht, da es mit Hilfe eines grafischen Verwaltungswerkzeugs konfiguriert werden kann. Unter Linux müssen dafür Startskripte ins System eingebunden werden, was je nach verwendeter Distribution auch durch Werkzeuge unterstützt wird (SuSE: YaST2).

Die anwendungsspezifische Konfiguration erfolgt im Fall des Anwendungsservers Jakarta Tomcat durch Editieren von XML-Konfigurationsdateien. Die Apache Software Foundation [58] stellt dazu Dokumentationen im WWW bereit. Unter Einbeziehung von Suchmaschinen und Internetforen gelingt es auch, vorgefertigte Konfigurationsdateien oder Ausschnitte daraus aufzufinden, die den Einstieg erleichtern. Trotz

etwaiger Funde war die Anpassung des Struts-Rahmenwerkes an den Anwendungsserver und die des touristischen Informationssystems an das Struts-Rahmenwerk sehr zeitaufwändig. Zur Anpassung der Konfigurationsdateien wurden manuell angefertigte Ablaufskizzen benutzt, aus denen sich auch die Anforderungen an etwaige Geschäftslogik- oder Aktionsklassen ergaben. Als nachteilig erweist sich bei der Arbeit mit XML-Dateien, dass kleine Änderungen oftmals unerwünschte Auswirkungen haben. Im vorliegenden Fall hat das falsche Setzen eines Kommentarende-Tags den Rest der Datei als Kommentar markiert. Dies wurde von den verwendeten Editoren nicht richtig farblich deutlich gemacht und hatte nach einem Neustart der Anwendung Fehlermeldungen zur Folge, die keinen Hinweis auf die wahre Fehlerursache gaben. Die falsche Art der Kommentierung ist `<!--Kommentar--->`, während `<!--Kommentar-->` einwandfrei funktioniert. Man muss genau darauf achten, dass Kommentare mit nur zwei --Zeichen beendet werden.
Zusätzlich zu solchen syntaktischen Feinheiten dürfen in Konfigurationsdateien keine Sonderzeichen verwendet werden. Auch in Kommentaren enthaltene Umlaute führen zu sonderbaren Fehlermeldungen. Dies liegt wahrscheinlich daran, dass die Konfigurationsdateien nicht unter Verwendung von UTF (Universal Transformation Format - Character Encoding) sondern gemäß sprachspezifischer Regeln kodiert sind.

Bei der Arbeit mit dem Anwendungsserver kam es sowohl unter Linux als auch Windows zu unerwarteten Nebenwirkungen. Beim Anwendungsstart initialisiert der Anwendungsserver alle vorhandenen Anwendungen sequentiell nacheinander. Deshalb benötigt er eine gewisse Zeit, bis er Anfragen von Klienten bearbeiten kann. Bei Verwendung der Java-Laufzeitumgebungen *JDK 1.5.0_beta* und *JDK 1.4.2_04* blockierten die virtuellen Maschinen (JVM) den Prozessor und führten zu einem starken Anstieg des Speicherbedarfs. Die CPU verharrte bei 100%-Auslastung, obwohl der Rechner keine Anfragen bearbeitete und die Initialisierung des Anwendungsservers abgeschlossen war. Anfangs bestand der Verdacht, dass es sich hierbei um Entwurfsfehler des TIP-Systems handelt. Dies konnte durch entsprechende Statusausgaben ausgeschlossen werden. Dieser Fehler in den Java-Implementierungen erschwerte die Arbeit erheblich, da andere Programme, die ebenfalls Java benutzen wollten, regelmässig abstürzten. Dazu gehört auch die Entwicklungsumgebung Eclipse. Auch nach einem Versionwechsel von Anwendungsserver und Entwicklungsumgebung bestanden die gleichen Probleme weiterhin.

Die Integration des Anwendungsservers in die Ant-Konfigurationsdatei basiert darauf, dass sich Ant mit der Konfigurationsoberfläche des Anwendungsservers verbindet und die gewünschte Aktion, z.B. einen Anwendungsneustart, ausführt. Dazu müssen Ant bestimmte tomcateigene Bibliotheken bekannt gemacht werden.
Bei schnell hintereinander ausgeführten Neustarts des Anwendungsservers funktionierte die Initialisierung der Anwendung nicht mehr richtig. Aus diesem Grund wurde stattdessen ein Linux-Shell-Skript benutzt, das den Anwendungsserver stoppt, die aktuelle von Ant generierte WAR-Datei einbindet und neu startet. Die Kombination von Ant und Shell-Skript erwies sich bei der TIP-Implementierung als zuverlässiger als die reine Benutzung von Ant. Die Erstellung des Ant-Skriptes `build.xml` war sehr zeitaufwändig, da die Einbindung externer JAR-Bibliotheken schwierig ist. Diese Dateien müssen sowohl dem Ant-Programm selbst als auch innerhalb der Konfigurationsdatei bekannt gemacht werden. Dieser Mechanismus ermöglicht eine einfache Einbindung von Java2html und StatCvs – siehe Abschnitt 5.3.2.2 (S. 65).

Die Struts-Konsole [19] wurde erst am Ende zur Konfiguration der `struts-config.xml`-Datei eingesetzt. Das Programm gibt zwar einen grafischen Überblick über die Konfiguration, ist aber beim Hinzufügen neuer Aktionen nur begrenzt hilfreich. Hier ist ein einfacher Texteditor nützlicher, da man mittels Copy&Paste eine bestehende Aktion wiederverwenden und anpassen kann. Die Struts-Konsole verändert beim Speichern den Dokumentinhalt und kann dabei die Lesbarkeit verringern, da Tabulatoreinrückungen oder Zeilenumbrüche entfernt werden. Ein Vorteil der Software ist, dass durch manuelles Editieren verursachte Fehler angezeigt werden können. Dazu wird die Konfigurationsdatei gegen eine Dokumenttypbeschreibung abgeglichen. Eine genaue Fehlerbeschreibung liefert die Struts-Konsole nicht, zeigt jedoch an, wenn die Konfigurationsdatei nicht mehr standardkonform aufgebaut ist.

Die Arbeit mit der Entwicklungsumgebung Eclipse erwies sich bis auf die Probleme durch die Java-Abstürze als sehr angenehm. Die zur Verfügung stehenden Quellcodegeneratoren vereinfachten die Erzeugung der Klassenrümpfe des touristischen Informationssystems. Zusätzlich bietet Eclipse eine grafische Oberfläche zur Integration mit CVS. Die bei der Arbeit aufgetauchten Probleme mit CVS sind auf strukturelle Schwächen von CVS zurückzuführen. Unter Linux verfügbare Zusatzinformationen zu Dateien (Zugriffsrechte und Eigentümer) werden nicht mit im Repositorium abgelegt. Dies führt beim Einchecken von Shell-Skripten, die unter Linux ein spezielles

Ausführungsattribut besitzen, zu Problemen.

Inzwischen existiert ein Versionierungssystem, das diese Schwächen umgeht und erweiterte Funktionalitäten bietet. In zukünftigen Projekten sollte dieses System Vorrang vor CVS haben. Subversion [51] bietet einen atomaren Versionswechsel im Repositorium. Im Vergleich zu CVS, das für jede einzelne Datei eine Historie mitführt, wird bei Subversion ein kompletter Versionswechsel des Moduls ausgeführt, sobald eine neue Datei hinzugefügt wird. Dies erleichtert die Fehlersuche, da man die gesamte Anwendung zu einem Versionszeitpunkt aus dem Repositorium auslesen kann. Für verteilt arbeitende Entwickler hat dies zur Folge, dass ein gleichzeitiges Einchecken der Projektänderungen ins Repositorium keinen inkonsistenten Zustand zur Folge hat. Die tägliche Arbeit erleichtert Subversion auch dadurch, dass gezielt Datei- und Verzeichnisstrukturen innerhalb des Repositoriums verschoben oder umbenannt werden können. Beim Einsatz von CVS musste eine Datei an der alten Stelle gelöscht und an der neuen manuell hinzugefügt werden. Dabei wurden die Versionsinformationen nicht automatisch von der alten an die neue Position mitkopiert und verhinderten damit ein Nachverfolgen der Projektarbeit.

Der Einsatz von CVS erwies sich zu Zwecken der Fehlersuche als sehr nützlich, da oftmals Vergleiche mit anderen Versionen des Repositoriums potentielle Fehlerquellen offenbart haben. Die Einbindung von CVS in Eclipse erleichterte die Arbeit mit dem Repositorium. Durch die Probleme mit dem Java-Compiler dauerte das erneute Einlesen eines Projekts umständlich lange. Hierbei erwies sich die Fähigkeit von CVS, über verschiedene Schnittstellen bedienbar zu sein, als sehr nützlich. Im Fehlerfall konnten die Änderungen über die Textkonsole ins Repositorium geschrieben werden. Anfangs wurde auf ein CVS-Repositorium an der FU Berlin zurückgegriffen. Da die Benutzer- und Gruppenhierarchie der Universitätsinfrastruktur nicht mit der auf dem Entwickler-PC vergleichbar ist, kam es zu Problemen beim Herunterladen der Repositoriumsinhalte. Diese konnten zwar mittels SSH verschlüsselt und sicher transportiert werden, erhielten aber auf dem lokalen Rechner beispielsweise nicht ausreichende Zugriffsrechte, so dass die Entwicklungsumgebung keine Schreibrechte hatte.

Für die gesamte Anwendungsentwicklung kam CVS auf einem lokalen Rechner zum Einsatz und diente in Zusammenarbeit mit den Visualisierungswerkzeugen als Zeugnis des Fortschritts der Arbeit am touristischen Informationssystem. Im Gegensatz zur naiven Herangehensweise den Quell-

code mit einem einfachen Texteditor zu entwickeln, wurde das TIP-System durch Anwendung entsprechender Softwarewerkzeuge vom Beginn an gut dokumentiert und strukturiert, was es von reinen prototypischen Entwicklungen qualitativ abhebt. Hinzu kommt die strikte Dokumentation innerhalb des Quellcodes, die durch die Beschreibung von Aktivitäten am Projekt, die im Logbuch des CVS-Repositoriums gespeichert sind, ergänzt wurde. Damit unterscheidet sich TIP von im Kapitel 3 erwähnten Implementierungen, über deren eigentliche Struktur wenig bekannt ist.

6.1.2 Inhaltliche Probleme

Als inhaltlich problematisch erwies sich die Zusammenarbeit des strutseigenen Datenbankpools mit Postgres und Postgis. Beim Zugriff mit JDBC auf eine Datenbank, in welche die Postgis-Erweiterung einkompiliert ist, müssen die speziellen geometrischen Datentypen der Datenbankverbindung manuell hinzugefügt werden. Dieser Zugang ist umständlich. Bei der Benutzung des Struts-Datenbankpools fiel auf, dass Datenbankverbindungen aus dem Pool die Fähigkeit verlieren, Postgis-Datentypen hinzufügen zu können. Die Datentypen müssen nur dann manuell hinzugefügt werden, wenn in den Ergebnismengen der SQL-Abfrage geometrische Daten enthalten sind. Aus diesem Grund stellen Komponenten, die geografische Daten anfragen, separat eine Verbindung zur Datenbank her und geben diese nach Benutzung wieder frei. Dies verhindert einen Leistungszuwachs, der durch Verwendung des Datenbankverbindungspools erreicht werden soll. Die anderen Komponenten arbeiten fehlerfrei und schnell mit der Datenbank, da sie auf bereits geöffnete Verbindungen des Pools zugreifen können. Schließt eine Komponente die Datenbankverbindung wieder, geht diese zur weiteren Benutzung an den Pool zurück; bleibt also technisch weiterhin geöffnet.

Die Konfiguration des Struts-Datenbankpools erwies sich als komplexer als erwartet. Die benötigten JAR-Bibliotheken und JDBC-Treiber der Datenbank müssen dem Anwendungsserver und Struts separat bekannt gemacht werden. Zusätzlich müssen die benötigten Bibliotheken dem touristischen Informationssystem als Anwendung bekannt gemacht werden. Dieser Umstand ist auf die modulare Struktur des Anwendungsservers zurückzuführen. Er bietet damit die Möglichkeit, dass Anwendungen ihre eigenen Bibliotheken oder die des Anwendungsservers benutzen können.

Die nicht ausreichende Dokumentation des bestehenden Datenmodells resultierte in einem hohen Einarbeitungsaufwand. Rückblickend betrachtet, wäre ein Neuentwurf mit entsprechender Dokumentation vielleicht effizienter gewesen, da so Anwendungsentwicklung und Datenbankentwurf mehr Hand in Hand hätten gehen können. Stattdessen wurde aus den eingepflegten Beispieldaten die Logik des Datenmodells abgeleitet und im Softwareentwurf in Java-Klassen entsprechend eingefügt.

Die durch das Rahmenwerk Struts eingebundenen Mechanismen zur Manipulation der Ansichtsschicht, wozu insbesondere die eigenen Tag-Bibliotheken zählen, erwiesen sich als sehr nützlich. Trotz des nötigen Aufwands der Einarbeitung wird die weitere Anwendungsentwicklung beschleunigt, wenn die allgemeinen Prinzipien verstanden und verinnerlicht sind.
Das Prinzip der Vermeidung von Java-Quellcode in den JSP-Seiten wurde strikt eingehalten. Zur Abbildung etwaiger Anzeigelogik wurden nur standardkonforme JSTL- und EL-Konstrukte eingesetzt. Die durch die Tag-Bibliothek bereitgestellte Nutzung von Auswahlelementen (checkbox) entspricht nicht ganz der Benutzung, die mit Wertobjektcontainern möglich ist. Am Beispiel der Themenprofilanzeige soll dieser Zusammenhang näher erläutert werden. Das Wertobjekt *ThemaVO* speichert, ob es vom aktuellen Benutzer abonniert ist. Diese Information im Wertobjekt zu speichern macht Sinn, da diese referentiell unabhängig von der Datenbank existieren und durch die Datenhaltungszugriffsschicht zur Verfügung gestellt werden. Jeder angemeldete Benutzer erhält also seine eigene Themenauswahl. Diese Themen sind in einem Container gespeichert. Eine sinnvolle Implementierung eines HTML-Checkbox-Tags sollte den Container iterieren und durch Auslesen des *abonniert*-Attributs bestimmen, ob in der Ansicht eine Auswahl zu treffen ist oder kein Häkchen gesetzt werden muss. Abbildung 7 (S. 130) zeigt das Aussehen des TIP-Systems bei der Änderung des Sehenswürdigkeitsprofils. Die logische Struktur gleicht der Änderung des Themenprofils. Diese Darstellung der Inhalte wird in der JSP-Seite unter Benutzung von if-Abfragen zur Verfügung gestellt und ermöglicht so eine korrekte Anzeige. Da bei einem Neuladen der jeweiligen HTML-Seite im Browser nur der entsprechende Objektcontainer ausgelesen wird, bleiben die gemachten Auswahlen erhalten. Traditionell löschen Browser beim Neuladen einer Seite alle Auswahlelemente in HTML-Formularen und erschweren die Benutzung, da sie bei komplexen Transaktionen einen hohen Mehraufwand durch doppeltes Eingeben verursachen.

Die strutseigene Implementierung des HTML-Checkbox-Tag-Elements geht davon aus, dass aus einer Menge von Objekten Teilmengen ausgewählt werden. Dazu muss eine Ergebnismenge und eine Gesamtmenge zur Verfügung gestellt werden. Aus der Gesamtmenge überträgt Struts dann bei Auswahl durch den Benutzer jeweils Elementkopien in die Ergebnismenge. Die Ergebnismenge wird beim Absenden des Formulars zur weiteren Verarbeitung zur Verfügung gestellt. Eine Benutzung der Wertobjekt-Container ist hierdurch erschwert. Diese würden Elemente aus sich selbst hin- und herkopieren und nicht im Sinne des Container-Konzepts einen vereinfachten Objektzugriff gestatten.
Der hier implementierte Ansatz, den Container zur Anzeige zu visualisieren, ermöglicht auch eine erweiterte Semantik beim Zurücksetzen (reset) des Formulars. Statt alle Häkchen bei den Auswahlboxen zu entfernen, wie dies die Standardimplementierung eines Formularzurücksetzens tut, wird der aus der Datenbank gelesene Container angezeigt. Dies vereinfacht die Profilbearbeitung, da der Benutzer im Fall von Irrtümern zum zuletzt in der Datenbank gespeicherten Zustand zurückkehren kann.

Im Allgemeinen erscheint eine Verwendung von Weboberflächen in echtzeitfähigen Systemen als nicht optimal. Im Fall eines touristischen Informationssystems wirken sich hohe Latenzzeiten negativ aus, da der Benutzer pro Zeiteinheit an den Dienstanbieter zahlen muss, was derzeit das Standardabrechnungsmodell für mobile Netzdienste ist. Prinzipiell lassen sich Ausführungszeiten von Weboberflächen nie mit absoluter Sicherheit bestimmen, da derzeit in den verwendeten Netzen kein zeitlich gesicherter Transport von Paketen (Quality of Service (QoS)) implementiert ist. Hinzu kommt, dass beim Erstaufruf von Aktionen in Struts die entsprechenden dynamischen Seiten vom Server kompiliert und bereitgestellt werden müssen. Diese Latenzzeit kann man jedoch durch einen gut ausgestattenen Anwendungsserver absenken, der die serverseitige Bearbeitung beschleunigt.

6.2 Erfahrungsgewinne

Die Arbeit an der Diplomarbeit mit ihren technischen und inhaltlichen Herausforderungen hat mir großen Spaß gemacht und einen hohen Wissenszuwachs bewirkt.

Die Einarbeitung in komplexe Rahmenwerke hat die Kraft und konzeptionelle Macht von Open-Source-Technologien und freien Standards gezeigt.

Diese bieten durch ihre Vielfalt und gute Dokumentation die Möglichkeit, das Verständnis der jeweiligen Projektentwickler in einem bestimmten Kontext besser zu verstehen. Durch die frei verfügbaren Quelltexte und Beispiele kann man somit ein konkretes Problem beim Entwurf oder der Implementierung eines Softwaresystems, durch die verschiedenen Rahmenwerke jeweils anders gelöst, betrachten.
Daraus resultiert ein hoher Grad von Freiheit bei der Erstellung eigener Anwendungen, da die vielen Rahmenwerke miteinander kombinierbar sind. Im Rahmen des Entwurfs des TIP-Systems Softwareentwicklung wurden hier vorwiegend Rahmenwerke und Werkzeuge der Apache Software Foundation eingesetzt.

Als sehr nützlich erwies sich auch das Vorhandensein vieler Internetforen und freiwillig ins Internet gestellter Ressourcen. Diese ließen sich durch Suchmaschinen ausfindig machen. Die im Internet verfügbaren Dokumente dienten als Hilfestellung und boten neue Ideen zur Lösung von Problemen während der Anwendungsentwicklung.

In Open-Source-Projekten mit hoher Verbreitung macht sich eine gewisse Ethik breit: möglichst gute Lösungen zur Verfügung zu stellen. Dies ist eher eine Philosophie, die aber trotzdem in die tägliche Arbeit integriert werden sollte, um Sicherheitslücken und Programmfehler zu minimieren. Der Open-Source-Ansatz sichert keine vollständige Fehlerfreiheit zu, bietet aber die Möglichkeit, durch direkten Blick in den Quellcode der verwendeten Werkzeuge potentiell kritische Programmstellen zu finden und zu deren Behebung beizutragen. Diese beim Entwurf und der Umsetzung von Anwendungssystemen beachtete Ethik führt langfristig zu hoher Qualität im Vergleich zu Lösungen, die naiv und unter hohem Geld- und Zeitdruck entstehen.
Bei der Implementierung des touristischen Informationssystems wurde auch aus diesem Grund auf eine gute Dokumentation der einzelnen Anwendungsteile geachtet. Andere Entwickler sollen schnell in die Lage versetzt werden, Änderungen am TIP-System durchzuführen oder dessen Struktur nachvollziehen zu können.

6.3 Zusammenfassung

Die Anforderung an die Softwarearchitektur, modular für zukünftige Erweiterungen zu sein und auf offenen Standards und freier Software zu basieren, wurden erfüllt. Die verwendeten Rahmenwerke ermöglichen die kon-

textabhängige Verbreitung von Informationen. Dazu wird ein ortsabhängiges verteiltes mehrschichtiges Anwendungssystem zur Verfügung gestellt, dessen Architektur durch einen hohen Grad der Modularisierung und Virtualisierung durch Schichten gekennzeichnet ist.

Das in dieser Arbeit entstandene Softwaresystem hat einen geringeren Funktionsumfang als bei Beginn der Arbeit erhofft, was auf den anfangs unterschätzten Aufwand zur Einarbeitung in das Datenmodell und die benutzten Technologien zurückzuführen ist. Die während der Arbeit gesammelten Erfahrungen haben gezeigt, wie vielfältig verteilte Informationssysteme implementiert werden können. Der Aspekt der Ortsabhängigkeit und die Verwendung mobiler Geräte stellen dabei große Herausforderungen an die Flexibilität der Anwendung. Die gemachten Annahmen, dass alle mobilen Geräte zukünftig eigene IP-Adressen haben werden und ihr Funktionsumfang zunehmen wird, lassen die Implementierung einer browserbasierten Lösung sinnvoll erscheinen, da diese somit auch auf zukünftigen Gerätegenerationen zum Einsatz kommen kann.

Mögliche mobile Erweiterungen und Verbesserungen am Gesamtsystem werden im nächsten Kapitel als Ausblick geboten.

Kapitel 7

Zusammenfassung und Ausblick

Dieses Kapitel fasst alle Diplomarbeitskapitel kurz zusammen und beschreibt mögliche inhaltliche Verbesserungen des TIP-Systems. Danach werden Ideen zur zukünftigen Weiterarbeit im Kontext dieser Diplomarbeit vorgestellt.

Im Rahmen der Diplomarbeit wurde eine erweiterbare Softwarearchitektur geschaffen, deren Sinn die Verbreitung kontextsensitiver Informationen ist. Dies setzt voraus, dass das System in der Lage ist, verschiedene Kontexte zu erkennen und zu verwalten. Hier geschieht dies einerseits basierend auf der aktuellen Position des Benutzers und andererseits durch Definition eines Interessenprofils. Damit lässt sich das entwickelte touristische Informationssystem in den Bereich ortsabhängiger Dienste einordnen.

Anfangs sollte die Anwendung aus zwei Komponenten bestehen. Eine mobile Erweiterung sollte die direkte Benutzer-Benutzer-Kommunikation nach dem P2P-Paradigma unterstützen. Im Gegensatz zu existierenden Ansätzen, die gerätespezifische Implementierungen bieten, sollte hier Java als Grundlage dienen. Die verfügbare mobile Ausgabe (J2ME) wurde getestet und eignete sich in der derzeitigen Ausführung nicht zur Entwicklung und Umsetzung komplexer Anwendungssysteme. Im aktuellen Standardisierungsprozess von J2ME finden sich Vorschläge, die J2ME um Fähigkeiten zur Netzprogrammierung erweitern und somit einen geeigneteren Funktionsumfang zur Entwicklung verteilter Informationssysteme zur Verfügung stellen.

Hauptproblem bei der Nutzung mobiler Geräte ist die geringere Leistungsfähigkeit im Vergleich zu Standgeräten bei der Anzeige grafischer Inhalte und der Kommunikation in Netzen. Aus diesem Grund wurde eine browserbasierte Lösung präferiert, die an das mobile Gerät geringe Anforderungen stellt. Läuft die Anwendung in einem Internetbrowser ab, muss keine

gerätespezifische Softwareimplementierung ausgeliefert werden. Beim Entwurf des touristischen Informationssystems wurden vorwiegend offene Standards und frei verfügbare Software eingesetzt. Zusätzlich bestand das Ziel, eine offene und modulare Architektur zu schaffen, in der es möglich ist, neue Funktionen und zukünftig verfügbare Geräteerweiterungen zu integrieren. Dazu wurde eine Mehr-Schichten-Architektur implementiert, deren Vorteile eine hohe Flexibilität und Modularisierung der Anwendung sind.

Bei der eigentlichen Umsetzung zeigten sich vorwiegend technische Schwierigkeiten bei der Konfiguration und Inbetriebnahme der einzelnen Anwendungskomponenten. Die Problemlösung erforderte ein hohes Maß an zeitlichem Aufwand. Inhaltliche Schwächen zeigten sich bei der Verwendung des Datenmodells und der geografischen Erweiterung. In einer zukünftigen Version sollte an diesen Stellen mehr Flexibilität geschaffen werden, um die Fähigkeit zur Internationalisierung des touristischen Informationssystems noch besser ausnutzen zu können. Die in Kapitel 4 vorgestellten Entwurfsmuster wurden erfolgreich in das TIP-System integriert.

Um die Anwendung erweiterbarer und lesbarer zu gestalten, wurden verschiedene Visualisierungswerkzeuge – Javadoc, Java2html [36] und StatCvs [49] – eingesetzt. Dadurch soll die gemachte Arbeit als Grundlage für weitere Tätigkeiten in diesem Interessengebiet dienen. Zur weiteren Bewertung der Diplomarbeitsinhalte werden im Folgenden systemeigene Verbesserungen und zukünftige Entwicklungen vorgeschlagen.

7.1 Systemimmanente Verbesserungen

Die Verwendung des Struts-Rahmenwerkes hat zur Folge, dass das touristische Informationssystem nur innerhalb eines Webbrowsers auf mobilen Geräten der Klienten funktioniert. Die übertragenen Inhalte sind technisch in Form von Dateien oder Texten realisiert. Dies erschwert eine einfache Integration zeichenbasierter Sprachen, wie sie im asiatischen Sprachraum vorherrschen.

Eine bessere Darstellung dieser Sprachen kann durch Verwendung von XML und XSLT umgesetzt werden. Dazu existiert eine Erweiterung des Struts-Rahmenwerks: StrutsCX [50], deren Ansichtskomponente komplett ohne JSP und Tag-Bibliotheken arbeitet. Stattdessen kommen XML und XHTML zum Einsatz. Zu Beginn der Diplomarbeit war das StrutsCX-Projekt noch in einer frühen Testphase, sodass mit der Struts-

Standardversion gearbeitet wurde. Durch die strikte Einhaltung der MVC-Architektur kann man jedoch die StrutsCX-Erweiterung in die bestehende TIP-Anwendung integrieren und das touristische Informationssystem für mehr Sprachen verfügbar machen. Ein weiterer Vorteil der Verwendung von XML als zu Grunde liegender Technologie ist die Fähigkeit, verschiedene Ausgabeformate zu generieren. Dazu gehören WML, PDF und XHTML. In Verbindung mit einer Verbesserung des Datenmodells kann die Anwendung somit für eine Benutzung unter realen Bedingungen tauglicher gemacht werden.

Die im Rahmen der Arbeit verworfene mobile Erweiterung kann weiter verfeinert werden. Basierend auf anderen Technologien lassen sich Forschungsergebnisse im Rahmen der Informationsverteilung in P2P-Netzen integrieren. Ein möglicher Ansatz ist die Verwendung von Rendezvous [4]. Dieses von der Firma Apple genutzte Protokoll wird derzeit zur Lokalisierung von Ressourcen in mobilen Netzen benutzt. Das an der FU entwickelte Heureka [34] stellt ein System zur Ressourcenentdeckung in Netzen zur Verfügung und basiert auf dem Namensdienst des Internets (DNS). Dessen Hierarchie wird zum Propagieren von neuen Netzelementen benutzt. Vielleicht gelingt es, dieses Prinzip zur Verteilung von touristischen Informationen einzusetzen. Die Hierarchie fungiert dabei auch als räumliche Abgrenzung und könnte im Fall eines Touristeninformationssystems zum Datenaustausch mit nahen Teilnehmern benutzt werden.

Um das implementierte System zu einem kompletten touristischen Informationssystem umzugestalten, bedarf es einer erweiterten Funktionalität der Anwendung. Derzeit hat das Informationssystem eher prototypischen Charakter, zeigt aber das Potenzial modular aufgebauter Anwendungen. Nach im Rahmen der Diplomarbeit erfolgter komplizierter Konfiguration können neue Funktionen und Benutzeraktionen nun einfach hinzugefügt werden. Die hier implementierte Mehr-Schichten-Architektur kann dabei von den neuen Komponenten benutzt werden, die auf die fertig gestellte Datenhaltungs- und Datenhaltungszugriffschicht zurückgreifen. Alle bisher implementierten Komponenten lassen sich wiederverwenden, da sie so allgemein wie möglich implementiert wurden.

7.2 Test und Vergleich mit anderen Ansätzen

Zum Vergleich der TIP-Anwendung mit bestehenden Informationssystemen sollten Belastungs- und Leistungstests von Datenbank- und Anwendungsserver durchgeführt werden. Dadurch könnte die technische Leistungsfähigkeit der ausgewählten Komponenten quantifiziert werden. Interessant wäre in diesem Umfeld auch, ob die gewählte Datenbank- und Anwendungsserverkombination bei den Tests noch zuverlässig arbeitet, da es sich um nichtkommerzielle Softwareprodukte handelt.

Andere verfügbare Systeme – siehe Kapitel 3, ab S. 15 – geben keine Auskunft über das zu Grunde liegende Design der Anwendung und lassen nur Vermutungen zu. Der hier umgesetzte Ansatz baut auf offene Standards und bietet eine umfangreiche Dokumentation sowohl des Quellcodes als auch der einzelnen Anwendungskomponenten. Damit ist eine schnelle Weiterentwicklung der Anwendung möglich. Zur weiteren Verbesserung der Anwendung sollten Benutzerakzeptanztests durchgeführt werden. Die daraus gewonnenen Erfahrungen können in eine Weiterentwicklung der Anwendung einfließen. Zur Verbesserung der grafischen Oberfläche und Navigation innerhalb der Anwendung sollten umfangreiche Tests auf mobilen Geräten durch reale Benutzer vorgenommen werden. Diese Benutzertests könnten auch Anregungen für zusätzlich zu implementierende Funktionen geben. Zusätzlich zur Benutzung durch Testbenutzer könnten geeignete Szenarien simuliert werden, um sowohl das Datenbankmodell als auch die gewählte Anwendungsserver-Datenbank-Kombination auf ihre Leistungsfähigkeit hin zu überprüfen. In Abhängigkeit vom gewählten Szenario kann damit auch die Fähigkeit der Anwendung getestet werden, in einem Netzcluster abzulaufen.

Durch die vielen verschiedenen Rahmenwerke existieren auf der Softwareseite unzählige Ansätze zur Implementierung verteilter Informationssysteme. Jede dieser Entwicklungen hat in einem bestimmten Kontext gegenüber anderen Vorteile. Im Zuge dieses vergrößerten Softwareangebots bleibt offen, inwieweit es für die einzelnen Produkte und Standards auch mobile Geräte gibt, die den Anforderungen der Software genügen. Die Hardwarehersteller haben somit großen Einfluss auf die Weiterentwicklung und Verbreitung plattform- und herstellerunabhängiger Standards und die Schaffung neuer Anwendungssysteme.

Das im Rahmen dieser Diplomarbeit entwickelte modulare touristische Informationssystem bietet die Verteilung kontextsensiviter Informationen und kann auch auf andere Netztopologien ausgeweitet werden, wenn entsprechende Hardware- und Softwareunterstützung zur Verfügung stehen.

Verzeichnisse

Abkürzungsverzeichnis

aGPS Assisted Global Positioning System

API Application Programming Interface

AWT Abstract Window Toolkit

CLDC Connected Limited Device Configuration

CPU Central Processing Unit

CSS Cascading Stylesheet

CVS Concurrent Versioning System

DNS Domain Name Server / Service

DTD Document Type Definition

EAR Enterprise Application Archive

EJB Enterprise Java Bean

EL Expression Language

FU Freie Universität Berlin

GIS Geographic Information System

GPRS General Packet Radio Service

GPS Global Positioning System

GSM Global System for Mobile Communications – Groupe Spécial Mobile

GUI Graphical User Interface

HTML HyperText Markup Language

HTTP HyperText Transfer Protocol

I/O Input / Output

IDE Integrated Development Environment

J2EE Java 2 Platform – Enterprise Edition

J2ME Java 2 Platform – Micro Edition

J2SE Java 2 Platform – Standard Edition

JAD Java Application Descriptor

JAR Java Archive

JDBC Java Database Connectivity

JPEG Joint Photographic Experts Group

JSP Java Server Pages

JSTL JSP Standard Tag Library

JVM Java Virtual Machine

KVM K Virtual Machine

LAN Local-Area Network

MIDP Mobile Information Device Profile

MVC Model View Controller

NAT Network Address Translation

ODBC Open Database Connectivity

ORDBMS Objekt-Relational Database Management System

P2P Peer-To-Peer

PDA Personal Digital Assistant

PDF Portable Document Format

PHP PHP: Hypertext Preprocessor

PNG Portable Network Graphics

QoS Quality of Service

RDF Resource Description Framework

RFC Request For Comments

RMI Java Remote Method Invocation API

SOAP Simple Object Access Protocol

SQL Structured Query Language

SSH Secure Shell

TIP Tourist Information Provider

TCP Transmission Control Protocol – IETF RFC 793

UDDI Universal Description, Discovery, and Integration

UDP User Datagram Protocol – IETF RFC 768

UML Unified Modeling Language

UMTS Universal Mobile Telecommunications System

URI Uniform Resource Identifier

URL Uniform Resource Locator

UTF Universal Transformation Format – Character Encoding

W3C World Wide Web Consortium

WAN Wide-Area Network

WAP Wireless Application Protocol

WAR Web Application Archive

WLAN Wireless LAN

WML Wireless Markup Language

WSDL Web Services Description Language

XFS High-performance Journaling File System

XHTML Extensible Hypertext Markup Language

XML Extensible Markup Language

XSL XML Style Language

XSLT XSL Transformations

YAST Yet Another Setup Tool

Abbildungsverzeichnis

1.1 TIP-Anwendungsszenario 2

3.1 Serverseitige Struktur verteilter Informationssysteme 16
3.2 Klassen möglicher mobiler Geräte 17

4.1 Singleton-Pattern als UML-Diagramm 37
4.2 Factory-Pattern in der UML-Darstellung 38
4.3 Komponenten einer MVC-Architektur in Java 39
4.4 Zusammenspiel von Iterator und Collection zum Zugriff auf Mengen von Objekten 41
4.5 Use Case-Diagramm: TIP-Anmeldung 42
4.6 Use Case-Diagramm: TIP-Profilbearbeitung 43
4.7 Logische Struktur des TIP-Systems 43
4.8 Schematischer Aufbau des TIP-Systems 45

5.1 TIP-Mehr-Schichten-Architektur 48
5.2 Benutzung von Skriptsprachen 58
5.3 Anwendung von JSTL/Tag-Bibliotheken 58
5.4 Beispiel für Internationalisierung 60
5.5 Negativbeispiel zur Benutzung von URL-Parametern 61
5.6 Sitzungsverwaltung mit Struts 61
5.7 Konfiguration eines Datenbankpools 63
5.8 MVC-Komponenten von Struts 73
5.9 TIP-Datenhaltungs- und Datenhaltungszugriffsschicht 74
5.10 Ausschnitt einer View-Komponente 77
5.11 Konfiguration einer Struts-Aktion 78
5.12 Ablaufschema der Kommunikation mit dem TIP-System . . . 85

1 Standardkonforme Kommentare (javadoc) 124

2 Visualisierung des Quellcodes durch Java2HTML 125
3 Deutscher Inhalt des TIP-Startbildschirms 126
4 Englischer Inhalt des TIP-Startbildschirms 127
5 TIP-Anmeldung: Fehler in Ortsangabe 128
6 TIP-Benutzermenü 129
7 TIP-Profilbearbeitung Sehenswürdigkeitsgruppen 130
8 TIP-Profilanzeige Themenbereiche 131
9 TIP-Umgebungsanzeige laut Profil 132
10 TIP-Informationsanzeige laut Profil 133
11 TIP-Debugging-Untermenü 134
12 TIP-Administration-Untermenü 135
13 TIP: Anlegen eines Benutzers 136
14 Übersicht über die Anzahl der Quellcodezeilen des TIP-Projektes 137
15 Gesamtinhalt des CVS-Repositoriums des TIP-Projektes . . . 138
16 Aktivität als Anzahl der COMMIT-Eingaben des TIP-Projektes 139

Die Abbildungen zu den Entwurfsmustern des Abschnitts 4.4 sind der Webseite `http://www.bit.umkc.edu/burris/pl/design-patterns/` [59] entnommen.

Tabellenverzeichnis

4.1 Eigenschaften mobiler Geräte im WTK-Simulator 30

5.1 Übersicht der Java-Paketstruktur 80

1 CD-Inhaltsverzeichnis 121

Literaturverzeichnis

[1] A. Hinze & A. Voisard. Location- and Time-Based Information Delivery in Tourism. In *Proceedings of the 8th Symposium on spatio-temporal databases (SSTD'2003) 2003*. Springer Verlag, Berlin, Heidelberg, New York, 2003. LNCS.

[2] A. Pashtan & R. Blatter & A. Heusser & P. Scheuermann. CATIS: A context-Aware Tourist Information System. In *Proceedings of the 4th International Workshop of Mobile Computing*, 2003.

[3] Apache Logging. *Webseiten des Apache Logging-Projektes (log4j)*, 2004. `http://logging.apache.org/log4j/`.

[4] Apple Coorperation. *Verwendung von Rendezvous in Mac OS X*, 2004-05. `http://www.apple.com/macosx/features/rendezvous/`.

[5] B. Rao & L. Minakakis. Evolution of Mobile Location-based Services. *Communications of the ACM, S. 61-65*, 12-2003.

[6] CVS-Versionsverwaltung. *Projekt-Webseite*, 2004-05-21. `http://www.cvshome.org/`.

[7] D. Gourley & B. Totty. *HTTP - The Definitive Guide*. O'Reilly & Associates, Inc., 09-2002.

[8] D. Lüders. Verführung mit System - Welches PDA-Betriebssystem für wen ? *ct-special - Mobile PCs, S.78-80*, 02-2004.

[9] D. Ray & E. Ray. *HTML 4 For Dummies Quick Reference*. IDG Books Worldwide, 1998.

[10] Die deutsche PostgreSQL-Seite. *PostgreSQL - das objektrelationale Open Source Datenbanksystem*, 2004-05-16. `http://www.postgres.de/`.

[11] E. Gamma & R. Helm & R. Johnson & J. Vlissides. *Design Patterns: Elements of Reusable Object-Oriented Software*. China Machine Press & Pearson Education, 2002.

[12] eclipse.org - offene erweiterbare integrierte Entwicklungsumgebung (IDE). *Homepage*, 2004. `http://www.eclipse.org/`.

[13] G. N. Pudy. *CVS Pocket Reference*. O'Reilly & Associates, Inc., 2000.

[14] H. Balzert. *Lehrbuch der Objektmodellierung - Analyse und Entwurf*. Spektrum Akademischer Verlag GmbH Heidelberg - Berlin, 1999.

[15] H. Balzert. *UML kompakt mit Checklisten*. Spektrum Akademischer Verlag, 2001.

[16] heise online. *Symbian dominiert bei Smartphones, 2004-04-27*, April 2004. `http://www.heise.de/newsticker/meldung/46901`.

[17] heise online. *Lycoris arbeitet an Linux-Distribution für PDAs*, 2004-02-03. `http://www.heise.de/newsticker/meldung/44303`.

[18] J. Christe & P. Ziegler. Leitfähig - Satellitennavigation mit Palm und Pocket PC. *ct-special - Mobile PCs, S.78-80*, 02-2004.

[19] J. Homes. *Struts Console – Struts-Konfigurationseditor*, 2004-05-21. `http://jamesholmes.com/struts/console/`.

[20] J. O'Connor. *Java Internationalization: Localization with ResourceBundles*, Oktober 1998. `http://java.sun.com/developer/technicalArticles/Intl/ResourceBundles/`.

[21] J. Roth. *Mobile Computing – Grundlagen, Technik, Konzepte*. dpunkt Verlag, 2002.

[22] J. Schiller. *Mobilkommunikation - Techniken für das allgegenwärtige Internet*. Addison-Wesley, 2000.

[23] J. Schiller. *Telematik - Vorlesungsskript*. Freie Universität Berlin, Institut für Informatik, WS 02/03 edition, 2002.

[24] J. Schiller & A. Voisard, editor. *Location-Based Services*. Morgan Kaufman, San Francisco & Elsevier, Oxford, 2004.

[25] J. Worsley & J. Drake. *Practical PostgreSQL*. O'Reilly & Associates, Inc., 2002.

[26] Jakarta HTTP Server. *Webseiten des Apache-HTTP-Projektes*, 2004. `http://httpd.apache.org/`.

[27] Jakarta Struts. *Webseiten des Struts-Projektes*, 2004. `http://jakarta.apache.org/struts`.

[28] Jakarta Tapestry. *Webseiten des Tapestry-Projektes*, 2004. `http://jakarta.apache.org/tapestry/`.

[29] Jakarta Tomcat Application Server. *Webseiten des Tomcat-Projektes*, 2004. `http://jakarta.apache.org/tomcat/`.

[30] Jakarta Velocity. *Webseiten des Velocity-Projektes*, 2004. `http://jakarta.apache.org/velocity`.

[31] K. Cheverst & N. Davies & K. Mitchell. The role of adaptive hypermedia in a context-aware tourist guide. In *Communications of the ACM*, volume 45, No. 5, pages 47–51, Mai 2002.

[32] K. Cheverst & N. Davies & K. Mitchell & A. Friday. Experiences of Developing and Deploying a Context-Aware Tourist Guide: The GUIDE Project. 2000.

[33] K. Löffler. User-Adapted Information Delivery in Context-Aware Systems. Master's thesis, Freie Universität Berlin, Germany, Jan 2004.

[34] K. Pauls & R. Hall. Eureka - A Resource Discovery Service for Component Deployment. In *Proceedings of the 2nd International Working Conference on Component Deployment (CD 2004)*, May 2004 (to appear).

[35] L. Prechelt. *Webseiten der Veranstaltung: Bau betrieblicher Informationssysteme mit Java2 Enterprise Edition.* Freie Universität Berlin, Institut für Informatik, AG Software Engineering, WS 2003/2004. `http://www.inf.fu-berlin.de/inst/ag-se/teaching/`.

[36] M. Gebhard. *Java2Html - Open source Java (and others) to (X)HTML (and TeX and RTF) converter*, 2004-05-16. `http://www.java2html.de`.

[37] M. Kroll & S. Haustein. *J2ME Developer's Guide - Java-Anwendungen für mobile Geräte*. Markt+Technik Verlag, 2003.

[38] M. Tian & T. Voigt & T. Naumowicz & H. Ritter & J. Schiller. Performance considerations for mobile web services. *Elsevier Computer Communications*, (27), 2004. 1097-1105.

[39] MySQL GmbH Deutschland. *MySQL - Die populärste Open-Source-Datenbank der Welt*, 2004. `http://www.mysql.de/`.

[40] Oracle. *Oracle 10g: Infrastructure for grid Computing - An Oracle White Paper*, September 2003.

[41] P. Holmes & S. Kristoffersen & T. Lunde & A. Larsen & T. Kristoffersen. Experiences with Mobile Application Development within MOPAS. Technical report, Norsk Regnesentral, Oslo, Norwegen, Oktober 1999.

[42] P. Ziegler. Grosse Freiheit - Satellitennavigation mit dem Handy. *ct-special - Mobile PCs, S.98-103*, 02-2004.

[43] PHP: Hypertext Preprocessor. *Deutsche Webseite des PHP-Projektes*, 2004. `http://www.php3.de/`.

[44] POSTGIS - geografische Erweiterung der PostgreSQL-Datenbank. *Geographic Objects for PostgreSQL*, 2004. `http://www.postgis.org/`.

[45] S. Abeck & P. Lockemann & J. Seitz & J. Schiller. *Verteilte Informationssysteme*. dpunkt Verlag, 2003.

[46] S. Haiges. *Struts - Java Framework für Webanwendungen*. Software & Support Verlag GmbH, Frankfurt, 2003.

[47] S. Haiges. Location API - Mobil geortet. *Javamagazin - Internet & Enterprise Technology*, Oktober 2003.

[48] S. Radtke & P. Pisani & W. Wolters. *Handbuch Visuelle Mediengestaltung*. Cornelsen Verlag, Berlin, 2001.

[49] StatCvs - stat your repository. *CVS Repository statistic analysis tool, written in Java*, 2004. `http://statcvs.sourceforge.net/`.

[50] StrutsCX Team. *StrutsCX - Struts with XSLT*, 2004-01. `http://it.cappuccinonet.com/strutscx/`.

[51] Subversion Versionierungssystem. *Projekt-Webseite*, 2004-05-21. `http://subversion.tigris.org/`.

[52] Sun Microsystems, Inc. *J2ME Wireless Toolkit 2.1 User's Guide*, 2004. `http://java.sun.com/products/j2mewtoolkit/download-2_1.html`.

[53] Sun Microsystems, Inc. *Java 2 Platform, Micro Edition (J2ME)*, 2004. `http://java.sun.com/j2me/`.

[54] Sun Microsystems, Inc. *Code Conventions for the Java Programming Language*, Revised April 20, 1999. `http://java.sun.com/docs/codeconv`.

[55] Symbian. *Symbian OS - the mobile operating system*, Mai 2004. `http://www.symbian.com/`.

[56] T. Narten (IBM) & E. Nordmark (Sun Microsystems) & W. Simpson (Daydreamer). *IPv6 RFC - The Internet Engineering Task Force (IETF)*, Dezember 1998. `http://www.ietf.org/rfc/rfc2461.txt`.

[57] The Apache Ant project. *Homepage*, 2004. `http://ant.apache.org/`.

[58] The Apache Software Foundation (ASF). *Webseiten des ASF-Projektes*, 2004-05-18. `http://apache.org/`.

[59] Virtual University - School of Interdisciplinary Computing and Engineering, University of Missouri - Kansas City. *CS451 - Software Engineering - Lesson 12 - Design Patterns*, Stand: 2004-05-11. `http://www.bit.umkc.edu/burris/pl/design-patterns/`.

[60] World Wide Web Consortium (W3C). *HyperText Markup Language (HTML) Home Page*, 2004. `http://www.w3.org/MarkUp/`.

[61] Y. Lee & I. Benbasat. Interface Design for Mobile Commerce. *Communications of the ACM, S.49-52*, 12-2003.

Linkverzeichnis

Zum vereinfachten Zugriff hier eine Liste der für die Diplomarbeitsimplementierung relevanten Internetseiten – Stand: 2004-06-14.

- `http://ant.apache.org/` – Apache Ant
- `http://apache.org/` – Apache Software Foundation
- `http://httpd.apache.org/` – Jakarta HTTP
- `http://it.cappuccinonet.com/strutscx/` – StrutsCX
- `http://jakarta.apache.org/struts/` – Jakarta Struts
- `http://jakarta.apache.org/tapestry/` – Jakarta Tapestry
- `http://jakarta.apache.org/tomcat/` – Jakarta Tomcat
- `http://jakarta.apache.org/velocity/` – Jakarta Velocity
- `http://jamesholmes.com/struts/console/` – Struts Konsole
- `http://java.sun.com/j2me/` – J2ME
- `http://java.sun.com/products/j2mewtoolkit/` – WTK 2.1
- `http://logging.apache.org/log4j/` – Apache Logging
- `http://statcvs.sourceforge.net/` – StatCvs
- `http://subversion.tigris.org/` – Subversion
- `http://www.cvshome.org/` – CVS
- `http://www.eclipse.org/` – Eclipse IDE

- `http://www.java2html.de/` – Java2html
- `http://www.postgis.org/` – Postgis
- `http://www.postgres.de/` – PostgreSQL
- `http://www.w3.org/MarkUp/` – W3C Markierungssprachen

Anhang

.1 CD-Inhalt

Der Diplomarbeit ist eine CD beigelegt, die folgenden Inhalt hat:

Verzeichnis / Datei	**Kurzbeschreibung**
diplomarbeit.war	enthält die komplett konfigurierte gesamte Anwendung zur Installation auf einem Tomcat-Anwendungsserver
/api	Javadoc-API des Projektes
build.xml	Apache Ant-Konfigurationsdatei aus dem Verzeichnis `/WEB-INF/`
datenbank.sql.gz	gepackter Inhalt der verwendeten Datenbank
/diplomarbeitstruts	kompletter Eclipse-Verzeichnisbaum des Gesamtprojekts
/java2html	Quellcode-Darstellung mittels Java2Html
/statcvs	komplette Visualisierung des Projektes mittels StatCvs
struts-config.xml	zentrale Struts-Konfigurationsdatei aus dem Verzeichnis `/WEB-INF/`
web.xml	Konfigurationsdatei des TIP-Projekts für den Anwendungsserver aus `/WEB-INF/`

Tabelle 1: Inhaltliche Gliederung der beiliegenden CD

.2 Abbildungen

Der Anhang enthält Bildschirmfotos zur Visualisierung des touristischen Informationssystems. Die Bilder zeigen das entwickelte Informationssystem im Webbrowser Mozilla Firefox 0.8 unter Linux.

- S. 126 – Start des Informationssystems in deutscher Sprache
- S. 127 – Start des Informationssystems in englischer Sprache
- S. 128 – Eingabe eines mehrdeutigen Ortsnamens beim Anmeldevorgang
- S. 129 – Benutzermenü nach erfolgreicher Anmeldung
- S. 130 – Bearbeitung des Sehenswürdigkeitsgruppenprofils
- S. 131 – Anzeige des Themenprofils
- S. 132 – Anzeige der Umgebungsinformationen zu einem Standort
- S. 133 – Informationen zu einem Standort anzeigen
- S. 134 – Inhalt des DEBUG-Untermenüs
- S. 135 – Inhalt des ADMINISTRATION-Untermenüs
- S. 136 – Anlegen eines neuen TIP-Benutzers

Die strikte Verwendung von Javadoc zum Kommentieren des Quellcodes in Verbindung mit der HTML-Darstellung (java2html) des Quellcodes macht es möglich, sich schnell in das Projekt einzuarbeiten. Die folgenden Abbildungen zeigen verwendete Softwarewerkzeuge und visualisieren das CVS-Projektrepositorium.

- S. 124 – Verwendung von Javadoc zur Dokumentation
- S. 125 – Visualisierung des Quellcodes mit Hilfe von Java2HTML [36]
- S. 137 – Visualisierung des CVS-Repositoriums mittels StatCVS [49]
  - S. 137 – Quellcode-Anzahl-Metrik des TIP-Systems
  - S. 138 – Projektstruktur als Baum – Gesamtansicht
  - S. 139 – Übersicht, wann am Projekt gearbeitet wurde

Abbildung 1: Standardkonforme Kommentare (javadoc)

Abbildung 2: Visualisierung des Quellcodes durch Java2HTML

Abbildung 3: Deutscher Inhalt des TIP-Startbildschirms

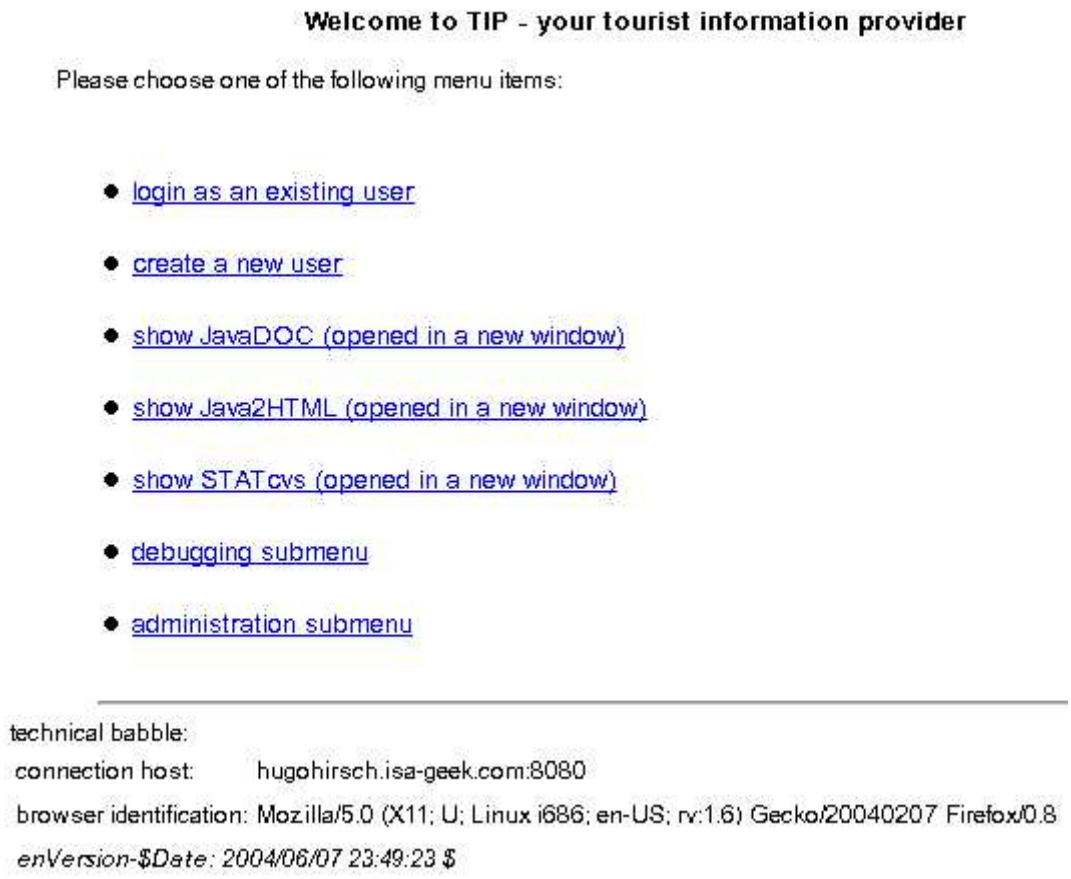

Abbildung 4: Englischer Inhalt des TIP-Startbildschirms

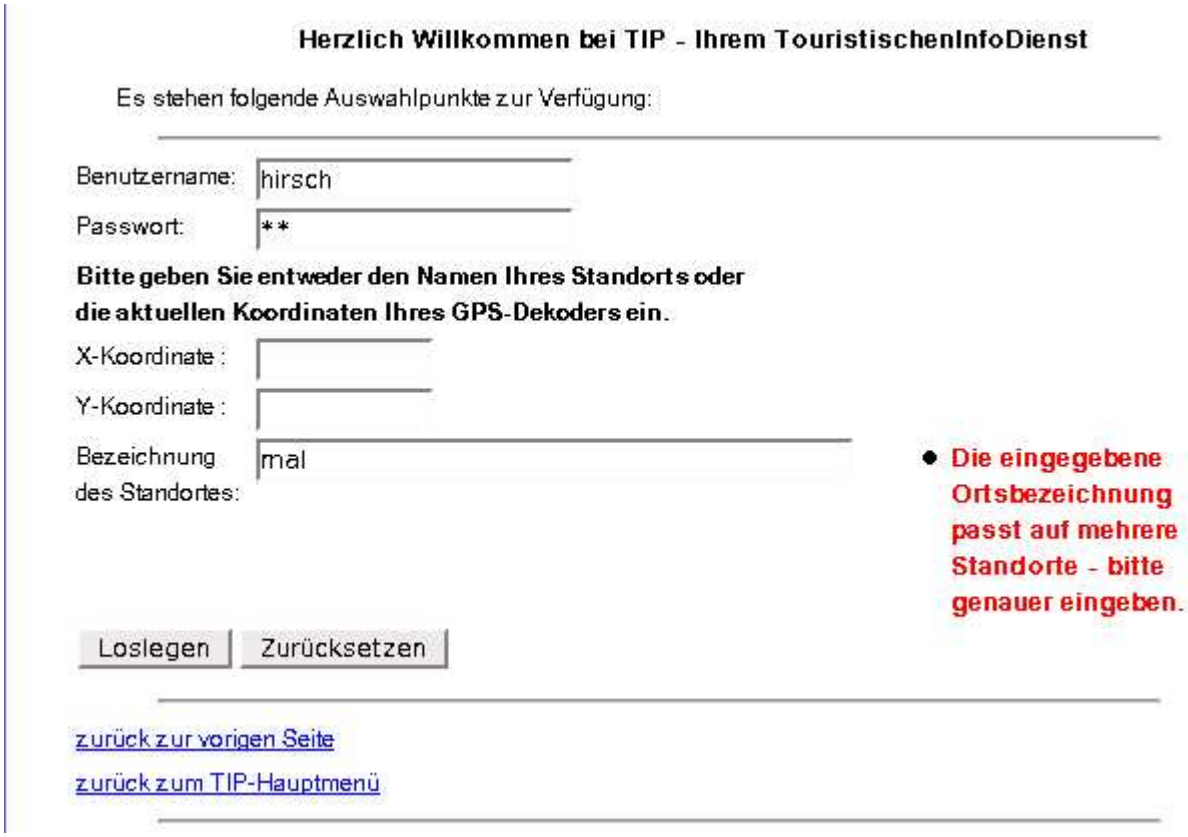

Abbildung 5: Eine mehrdeutige Ortsbezeichnung wird vom TIP-System erkannt.

Hauptmenü des touristischen Infodienstes für angemeldete Benutzer

Inhalte ausgeben:

hirsch @ (13.39523, 52.51909)

Sie befinden sich am Standort: Mommsendenkmal

Willkommensnachrichten testen:

Hugo Hirsch - Herzlich Willkommen am TIP - System

Ihr gemeldeter Standort ist (13.39523 / 52.51909).
Es stehen folgende Punkte zur Auswahl:

Informationen zum aktuellen Standort anzeigen

Anzeigen Ihrer Ereignishistorie

Anzeigen und Bearbeiten Ihres Profils

zurück zur vorigen Seite

zurück zum Benutzermenü

Abmelden

zurück zum TIP-Hauptmenü

Abbildung 6: TIP-Benutzermenü nach erfolgreicher Anmeldung

Abbildung 7: Bearbeitung des Sehenswürdigkeitsgruppenprofils

Ihr Abonnementprofil für Themen gliedert sich wie folgt:

Folgende Themengruppen sind in Ihrem Profil aktiviert:

Themenbeschreibung	Inhalt, der von diesem Thema abgedeckt wird
architecture (2)	Information concerning the architecture of a building
art (5)	Information concerning the art of a monument
general (1)	General Information
persons (3)	Information about the person(s) which has/have created a sight

---

Folgende Themengruppen sind in Ihrem Profil deaktiviert:

Themenbeschreibung	Inhalt, der von diesem Thema abgedeckt wird
history (4)	What has happend over the years to that sight?

---

zurück zur vorigen Seite

zurück zum Benutzermenü

Abmelden

zurück zum TIP-Hauptmenü

Abbildung 8: Anzeige des aktuellen Themenprofils

Umgebungsinfo

In Ihrer Umgebung befinden sich folgende Sehenswürdigkeitsgruppen:

- Squares
- Sculptures in general
- Sculptures in general

In Ihrer Umgebung befinden sich folgende Sehenswürdigkeiten:

Name	Beschreibung	
Helmholtzdenkmal		>> weitere Details zu Helmholtzdenkmal
Lustgarten		>> weitere Details zu Lustgarten
Mommsendenkmal		>> weitere Details zu Mommsendenkmal

zurück zur vorigen Seite

zurück zum Benutzermenü

Abmelden

zurück zum TIP-Hauptmenü

Abbildung 9: Beispielanzeige der aktuellen Umgebung
in Abhängigkeit vom eingestellten Profil

Abbildung 10: Anzeige aller Informationenzur Sehenswürdigkeit Lustgarten

Abbildung 11: Inhalt des DEBUGGING-Untermenüs

Abbildung 12: Inhalt des ADMINISTRATION-Untermenüs

Abbildung 13: Einmaliges Anlegen eines neuen Benutzers

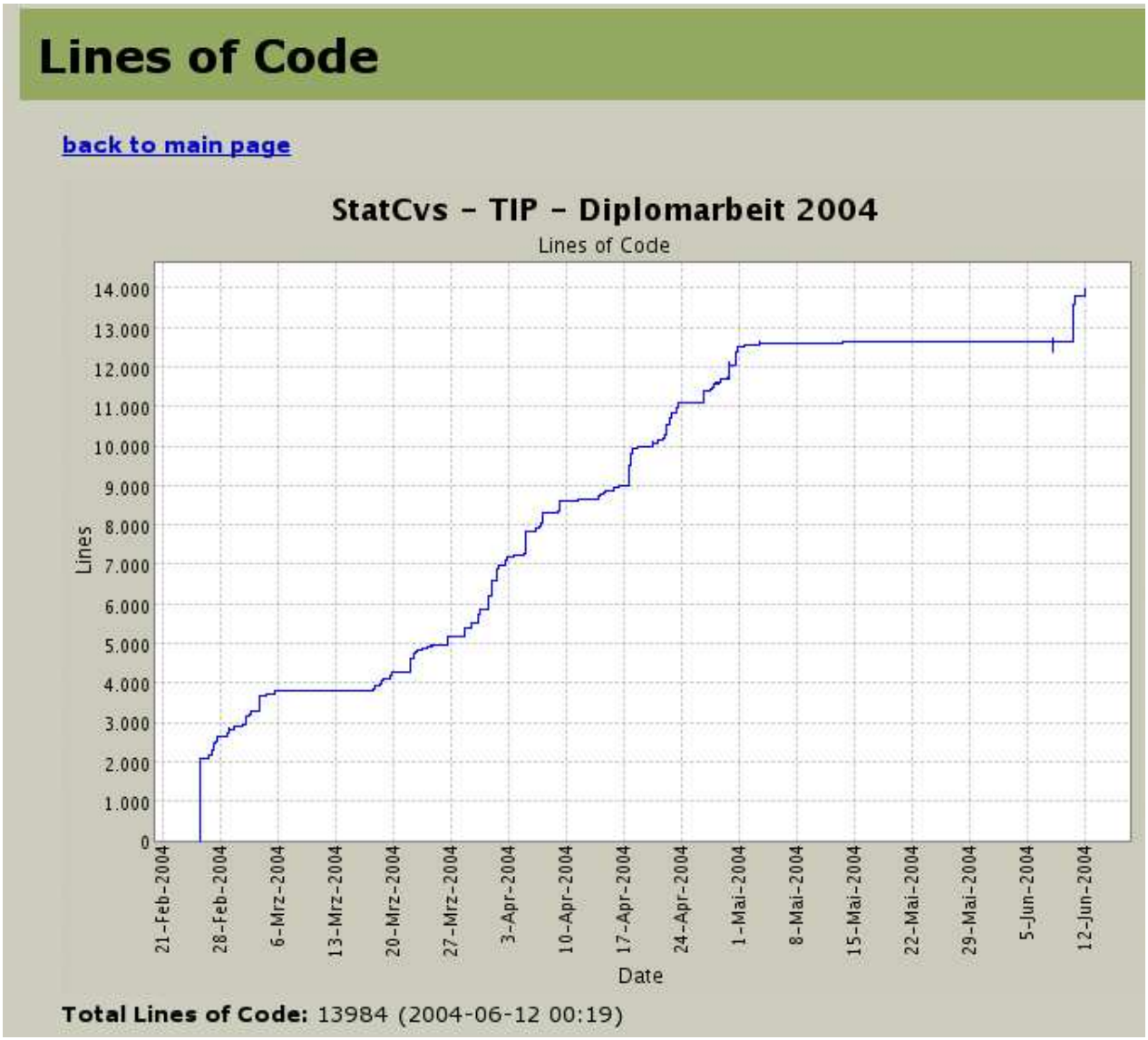

Abbildung 14: Übersicht über die Anzahl der Quellcodezeilen des TIP-Projektes

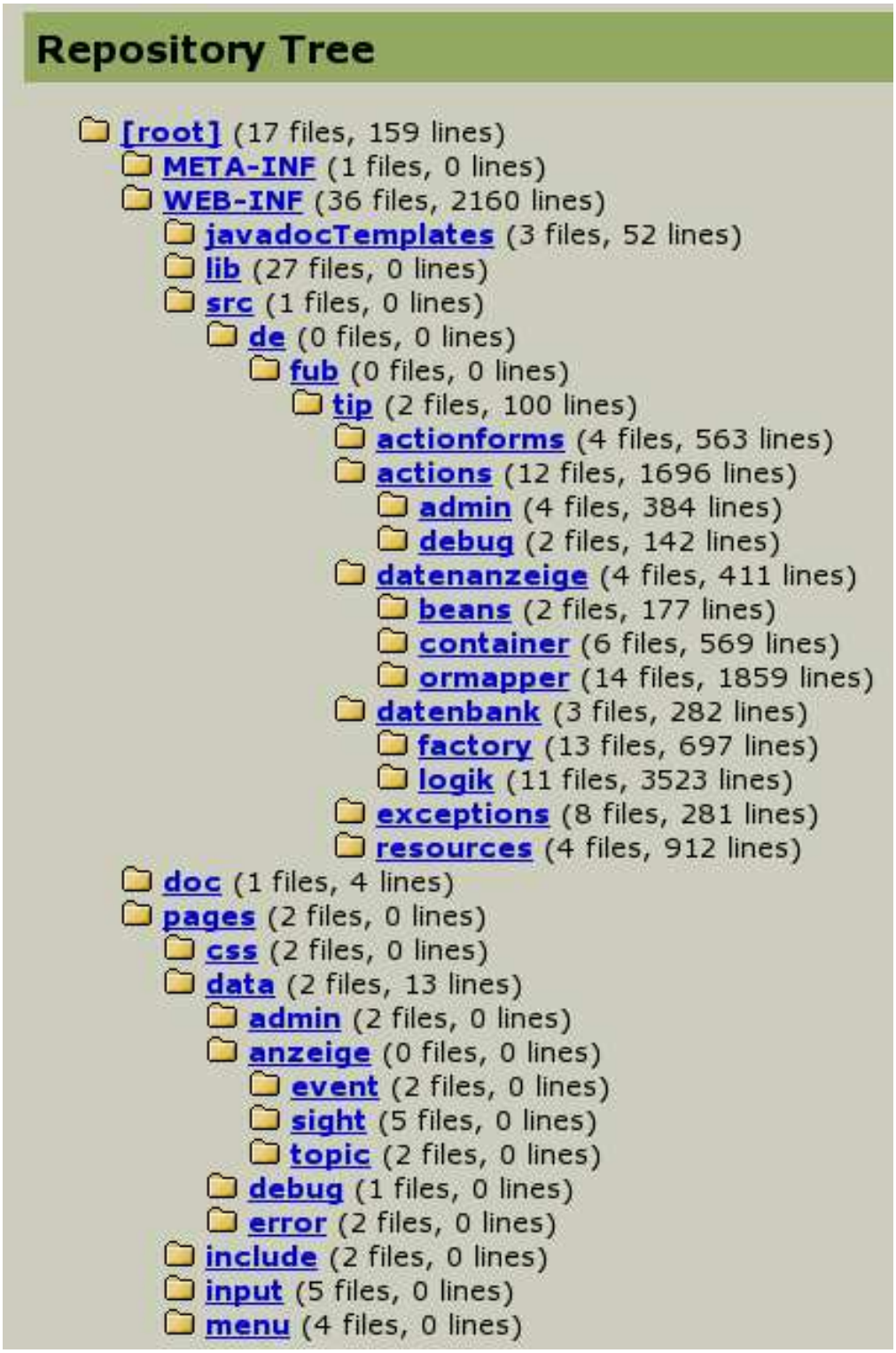

Abbildung 15: Gesamtinhalt des CVS-Repositoriums des TIP-Projektes

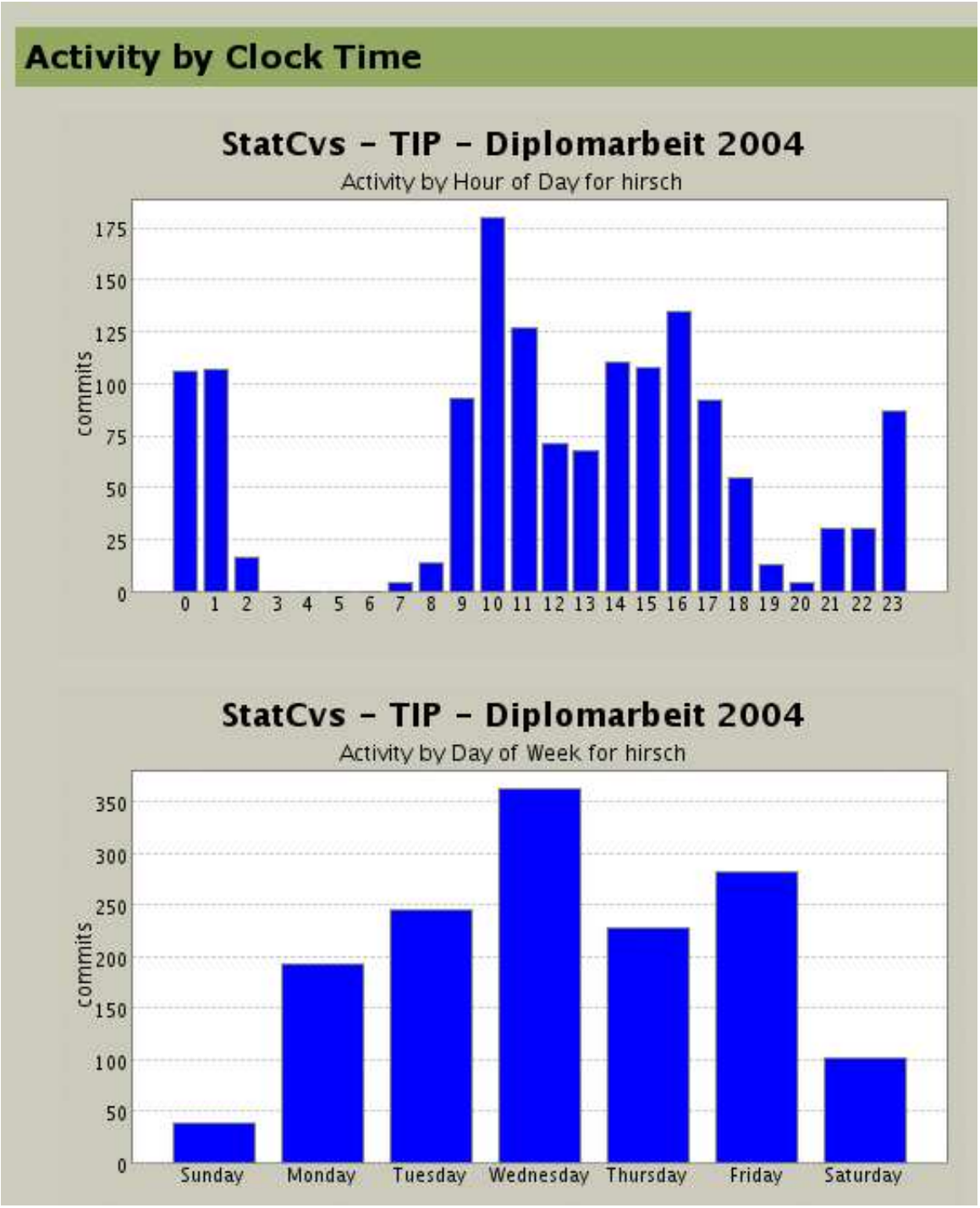

Abbildung 16: Aktivität als Anzahl der COMMIT-Eingaben des TIP-Projektes