Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Bachelorarbeit

Dirk Parlowski

Entwicklung eines mobilen Navigationssystems für den öffentlichen Nahverkehr

Fakultät Technik und Informatik
Studiendepartment Informatik

Faculty of Engineering and Computer Science
Department of Computer Science

Dirk Parlowski

Entwicklung eines mobilen Navigationssystems für den öffentlichen Nahverkehr

Bachelorarbeit eingereicht im Rahmen der Bachelorprüfung

im Studiengang Angewandte Informatik
am Studiendepartment Informatik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer: Prof. Dr. Birgit Wendholt
Zweitgutachter: Prof. Dr. Kai von Luck

Abgegeben am 11.06.2009

Dirk Parlowski

Thema der Bachelorarbeit
Entwicklung eines mobilen Navigationssystems für den öffentlichen Nahverkehr
Stichworte
Location-Based Services, öffentlicher Nahverkehr, mobile Navigation, J2ME, GPS, Webservices
Kurzzusammenfassung
Die Nutzung von öffentlichen Verkehrsmitteln für ortsansässige und ortsunkundige Personen wird durch komplexe Streckennetze und kombinierte Verkehrsmittel immer schwieriger. In dieser Arbeit wird ein mobiles Navigationssystem NavMobile für den öffentlichen Nahverkehr entwickelt, welches Benutzern von mobilen Endgeräten vor und während einer Fahrt mit notwendigen Fahrplan- und Navigationsinformationen versorgt, um das Erreichen eines bestimmten Zieles mit öffentlichen Verkehrsmitteln von beliebigen Standorten in einer unbekannten Stadt einfacher zu gestalten. Um eine möglichst optimale Orientierung zu gewährleisten, wird die Umgebung in realen Karten abgebildet. Die in der Arbeit verwendeten Technologien zur Entwicklung von NavMobile umfassen GPS, J2ME und Webservices.

Dirk Parlowski

Title of the paper
Development of a mobile navigation system for public transport
Keywords
location-based services, public transport, mobile navigation, J2ME, GPS, webservices
Abstract
Because of complex route systems the use of public transportation for residents and non-residents is getting more and more challenging. This paper presents the mobile navigation system NavMobile which provides a user of a portable device with schedule and navigation information before and during a journey. NavMobile improves the finding of the final destination combining public transport and pedestrian route guidance starting from any location even in unknown cities. To provide the user with as much orientation as possible the surrounding area is show in real maps. Among technologies used to develop NavMobile are GPS, J2ME and webservices.

Inhaltsverzeichnis

Abbildungsverzeichnis

Kapitel 1

Einleitung

1.1 Motivation

Im 21. Jahrhundert ist die Mobilität des Menschen so hoch wie nie und damit auch sein Bedarf nach mobilen Kommunikations- und Informationssystemen. Obwohl das mobile Endgerät seinen weltweiten Siegeszug schon vor Jahren begonnen hat, ist ein Ende noch nicht abzusehen und die Verbreitung steigt stetig weiter. Heutzutage besitzen und nutzen über 80 Prozent der Deutschen ein mobiles Endgerät und jedes Jahr werden über eine Milliarde Geräte weltweit verkauft. Durch diese hohe Verbreitung und Akzeptanz ist das mobile Endgerät zu einem ständigen Begleiter geworden. Technologische Weiterentwicklungen führten dazu, dass die Funktionen der mobilen Endgeräte ständig erweitert und die Leistungsfähigkeit erhöht werden konnte.

Spätestens seit die Bundesrepublik Deuschland ca. 100 Milliarden DM (ca. 51 Mill Euro) für den Verkauf der UMTS (Universal Mobile Telecommunications System) -Lizenzen an verschiedene Mobilfunkbetreiber eingenommen hat, steht fest, dass in Zukunft mobile Endgeräte mehr sein werden als nur ein einfaches Mittel zur Telekommunikation. Durch den Einsatz der UMTS-Technologie sind schon heute Datenübertragungsraten von bis zu 7,2 Mbit/s (download) möglich. Für die immer leistungsstärker werdenden mobilen Endgeräte öffnen sich somit neue Bereiche der mobilen Kommunikation sowie der Integration von multifunktionalen, multimedialen und internetbasierten Anwendungen.

Die zunehmende Wettbewerbsintensität auf dem umsatzstarken Mobilfunkmarkt veranlassen die Mobilfunknetzbetreiber dazu, die Preise für Datenübertragungen kontinuierlich zu

senken. Zudem werden relativ günstige Flatrate-Angebote offeriert, die eine ständige Verbindung zum Internet (always connected), ohne die Berücksichtigung des übertragenen Datenvolumens und der Verbindungszeit, erlauben. Diese speziellen Angebote bewirken ein zunehmendes Wachstum der Zielgruppen sowie des Marktes für internetbasierte Anwendungen.

Darüber hinaus sorgen die technischen Möglichkeiten mobiler Endgeräte dafür, dass der Bedarf an alltäglichen Anwendungen enorm wächst. Dafür können verschiedene Faktoren und Informationen miteinbezogen werden, um ein möglichst breites Spektrum an Funktionen zur Verfügung zu stellen. Diese sollen den Benutzer bei alltäglichen Situationen transparent unterstützen. In diesem Kontext lassen sich die Begriffe *Ubiquitous Computing* [Weiser (1999)] und *Ambient Intelligence* [de Ruyter und Aarts (2004)] einordnen. Diese besagen, dass Alltagsgegenstände, wie z.B. ein mobiles Endgerät, um Kommunikationsfähigkeit und eine gewisse Intelligenz erweitert werden, um den Menschen bei alltäglichen Situationen zu unterstützen. Dabei sollen diese Alltagsgegenstände ihre speziellen computergestützten Funktionen vor dem Anwender verbergen und eine Nutzung so einfach wie möglich gestalten.

Ein spezielles Anwendungsgebiet von *Ambient Intelligence* ist die effizientere Nutzung der Verkehrsinfrastruktur. In diesem Zusammenhang werden unter anderem Location-Based Services entwickelt, die standortbezogene Informationen und die aktuelle Position des Benutzers verwenden, um personalisierte Anwendungen auf dem mobilen Endgerät zur Verfügung zu stellen. Solche Anwendungen sind zum Beispiel *deep map* [Zipf und Malaka (1999)]und *We-Travel* [weTravel (2008)], welche unter anderem als mobiles Navigationssystem sowie als Touristeninformation für Fussgänger entwickelt wurden. Diese können einem Benutzer ortsbezogene Informationen zur Verfügung stellen und über eine dynamisch generierte Kartendarstellung zu einem definierten Ziel navigieren.

Allerdings sind insbesondere Touristen und Fussgänger meistens in ihrer Mobilität eingeschränkt und somit auf andere Verkehrsmittel, wie die des öffentlichen Nahverkehrs, angewiesen. Das kann ganz besonders in Großstädten häufig zu Problemen führen, weil komplexe Nahverkehrsnetze, kombinierte Verkehrsmittel, Beförderungs- und Tarifbestimmungen die Übersichtlichkeit und Nutzung erheblich erschweren. Die Anforderungen an öffentliche Verkehrsmittel können sich allerdings unterscheiden, wodurch eine Aufteilung in zwei verschiedene Personengruppen notwendig ist.

Zu der ersten Gruppe gehören die ortsunkundigen Personen, wie Touristen, die sich in einer unbekannten Stadt bewegen und über keinerlei Informationen zu Haltestellenpositionen, Verkehrslinien, Tarifbestimmungen, Abfahrtszeiten, etc. verfügen. Darüber hinaus sind ebenfalls keine Ortskenntnisse vorhanden, die es ihnen ermöglichen würde, eine bestimmte Haltestelle, Strasse oder Sehenswürdigkeit zu Fuss zu erreichen. Für diese Personen ergeben sich

somit die Anforderungen an umfangreiche standortbezogene Fahrplan- sowie Navigationsinformationen, um zu einem bestimmten Ziel mit öffentlichen Verkehrsmitteln zu gelangen.

Die zweite Gruppe sind ortsansässige Personen, die mit der Umgebung vertraut sind und regelmäßig öffentliche Verkehrsmittel benutzen. Diese Personen kennen ihre nähere Umgebung und sind in der Lage, bestimmte Ziele, wie den Arbeitsplatz, die Wohnung, Freunde, Bekannte, Geschäfte, etc., mit öffentlichen Verkehrsmitteln zu erreichen. Dazu kennen sie ebenfalls die für sie notwendigen Fusswege sowie Haltestellenpositionen. Für diese Personengruppe sind hauptsächlich standortbezogene Fahrplaninformationen relevant. Darüber hinaus können auch Navigationsinformationen benötigt werden, wenn sich diese Personen außerhalb der ihnen bekannten Umgebung bewegen (z.B. spezielle Restaurants oder Geschäfte).

Für beide Personengruppen wird eine mobile Navigationsanwendung benötigt, die Fusswege und Fahrplandaten als intermodale Routenführung kombiniert und die Navigation mit Hilfe von dynamischen Kartendarstellungen unterstützt. Berücksichtigt man die Personen, die pro Jahr öffentliche Verkehrsmittel nutzen - allein beim Hamburger Verkehrsverbund wurden z.B. im Jahr 2007 618 Millionen Fahrgäste befördert [HVV]- wird die Relevanz einer solchen Anwendung deutlich.

1.2 Ziel dieser Arbeit

In dieser Arbeit soll eine mobile Anwendung entwickelt werden, die einen Benutzer möglichst einfach und intuitiv bei der Nutzung von öffentlichen Verkehrsmitteln unterstützt. Der Benutzer soll in der Lage sein, durch die Angabe eines Zieles eine automatische Fahrplanauskunft zu erhalten. In einer dynamischen und realitätsnahen Kartendarstellung sollen alle notwendigen Navigationsinformationen übersichtlich dargestellt werden, um den Benutzer von seiner aktuellen Position bis zum Ziel navigieren zu können. Dazu werden alle Haltestellen, die zum Erreichen des Zieles notwendig sind, auf der Karte angezeigt. Zudem werden die Fusswege von der aktuellen Position bis zur Starthaltestelle und von der Endhaltestelle bis zum Ziel ebenfalls angezeigt. Um dem Benutzer die Orientierung zu erleichtern, soll er jederzeit in der Lage sein, seine aktuelle Position auf der Karte zu verfolgen. Während der gesamten Reise sollen darüber hinaus Fahrplaninformationen zu Abfahrts- und Ankunftszeiten sowie zu benötigten Verkehrsmitteln bereit gestellt werden.

Um ein klar definiertes Konzept und Design der Anwendung zu erhalten, werden bei der Enwicklung Verfahren und Methoden der objektorientierten Softwareentwicklung angewendet. Außerdem soll die Anwendung auf einer Architektur aufbauen, die zukünftige Erweiterungen, z.B. zu anderen öffentlichen Nahverkehrsnetzen, ermöglicht.

1.3 Gliederung

In dem Kapitel 2 werden grundlegende Technologien vorgestellt, die für den weiteren Verlauf der Arbeit notwendig sind und die technischen Grundlagen für das Design und die Realisierung darstellen. Dazu werden Location-Based Service (LBS), Lokalisierungsdienste sowie Datenübertragungstechniken vorgestellt. Als verwendete Laufzeitumgebung wird dabei J2ME auf Grund der weiten Verbreitung und der Portabilität verwendet. Darüber hinaus wird die GPS-Technologie eingesetzt, weil es die notwendige Genauigkeit für ein Navigationssystem zur Verfügung stellt. Als Datenübertragungstechnik wird hauptsächlich UMTS eingesetzt, da es hohe Übertragungsraten bietet.

Das Kapitel 3 stellt vergleichbare Arbeiten und deren verwendete Ansätze vor. Dabei wird am Ende eine Bewertung der vorgestellten Arbeiten abgegeben. Für die Entwicklung der Anwendung werden dann die positiven Ansätze miteinbezogen und die negative Ansätze vermieden.

Kapitel 4 beschreibt die erste Phase der Softwareentwicklung und definiert die funktionalen und nichtfunktionalen Anforderungen. Dazu wird ein Use-Case Diagramm dargestellt, dass die Abhängigkeiten verdeutlicht. Die einzelnen Use-Cases werden in tabellarischer Form erläutert und beschrieben. Dadurch können widersprüchliche oder inkonsistente funktionale Anforderungen vermieden werden.

In dem Kapitel 5 wird aus den funktionalen und nichtfunktionalen Anforderungen ein Konzept entwickelt. Dieses wird bei der Realisierung der Client- und Server Komponente umgesetzt. Für den Client und Server werden jeweils Klassen- und Sequenzdiagramme modelliert, die spezielle Aufgaben, Zuständigkeiten und Abhängigkeiten aufzeigen.

Das Kapitel 6 gibt eine kurze Zusammenfassung der Arbeit sowie eine Ausblick darüber, wie die entstandende Anwendung verbessert oder erweitert werden könnte.

Kapitel 2

Grundlagen

In diesem Kapitel werden grundlegende Technologien vorgestellt, die als Basis für den weiteren Verlauf der Arbeit dienen. In dem Kapitel 2.1 wird diese Arbeit als Location-Based Service (LBS) eingeordnet und der Begriff kurz erläutert. Kapitel 2.2 stellt verschiedene Systeme und Dienste vor, die für eine Lokalisierung notwendig sind. Da es sich in der vorliegenden Arbeit um eine Anwendung auf einem mobilen Endgerät mit hoher Datenübertragungsrate handelt, werden im Kapitel 2.3 verschiedene Datenübertragungstechniken gegenübergestellt. In dem Kapitel 2.4 wird ein Einblick in die Grundlagen von J2ME gegeben, welche als Programmiersprache für die mobile Anwendung verwendet wird. Die für die Realisierung einer verteilten Anwendung notwendige Technologie wird im Kapitel 2.5 vorgestellt. Dabei wird erläutert, aus welchen Gründen Webservices verwendet werden.

2.1 Location-Based Services

Location-Based Services (LBS) stellen einem Benutzer ortsbezogene Informationen über ein mobiles Endgerät zur Verfügung. Dabei spielt jeweils unsere aktuelle geografische Position für die Nutzbarkeit der Informationen eine große Rolle (vgl. [Jochen Schiller (2004)]). Die Kombination von Information und aktueller geographischer Position ermöglicht es, mobile Dienste zu personalisieren. Um standortabhängige Informationen zu erhalten, werden meistens externe Internetdienstleister in Anspruch genommen.

In dieser Bachelorarbeit soll ein Location-Based Service entwickelt werden, der dem Benutzer standortabhängige Informationen zur Navigation bereitstellt. Dafür ist die Kommuni-

kation mit externen Dienstleistern wie z.B. einem Geoinformationsdienst[1] und einem Informationssystem, das ortsabhängige Daten anbietet, notwendig. Der Geoinformationsdienst liefert die geographischen Daten, die mit den ortsabhängigen Daten des Informationssystems verknüpft werden. Um den aktuellen Ortskontext einer Person zu ermitteln, muss ein Ortungsverfahren (Lokalisierung) zur Verfügung stehen, dass die geographische Position berechnet.

In den folgenden Kapiteln werden verschiedene Technologien zur Lokalisierung (2.2) und Datenübertragung (2.3) vorgestellt, welche für die Entwicklung eines mobilen Navigationssystems erforderlich sind.

2.2 Lokalisierung

Lokalisierung ist die Bestimmung einer Position in Bezug auf ein festgelegtes Koordinatensystem. Innerhalb eines gewählten Koordinatensystems wird die Position durch Angabe von Koordinaten eindeutig bestimmt. Kapitel 2.2.1 stellt zwei Koordinatensysteme mit unterschiedlichen Verfahren der Positionsangabe vor. Kapitel 2.2.2 erläutert verschiedene Ortungsverfahren von Lokalisierungsdiensten, die zur Positionsbestimmung verwendet werden.

2.2.1 Koordinatensysteme

Die bei weitem bekanntesten Weltkoordinatensysteme sind das geographische- und das Gauss-Krüger Koordinatensystem. Während ersteres im Global Positioning System (GPS) Verwendung findet, ist letzteres Grundlage für nahezu alle geographischen Informationssysteme.

2.2.1.1 Geographisches Koordinatensystem

Das geographische Koordinatensystem baut auf der dreidimensionalen kugelförmigen Darstellung der Erde auf. Dabei sind die Koordinaten als Winkel definiert und werden durch die

[1] Karten-Server (Map-Server), von denen Auszüge von Landkarten über das Internet abgerufen werden kann (z.B. Google Maps, MSN, OpenStreetMap)

geographische Länge (Longitude) und die geographische Breite (Latitude) angegeben. Diese geographischen Koordinaten beziehen sich auf den Nullmeridian[2] und die Äquatorebene. Als Nullmeridian wurde der Meridian, der durch die Sternwarte von Greenwich verläuft, international anerkannt. Die Längengrade werden von dem Nullmeridian nach Osten und Westen bis 180 Grad gezählt. Die Breitengrade werden vom Äquator aus bis 90 Grad Nord oder Süd gezählt (vgl. [Kahmen (2006)]).

Wie in der Abbildung 2.1 zu sehen ist, wird die geographische Länge eines Punktes P durch den Winkel λ zwischen der Ebene durch den Nullmeridian und der Meridianebene im Punkt P bestimmt. Die geographische Breite ist der Winkel ϕ im Erdmittelpunkt zwischen dem Äquator und dem gesuchten Punkt.

Da die Erde an den Polen etwas abgeplattet ist, stellt sie keine richtige Kugel dar. Deshalb kann die Positionsangabe aus Abbildung 2.1 nur verwendet werden, wenn die Genauigkeit nur geringen Ansprüchen genügen muss. Aus diesem Grund können Referenzsysteme eingesetzt werden, die ein geometrisches Modell der Erdoberflächenform mathematisch definieren und zuverlässigere Positionsangaben ermöglichen. Das weltweit am häufigsten verwendetete Referenzsystem ist das World Geodetic System 1984 (WGS 84), welches unter anderem die Grundlage für das Global Positioning System (siehe Kapitel 2.2.2.1) ist.

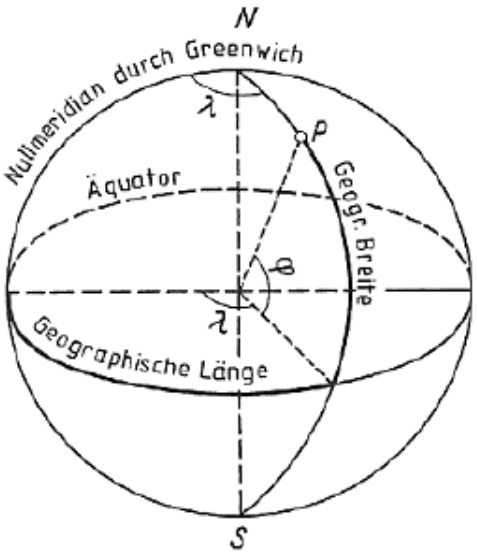


Abbildung 2.1: Geographische Koordinaten

[2]senkrecht zum Äquator stehender, von Nord- nach Südpol verlaufender Halbkreis

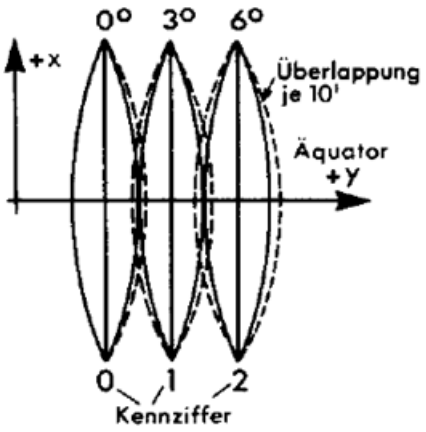
2.2.1.2 Gauss-Krüger Koordinatensystem

Das Gauss-Krüger Koordinatensystem wurde von Carl Friedrich Gauß und von Johann Heinrich Louis Krüger entwickelt, um die gekrümmte Erdoberfläche in eine zweidimensionale Ebene abzubilden. Dabei wird die Positionsangabe mit Hilfe der metrischen Koordinaten Rechtswert und Hochwert bestimmt. Das Gauss-Krüger Koordinatensystem wird hauptsächlich im deutschsprachigen Raum und von sehr vielen amtlichen topografischen Kartenwerken genutzt (vgl. [Kahmen (2006)]).

Die Notwendigkeit von zweidimensionalen Koordinatensystemen ist darin begründet, dass dreidimensionale Koordinatensysteme für praktische Anwendungen (z.B. Vermessungswesen, Nutzung von topographischen Landkarten) ungeeignet sind. Denn in zweidimensionalen Koordinatensystemen sind Berechnungsmethoden der ebenen Geometrie anwendbar und somit werden zum Beispiel einfache Distanzberechnungen ermöglicht.

Bei der Abbildung der gekrümmten Erdoberfläche in eine Ebene wird sich im Gauss-Krüger Koordinatensystem auf kleine Teile der Erdoberfläche beschränkt, damit die prinzipiell nicht vermeidbaren Abbildungsverzerrungen in Grenzen bleiben. Dabei bezieht sich das Gauss-Krüger Koordinatensystem auf das Bessel- (im Westen Deutschlands) oder das Krassowski-Ellipsoid (im Osten Deutschlands). Das sind Referenzsysteme, die eine Annäherung an die ideale Erdform vorgeben. Die gesamte Erdoberfläche wird bei der Abbildung mit Hilfe des geographischen Koordinatensystems in 3 Grad breite Meridianstreifen[3] aufgeteilt (siehe Abbildung 2.2).

[3] ein geographischer halber Längenkreis, der vom Nord- zum Südpol verläuft



Meridianstreifen	6°	9°	12°	15°
Kennziffer	2	3	4	5

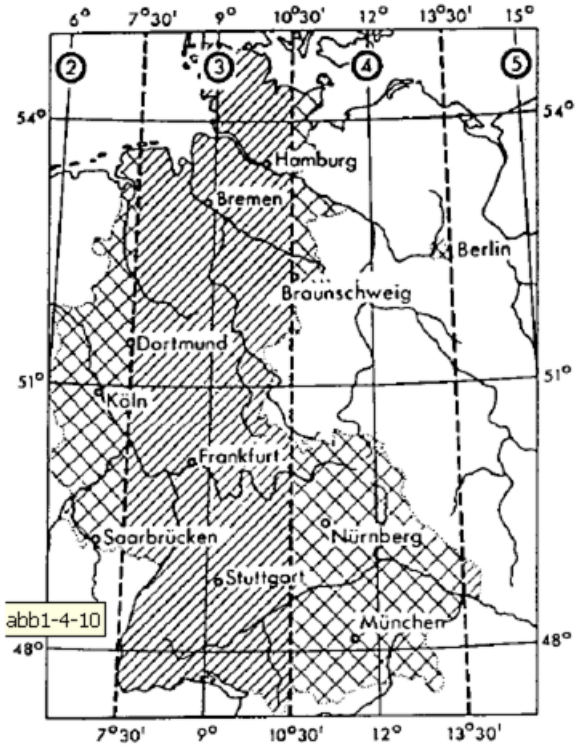


Abbildung 2.2: Meridianstreifen des Gauss-Krüger Koordinatensystems

Um einen Punkt innerhalb eines Meridianstreifens zu bestimmen, werden der Rechtswert (X) und der Hochwert (Y) wie in einem karthesischen Koordinatensystem abgelesen. Die Gauss-Krüger Koordinaten werden dabei in Metern angegeben. Den Meridianstreifen werden bestimmte Kennziffern zugeordnet, welche die erste Stelle des Rechtswertes darstellen. Dazu werden 500000 Meter addiert, um keine negativen Werte zu bekommen (z.B. X**=3**500000). Um den Rechtswert (X) des Punktes genau festzulegen, wird die Entfernung vom jeweiligen Mittelmeridian zu diesem Punkt addiert (z.B. X=3567870). Dabei ist zu beachten, dass die Entfernung nur in östlicher Richtung bestimmt werden darf. Der Hochwert (Y) gibt im Prinzip nur die Entfernung vom Äquator bis zu diesem bestimmten Punkt in Metern an (z.B. Y=5936661).

Diese als Beispiel angegebenen Gauss-Krüger Koordinaten (X=3567870,Y=5936661) stellen den Standort der HAW-Hamburg dar. Konkret befindet sich die HAW-Hamburg somit 67,87 Km östlich des 9° Meridianstreifens und 5936,661 Km nördlich des Äquators. Weitere Informationen sind dem [Kahmen (2006)] zu entnehmen.

2.2.2 Lokalisierungsdienste

In diesem Abschnitt werden zwei verschiedene Lokalisierungsdienste vorgestellt. Dabei ist zu beachten, dass diese Dienste nicht die einzigen möglichen Alternativen darstellen, sondern nur nach ihrer Relevanz für diese Arbeit ausgewählt wurden.

2.2.2.1 Global Positioning System

Als Global Positioning System (GPS) wird jedes globale satellitengestützte Navigationssystem bezeichnet. Konkret ist damit meistens das NAVSTAR (Navigational Satellite Timing and Ranging) System gemeint, welches vom Verteidigungsministerium der Vereinigten Staaten entwickelt und 1995 offiziell in Betrieb genommen wurde. Dieses System ist weltweit das wichtigste Ortungs- und Navigationssystem. In dieser Arbeit wird der Begriff GPS als Synonym für das NAVSTAR System verwendet.

Das GPS-System besteht aus 24 Satelliten (heute sind es meist um die 30 Satelliten), welche die Erde in 20200 Metern Höhe in sechs verschiedenen Umlaufbahnen umkreisen (siehe Abbildung 2.3). Dabei senden die Satelliten ständig ihre aktuelle Position und die genaue Uhrzeit. Ein GPS-Empfänger kann diese Signale empfangen und an Hand der Signallaufzeiten seine eigenen geographischen Koordinaten berechnen. Für die Positionsbestimmung sind theoretisch drei Satelliten notwendig, um eine korrekte Trilateration[4] durchzuführen. Außerdem muss der GPS-Empfänger über eine sehr genaue Uhrzeit verfügen, um die Signallaufzeiten der Satelliten präzise bestimmen zu können. Da das in der Praxis aber nicht der Fall ist, wird ein vierter Satellit benötigt, mit dem der GPS-Empfänger die Uhrzeit exakt bestimmen kann.

Abbildung 2.3: Umlaufbahnen der GPS-Satelliten

[4] Entfernungsmessung von drei Punkten aus

In der praktischen Anwendung spielt die Anzahl der empfangenen Satellitensignale ein große Rolle. Denn je mehr Messwerte zur Verfügung stehen, umso genauer kann die aktuelle Position bestimmt werden. Bei sieben verschiedenen Satellitensignalen können sehr präzise Positionsbestimmungen durchgeführt werden, die bei einer Genauigkeit von unter 10 Metern liegen.

Die sehr große Genauigkeit des GPS-Systems ist der Grund, dass es weltweit das am meisten eingesetzte Positionsbestimmungsverfahren ist. Darüber hinaus wird das GPS-Signal jedem kostenlos zur Verfügung gestellt, der einen GPS-Empfänger besitzt. Ein wesentlicher Nachteil ist aber, dass der GPS-Empfänger eine uneingeschränkte Sicht auf die Satelliten haben muss und es sich deshalb nur für die Outdoor Navigation eignet. Detailliertere Informationen zum GPS-System sind [Bauer (2002)] zu entnehmen.

2.2.2.2 Zellbasierte Ortung (Cell of Origin)

Die zellbasierte Ortung ist ein Verfahren, bei dem die Position an Hand der Funkzellen eines Mobilfunknetzes bestimmt wird. Dabei wird vorausgesetzt, dass das mobile Endgerät eines Mobilfunkteilnehmers bei einer Funkzelle angemeldet ist. Diesen Funkzellen sind Cell-ID's und geographische Koordinaten zugeordnet. Das mobile Endgerät kann diese Funkzelle durch ihre Cell-ID identifizieren und die geographischen Koordinaten der Funkzelle bestimmen. Somit kann die aktuelle Position des mobilen Endgerätes auf den Radius des Sendebereichs der jeweiligen Funkzelle eingegrenzt werden.

Der Sendebereich kann in ländlichen Gebieten allerdings mehrere Kilometer betragen, während in dichtbesiedelten Städten mehr Funkzellen zur Verfügung stehen und der Sendebereich meist unter 300 Metern liegt. Auch die Funkzellenanordnung spielt eine gewisse Rolle, denn die Funkzellen sollten sich so wenig wie möglich überschneiden, damit die Funkzellen eindeutig voneinander getrennt werden können. Somit ist die Positionsbestimmung bei diesem Verfahren selbst in dichtbesiedelten Städten sehr ungenau.

Die Vorteile der zellbasierten Ortung liegen darin, dass keine zusätzliche Hardware und keine Modifikation der mobilen Endgeräte notwendig sind. Außerdem kann die schon vorhandene Netzwerkinfrastruktur der Mobilfunkbetreiber genutzt werden. Ein weiterer sehr wichtiger Vorteil ist, dass kein direkter Sichtkontakt (wie beim GPS-System) erforderlich ist und dieses Verfahren somit auch für die Indoor-Navigation genutzt werden kann. Der Nachteil der zu ungenauen Positionsbestimmung macht trotz dieser Vorteile praktische Anwendungen im Navigationsbereich momentan unmöglich. Allerdings werden zunehmend neue Verfahren und Berechnungsmethoden entwickelt und getestet, die zum Beispiel mehrere Funkzellen,

Signalstärken und Signallaufzeiten verwenden. Für weitergehende Informationen wird allerdings auf [Dietz (2005)] verwiesen. Obwohl die zellbasierte Ortung für den Navigationsbereich zu ungenau arbeitet, wäre eine Kombination mit dem GPS-System vorstellbar. Damit könnten Nachteile beider Verfahren ausgeglichen werden, wie zum Beispiel der benötigte Sichtkontakt zu den Satelliten des GPS-Systems.

2.3 Datenübertragungstechniken

Da in dieser Arbeit ein mobiles Navigationssystem entwickelt werden soll, sind dafür standortabhängige Geoinformationen notwendig. Diese müssen von einem externen Internetdienstleister bezogen werden und machen eine effiziente und zuverlässige Datenübertragungstechnik erforderlich. Aus diesem Grund werden nachfolgend verschiedene Datenübertragungstechniken vorgestellt, die sich in ihren Leistungsmerkmalen erheblich unterscheiden (vgl. [Sauter (2008)]). Auf die GSM[5]-Technik wird an der Stelle nicht eingegangen, weil sie hauptsächlich für Sprachverbindungen vorgesehen ist.

2.3.1 General Packet Radio Service (GPRS)

Der General Packet Radio Service ist ein packetorientiertes Datenübertragungsverfahren. Dieser Service nutzt die bereits vorhandene Infrastruktur der GSM-Mobilfunknetze und baut auf der vorhanden GSM-Technik auf. Durch diese gut ausgebaute Infrastruktur ist die Verfügbarkeit sehr hoch und darüber hinaus wird GPRS von allen neuen Mobiltelefonen standardmäßig unterstützt. Zur Datenübertragung wird das Internet Protocol (IP) verwendet, welches die Daten in Paketen versendet. Um die Kapazität des GSM-Mobilfunknetzes effektiv nutzen zu können, setzt GPRS auf die Bündelung der GSM-Kanäle. Durch den maximalen Datendurchsatz von 21,4 kbit/s pro Kanal ergibt sich bei acht Kanälen eine theoretische Datenübertragungsrate von 171,2 kbit/s. Allerding beschränken die Netzbetreiber im praktischen Einsatz die Datenübertragungsrate auf bis zu 53,6 kbit/s .

Ein Vorteil von GPRS ist, dass keine langen Einwahlzeiten notwendig sind, weil ab der Aktivierung eine virtuelle Verbindung zur Gegenstelle besteht. Sobald Daten übertragen werden, wird ein Funkkanal für die Datenübertragung reserviert. Die dabei entstehenden Kosten werden meistens an Hand der übertragenen Datenmengen ermittelt. Allerdings ist zu beachten, dass ein Großteil der Mobilfunkbetreiber diese Datenmengen in kleinen Datenblöcken (meist 10 oder 100 Kb) abrechnet. Dabei können sehr leicht hohe Kosten entstehen und bei einem

[5]Global System for Mobile Communications- Standard für volldigitale Mobilfunknetze

zu erwartenden hohen Datenvolumen, sollten Tarife ohne Volumenbegrenzung (Flatrate) vorgezogen werden.

Zusammenfassend lässt sich sagen, dass GPRS durch die hohe Verfügbarkeit und die durchschnittliche Datenübertragungsrate ein mögliche Alternative für den Gelegenheitssurfer darstellt. Da das zu erwartende Datenvolumen in dieser Arbeit durch die benötigten Geoinformationen (Kartenmaterial) aber relativ hoch sein wird, wird von einem Einsatz dieses Verfahrens abgesehen.

2.3.2 Universal Mobile Telecommunications System (UMTS)

Das Universal Mobile Telecommunications System ist ein paketorientiertes Datenübertragungsverfahren der dritten Generation (3G) und verwendet das Internet Protocol (IP). Es wurde dafür entwickelt, einen schnellen mobilen Internet-Breitbandanschluss zur Verfügung zu stellen, mit dem multimediale Dienste (z.B. Video- und Datenanwendungen, Bildtelefonie) ermöglicht werden.

Der Start von UMTS begann im August 2000, als die UMTS-Lizenzen für Deutschland versteigert wurden und der Staat etwa 100 Milliarden DM (ca. 50 Milliarden Euro) von verschiedenen Mobilfunkbetreibern dafür erhalten hat. Seit dem 1. Quartal 2004 kann UMTS deutschlandweit in allen Mobilfunk-Netzen genutzt werden. Die UMTS-Infrastruktur wird allerdings immer noch ausgebaut, ist aber zum größten Teil in dichtbesiedelten Gebieten und Kleinstädten verfügbar. Darüber hinaus ist es dafür konzipiert in absehbarer Zeit seine Datenübertragungsrate durch verschiedene Modulationsverfahren und Anpassungen der UMTS-Infrastruktur auf bis zu 50 Mbit/s zu steigern.

Die momentan von allen deutschen Mobilfunkbetreibern unterstützte High-Speed-Downlink-Packet-Access (HSDPA) Technik der Kategorie 8 ermöglicht eine Empfangs-Datenübertragungsrate (download) von maximal 7,2 Mbit/s. Durch die Technik High Speed Uplink Packet Access (HSUPA) wird eine Sende-Datenübertragungsrate (upload) von maximal 1,4 Mbit/s bereitgestellt. Dabei ist allerdings zu beachten, dass die praktische Datenübertragungsrate auch von den technischen Fähigkeiten (unterstützte Modulationsarten) des Mobiltelefons abhängig sein können.

Die entstehenden Kosten für UMTS-Verbindungen werden wie bei GPRS meisten in Datenmengen abgerechnet. Allerdings machen die zunehmende Verfügbarkeit und der Wettbewerbsdruck der Mobilfunkbetreiber spezielle UMTS-Tarife mittlerweile erschwinglich. Die stetig zunehmende Verfügbarkeit und die hohen Datenübertragungsraten machen UMTS zu

der zukünftigen mobilen Datenübertragungstechnik. Somit erfüllt es alle notwendigen Anforderungen an ein Datenübertragungsverfahren und kann ohne Bedenken in dieser Arbeit verwendet werden.

2.3.3 Wireless Local Area Network (WLAN)

Das Wireless Local Area Network ist eine durch das IEEE[6] standardisierte Netzwerktechnologie, die einem Benutzer einen drahtlosen Zugang zu einem kabelgebundenen Netzwerk ermöglicht. Dieser drahtlose Zugang wird über einen Access Point (Funkzugangsknoten) realisiert. Clientseitig benötigt der Benutzer eine WLAN-Netzwerkkarte zur Herstellung einer Verbindung. Mit der aktuellen Version 802.11g der WLAN-Technologie wird eine Datenübertragungsrate von 54 Mbit/s ermöglicht. Allerdings soll Ende 2009 der neue Standard 802.11n schon einen Durchsatz von 74 Mbit/s zur Verfügung stellen. Die Funkreichweite des aktuellen Standards beschränkt sich dabei innerhalb von Gebäuden auf maximal 35 Meter und im Freien auf etwa 120 Meter.

Die WLAN-Technologie ist weltweit äußerst beliebt, weil der wirtschaftliche und technische Aufwand für die Installation sehr gering ist. Aus diesem Grund haben sehr viele Unternehmen, wie Flughäfen, Restaurants oder Hotels, sogenannte Hot Spots als zusätzlichen Service eingerichtet. Diese erlauben dem Kunden meist gegen Bezahlung einen drahtlosen öffentlichen Zugang zum Internet. Auch im privaten Gebrauch ist diese Technologie sehr beliebt, da Internetdienstleister bei Vertragsabschluss günstige Kombi-Geräte (WLAN-DSL-Router) anbieten. Da die WLAN's auch gewisse Sicherheitsrisiken beinhalten, wie das Abhören von Daten oder das Einbrechen in Netzwerke, wird die Funkverbindung meistens mit gängigen Verschlüsselungsmechanismen (z.B. WPA2) gesichert.

Die WLAN-Technologie ist eine sehr gute und sehr effektive Möglichkeit einen Zugang zum Internet zu realisieren. Darüber hinaus sind zwar viele WLAN's verfügbar aber diese sind größtenteils eigenständig, voneinander unabhängig und nicht öffentlich zugänglich, was eine flächendeckende Lösung unmöglich macht. Aus diesem Grund ist WLAN keine wirkliche Konkurrenz zur UMTS-Technologie. Da WLAN aber eine kostenlose und effektive Möglichkeit für eigene Netzwerke darstellt, wird sie in dieser Arbeit zu Testzwecken eingesetzt.

[6] Institute of Electrical and Electronics Engineers - weltweiter Berufsverband von Ingenieuren aus den Bereichen Elektrotechnik und Informatik

2.4 Laufzeitumgebung für mobile Anwendungen

Heutzutage sind auf dem freien Markt eine Reihe von Laufzeitumgebungen für mobile Endgeräte vorhanden. Dazu zählen das .NET Compact Framework, Google Androit und das von iPhone verwendete Objective-C. Alle genannten Umgebungen sind an eine spezielle Softwarefirma gebunden und erfüllen darüber hinaus nicht die Kriterien nach Plattformunabhängigkeit, Open-Source und Portabilität. Daher ist die Entscheidung auf die Java 2 Micro Edition (J2ME) von Sun Microsystems als Laufzeitumgebung für diese Arbeit gefallen, das im Folgenden vorgestellt wird.

Die Java 2 Micro Edition (J2ME) ist eine Umsetzung der objektorientierten Programmiersprache Java für sogenannte Embedded Devices, wie Mobiltelefone, PDAs, Autoradios und Settop-Boxen. J2ME basiert auf der Java 2 Standard Edition (J2SE) und stellt ebenso eine virtuelle Maschine sowie eine Sammlung von Bibliotheken und Programmierschnittstellen zur Verfügung. Diese Bibliotheken sind im Vergleich zu J2SE nur sehr begrenzt vorhanden, weil J2ME nur für Geräte mit eingeschränkten Ressourcen, wie geringer Rechenleistung, Speicherkapazität, Eingabe- und Darstellungsmöglichkeiten, konzipiert und optimiert worden ist.

Da J2ME von allen wichtigen Herstellern unterstützt wird und bei den meisten moderen Mobiltelefonen als Laufzeitumgebung bereits integriert ist, stellt es die ideale Programmiersprache zur Entwicklung der mobilen Anwendung dar. Darüber hinaus bietet J2ME die notwendige Plattformunabhängigkeit, Portabilität, ein umfangreiches Sicherheitskonzept sowie praxistaugliche Verfahren zur Installation von Anwendungen. Aus diesem Grund wird J2ME zur Entwicklung der mobilen Anwendung in dieser Arbeit verwendet. In den nachfolgenden Unterkapiteln wird eine kurze Einführung in die Grundlagen von J2ME gegeben, um einen Überblick über die Architektur und wichtige Bestandteile zu erhalten.

2.4.1 Architektur von J2ME

„Verglichen mit den größeren Editionen deckt die J2ME ein heterogenes Feld von Geräten ab. Der Versuch, ein einheitliches Softwareprodukt zu schaffen, das für alle Geräte gleichermaßen gut geeignet ist, wäre vermutlich zum Scheitern verurteilt gewesen. Die J2ME definiert sich durch verschiedene Spezifikationen, die zusammen mehrere Plattformen bilden. Jede dieser Plattform ist für eine bestimmte Kategorie von Endgeräten ausgelegt“(vgl.[Schmatz (2004)]).

In Abbildung 2.4 ist der strukturelle Aufbau von J2ME dargestellt. Dabei ist zu beachten, dass J2ME aus drei aufeinander aufbauenden Schichten besteht. Die unterste Schicht beinhaltet

die Konfigurationen, welche die Hardwareanforderungen an ein Gerät und eine Java Virtual Machine (JVM) definiert. Auf diesen Konfigurationen bauen verschiedene Profile auf, die gerätespezifische Bibliotheken zur Verfügung stellen. Die oberste Schicht enthält die optionalen Pakete, welche zusätzliche Funktionalität bereitstellen.

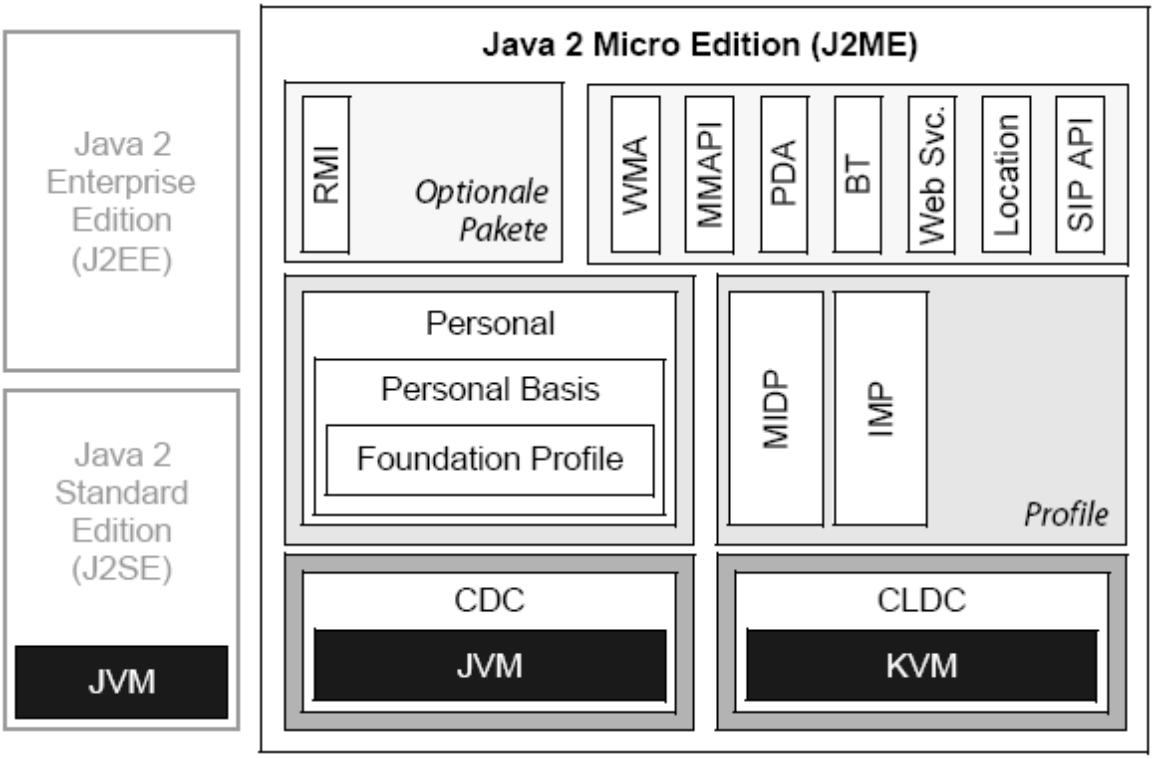


Abbildung 2.4: Strukturdiagramm von J2ME

Konfiguration

Die Konfiguration stellt basierend auf den Hardwaremerkmalen einer bestimmten Gruppen von Endgeräten eine Grundfunktionalität für die Java Plattform zur Verfügung. Diese Grundfunktionalität müssen die Hersteller implementieren, damit J2ME-Anwendungen die spezifischen Funktionen der Endgeräte nutzen können. Darüber hinaus wird für die verschiedenen Gruppen von Endgeräten der Funktionsumfang der Java Virtual Machine sowie die nutzbaren Java Klassenbibliotheken definiert.

Momentan stehen für J2ME nur zwei Konfigurationen zur Verfügung. Die Connected Limited Device Configuration (CLDC), welche auf mobile Endgeräte mit begrenzten Ressourcen ausgelegt ist. Als Zweites die Connected Device Configuration (CDC), welche für leistungsfähigere Endgeräte, wie high-end PDA's und Set-Top-Boxen eingesetzt werden kann. Darüber hinaus wird der CDC eine vollständige Java Virtual Machine bereitgestellt, während die

CLDC nur eine Kilobyte Virtual Machine (KVM) enthält, welche über begrenzte Fähigkeiten verfügt aber wenig Speicher verbraucht. Da in dieser Arbeit die CLDC für mobile Endgeräte genutzt wird, folgt in Kapitel 2.4.2 eine detailliertere Erläuterung zu dieser Konfiguration.

Profile

Die Profile bauen auf der Konfiguration auf und erweitern es um zusätzliche gerätespezifische API's, weil die unterschiedlichen Merkmale und Eigenschaften der mobilen Endgeräte eine einheitliche Nutzung unmöglich machen. Zum Beispiel haben die meisten Mobiltelefone unterschiedliche Displays. Mit einem Profil wird dem jeweiligen Hersteller die Möglichkeit gegeben, J2ME an die speziellen Eigenschaften anzupassen. Das wichtigste und meistgenutzte Profil ist das Mobile Information Device Profile (MIDP, siehe Kapitel 2.4.3).

Optionale Pakete

Mit optionale Paketen können die Profile um zusätzliche API's erweitert werden. Damit sind die Hersteller in der Lage, spezielle oder neue Leistungsmerkmale in die Endgeräte zu integrieren. Die bekanntesten optionalen Pakete für CLDC sind:

- Location API
- Web Service API
- Wireless Messaging API
- Bluetooth API
- PDA Optional Packages
- Mobile Media API
- Session Initiation Protocol API

Um die Funktionen der optionalen Pakete nutzen zu können, müssen die Endgeräte diese Pakete integriert haben. Eine nachträgliche Installation ist allerdings auch möglich. In dieser Arbeit werden die optionalen Pakete Location API, Web Service API und das Wireless Messaging API genutzt. Das Location API [JSR179 (2003)] kapselt den ganzen Lokalisierungsprozess und stellt die benötigten Informationen durch einige einfache Methoden zur Verfügung. Zum Beispiel kann es einen externen GPS-Empfänger über Bluetooth anbinden und automatisch die Daten abrufen, parsen und bereitstellen.

Das Web Service API [JSR172 (2004)] ermöglicht die Nutzung von entfernten Diensten (Webservice) durch eine mobile Anwendung. Dafür beinhaltet das API eine eingeschränkte Version des JAX-RPC (siehe Kapitel 2.5.2) von J2SE. Da JAX-RPC für XML-basierte Kommunikation konzipiert wurde, wird als Netzwerkprotokoll SOAP verwendet. Eine sehr wichtige Einschränkung stellt die Codierung der XML-Dokumente dar. Diese darf nur im Dokument/Literal- Format codiert sein, um eine korrekte Interpretation der Daten zu gewährleisten.

Mit dem Wireless Messaging API [JSR205 (2004)] ist das Versenden und Empfangen von Kurznachrichten (SMS) möglich. Ab der Version 2.0 werden auch multimediale Mitteilungen (MMS) von dem API unterstützt.

2.4.2 Connected Limited Device Configuration (CLDC)

Die CLDC ist eine Konfiguration für Endgeräte mit begrenzten Ressourcen, wie Mobiltelefone. Dafür definiert die CLDC eine Java Virtual Machine mit eingeschränktem Funktionsumfang, die sogenannte Kilobyte Virtual Machine (KVM). Außerdem wird nur eine minimale Klassenbibliothek bereitgestellt, die den Hardwareanforderungen dieser Konfiguration entspricht.

In dieser Arbeit wird die CLDC Version 1.1 verwendet, die in allen modernen Mobiltelefonen installiert ist. Die Klassenbibliotheken der CLDC bestehen aus zwei Teilbereichen: Der Erste bildet eine Teilmenge der J2SE Bibliothek, die mit den Packages *java.lang*, *java.util* und *java.io* die Kernfunktionen von Java bereitstellen. Der Zweite enthält mit dem Package *javax.microedition.io* das Generic Connection Framework der CLDC. Dieses Framework definiert an Stelle von *java.net* eine Sammlung von Interfaces zur Nutzung von gängigen Netzwerkprotokollen, wie zum Beispiel HTTP.

Die CLDC stellt somit die Grundkonfiguration für eine lauffähige Anwendung auf einem mobilen Endgerät zur Verfügung. Um gerätespezifische Eigenschaften der mobilen Endgeräte zu definieren ist ein Profil notwendig, das auf dieser Konfiguration aufbaut. Dieses Profil ist das Mobile Information Device Profile, welches im Folgenden erläutert wird.

2.4.3 Mobile Information Device Profile (MIDP)

Das am weitesten verbreitete und wichtigste Profil, welches auf der CLDC aufbaut, ist das Mobile Information Device Profile (MIDP). Die aktuelle Version MIDP 2.0 ist in allen modernen Mobiltelefonen vorinstalliert. Die Klassenbibliothek des MIDP 2.0 ergänzt die CLDC um

umfangreiche Programmierschnittstellen, mit denen sich vollwertige Anwendungen für mobile Endgeräte implementieren lassen. Die zusätzlichen Bibliotheken decken dabei folgende funktionale Bereiche ab:

- Steuerung des Lebenszyklus einer Anwendung
- Bedienoberflächen
- Persistente Datenhaltung (Recorde Store)
- Sicherheitsmodell und Signierung
- Unterstützung von Kommunikationsdiensten
- Audiounterstützung

In der nachfolgenden Abbildung 2.5 ist dargestellt, wie mit MIDP erstellte Anwendungen (Applikationen) einzuordnen sind. Die nativen Applikationen stellen systemeigene Funktionen zur Verfügung und können parallel zur MIDP Anwendungen laufen. Die Hersteller haben zusätzlich die Möglichkeit, eigene gerätespezifische API's zu integrieren und darauf aufbauend eigene Java Anwendungen zur Verfügung zu stellen, die den speziellen Eigenschaften und Merkmalen des mobilen Endgerätes entsprechen.

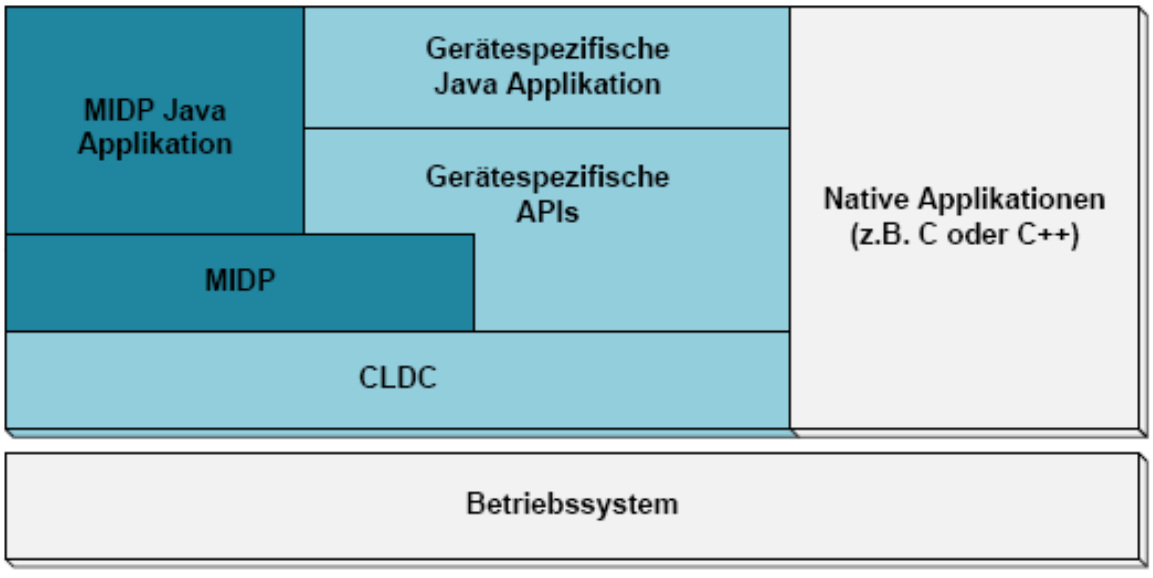


Abbildung 2.5: Einordung des MIDP

Die mit MIDP erstellten Anwendungen werden als MIDlets bezeichnet. Eine MIDlet ist eine Java-Klasse, die von der abstrakten Klasse MIDlet abgeleitet ist. Ein oder mehrere MIDlets bilden eine MIDlet-Suite, welche unter anderem benutzerdefinierte Klassen und Interfaces enthalten kann. Diese MIDlet-Suite kann als jar-Archiv verpackt werden und stellt die installierbare Anwendung dar. Ein MIDlet kann auf dem mobilen Endgerät gestartet, beendet oder in einen pausierenden Zustand versetzt werden. Die verschiedenen Zustände sind für die Steuerung des Lebenszyklus einer Anwendung notwendig. Der pausierende und der Start-Zustand wird zum Beispiel benötigt, damit grundlegende priorisierte Funktionen des mobilen Endgerätes, wie das Eingehen eines Anrufes oder einer SMS, die MIDP Anwendung unterbrechen und später fortsetzen können.

Die Sicherheit der MIDP-Anwendungen ist durch das sogenannte Sandbox[7]-Modell gewährleistet. Darüber hinaus sind sicherheitskritische Funktionen, wie zum Beispiel die Netzwerkkommunikation nur mit Zustimmung des Benutzers möglich. Außerdem wurde mit der Version MIDP 2.0 ein umfangreiches Berechtigungsmodell für MIDlets eingeführt, welches detailliert im [Schmatz (2004)] erläutert wird.

2.4.4 Entwicklungswerkzeuge

Das J2ME Wireless Toolkit (WTK) für CLDC ist ein Entwicklungwerkzeug von Sun Microsystems. Damit können Midlet-Suites übersetzt, paketiert und ausgeführt werden. Dafür stellt sie MIDP und CLDC Bibliotheken sowie eine Software-Emulation eines mobilen Endgerätes zur Verfügung. Darüber hinaus wird der Entwickler durch viele zusätzliche Funktionen unterstützt, wie zum Beispiel durch einen Stub-Generator für Webservices (siehe Kapitel 2.5) oder durch die Simulation von GPS-Koordinaten. Detaillierte Informationen dazu sind unter [WTK (2008)] zu finden.

Um das WTK innerhalb der Eclipse Entwicklungsumgebung nutzen zu können, kann das Plugin EclipseME [EclipseME (2008)] installiert werden, welches zusätzlich weitere Emulatoren und Debug-Möglichkeiten bereitstellt. Da das WTK in Verbindung mit Eclipse die Entwicklung einer J2ME-Anwendung erheblich vereinfacht, werden diese Entwicklungswerkzeuge in dieser Arbeit verwendet.

[7] Besonderheit der Laufzeitumgebung einer Software, um sie aus Sicherheitsgründen vom Rest des Systems abzuschirmen

2.5 Webservices

Ein Webservice ist eine Software-Anwendung, die definierte Dienste (z.B. Anwendungslogik) über standardisierte Internet Protokolle wie HTTP oder SMTP zur Verfügung stellen kann. Diese Dienste sind durch einen Uniform Resource Identifier (URI)[8] eindeutich identifizierbar. Durch ein XML-gekapseltes Protokoll gewährleisten Webservices eine zuverlässige Kommunikation und ermöglichen, heterogene Systeme miteinander zu kombinieren. Die Methoden eines Webservices, die von außen zugänglich sind sowie deren Parameter und Rückgabewerte, werden in der Web Service Definition Language (WSDL) detailliert beschrieben. Die am weitesten verbreitete Möglichkeit zur Kommunikation zwischen den verschiedenen Systemen ist SOAP über HTTP und TCP.

Das Simple Object Access Protocol (SOAP) ist ein plattformunabhängiges Netzwerkprotokoll zum Austausch von XML-basierten Nachrichten. Dafür kodiert es die Nachricht in XML und überträgt die Aufrufe an die Methode des Webservices sowie das Ergebnis der Anfrage.

Durch den Einsatz des HTTP Protokolls, welches den Standard-Port 80 verwendet, kann schon im Vorfeld ein sehr bekanntes Kommunikationsproblem vermieden werden. Dieses Problem kann durch spezielle Sicherkeitskonfigurationen, wie zum Beipiel von Firewalls, verursacht werden. Dabei werden nur Kommunikationsverbindungen von explizit freigegebenen Ports zugelassen. Zu diesen freigegebenen Ports gehört normalerweise auch der Standard-Port 80, weil der auch von Webbrowsern verwendet wird.

Um einen Webservice aufrufen zu können, sind Client-Stubs und Server-Skeletons notwendig. Diese dienen als Schnittstellen zwischen dem Client und dem Server. Sie kapseln den entfernten Methodenaufruf und stellen dem Entwickler die vom Webservice bereitgestellte Funktionalität transparent zur Verfügung. Dafür wandeln die Stubs oder Skeletons einen Methodenaufruf in eine SOAP Nachricht um oder entschlüsseln eine.

Es gibt in der Java Welt eine Reihe von Frameworks, die die Entwicklung von Webservices erleichtern. Eines der bekanntesten ist Apache Axis (Kapitel 2.5.1), die für den Betrieb von Webservices eine eigene Implementierung des JAX-RPC (Kapitel 2.5.2) Standards verwendet.

2.5.1 Apache AXIS

Apache AXIS für Java ist eine Open-Source-Implementierung des Webservice Standards SOAP unter der Lizenz der Apache Software Foundation. Mit AXIS können sehr einfach

[8] eine Zeichenfolge, die zur Identifizierung einer abstrakten oder physischen Ressource dient

Client- und Server-Anwendungen für Webservices entwickelt werden. Außerdem ist der Betrieb und das Anbieten von Webservices ebenfalls möglich.
Um die Entwicklung von Webservices zu unterstützen, werden unter anderem die Tools JAVA2WSDL und WSDL2JAVA zur Verfügung gestellt. Mit JAVA2WSDL kann aus einer definierten Schnittstelle und den dazugehörigen Methodenrümpfen und Parametern eine WSDL generiert werden. Mit dem Tool WSDL2JAVA können aus der WSDL die notwendigen Stubs oder Skeletons automatisch generiert werden. Dadurch kann sich der Entwickler ganz auf die Implementierung der Anwendungsfunktionalität konzentrieren. Apache Axis wird in dieser Arbeit zur Entwicklung und Bereitstellung von Webservices auf dem Server genutzt. Weitergehende Informationen sind [Dapeng Wang (2004)] zu entnehmen.

2.5.2 JAX-RPC

Die JAX-RPC ist eine API für die Webservice Entwicklung auf der Java Plattform. Dafür enthält es Bibliotheken, um Remote Procedure Calls auf XML-Basis ausführen zu können. Da JAX-RPC in einer eingeschränkten Version in dem J2ME WEB Service API enthalten ist, wird es für die Entwicklung der mobilen J2ME-Anwendung eingesetzt. Leider können mit dieser eingeschränkten Version keine Client-Stubs generiert werden. Aus diesem Grund wird der Stub-Generator des Sun-WTK für die automatische Generierung der Stubs verwendet. Detailliertere Informationen zu JAX-RPC sind im [Tobias Hauser (2003)] zu finden.

2.6 Zusammenfassung

In diesem Kapitel wurden Technologien vorgestellt, die zum Verständnis für den weiteren Verlauf dieser Arbeit notwendig sind und insbesondere die technischen Grundlagen für das Kapitel Design und Realisierung stellen. Dazu gehören Location-Based Services sowie das Gauss-Krüger- und geographische Koordinatensystem, Webservices und die mobile Laufzeitumgebung J2ME. Darüber hinaus wurden die Lokalisierungsdienste vorgestellt, die mit der Ausstattung eines modernen Mobiltelefons sinnvoll genutzt weden können. Dazu zählen GPS und die Zellbasierte Ortung. Der Vergleich der Datenübertragungstechniken ergab, dass für mobile Anwendungen mit hohem Datenvolumen mindestens GPRS, noch besser UMTS als Datenübertragungstechnik vorausgesetzt werden muss.

Kapitel 3

Vergleichbare Arbeiten

In diesem Kapitel findet eine Einordnung der Arbeit in den wissenschaftlichen Kontext statt und vergleichbare Ansätze werden vorgestellt.

3.1 Open-SPIRIT

Open-SPIRIT ist ein Indoor-Outdoor-Reiseinformations- und Navigationssystem für Smartphones, das sich insbesondere an Fahrgäste des öffentlichen Verkehrs richtet. Das Pilotprojekt Open-SPIRIT wurde im Juli 2004 in Österreich gestartet und im November 2005 beendet. Entwickelt wurde es von einem Konsortium aus Verkehrsverbünden, Forschungseinrichtungen und Softwarelieferanten, wobei die Projektkoordination von der ARC Seibersdorf research GmbH übernommen wurde. Darüber hinaus wurde das Projekt im Impulsprogramm "I2 - Intelligente Infrastruktur" des Bundesministeriums für Verkehr, Innovation und Technologie (BMVIT) gefördert [Stefan Bruntsch (2006)]. Folgende Ziele wurden zu Beginn des Projektes definiert:

- Intermodale und überregionale Tür-zu-Tür-Routenplanung
- Kontinuierliche und positionsabhängige Navigation entlang der geplanten Route sowie in Gebäuden und Umsteigeknotenpunkten
- Personalisierung und Speicherung von Routen in einem persönlichen Benutzerprofil
- Elektronische Fahrpreisauskunft und Ticketkauf (E-Ticketing)

In Abbildung 3.1 werden Screenshots der Open-SPIRIT-Anwendung dargestellt.

Abbildung 3.1: Open-SPIRIT Screenshots der Routenergebnisliste und der Navigationsdarstellungen

Das Open-SPIRIT Projekt ist als Client- Server-Architektur umgesetzt worden. Serverseitig wird die intermodale Routenplanung, die Berechnung von Indoor-Fußwegen und die Bereitstellung von Karten und Textanweisungen zur Indoor-Navigation bereitgestellt. Die Kommunikation zwischen dem Client und Server erfolgt über XML.

Der Client wurde mit J2ME umgesetzt und muss auf dem Smartphone installiert sein. Für alle Routenplanungsvorgänge und die Benutzereinstellungen wird ein Microbrowser benutzt, der die am Server generierten, XML-basierten Seiten darstellt. Routen- und Kartendaten können entweder lokal abgerufen oder vom Server mittels GPRS oder UMTS heruntergeladen werden. Damit diese Daten auch lokal abgerufen werden können, müssen sie zu Beginn komplett vom Server heruntergeladen und zwischengespeichert werden. Somit wird die Navigation auch ermöglicht, wenn keine Datenverbindung zum Server verfügbar ist.

Nachfolgend wird kurz erläutert, wie in Open-SPIRIT der Routenplaner, die Orientierung und Navigation sowie das Benutzerprofil und die Personalisierung umgesetzt wurden.

Routenplaner

Der Routenplaner ermöglicht dem Benutzer eine intermodale Tür-zu-Tür-Routenplanung. Er kann den Start- und Zielort, die Abfahrts- und Ankunftszeit sowie spezielle Einstellungen, wie z.B. kürzeste Verbindung oder bevorzugte Verkehrsmittel, eingeben. Als Ergebnis der Routenabfrage werden alternative Routen und Details zu jedem Abschnitt angezeigt. Darüber hinaus kann die Gesamtroute oder einzelne Abschnitte auf einer Karte angezeigt werden. Für große, komplexe Umstiegsgebäude ist es dann möglich, den in Open-SPIRIT entwickelten Orientierungs- und Navigationsdienst zu starten.

Orientierung und Navigation

Da es sich bei Open-SPIRIT um ein Indoor-Outdoor-System handelt, werden für die Positionsbestimmung verschiedene Lokalisierungsdienste verwendet. Im Außenbereich wird auf die weit verbreitete GPS-Technologie zurückgegriffen, die allerdings für die Positionsbestimmung innerhalb von Gebäuden ungeeignet ist. Deshalb wurden dafür zwei andere Ansätze verfolgt. Zum einen wurde die Bluetooth-Positionierung eingesetzt und zum anderen sollten Wegbeschreibungen so detailliert erarbeitet werden, dass der Benutzer mit Hilfe der Selbstpositionierung zum Ziel finden kann. Ebenso musste beachtet werden, dass nur für die Fußgängernavigation im Außenbereich Kartenmaterial von externen Dienstleistern zur Verfügung steht. Für den Innenbereich musste deshalb eine manuelle Erfassung durchgeführt werden. Da die Umstiegsnavigation der Hauptschwerpunkt der Entwicklungsarbeit in Open-SPIRIT gewesen ist, wird darauf etwas detaillierter eingegangen.

Um eine Bluetooth-Positionierung innerhalb eines Umsteigegebäudes zu ermöglichen, sind Bluetooth-Beacons[9] an definierten Stellen des Gebäudes angebracht (siehe Abbildung 3.2) worden. Diese Beacons senden dabei ihre Kennung aus und sind im System georeferenziert. Die Größe einer Bluetooth-Zelle (Senderadius) liegt bei 4 Metern, damit eine genaue Positionsbestimmung durchgeführt werden kann. Die Liste aller Bluetooth-Beacons eines Gebäudes kann der Client vom Server herunterladen. Zur Positionierung sucht der Client dann ständig nach den Bluetooth-Beacons und vergleicht die sichtbaren Beacons mit denen aus der Liste. Dadurch kann eine zellbasierte Positionierung durchgeführt werden. Mit Hilfe der Georeferenz der Beacons wird eine zugeordnete Karte ausgewählt und die Position des Benutzers eingezeichnet. Im Rahmen des Open-SPIRIT Projektes wurde zu Testzwecken die Station Matzleinsdorfer Platz in Wien mit Bluetooth-Beacons versehen.

[9] Bluetooth-Sender mit einer regulierbaren Signalstärke und autonomen Stromversorgung

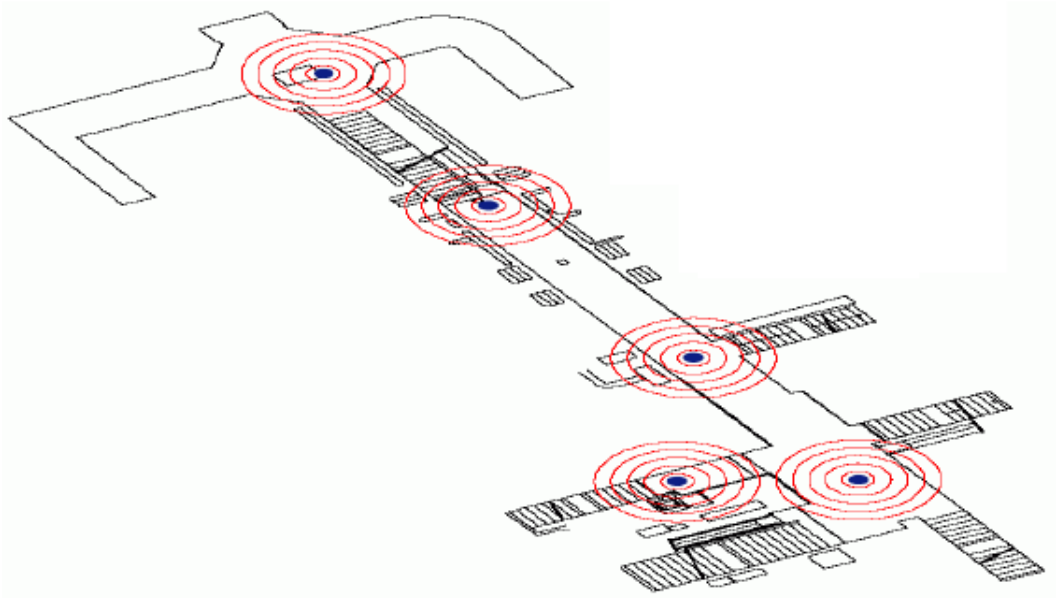


Abbildung 3.2: Plan der Bluetooth-Positionen am Matzleinsdorfer Platz in Wien

Damit die Navigation innerhalb von einem Umsteigegebäude ermöglicht wurde, musste die logische Struktur des Gesamtgebäudes manuell erfasst werden, da in der Regel keine geeigneten Daten in elektronischer Form vorhanden sind. Dafür mußten unter anderem unterschiedliche Ebenen, Bereiche, Eingänge, Fusswege, Treppen oder wichtige Hinweistafeln erfasst werden. Außerdem sind Meterangaben für Wegstrecken und spezielle Anweisungen, die einen konkreten Bezug auf die Umgebung und vorhandene Orientierungselemente haben („den Gang entlang bis zur Treppe"), erstellt worden.
Diese Daten wurden in ein Routingnetzwerk integriert, um Fusswege (Routen) in Gebäuden automatisch berechnen zu können. Dabei sollte die Orientierung und Navigation innerhalb eines Gebäudes durch visuelle und textuelle Präsentation sowie geeignete Leit- und Umgebungsinformationen unterstützt werden. Die textuelle Beschreibung für die Fusswege wurden aus den erfassten Daten sowie vordefinierten Textbausteinen zusammengesetzt (z.B."gehe die Treppe hinauf"). Um die visuelle Präsentation zu ermöglichen, wird eine dynamische, optimierte Karte aus dem geographischen Modell des Gebäudes vom Server generiert. Diese kann dann für einen kommenden Wegabschnitt auf dem Display dargestellt werden.

Die grafische Darstellung der geplanten Route wurde in Open-SPIRIT in einer 2D- und 3D- (perspektivischer Vektordarstellung) Vaiante entwickelt. Dabei wird der Benutzer auf dem Weg entlang der Route durch Anweisungen in Textform begleitet. Wenn kein Indoor-Positionierungsverfahren zur Verfügung steht, wird durch die textuellen Anweisungen sowie Leit- und Umgebungsinformationen dem Benutzer die Selbstpositionierung ermöglicht. Da-

bei kann er sich durch manuelles Weiterklicken zu den jeweiligen Streckenabschnitten orientieren und dem Verlauf der berechneten Route folgen (siehe Abbildung 3.3). Damit die berechnete Route auch offline angezeigt werden kann, ist das Herunterladen der benötigten Informationen zu Beginn der Reise möglich.

Abbildung 3.3: 2D-Indoor-Kartendarstellung

Benutzerprofil und Personalisierung

In dem Benutzerprofil kann der Benutzer persönliche Einstellungen und Daten speichern. Das ist insbesondere für den Routenplaner notwendig, da zum Beispiel oft benutzte Routen, bevorzugte Verkehrsmittel, schnellste oder kürzeste Verbindung und eigene POI's gespeichert werden können. Desweiteren ist es auch für den Ticketkauf (E-Ticketing) vorgesehen. Für das elektronischem Fahrgeldmanagement wurde das technische Konzept erarbeitet, welches allerdings nicht umgesetzt wurde.

Für Open-SPIRIT wurde am 20. und 21. Oktober 2005 an der Station Matzleinsdorfer Platz in Wien ein Benutzertest mit 20 Testpersonen durchgeführt. Dabei wurde eine Testroute mit öffentlichen Verkehrsmitteln geplant, bei der besonders viel Wert auf die Umstiegsnavigation an der Station Matzleinsdorfer Platz gelegt wurde. Das Feedback der Testpersonen ist dabei mit einer Bewertung von 2,43 (auf einer Skala von 1 bis 6) überwiegend positiv ausgefallen.

3.2 MONA

MONA bedeutet „Mobile Navigation mit öffentlichen Verkehrsmitteln" und ist ein Forschungsprojekt der Friedrich-Alexander-Universität Erlangen-Nürnberg. Das Ziel dabei ist, die Fahrgäste bei der Verwendung von öffentlichen Verkehrsmitteln zu unterstützen und sie bei der kompletten Fahrt zu begleiten. Im Rahmen dieses Forschungsprojektes sind zwischen 2006 und 2007 zwei Arbeiten von Studenten erstellt worden, die verschiedene Ansätze (online und offline) zur Umsetzungen dieses Zieles verfolgt haben. Die erste Arbeit *Mobile Fahrplanauskunft mit GPS und Webservices* (online-Ansatz) wird in Kapitel 3.2.1 vorgestellt und darauf folgend wird in Kapitel 3.2.2 die Arbeit *Autarke mobile Echtzeit-Fahrplannavigation* (offline-Ansatz) erläutert.

3.2.1 Mobile Fahrplanauskunft mit GPS und Web Services

Diese Diplomarbeit ist mit dem Ziel, eine vollautomatische Fahrplanauskunft für die mobile Navigation mit öffentlichen Verkehrsmitteln zu realisieren, zwischen 06.09.2006 und 06.03.2007 von Tobias Schwab erstellt worden [Schwab (2007)]. Der Testbereich beschränkt sich dabei auf den öffentlichen Nahverkehr der Stadt Erlangen im Verkehrsverbund Großraum Nürnberg (VGN). Zur Umsetzung wurde dafür eine Client-Server-Architektur verwendet. Der Client ist dabei mit Hilfe von J2ME realisiert worden, während auf dem Server das Framework Ruby on Rails zum Einsatz gekommen ist. Als Datenübertragungsverfahren für das mobile Endgeräte wurde GPRS und UMTS verwendet.

Mit der Client-Anwendung hat ein Benutzer die Möglichkeit die Adresse oder geographische Koordinaten eines Zieles einzugeben. Darüber hinaus kann er eine ortsbezogene Suche nach speziellen POI's, wie Sehenswürdigkeiten oder Geschäften, durchführen und diese dann als Ziel auswählen. Die aktuelle Position, welche mit der GPS-Technologie bestimmt wird, markiert dabei den Startpunkt. Mit dem Start- und Zielpunkt kann eine Fahrplanauskunft der Deutschen Bahn ermittelt und angezeigt werden. Zur Orientierung und Navigation wird die Entfernung und die Richtung zur jeweiligen Haltestelle oder zum Ziel angezeigt (siehe Abbildung 3.4).

Abbildung 3.4: Zielführung der mobilen Fahrplanauskunft

Nachfolgend wird kurz erläutert, wie die wichtigsten Konzepte der Arbeit umgesetzt wurden.

Haltestellenpositionen

Da für die Navigation die geographischen Koordinaten der Haltestellen notwendig sind und diese nicht verfügbar waren, wurde eine manuelle Erfassung der Haltestellenpositionen (mit Name und Adresse) durchgeführt. Zur groben Positionsbestimmung wurden dafür unter anderem topographische Liniennetzpläne des VGN verwendet. Eine genauere Position der Haltestellen kann durch charakteristische Eigenschaften wie Straßennamen oder Kreuzungen mit Hilfe von Google Earth identifiziert werden. Darüber hinaus bietet Google Earth die Möglichkeit, Punkte in den einzelnen Karten zu markieren und diese in Form von XML–Dateien zu exportieren. Für den Bereich des VGN wurden alle Haltestellen mit dieser Methode identifiziert und als XML-Dateien exportiert. Diese Dateien enthalten dann die Namen sowie die Koordinaten der Haltestellen und wurden auf dem Server hinterlegt.

Fahrplanauskunft

Um eine vollautomatische Fahrplanauskunft zu realisieren, wurde die elektronische Fahrplanauskunft der Deutschen Bahn verwendet. Die Deutsche Bahn bietet neben der üblichen HTML-Variante für Webbrowser auch eine für mobile Endgeräte optimierte Fahrplanauskunft

im WML[10]- Format mit dem Protokoll WAP an. Da die Ergebnisseiten der WML[11]- Variante viel einfacher gehalten sind, sich nur auf die wesentlichen Informationen beschränken und einen geringeren Kommunikationsaufwand benötigen, wurde diese Variante für eine Fahrplanauskunft ausgewählt. Dafür ist eine Schnittstelle auf dem Server entwickelt worden, die die Start- und Zieldaten an die Fahrplanauskunft der Deutschen Bahn überträgt und die benötigten Informationen aus der Ergebnisseite extrahiert.

Da als Startwert für die Fahrplansuche eine Haltestelle vorgesehen ist, werden in einer festgelegten Distanz zur aktuellen Position alle möglichen Haltestellen erfasst und zur Fahrplansuche verwendet. Mit jeder dieser Haltestellen führt der Server eine Fahrplananfrage durch und bewertet sie hinsichtlich der Fahrtdauer. Die beste Verbindung wird dann zum Client zurückgesendet.

Ortsbezogene Suche

Mit der ortsbezogenen Suche kann der Benutzer neben der eigentlichen Verbindungsauskunft nach bekannten Sehenswürdigkeiten suchen oder sich interessante Orte in seiner Umgebung auflisten lassen. Dafür wird in Wikipedia-Artikeln nach gespeicherten Ortsinformationen gesucht. Diese aufgelisteten Orte können dann als Ziel ausgewählt werden. Außerdem können mit GoYellow Telefonbucheinträge von Personen gesucht und deren Adressen ebenfalls als Ziel verwendet werden.

Zielführung

Die Zielführung, welche schon in Abbildung 3.4 dargestellt wurde, soll den Benutzer auf dem Weg zur nächsten Haltestelle bzw. zum Ziel navigieren und mit zusätzlichen Informationen versorgen. Mit dem schwarzen Pfeil wird dem Benutzer die Richtung zur Haltestelle relativ zur aktuellen Bewegungsrichtung angezeigt. Außerdem wird die Abfahrts- und Ankuftszeit sowie die Entfernung zur Haltestelle angezeigt. In einem Textfenster sind die detaillierten Informationen zum Fahrplan dargestellt.

Damit eine Zielrichtung bestimmt werden kann, wird aus den aktuellen Positions– und Bewegungsdaten eine Bewegungsrichtung bestimmt. Mit dieser Bewegungsrichtung und den Koordinaten des Zielpunktes kann dann die Zielrichtung berechnet werden. Um eine verwertbare Bewegungsrichtung zu erhalten, sind zwei Positionswerte (letzte Position, aktuelle Position) notwendig, die ca. 30 Meter auseinander liegen. Der Benutzer entscheidet dabei mit dem Menüpunkt *Aktualisieren*, wann ein neuer Wert ermittelt und die Bewegungsrichtung berechnet werden soll.

[10] Wireless Markup Language- ist eine XML-basierte Seitenbeschreibungssprache, die eine reduzierte Fassung von HTML darstellt

[11] Wireless Application Protocol- besteht aus Techniken und Protokollen, die für Mobiltelefone mit begrenzten Ressourcen optimiert wurden

Wenn die Adressen bzw. Koordinaten des Zieles oder eines anderen POI's benötigt aber nicht verfügbar waren, wurde das Geocoding[12] bzw. Reverse-Geocoding[13] von Google Maps zur Übersetzung verwendet.

3.2.2 Autarke mobile Echtzeit-Fahrplannavigation

Diese Studienarbeit hat das Ziel, ein Fahrplanauskunftssystem für öffentliche Verkehrsmittel zu erstellen. Dabei soll die Benutzung überwiegend offline auf mobilen Endgeräten erfolgen. Diese Arbeit wurde von Christian Schmidt bis August 2007 erarbeitet und beschränkt sich nur auf den öffentlichen Nahverkehr der Stadt Erlangen im Verkehrsverbund Großraum Nürnberg (VGN) [Schmidt (2007)].

Damit ein Fahrplanauskunftsystem entwickelt werden konnte, das ohne Netzwerkanbindung und Server auskommt, mussten alle notwendigen Daten (Kartenmaterial, Fahrpläne) auf dem mobilen Endgerät verfügbar gemacht werden. Darüber hinaus musste ein algorithmischer Hintergrund geschaffen werden, mit dem die Berechnung einer Fahrplanverbindung möglich ist. Mit Hilfe der GPS-Technologie wurde eine Bestimmung der eigenen aktuelle Position durchgeführt und zur Orientierung auf einer Karte zusammen mit der berechneten Fahrplanroute dargestellt.

Da es nicht möglich war, Fahrpläne von öffentlichen Verkehrsmitteln der Stadt Erlangen in maschinenlesbarer Form zu bekommen, wurden diese mit sehr großem Aufwand manuell erstellt und auf dem mobilen Endgerät hinterlegt. Bei möglichen Änderungen der Fahrpläne, müssen diese regelmäßig kontrolliert und aktualisiert werden, was sich sehr nachteilig auf den benötigten Wartungsaufwand auswirkt. Für eine Kartendarstellung im offline-Modus mussten die benötigten Kartendaten für den Bereich Erlangen vollständig von Google Maps heruntergeladen und ebenfalls auf dem mobilen Endgerät gespeichert werden. Die Kartendaten liegen dabei als Bilddaten im png-Format vor. Für das Kartenmaterial wurden Schnittstellen entwickelt, die Ausschnitte zu bestimmten Koordinaten und einer definierten Zoomstufe zurück geben und bestimmte Punkte und Routen auf der Karte graphisch darstellen können.

Zur Berechnung einer Fahrplanroute muss sowohl die Start- als auch die Zielhaltestelle angegeben werden. Bei der Eingabe von Haltestellen ist das Benutzerinterface relativ tolerant gegenüber Eingabefehlern bzw. vollständigen Haltestellennamen. Dafür wurde der

[12] mit Geocoding kann einer Adresse oder eines Namens (POI) die entsprechenden Koordinaten zugewiesen werden

[13] mit Reverse-Geocoding kann man mittels Angabe der Koordinaten den genauen Namen des Standortes feststellen

Damerau- Levenshtein-Abstand[14] benutzt, um die Ähnlichkeit zwischen Eingabe zu Haltestellennamen zu vergleichen, zu gewichten und die Ergebnisse entsprechend anzuzeigen (siehe Abbildung 3.5).

Abbildung 3.5: Kartenausschnitt und Eingabeformular

Für die Berechnung einer Fahrplanroute von einem Startpunkt zu einem Zielpunkt auf vorgegebenen Strecken wurden Methoden der Graphentheorie verwendet. Dabei werden die Haltestellen durch Knoten, die Verbindungen zwischen den Haltestellen durch Kanten und die zeitliche Entfernung durch die Gewichte des Graphen abgebildet. Zur Berechnung des kürzesten Pfades wurde der Algorithmus von Dijkstra verwendet. Die berechnete Fahrplanverbindung kann dann als Route mit zusätzlich annotierten Informationen (Linien, Abfahrtszeiten,etc.) auf der Karte dargestellt werden. Die Routensuche hat in Tests gezeigt, dass sie im Vergleich zum VGN den gesetzten Anforderungen genügt.

Da der Algorithmus, sowie die Speicherung der Fahrplan- und Kartendaten relativ aufwendig ist, muss das mobile Endgerät hohe Mindestanforderungen an Speicherkapazität und Rechenleistung erfüllen, damit das Programm überhaupt lauffähig ist. Darüber hinaus wurde die mobile Anwendung in ihrer Funktionalität sehr eingeschränkt, weil jede potentiell unsichere Aktion von dem Benutzer explizit bestätigt werden muss. Da für jede Buslinie und Variante eine eigene Datei benutzt wurde und das Laden einer Datei eine potentiell unsichere Aktion darstellt, wird der Benutzer mit Sicherheitsabfragen überhäuft. Die Lösung dieses Problems

[14] ein Maß für den Unterschied zwischen zwei Zeichenketten

benötigt eine gültige Lizenz für die Rechtevergabe, mit dem das MIDlet signiert sein muss. Da durch den Erwerb relativ hohe laufende Kosten entstehen, wurde dieses Problem bis zum Abschluss der Arbeit nicht gelöst.

3.3 Bewertung der bestehenden Arbeiten

Open-SPIRIT ist ein Indoor-Outdoor-Reiseinformationssystem und wurde als umfangreiches Forschungsprojekt von verschiedenen Verkehrsverbünden, Forschungseinrichtungen und Softwarelieferanten umgesetzt und finanziell vom Bundesministerium für Verkehr in Östereich unterstützt. Der Schwerpunkt dieses Projektes lag dabei auf der Indoor-Navigation. Dazu kann die Realisierung der Bluetooth-Positionierung angeführt werden sowie die Selbstpositionierung durch Leit- und Umgebungsinformationen. Die Navigation wurde durch übersichtliche graphische 2D- und 3D Umgebungsdarstellungen und ergänzende textbasierte Leitinformationen umfangreich unterstützt. Dabei muss allerdings beachtet werden, dass ein sehr hoher Aufwand für die Realisierung der Indoor-Navigation notwendig war. Dazu gehört zum Beispiel die manuelle Erfassung der logischen Struktur des Gesamtgebäudes.
Auf Grund der Tatsache, dass für dieses Projekt keine detaillierten Informationen zu personellen- und finanziellen Mitteln sowie zur Fahrplanauskunft und Outdoor-Navigation publiziert worden sind, ist an dieser Stelle eine umfassende Bewertung leider nicht möglich. Wenn von den aufgewendeten Ressourcen abgesehen wird, lässt sich zur Indoor-Navigation feststellen, dass richtigungsweisende Ansätze umgesetzt wurden.

Die *Mobile Fahrplanauskunft mit GPS und Web Services* ist die Online-Variante des Forschungsprojektes MONA und bietet ein gut umgesetztes Konzept einer vollautomatischen Fahrplanauskunft. Bei dieser Diplomarbeit sind die entwickelten Ideen, wie das Realisieren der Fahrplanauskunft über das WAP-Protokoll der Deutschen Bahn oder die manuelle Erfassung der Haltestellenpositionen mit Hilfe von Liniennetzplänen und Google Earth sehr positiv zu bewerten, weil mit den wenigen zur Verfügung stehenden Mitteln eine funktionsfähige Fahrplanauskunft realisiert worden ist. Darüber hinaus ist die ortsbezogene Suche nach Sehenswürdigkeiten, Geschäften und Personen mit Hilfe von Wikipedia-Artikel und GoYellow ebenfalls positiv zu bewerten. Das Zielführungsmenü macht einen sehr funktionalen Eindruck und stellt die notwendigen Fahrplaninformationen übersichtlich dar. Der zur Navigation vorgesehene Zielrichtungspfeil, könnte eine äußerst einfache Orientierungshilfe sein. Da er jedoch nur die richtige Richtung anzeigt, wenn man ca. 30 Meter geradeaus geht und die *Aktualisierung* betätigt, ist der Pfeil für die praktische Anwendung eher störend und irreführend. Darüber hinaus sind mögliche Umwege auf dem Weg zum Ziel auch sehr wahrscheinlich. Negativ aufgefallen ist unter anderem, dass die erarbeiteten Methoden und Konzepte nicht repräsentativ und übertragbar auf verschiedene Umgebungen sind, weil der

zugrundeliegende Testbereich (Erlangen) nicht die dafür notwendige Komplexität des vorhandenen Streckennetzes in Verbindung mit verwendeten Verkehrsmitteln (nur Bus) bietet. So ist zum Beispiel der Bereich von Hamburg (nur die Fläche) ca. 10 mal so groß, wie der von Erlangen. Da Hamburg über ca. 9600 Haltestellen und Stationen verfügt, würde darüber hinaus der Aufwand für eine manuelle Erfasssung der Positionen und Namen bzw. Adressen viel zu hoch werden. Obwohl die entwickelte Fahrplanauskunft sehr ideenreich ist, kann die Zuverlässigkeit und Verfügbarkeit dieses Verfahrens durch mögliche serverseitige Änderungen nicht garantiert werden. Leider wurde in der Diplomarbeit nicht darauf eingegangen aber es ist durchaus vorstellbar, dass rechtliche Datenschutzrichtlinien das automatische Abrufen von Fahrplaninformationen über das WAP-Protokoll untersagen, damit die Server der Deutschen Bahn nicht überlastet werden. Insbesondere, da diese Auskunft nur für mobile Endgeräte vorgesehen ist.
Im Großen und Ganzen birgt dieser Ansatz zwar einige gute Ideen, ist aber eher für ortsansässige Personen ohne Navigationsansprüche und für Städte mit relativ gering ausgebauten öffentlichen Nahverkehrsnetzen geeignet.

Die *Autarke mobile Echtzeit-Fahrplannavigation* ist die Offline-Variante des Forschungsprojektes MONA, bei der alle benötigten Informationen zu Fahrplänen und Navigation (Kartenmaterial) für den Bereich Erlangen auf dem mobilen Endgerät gespeichert wurden. Dabei wurden gut durchdachte algorithmische Verfahren zur Ermittlung der Fahrplanverbindung entwickelt und umgesetzt. Da diese Studienarbeit keine detaillierten Informationen enthält, wie die notwendigen Haltestellenpositionen und -namen ermittelt worden sind, ist davon auszugehen, dass diese Daten aus der Diplomarbeit *Mobile Fahrplanauskunft mit GPS und Web Services* übernommen wurden. Somit sind auch die Nachteile, die aus den manuell erfassten Daten entstehen und bei der Bewertung der Diplomarbeit schon erläutert wurden, auf diese Studienarbeit übertragbar. Dieser Nachteil wird noch dadurch verstärkt, dass ein sehr großer Aufwand betrieben wurde, um die Daten der Fahrpläne auf dem mobilen Endgerät abspeichern zu können. Trotzdem kann positiv hervorgehoben werden, dass die Ermittlung und Darstellung der Fahrplanroute auf Grund von lokal abgespeicherten Daten durchaus gelungen ist. Allerdings wäre eine zusätzliche textbasierte Darstellung des Fahrplans sinnvoller gewesen. Das bei dieser Arbeit entwickelte Konzept ist zwar gelungen aber es ist ebenfalls auf Grund des sehr hohen Aufwandes nicht einfach auf andere öffentliche Verkehrsnetze übertragbar.

Unter Berücksichtigung der vorgestellten Arbeiten wird im Folgenden ein mobiles Navigationssystem (NavMobile) konzipiert, welches aufgezeigte Schwachstellen der diskutierten Ansätze vermeidet und positive Aspekte miteinbezieht.

Kapitel 4

Analyse

In diesem Kapitel wird die erste Stufe in der Softwareentwicklung beschrieben. Dabei werden die Anforderungen an das Navigationssystem NavMobile in einem fachlichen Rahmen analysiert und spezifiziert. Die Anforderungen werden dabei in Funktionale und Nichtfunktionale unterteilt. Als Werkzeug wird eine standardisierte Modellierungssprache für Software verwendet, die Unified Modeling Language (UML). Damit wird in diesem Kapitel ein Anwendungsfalldiagramm modelliert, mit deren Hilfe sich die Anforderungen, Beziehungen und Zusammenhänge sehr gut darstellen und verdeutlichen lassen. Detailliertere Informationen dazu sind im [Störrle (2007)] zu finden.

Da in diesem Kapitel die Anforderungen vollständig und widerspruchsfrei definiert werden sollen, ist es unabdingbar die Komplexität so gering wie möglich zu halten, damit die gesamten Problemstellungen besser erfasst und in zweckmäßige Lösungen umgesetzt werden können. Deshalb wird auch sehr viel Wert darauf gelegt, sie nicht zu technisch darzustellen. Auf dieser Grundlage kann dann im nächsten Kapitel das technische Konzept bzw. Design entwickelt werden.

Im folgenden Abschnitt wird die Systembeschreibung einen Einblick in die Ideen und Vorstellungen für die NavMobile-Anwendung geben. Die daraus resultierenden funktionalen und nicht funktionalen Anforderungen werden in den darauf folgenden Abschnitten noch detailliert erläutert.

4.1 Systemidee

Es soll ein mobiles Navigationssystem „NavMobile” entwickelt werden, das auf vielen verschiedenen und weit verbreiteten mobilen Endgeräten lauffähig ist. Dazu soll es in der Lage

sein, eine nahtlose Route vom aktuellen Standort bis zu einem beliebigen Standort unter Verwendung des Fussweges und öffentlicher Verkekehrsmittel bereit zu stellen. Der Fussweg und alle benötigten Haltestellen sowie relevante Informationen (Verkehrsmittel, Abfahrts- und Ankunftszeiten) sollen dabei auf einer Karte dargestellt werden.

Da für ein Navigationssystem die Positionsbestimmung notwendig ist, muss NavMobile seinen eigenen Standort selbstständig ermitteln können. Dieser wird dann als Startpunkt für die Navigation verwendet. Den Zielpunkt, bei dem es sich um eine Straße, eine Haltestelle oder eine Besondere Stätte[15] handeln kann, soll der Benutzer über ein Eingabeformular eingeben können. Aus dem Start- und Zielpunkt wird dann der notwendige Fahrplan ermittelt und detailliert auf dem Display dargestellt.

Zur einfachen Orientierung soll es außerdem möglich sein, den eigenen Standort, das Ziel und alle Zwischenhaltestellen als POI's auf einer Karte darstellen zu lassen. Der eigene Standort muss dabei zeitnah aktualisiert und auf der Karte verfolgt werden können. Die Haltestellen werden zur besseren Orientierung mit einer kurzen Beschreibung versehen, auf der die Abfahrtszeit, die Ankunftszeit und benötige Verkehrsmittel hinterlegt sind.
Um dann noch die Navigation zur jeweiligen Haltestelle bzw. zum Zielort für den Benutzer zu vereinfachen, soll der geplante Fussweg auf der Karte als Route angezeigt werden.

Somit werden dem Benutzer alle Informationen, die zur Erreichung des Ziels notwendig sind, übersichtlich auf der Karte präsentiert, so dass er folgende Fragen jederzeit beantorten kann:

- Wo befinde ich mich gerade?
- Wie komme ich zur jeweiligen Haltestelle oder zum Ziel?
- Wann komme ich am Ziel an und was sind die Abfahrts- und Ankuftszeiten der Verkehrsmittel?
- Mit welchem Verkehrsmittel muss ich fahren?
- Wo muss ich evtl. in welches Verkehrsmittel umsteigen und wieviel Zeit bleibt mir dafür?
- Wieviel Geld muss ich für einen Fahrschein bezahlen?

[15] besondere Orte, wie Hotels, Behörden, Firmen, touristisch interessante Orte (z.B. Stadion des HSV)

4.2 Funktionale Anforderungen

Funktionale Anforderungen legen grundsätzlich fest, welche Aufgaben oder Dienste ein Produkt erledigen bzw. bereitstellen soll.
In dem folgendem Anwendungsfalldiagramm wird eine Gesamtübersicht über alle Aufgaben und Dienste, welche die Anwendung NavMobile zur Verfügung stellen soll, gegeben. Die beteiligten Akteure aus dem Diagramm werden danach kurz beschrieben und die einzelnen Anwendungsfälle werden identifiziert und deren Verhalten spezifiziert.

4.2.1 Anwendungsfalldiagramm

Das folgende Diagramm zeigt die fachlichen funktionalen Anforderungen an das Gesamtsystem mit allen beteiligten Akteuren an. Danach werden die Zusammenhänge in kurzer Form erläutert.

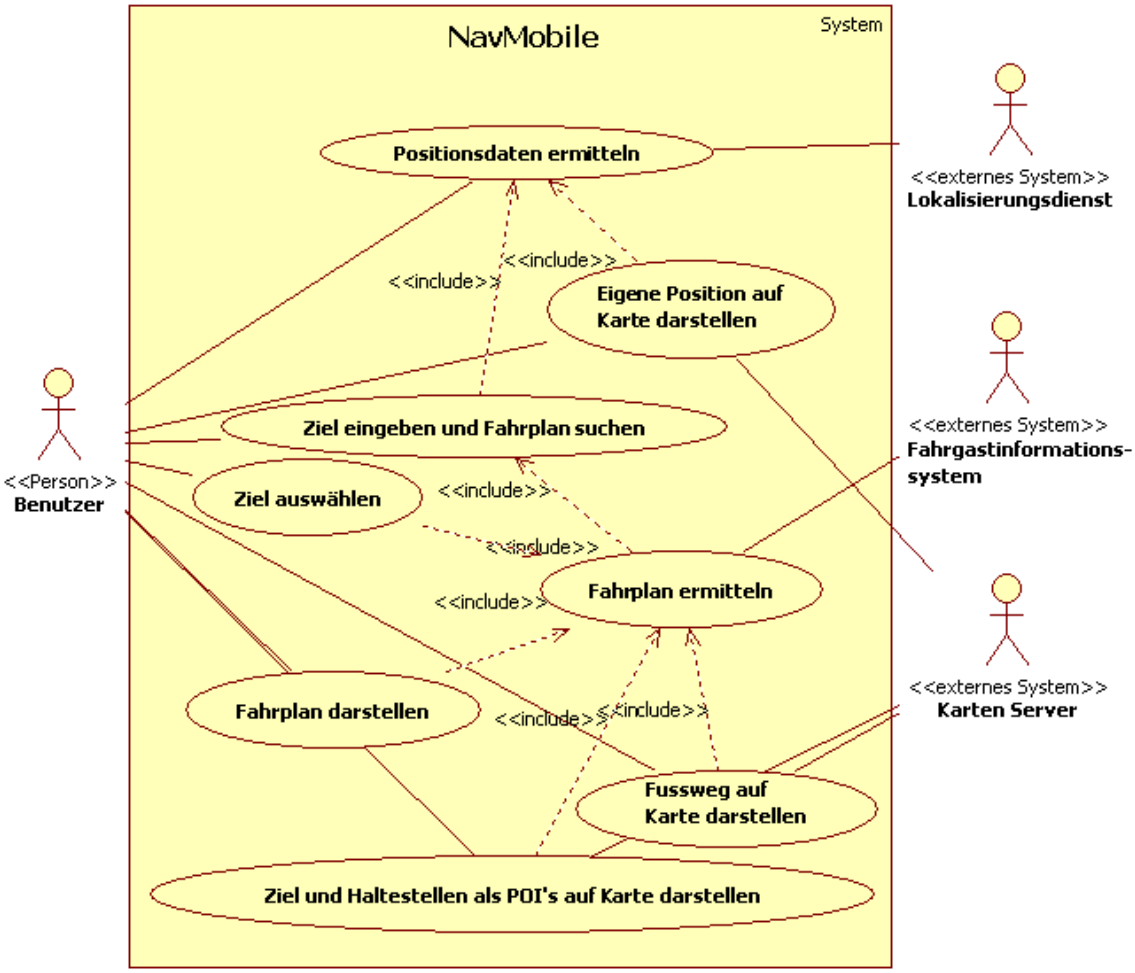


Abbildung 4.1: Anwendungsfälle des mobilen Navigationssystems „NavMobile”

Der Benutzer ist der Akteur, der mit dem Navigationssystem NavMobile interagieren kann. Die anderen Akteure, wie der Lokalisierungsdienst, Fahrgastinformationsdienst und der Karten Server sind externe Systeme, die benötigte Dienste für die jeweiligen Anwendungsfälle bereitstellen.

Als erstes muss der Benutzer die *Positionsdaten ermitteln*, damit die Voraussetzung für alle anderen Anwendungsfälle geschaffen wird. Dieser Anwendungsfall beinhaltet, dass in regelmäßigen Abständen, in Zusammenarbeit mit einem Lokalisierungsdienst, die Positionsdaten maschinell ermittelt werden. Als nächstes kann der Benutzer das *Ziel eingeben und den Fahrplan suchen* oder seine *Eigene Position auf der Karte darstellen* lassen. Für letzteren Anwendungsfall muss außer den Positionsdaten noch zusätzlich Kartenmaterial vom Karten

Server bezogen werden. Der Anwendungsfall *Ziel eingeben und Fahrplan suchen* benötigt ebenfalls die Positionsdaten, weil für die Ermittlung einer optimierten Fahrplanverbindung auch der Startpunkt notwendig ist. Wenn das Ziel eingegeben und die Suche bestätigt wurde, kann in Zusammenarbeit mit einem Fahrgastinformationsdienst der optimierte *Fahrplan ermittelt* werden.

Wird der Fahrplan ermittelt, kann es dazu kommen, dass das Ziel nicht eindeutig ist. In diesem Fall werden die verschiedenen Alternativen zur Auswahl bereitgestellt und der Benutzer muss das *Ziel auswählen*. Nach der Bestätigung wird *Fahrplan ermitteln* erneut ausgeführt. Sobald das Ziel eindeutig ist, wird der Anwendungsfall *Fahrplan darstellen* automatisch ausgeführt.

In diesem Fall sind dann auch die letzten Anwendungsfälle möglich. *Fahrplan ermitteln* stellt die benötigten Haltestellen mit ihrem genauen Standort zur Verfügung, um die Anwendungsfälle *Fussweg auf Karte darstellen* und *Ziel und Haltestellen als POI's auf Karte darstellen* zu ermöglichen. Diese benötigen ebenfalls zusätzliches Kartenmaterial von dem Karten Server.

Als Letztes ist zu beachten, dass alle Anwendungsfälle, die sich auf die Kartendarstellung beziehen (*Eigene Position auf Karte darstellen*, *Fussweg auf Karte darstellen, Ziel und Haltestellen als POI's auf Karte darstellen*) gleichzeitig ausführbar sind, während die anderen Anwendungsfälle jeweils nur einzeln nacheinander ausführbar sind. Die Ausnahme bildet der Anwendungsfall *Positionsdaten ermitteln*, welcher während der gesamten Laufzeit des Systems aktiv ist.

Die genaue Spezifikation der Akteure und Anwendungsfälle folgt dann in den Kapiteln 4.2.2 und 4.2.3.

4.2.2 Beteiligte Akteure

Die in dem Anwendungsfalldiagramm dargestellten Akteure werden hier kurz in tabellarischer Form erläutert.

Akteur	Beschreibung
Benutzer	Der Benutzer agiert von einem mobilen Endgerät aus mit der Anwendungssoftware. Er macht Angaben zum Ziel und sucht nach einen auf ihn zugeschnittenen Fahrplan und möchte mit Hilfe einer Kartendarstellung zum Ziel navigiert werden.
Lokalisierungsdienst	Ein Dienst, der mit Hilfe von einem Ortungsverfahren die geographischen Koordinaten berechnen und zur Verfügung stellen kann.
Fahrgastinformationssystem	Ein elektronisches Informationssystem für den öffentlichen Personennahverkehr, das es erlaubt, detaillierte und optimierte Fahrplanverbindungen zu berechnen und abzufragen.
Karten-Server	Ein Service-Provider wie Google oder MSN, der es erlaubt, zu bestimmten Koordinaten die dazu passenden Kartenausschnitte herunterzuladen. Darüber hinaus ist er in der Lage durch Angabe eines Start- und Zielpunktes Routen zu berechnen und zur Verfügung zu stellen.

Tabelle 4.1: Beteiligte Akteure des mobilen Navigationssystems

4.2.3 Spezifikation der Anwendungsfälle

In diesem Abschnitt werden die Spezifikationen der Anwendungsfälle aus der Abbildung 4.1 detailliert in tabellarischer Form erläutert.

ID und Name	**UC1: Positionsdaten ermitteln**
Beschreibung	Eine Verbindung mit dem Lokalisierungsdienst soll hergestellt werden und die genauen Positionsdaten (geographische Koordinaten) sollen in regelmäßigen Abständen abrufbar sein.
Beteiligte Akteure	Benutzer, Lokalisierungsdienst
Auslöser	Benutzer startet die NavMobile Anwendung
Vorbedingungen	Lokalisierungsdienst ist betriebsbereit
Nachbedingungen	Verbindung zum Lokalisierungsdienst ist hergestellt und die Positionsdaten werden geliefert.
Ausnahmen	Der Benutzer soll darüber informiert werden,wenn der Lokalisierungsdienst nicht betriebsbereit ist oder keine Positionsdaten ermitteln kann.

Tabelle 4.2: Anwendungsfall „Positionsdaten ermitteln”

ID und Name	**UC2: Eigene Position auf Karte darstellen**
Beschreibung	Eine Karte soll dargestellt und die eigene aktuelle Position angezeigt und in regelmäßigen Abständen aktualisiert werden.
Beteiligte Akteure	Benutzer, Karten-Server, Lokalisierungsdienst
Auslöser	Benutzer wählt Kartendarstellung aus
Vorbedingungen	Der Anwendungsfall „Positionsdaten erstellen(UC1)“ wurde erfolgreich ausgeführt und ein Karten-Server ist über eine Internetverbindung verfügbar.
Nachbedingungen	Die eigene Position wird auf der Karte eingetragen und die Karte wird dargestellt. Änderungen der eigenen Position werden dabei in regelmäßigen Abständen überprüft und angezeigt.
Ausnahmen	Wenn keine aktuellen Positionsdaten verfügbar sind, soll der Benutzer darauf hingewiesen und die letzte bekannte Position auf der Karte dargestellt werden.

Tabelle 4.3: Anwendungsfall „Eigene Position auf Karte darstellen”

ID und Name	**UC3: Ziel eingeben und Fahrplan suchen**
Beschreibung	Ein Eingabeformular soll dargestellt werden, indem man das Ziel eingeben und anschließend die Fahrplansuche starten kann.
Beteiligte Akteure	Benutzer
Auslöser	Benutzer wählt Zieleingabe aus
Vorbedingungen	Der Anwendungsfall „Positionsdaten erstellen(UC1)“ wurde erfolgreich ausgeführt und eine Internetverbindung ist verfügbar.
Nachbedingungen	Die Anwendung wechselt automatisch in den Anwendungsfall „Fahrplan ermitteln(UC4)“.
Ausnahmen	Bei falschen Eingabedaten sowie bei fehlerhafter Internetverbindung soll der Benutzer darüber informiert werden.

Tabelle 4.4: Anwendungsfall „Ziel eingeben und Fahrplan suchen”

ID und Name	**UC4: Fahrplan ermitteln**
Beschreibung	Der Fahrplan soll gemäß den Zieleingaben ermittelt werden.
Beteiligte Akteure	Fahrgastinfomationssystem
Auslöser	Der Benutzer hat die Fahrplansuche ausgewählt.
Vorbedingungen	Der Anwendungsfall „Ziel eingeben und Fahrplan suchen(UC3)“ wurde erfolgreich ausgeführt.
Nachbedingungen	Der ermittelte Fahrplan wird an den Anwendungsfall „Fahrplan darstellen(UC6)“ weitergeleitet.
Ausnahmen	Ist das Ziel des ermittelten Fahrplans nicht eindeutig, wird das Ergebnis an den Anwendungsfall „Ziel auswählen(UC5)” weitergeleitet. Bei fehlerhafter Fahrplanermittlung wird der Benutzer darüber informiert.

Tabelle 4.5: Anwendungsfall „Fahrplan ermitteln”

ID und Name	**UC5: Ziel auswählen**
Beschreibung	Die verschiedenen Möglichkeiten des Ziels werden dargestellt, von denen dann eine ausgewählt werden kann.
Beteiligte Akteure	Benutzer
Auslöser	Bei der Ermittlung des optimierten Fahrplans(UC4) tritt eine Mehrdeutigkeit des Ziels ein.
Vorbedingungen	Der Anwendungsfall „Fahrplan ermitteln(UC4)“ wurde erfolgreich ausgeführt.
Nachbedingungen	Der Anwendungsfall „Fahrplan ermitteln(UC4)“ wird mit dem ausgewählten Ziel als Parameter ausgeführt.
Ausnahmen	-

Tabelle 4.6: Anwendungsfall „Ziel auswählen”

ID und Name	**UC6: Fahrplan darstellen**
Beschreibung	Der ermittelte Fahrplan soll übersichtlich und strukturiert dargestellt werden. Er soll nach der Fahrplansuche automatisch angezeigt und zwischengespeichert werden, damit er jederzeit verfügbar ist.
Beteiligte Akteure	Benutzer
Auslöser	Benutzer wählt Fahrplandarstellung aus oder der Anwendungsfall „Fahrplan ermitteln(UC4)“ wurde erfolgreich ausgeführt. Dann folgt eine automatische Weiterleitung.
Vorbedingungen	Der Anwendungsfall „Fahrplan ermitteln(UC4)“ wurde erfolgreich ausgeführt.
Nachbedingungen	Der Fahrplan wird dargestellt.
Ausnahmen	Ist noch kein Fahrplan verfügbar,soll der Benutzer darauf hingewiesen werden.

Tabelle 4.7: Anwendungsfall „Fahrplan darstellen”

ID und Name	**UC7: Fussweg auf Karte darstellen**
Beschreibung	Der geplante Fussweg von der eigenen Position bis zur Haltestelle bzw. zum Ziel soll auf der Karte dargestellt werden.
Beteiligte Akteure	Benutzer, Karten-Server, Lokalisierungsdienst
Auslöser	Benutzer wählt Wegdarstellung aus
Vorbedingungen	Die Anwendungsfälle „Positionsdaten ermitteln(UC1)“und „Fahrplan ermitteln(UC4)“ wurden erfolgreich ausgeführt.
Nachbedingungen	Die eigene Position, die Ziel-Position und der dazugehörige Fussweg wird auf der Karte dargestellt.
Ausnahmen	Wenn der Fussweg nicht verfügbar oder der Karten-Server nicht erreichbar ist, soll der Benutzer darüber informiert werden.

Tabelle 4.8: Anwendungsfall „Fussweg auf Karte darstellen”

ID und Name	**UC8: Ziel und Haltestellen als POI’s auf Karte darstellen**
Beschreibung	Das Ziel und alle benötigten Haltestellen sollen als POI’s auf der Karte dargestellt werden. Zusätzlich sollen Informationen wie Abfahrts-, Ankunftszeit und Verkehrsmittel annotiert werden.
Beteiligte Akteure	Benutzer
Auslöser	Benutzer wählt Wegdarstellung aus
Vorbedingungen	Der Anwendungsfall „Fahrplan ermitteln(UC4)“ wurde erfolgreich ausgeführt.
Nachbedingungen	Das Ziel, alle ermittelten Haltestellen aus UC4 und deren zusätzliche Informationen werden auf der Karte dargestellt.
Ausnahmen	Wenn der Karten- Server nicht erreichbar ist, soll der Benutzer darüber informiert werden.

Tabelle 4.9: Anwendungsfall „Ziel und Haltestellen als POI’s auf Karte darstellen”

4.3 Nichtfunktionale Anforderungen

Nichtfunktionale Anforderungen beziehen sich meist auf ein System als Ganzes, nicht auf einzelne Funtionen (vgl.[Buth (2006)]). Der Schwerpunkt liegt dabei in der Art und Weise, wie etwas gemacht wird und nicht auf deren Funktion.
Im folgenden Abschnitt werden die verschiedenen Arten von nichtfunktionalen Anforderungen im Bezug auf die zu entwickelnde Anwendung klassifiziert und spezifiziert.

Benutzbarkeit/Benutzerfreundlichkeit: Da sich die Anwendungssoftware auch an Menschen mit geringem technischen Verständnis richtet, ist die Benutzerfreundlichkeit ein wichtiger Faktor bei der Softwareentwicklung. Um dieses zu gewährleisten muss die Software einfach und intuitiv zu bedienen sein. Dazu gehört auch eine einfache Installation der mobilen Anwendung und eine strukturierte Menüführung, die leicht verständlich und zum größten Teil selbsterklärend sein soll.

Zuverlässigkeit: Damit die Zuverlässigkeit und Stabilität verbessert wird, muss der Anteil der Ereignisse, die zu Fehlfunktionen führen können, minimiert werden. Außerdem sollte die Verfügbarkeit aller Systemkomponenten überprüft und sichergestellt werden. So können z.B. redundante Komponenten auftretende Störungen oder Ausfälle kompensieren. Um die Zuverlässigkeit noch weiter zu verbessern, sollte auch die Fehlertoleranz nicht außer Acht gelassen werden. Deshalb muss darauf geachtet werden, dass bei fehlerhaften Abläufen oder Eingaben ein Programmabsturz verhindert und stattdessen der Benutzer durch gut platzierte und formulierte Info- oder Warnmeldungen unterstützt wird.

Korrektheit: Bei der Entwicklung von Methoden und Funktionen ist es besonders wichtig, dass korrekte Ergebnisse zurückgeliefert werden. Diese sind gegebenfalls durch umfangreiche Tests sicherzustellen. Sind keine Tests im herkömmlichen Sinne möglich, z.B. wenn Ergebnisse von externen Komponenten genutzt werden, so muss deren Korrektheit durch aussagekräftige Referenzen (Fachartikel, Erfahrungsberichte) überprüft werden.

Portierbarkeit/Übertragbarkeit: Damit die Portierbarkeit und Übertragbarkeit gewährleistet wird, sollte die Wahl auf eine plattformunabhängige Programmiersprache, deren Verbreitungsgrad dazu noch sehr hoch ist, fallen. Eine standardisierte Kommunikation zwischen den beteiligten Systemkomponenten unterstützt dazu noch die nötige Flexibilität.

Wartbarkeit/Änderbarkeit/Erweiterbarkeit: Bei der Softwareentwicklung sollte auf die Modularität der einzelnen Softwarekomponenten geachtet werden, um sie leichter austauschbar,wartbar und erweiterbar zu machen. Eine gute Dokumentation sowie eine

klar definierte mehrschichtige Softwarearchitektur sollten diese Anforderungen noch weiter unterstützen.

Effizienz/Performance: Da gerade für Anwendungen auf mobilen Endgeräten die Effizienz eine sehr große Rolle spielt, muss unbedingt darauf geachtet werden, rechenintensive Aufgaben zu vermeiden oder sie auf externe, leistungsstarke Komponenten zu verlagern. Des weiteren müssen bei der Kommunikation mit externen Systemkomponenten die ausgetauschten Informationen nur auf das Notwendigste beschränkt werden, weil nicht davon auszugehen ist, dass ein mobiles Endgerät überall und jederzeit über eine hohe Bandbreite verfügen kann. Diese Reduzierung des Overheads fördert nicht nur die Effizienz des mobilen Endgerätes, sondern auch die der externen Systemkomponenten und verbessert noch dazu deren Skalierbarkeit.

Datenschutz/Sicherheit: Als letzten Punkt der nichtfunktionalen Anforderungen wird auf den Datenschutz und die Sicherheit eingegangen. Dabei ist zu beachten, dass Services von externen Dienstleistern genutzt und Positionsdaten an sie gesendet werden sollen. Somit können aus Datenschutzgründen die externen Karten-Server als sicherheitskritisch eingestuft werden. Wenn positionsbezogenes Kartenmaterial oder Routeninformationen direkt von einem mobilen Endgerät abgerufen werden, so ist dieser eindeutig identifizierbar und lokalisierbar. Aus diesem Grund müssen die Datenschutzrichtlinien der jeweiligen Anbieter überprüft werden, damit eine unsachgemäße Verwendung der Positionsdaten ausgeschlossen werden kann. So könnten z.B. Bewegungsprofile des Benutzers erstellt und gespeichert werden.
Aus diesem Grund ist es auch sinnvoll, die Kommunikation mit den beteiligten Systemkomponenten über eine verschlüsselte Verbindung abzuwickeln. Das erhöht die Sicherheit, benötigt aber auch mehr Rechenleistung und erzeugt mehr Overhead. Deshalb sollte dem Benutzer die Möglichkeit gegeben werden, zwischen einer verschlüsselten und einer unverschlüsselten Verbindung zu wählen. Damit ist er dann auch in der Lage die Sicherheitsrelevanz der übertragenen Daten selbst zu beurteilen und dementsprechend die Art der Verbindung zu wählen.

4.4 Zusammenfassung

In diesem Kapitel wurden umfassend die funktionalen Anforderungen mit Hilfe von Use Cases dargestellt und die notwendigen nichtfunktionalen Anforderungen erläutert. Diese bilden die Grundlage für das Design und die Realisierung.

Kapitel 5

Design und Realisierung

In diesem Kapitel wird das technische Konzept bzw. Design der Anwendung entwickelt. Dafür werden die in Kapitel 4 formulierten und definierten Anforderungen zu Hilfe genommen. In Kapitel 5.4 wird das entwickelte Design bei der Realisierung der Softwareanwendung umgesetzt.

5.1 Designentscheidung

Für die Entwicklung der NavMobile Anwendung, die den Anforderungen aus Kapitel 4 genügt, sind verschiedene komplexe Aufgaben zu bewältigen. Das Zusammenspiel und die Integration der verschiedenen Aufgaben werden dabei das Design der Anwendung entscheidend beeinflussen. Da es aber aus zeitlichen Gründen nicht möglich sein wird, für jede einzelne Aufgabe eine Anwendung neu zu entwickeln, werden in diesem Kapitel unter anderem schon bestehende Lösungen und Anwendungen beurteilt und im Hinblick auf die gestellten Anforderungen, die notwendigen Designentscheidungen getroffen. Außerdem werden die Gründe für den Einsatz von bestimmten Technologien, wie der Programmiersprache, kurz dargelegt.

5.1.1 Laufzeitumgebung für den mobilen Client

Damit die Anwendung portierbar und plattformunabhängig ist, wird die mobile Version der Programmiersprache Java (J2ME) als Laufzeitumgebung für den mobilen Client eingesetzt.

Diese Entscheidung ist leicht zu vertreten, da J2ME mittlerweile von jedem Hersteller für mobile Endgeräte unterstützt wird, ein ausgefeiltes Sicherheitskonzept bietet und seine Praxistauglichkeit schon unter Beweis gestellt hat. Darüber hinaus ist es Open-Source und bietet zusätzlich eine große Anzahl von API's, die unter anderem die Implementierung des UI's wesentlich vereinfacht. Desweiteren werden von vielen Herstellern noch weitere optionale Pakete für J2ME bereitgestellt, die zum Beispiel das Arbeiten mit Webservices oder Lokalisierungsdiensten sehr vereinfachen.

5.1.2 Anbinden externer Systeme

5.1.2.1 Karten-Server

Da die mobile Anwendung in der Lage sein soll, Kartenmaterial von externen Karten-Servern zu beziehen und anzeigen zu lassen, werden nachfolgend verschiedene Konzepte betrachtet, die diese Funktionalität ermöglichen. Dabei ist noch zu beachten, dass zusätzliche Informationen wie POI's (z.B. Haltestellen) oder ein Fussweg mit dem Kartenmaterial verknüpft und dargestellt werden müssen. Außerdem wird sehr viel Wert darauf gelegt, dass die Anwendung Schnittstellen zu verschiedenen Karten Providern unterstützt, um die Abhängigkeiten zu reduzieren und die Zuverlässigkeit zu erhöhen.

Die erste Möglichkeit wäre einen zentralen Server in das Konzept mit einzubeziehen, der das Kartenmaterial von den externen Karten-Servern bezieht, zusätzliche Informationen und POI's annotiert und das Ergebnis nur noch an die mobile Anwendung sendet, wo es dann dargestellt wird. Das würde zum einen die mobile Anwendung in Hinsicht auf die Rechenkapazität sehr entlasten und zum anderen würde es die Wartbarkeit und Erweiterbarkeit der gesamten Anwendung fördern, da z.B. Änderungen nur an dem zentralen Server nötig wären. Da es aber für diese Server-basierte Lösung keine ausreichenden Bibliotheken oder Frameworks gibt, die diese Funktionalität unterstützen, müßten entsprechende Entwicklungen und Implementierungen selbst vorgenommen werden. Die notwendigen Entwicklungen sind nachfolgend stichpunktartig aufgelistet:

- Entwicklungen auf dem Server:
  - Schnittstellen zu verschiedenen Karten-Servern, um auf deren Kartenmaterial zugreifen zu können
  - Programmlogik zur Interpretation und Annotation von Kartenmaterial, Routen, POI's, etc.

  - Programmlogik zur Interpretation und Konvertierung zu verschiedenen Formaten für Geodaten (z.B. KML)

- Entwicklungen auf dem mobilen Client:

  - Programmlogik zur Interpretation von Geodaten (z.B. KML)
  - User Interface für die Kartendarstellung
  - Georeferenzieren in der Routen -und Kartendarstellung

Auf Grund des hohen zeitlichen Aufwands ist die skizzierte Alternative, auch wenn architektonisch vorzuziehen, für diese Arbeit leider ungeeignet. Deshalb wird nun ein zweiter Client-basierter Ansatz vorgestellt, der die vorgestellte Funktionalität in bereits fertigen Komponenten integriert. Die Java-basierte Software J2MEMap von Thomas Landspurg [Landspurg] kann auf Kartenmaterial von verschiedenen Karten-Servern zugreifen und auch die benötigten zusätzlichen Informationen annotieren und verfügt somit über die notwendigen Anforderungen. Nachfolgend werden die Vorteile von J2MEMap stichpunktartig erläutert.

J2MEMap

- kann Kartenmaterial von Google, MSN, Ask, Yahoo beziehen und ist für weitere Server erweiterbar
- ist frei verfügbar (für ein SDK ist eine Registrierung nötig)
- unterstützt verschiedene Geodatenformate KML, GPX, LOC
- hat ein graphisches User Interface für Kartendarstellung (Zwischenspeicherung von lokalem Kartenmaterial, was Benutzerfreundlichkeit erhöht) siehe Bild 5.1
- ermöglicht einfache Annotation von POI's, Routen,etc. sowie einfache Darstellung von Positionsdaten (geographische Koordinaten)
- wurde schon in verschiedenen Arbeiten erfolgreich eingesetzt (vgl. [Zohurian (2007)] und [Buchholz (2007)])

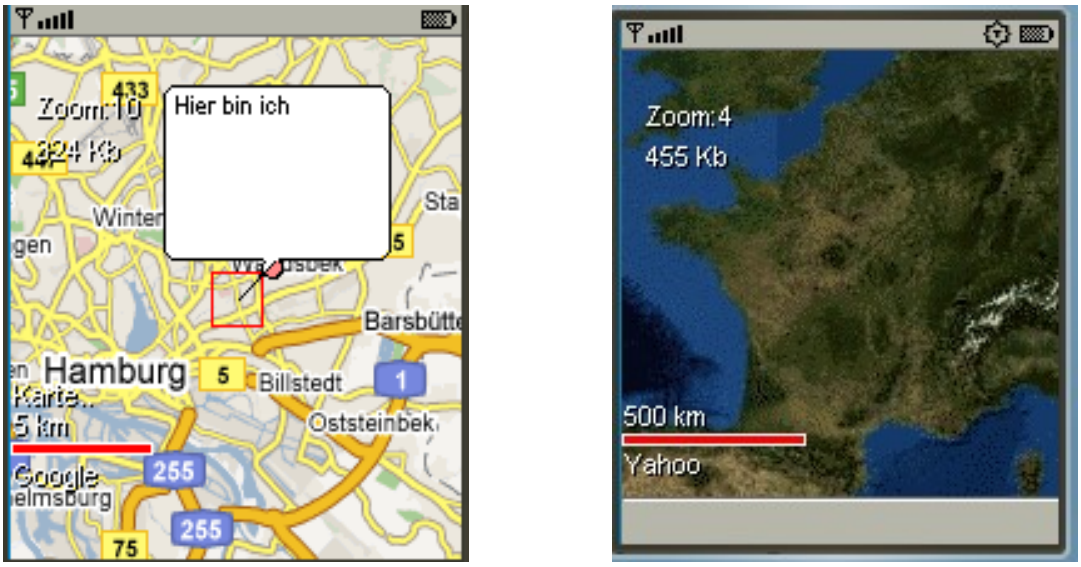


Abbildung 5.1: Kartendarstellung mit J2MEMap

Allerdings hat J2MEMap auch wesentliche Nachteile. So verbraucht die Annotation der zusätzlichen Informationen auf dem mobilen Endgerät mehr Rechenkapazität. Außerdem wird der Speicherverbrauch und der Overhead durch die Tatsache erhöht, dass Kartenmaterial der näheren Umgebung heruntergeladen und zwischengespeichert wird. Wie schon beim Datenschutz in den nichtfuntionalen Anforderungen aus Kapitel 4.3 angesprochen wurde, sind externe Karten-Server aus Datenschutzgründen als sicherheitskritisch einzustufen, weil durch positionsbezogenes Kartenmaterial oder Routeninformationen das mobile Endgerät eindeutig identifizierbar und lokalisierbar ist. Trotz dieser Nachteile wird die Entscheidung zu Gunsten von J2MEMap getroffen, weil die umfangreiche integrierte Funktionalität überzeugend ist.

5.1.2.2 Nutzung des Fahrgastinformationssystems

Die nächste Designentscheidung betrifft die Nutzung eines Fahrgastinformationsdienstes. Dafür wird kurz erläutert, welche Voraussetzungen für eine eigene Implementierung notwendig wären und welche Vor- und Nachteile schon bestehende Lösungen haben.

Wie in der Tabelle 4.1 schon kurz erläutert wurde, ist ein Fahrgastinformationsdienst ein elektronisches Informationssystem für den öffentlichen Personennahverkehr, das es erlaubt, detaillierte und optimierte Fahrpläne zu berechnen und abzufragen.
Um ein Fahrgastinformationssystem selbst zu implementieren, sind umfangreiche und detaillierte Datenbestände über Verkehrslinien, Fahrpläne, Haltestellen, etc. notwendig. Diese

Datenbestände müssen immer auf dem neusten Stand gehalten werden, um Fehler bei der Fahrplanberechnung zu vermeiden. Für die Ermittlung von optimierten Fahrplänen sind spezielle Suchalgorithmen, die mit Hilfe von KI- Methoden und der Graphentheorie entwickelt werden, notwendig. Detailliertere Informationen dazu sind im [Ralf Möller (1991)] zu finden.

Die einzelnen Verkehrslinien und Fahrpläne von den Verkehrsverbünden sind meistens über ihre jeweilige Webseite als PDF Dokument verfügbar. Dadurch wäre es möglich, die benötigten Daten aus den Dokumenten zu extrahieren und sie in einen eigenen Datenbestand zu transferieren. Allerdings wäre es notwendig, die Daten der Verkehrsverbünde regelmäßig mit dem eigenen Datenbestand zu synchronisieren, wenn sich die Fahrpläne ändern. Erschwerend kommt hinzu, dass noch ein Suchalgorithmus entwickelt werden müßte, der die Berechnung von Fahrplänen aus den vorhandenen Datenbeständen ermöglicht.

Auf der anderen Seite stellen die meisten Verkehrsverbünde webbasierte Fahrgastinformationsdienste bereit, die schon über ausgefeilte Algorithmen für die Ermittlung von optimierten Fahrplanverbindungen verfügen. Der Vorteil liegt dabei natürlich darin, dass diese Systeme sich im Einsatz schon bewährt haben und auch Änderungen oder Wartungsarbeiten regelmäßig durchgeführt werden. Der Nachteil ist die Abhängigkeit von dem jeweiligen Fahrgastinformationsdienst und die damit verbundene Möglichkeit, nicht auf die benötigten Informationen zugreifen zu können. Allerdings überwiegen die Vorteile dieser Möglichkeit, weil das Nutzen und Beziehen von korrekten Informationen eines externen Dienstes den nichtfunktionalen Anforderungen aus Kapitel 4.3 entspricht.

Somit lässt sich feststellen, dass eine eigene Implementation eines Fahrgastinformationsdienstes und der dafür benötigte zeitliche Aufwand den Nutzen für diese Arbeit bei weitem übersteigen würde. Deshalb wird die Nutzung eines bereits vorhandenen Fahrgastinformationsdienstes in das Design mit einbezogen. Für den Großraum Hamburg bietet sich der Fahrgastinformationsdienst Geofox an, da dieser auch vom Hamburger Verkehrverbund für die Ermittlung von optimierten Fahrplanverbindungen eingesetzt wird. Detailliertere Informationen sind im Kapitel 5.3.2 zu finden.

5.1.2.3 Einsatz eines zentralen Servers

Eine äußerst wichtige Designentscheidung ist die Frage, ob ein zentraler Server in das Design mit einfließen soll oder ob nur eine mobile standalone Anwendung entwickelt werden soll. Der zentrale Server könnte das mobile Endgerät in vielen Bereichen entlasten. So kann der Server als Adapter zwischen dem mobilen Endgerät und dem Fahrgastinformationsdienst fungieren und ihm die benötigten Informationen in komprimierter Form zukommen

lassen, was die beschränkte Bandbreite und Rechenkapazität des mobilen Endgerätes erheblich entlasten würde. Das Auslagern von rechenintensiver Anwendungslogik auf den zentralen Server würde noch weitere Entlastung mit sich bringen und zusätzlich die Wartbarkeit und Änderbarkeit erhöhen, da die ausgelagerte Funktionalität auf dem zentralen Server wesentlich zugänglicher ist als auf vielen verschiedenen mobilen Endgeräten. Auf Grund dessen sind auch Erweiterungen wesentlich einfacher umzusetzen und in das Gesamtsystem zu integrieren.

Der zentrale Server bietet allerdings auch einen entscheidenen Nachteil, der sich Single Point of Failure nennt. Das bedeutet, dass das Gesamtsystem entscheidend von der Zuverlässigkeit und der Verfügbarkeit dieses Servers abhängig ist und ohne diesen nicht funktionsfähig wäre. Redundante Computertechnologien wie Cluster Server, die im kommerziellen Bereich häufig zum Einsatz kommen, können diesen Nachteil zwar ausgleichen, was allerdings mit erhöhtem Aufwand und Kosten verbunden ist. Ein weiter wichtiger Nachteil ist der Datenschutz und die Sicherheit, da ein zentraler Zugangspunkt von außen wesentlich besser angreifbar ist. Dadurch ist es auch schwieriger die Daten vor versehentlichem Löschen, unberechtigtem Zugriff und Manipulation zu schützen. Allerings gibt es auch dafür Technologien, wie z.B. Backup-und RAID Systeme, eine Firewall, Benutzerauthentifikation und Zugriffskontrollsysteme, die diese Gefahren minimieren können.
Zusammenfassend läßt sich sagen, dass ein zentraler Server, ganz besonders im Hinblick auf zukünftige Erweiterungen, wesentlich mehr Vorteile für das Navigationssystem NavMobile zu bieten hat, als es Einschränkungen verursacht.

5.1.3 Auswahl des Lokalisierungsdienstes

Eine weitere Designentscheidung betrifft die Auswahl eines Lokalisierungsdienstes. Da in Kapitel 2.2.2 schon die verschiedenen Lokalisierungsdienste erläutert wurden, werden sie anschließend nur noch nach den funktionalen und nichtfunktionalen Anforderungen aus Kapitel 4 beurteilt und dementsprechend ein Lokalisierungsdienst ausgewählt.

Die zellbasierte Ortung bietet eine gute Möglichkeit, die eigene Position zu bestimmen. Der Vorteil dabei ist, dass sie so gut wie überall verfügbar ist und keine besonderen Anforderungen an spezieller Hardware mit sich bringt. Der große Nachteil ist aber, dass die Genauigkeit oder Korrektheit der Positionsdaten nicht den gestellten Anforderungen entspricht und somit für ein mobiles Navigationssystem ungeeignet ist.

Das Global Positioning System bietet eine sehr große Genauigkeit, die durch die vielen Positionssatelliten gewährleistet wird. Leider ist dafür spezielle Hardware wie ein integrierter oder externer GPS-Empfäger notwendig. Außerdem wird die Zuverlässigkeit des Systems durch

den Sichtkontakt zu den Positionssatelliten beschränkt. Das bedeutet, dass diese Möglichkeit für die Indoor Navigation, also innerhalb von Gebäuden, U-Bahnen, Bahnhöfen,etc. ungeeignet ist.

Auf Grund der Tatsache, dass das Global Positioning System momentan die einzige kommerziell nutzbare Technologie ist, die präzise und zuverlässige Positionangaben für die Outdoor Navigation liefern kann, wird es als Lokalisierungsdienst für diese Arbeit eingesetzt. In diesem Zusammenhang kann in Erwägung gezogen werden, die zellbasierte Ortung als redundanten Lokalisierungsdienst zu nutzen, um Nachteile des Global Positioning Systems ausgleichen zu können. In dieser Arbeit wird auf diese Möglichkeit aus zeitlichen Gründen allerdings nicht eingegangen.

5.1.4 Technische Schnittstellen des zentralen Servers

Die letzte Designentscheidung bezieht sich auf die Interaktion des mobilen Endgerätes mit dem zentralen Server. Wie schon im Kapitel 2.5 näher erläutert wurde, stellen die spezifischen Eigenschaften von Webservices die optimalen Vorteile einer Internet Kommunikationstechnologie zur Verfügung. Deshalb werden nachfolgend nur noch die verwendeten Standards in Zusammenhang mit den nichtfunktionalen Anforderungen aus Kapitel 4.3 kurz erläutert.

Da Webservices sehr viele offene Standards unterstützt, sind verschiedene Konstellationen möglich. In dieser Arbeit wird SOAP als Netzwerkprotokoll, XML zur Repräsentation und HTTP zur Datenübertragung verwendet. Durch diese Standards wird eine zuverlässige Kommunikation ermöglicht, um heterogene Systeme miteinander zu kombinieren. Das entspricht den gestellten Anforderungen im Hinblick auf die Portabilität und Plattformunabhängigkeit. Da das Protokoll HTTP über den Port 80, welcher standardmäßig für das Laden von Webseiten in Webbrowsern geöffnet ist, Daten überträgt, werden eventuell auftretende Konfigurationsprobleme mit Firewalls oder Routern von vornherein vermieden. Denn bei der Verwendung von anderen Übertragungsprotokollen müssen eventuell spezielle Ports explizit freigegeben werden, um Übertragungsfehler zu vermeiden. Desweiteren werden Webservices durch umfangreiche, frei verfügbare, Java-basierte Bibliotheken (J2ME Webservice API ([JSR172 (2004)]), AXIS) für Client- und Server-basierte Software Anwendungen unterstützt.

Die Nachteile von Webservices betreffen die Sicherheit und die Performance. So sind die Sicherheitsaspekte von SOAP- Verbindungen noch nicht hinreichend gelöst (vgl.[Tobias Hauser (2003)]). Dieses kann allerdings durch vorhandene Sicherheitslösungen wie SAML (Authentifizierung und Autorisierung) erweitert werden (vgl.[Chanliau (2004)]). Eine niedrigere

Performance wird durch die XML-basierten Nachrichten verursacht. Denn durch die XML-Repräsentation werden die Dokumente erheblich größer und somit steigt auch der zu übertragene Overhead. Erschwerend kommt hinzu, dass die XML-Nachrichten client- und serverseitig geparsed und die Informationen extrahiert werden müssen. Aus diesem Grund wird auf die Verwendung von komplexen Objekten bei der Erstellung der Webservices verzichtet, weil diese Objekte für die Datenübertragung aufwendig serialisiert und in XML-Dokumenten gekapselt werden müssen. Das erhöht allerdings noch zusätzlich den Overhead erheblich und wirkt sich negativ auf die zur Verfügung stehende Bandbreite des mobilen Endgerätes aus. Deshalb werden an der Schnittstelle des mobilen Clients sowie der des zentralen Servers die komplexen Objekte in ein flacheres Format (Java Basistypen z.B. `String`) transformiert.

5.2 Systemarchitektur

In diesem Kapitel werden die Anforderungen aus Kapitel 4 und die Designentscheidungen zur Konzeptionierung der Systemarchitektur verwendet. In der Abbildung 5.2 wird die Systemarchitektur dargestellt und daraufhin werden die Zusammenhänge noch detailliert erläutert.

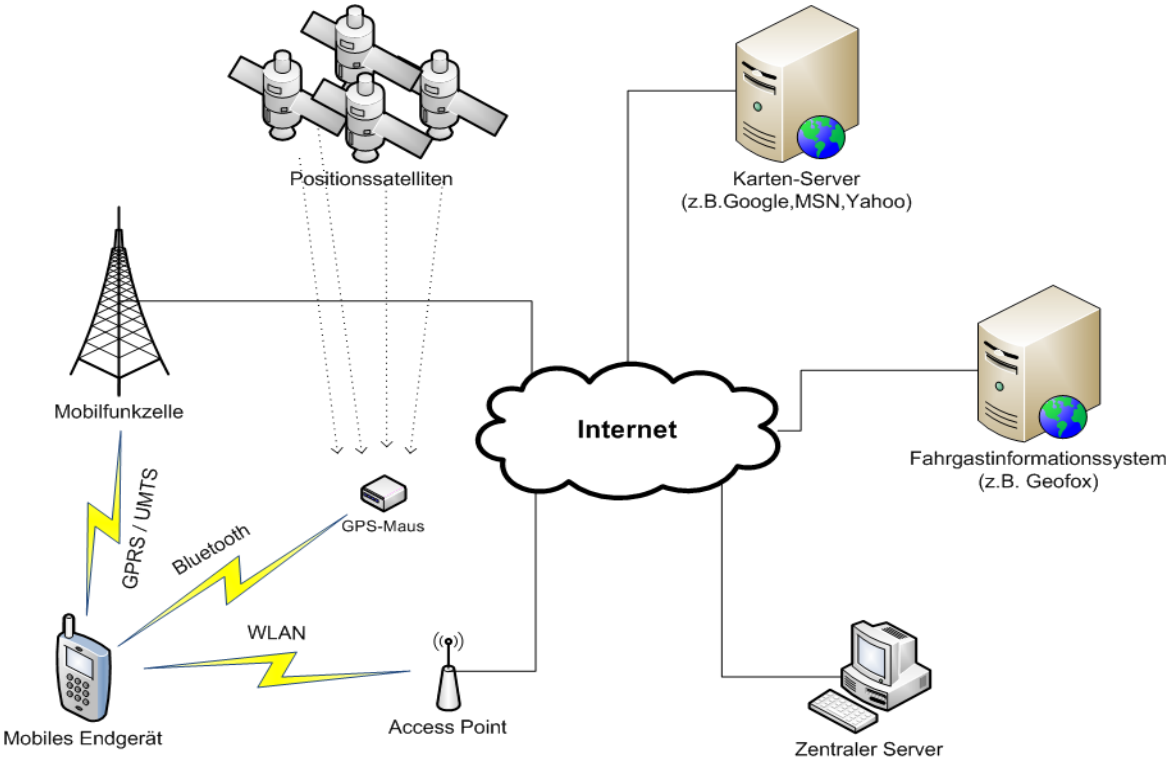


Abbildung 5.2: Systemarchitektur

Die Systemarchitektur besteht aus vielen unabhängigen, vernetzten Komponenten, die unterschiedliche Aufgaben erfüllen bzw. benötigte Ressourcen zur Verfügung stellen und miteinander interagieren. Aus diesem Grund handelt es sich um ein verteiltes System. Das mobile Endgerät dient dem Benutzer dabei als Schnittstelle zur Eingabe von Zieldaten und zur Darstellung der ermittelten Fahrplanverbindungen und Navigationsinformationen. Die anderen Komponenten des verteilten Systems, wie die GPS-Maus, der zentrale Server, die Karten-Server und der Fahrgastinformationsdienst, stellen dabei für den Benutzer transparente Funktionalität zur Verfügung.

Um die Kommunikationsverbindungen mit den beteiligten Komponenten herzustellen, werden verschiedene Funk-Datenübertragungstechniken verwendet, die schon im Kapitel 2.3 näher erläutert wurden. Die Lokalisation des eigenen Standorts wird mit Hilfe einer externen GPS-Maus, welche über eine Bluetooth-Schnittstelle angesprochen wird, bewerkstelligt. Diese GPS-Maus berechnet ständig geographische Koordinaten durch das Auswerten von Signalen verschiedener Positionssatelliten (siehe Kapitel 2.2.2.1). Die benötigte Internetverbindung kann über die mobilen Datenübertragungstechniken GPRS oder UMTS hergestellt werden, welche direkt mit der jeweiligen Mobilfunkzelle des Netzbetreibers kommunizieren und über ein Internet-Gateway weitervermittelt werden. Wenn das mobile Endgerät über eine WLAN-Schnittstelle verfügt, kann die Verbindung auch über eine WLAN-Netz mit dem dazugehörigen Access Point erfolgen. Der Access Point, für den meistens eine Zugangsberechtigung vorhanden sein muss, kann dann die Verbindung über einen Internet-Gateway herstellen.

Das mobile Endgerät benötigt die Internetverbindung, um Funktionen und Ressourcen des zentralen Servers oder des Karten-Servers zu nutzen. Der zentrale Server stellt dabei den Adapter (Adapter-Pattern, vgl. [Erich Gamma (1996)]) zum Fahrgastinformationssystem dar, denn das mobile Endgerät sendet Anfragen zur Ermittlung von optimalen Fahrplanverbindungen nur direkt an den zentralen Server. Auf Grund der Argumentation im Kapitel 5.1, Designentscheidungen, stellt der zentrale Server die Funktionalität als Webservice zur Verfügung. Dieser Webservice verarbeitet automatisiert die Anfrage und leitet sie an den Fahrgastinformationsdienst weiter. Von diesem erhält der zentrale Server dann die benötigten Fahrplanverbindungen, welche in komprimierter Form an das mobile Endgerät gesendet werden. Aus Performace-Gründen werden zusätzlich aufwendige Rechenoperationen auf den zentralen Server ausgelagert und auch als Webservice bereitgestellt.
Das mobile Endgerät kommuniziert außerdem mit Karten-Servern verschiedener Provider, um Kartenmaterial und Routeninformationen abrufen zu können. Diese Informationen werden dem Benutzer dann auf dem Display zur besseren Orientierung dargestellt.

5.3 Softwarearchitektur

Eine Softwarearchitektur ist die grundlegende Organisation eines Softwaresystems, bei welchem das Gesamtsystem in verschiedene Komponenten zerlegt und die Beziehungen zueinander dargestellt werden (vgl. [Oestereich (2004)]). Eine Komponente repräsentiert dabei einen modularen Systemteil, der aus Elementen mit klar definierter Funktionalität besteht. Außerdem kann sie eine eigenständige Anwendung sein und wird über Schnittstellen beschrieben. Eine strukturierte Anordnung der Komponenten und die Definition eines hierarchischen Schichtenmodells sorgt für eine lose Kopplung und unterstützt damit die Separierung von Verantwortlichkeiten und Abhängigkeiten. Nachfolgend werden die Vor- und Nachteile einer Softwarearchitektur stichpunktartig dargestellt.

- Vorteile:
  - duch Zerlegung des Gesamtsystems ist eine bessere Arbeitsteilung und Übersichtlichkeit gegeben
  - langfristige Flexibilität in der Systementwicklung und Projektorganisation
  - Verbesserung der nichtfunktionalen Anforderungen (siehe Kapitel 4.3) Portierbarkeit, Wartbarkeit, Änderbarkeit, Erweiterbarkeit, Performace und Sicherheit
  - vereinfacht die Definition von Klassen und Schnittstellen
- Nachteile:
  - relativ hohe Einarbeitungszeit notwendig
  - eine eingerichtete Softwarearchitektur läßt sich später sehr schwer abändern
  - bei schwierigen Problemsituationen ist ein Konzept für die Softwarearchitektur (Zerlegung des Gesamtsystems in Komponenten) evtl. mit erhöhtem Zeit- oder Ressourcenverbrauch verbunden
  - Problem der Abstraktionsebenen -> bei Software ist es schwierig, Ebenen festzulegen und sich während der Entwicklung daran zu halten

Zu den Anforderungen, die heute an eine Softwarearchitektur gestellt werden, gehört auch die Definition einer mehrschichtigen Architektur. Aus diesem Grund wird eine Zwei-Schichten-Architektur (siehe Abbildung 5.3.1) für das Navigationssystem NavMobile mit einer Präsentationsschicht und einer Schicht für die Anwendungslogik entworfen. Die Präsentationsschicht ist die obere Schicht und dient als Benutzerschnittstelle für Benutzereingaben. Außerdem ist sie für die visuelle Repräsentation der Daten verantwortlich. Die Anwendungslogikschicht nimmt Interaktionen der Präsentationsschicht entgegen, stellt verschiedene Dienste zur Verfügung und ermöglicht die Informations- und Datenverarbeitung. Eine

dritte mögliche Schicht, die ganz unten angeordnet werden würde, umfasst das reine Laden und Speichern von Daten und nennt sich zentrale Datenhaltungsschicht. Da das Navigationssytem NavMobile aber nur die externen Dienstleistungen eines Fahrgastinformationsdienstes und der Karten-Server benutzt, ist keine zentrale Datenhaltung notwendig. Aus diesem Grund entfällt die dritte Schicht der Softwarearchitekur. Wenn mögliche Erweiterungen diese allerdings erforderlich machen, ist das nachträgliche Einfügen der zentralen Datenhaltungsschicht leicht möglich.

Den einzelnen Schichten der Architektur werden jeweils verschiedene Funktionalitäten, Komponenten oder Klassen zugeordnet. Außerdem werden die Abhängigkeiten der beiden Schichten darauf beschränkt, dass nur Module der höheren Schicht die Module der niedrigeren Schicht benutzen dürfen (gemäß Dependency Inversion Principle [Martin (2003)]). Dadurch wird die Komplexität der Abhängigkeiten innerhalb des Systems reduziert und somit eine geringere Kopplung bei gleichzeitig höherer Kohäsion der einzelnen Schichten erreicht. Das macht die Architektur leichter portierbar und zusätzlich ist der Austausch von Schichten ohne weiteres möglich, so lange die Schnittstellen nicht verändert werden.

5.3.1 Zwei-Schichten-Architektur-Modell

Nachfolgend wird das Modell der Zwei-Schichten-Architektur in der Abbildung 5.3 dargestellt und kurz erläutert. Eine detailliertere Beschreibung der einzelnen Komponenten sowie ihrer Aufgaben, Beziehungen und Zugehörigkeit zu den einzelnen Prozessen des verteilten Systems folgt in Kapitel 5.3.3.

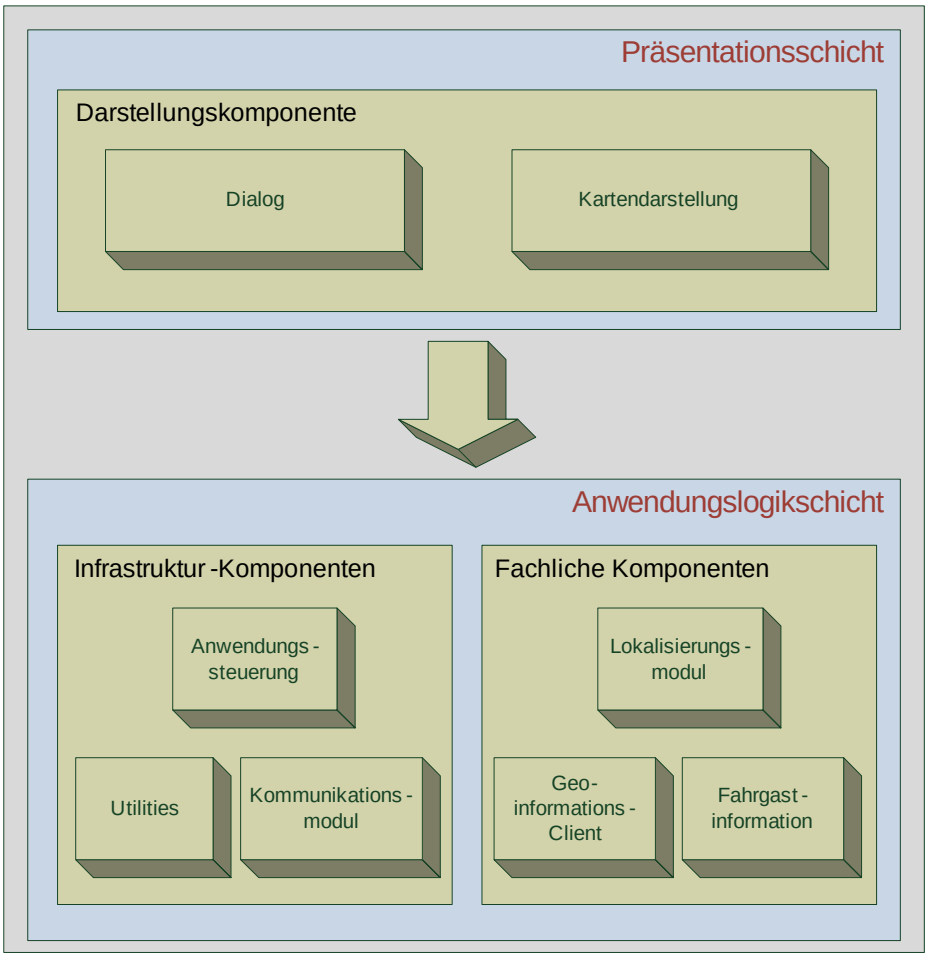


Abbildung 5.3: Zwei-Schichten-Architektur-Modell

Die Zwei-Schichten-Architektur besteht aus einer Präsentationsschicht und einer Anwendungslogikschicht. Die Zugehörigkeit der verschiedenen Komponenten zur jeweiligen Abstraktionsschicht spielt dabei eine entscheidene Rolle, weil nur Komponenten der oberen Schicht auf die einer unteren Schicht zugreifen dürfen. Während die Darstellungskomponente architektonisch zur Präsentationsschicht gehört, werden der Anwendungslogikschicht die Infrastruktur- und fachlichen Komponenten zugeordnet. Die Aufteilung der Anwendungslogikschicht in Infrastruktur- und fachliche Komponenten schafft eine zusätzliche Abstraktion der Funktionalität, was die Übersichtlichkeit erhöht und bei der Implementierung die Definition von Klassen und Schnittstellen einfacher gestaltet.

Die Darstellungskomponente besteht aus den Modulen Dialog und Kartendarstellung. Der Dialog bildet die Schnittstelle zum Benutzer, nimmt Eingaben entgegen und sorgt für die Präsentation von Informationen. Das Modul Kartendarstellung ist für das Aufbereiten und

Darstellen des Kartenmaterials sowie für die Annotation der Routen und zusätzlichen Informationen, wie POI's zuständig.

Die Infrastruktur-Komponenten der Anwendungslogikschicht stellen hauptsächlich Basisfunktionalitäten zur Verfügung und sind für die Koordination und Steuerung des Gesamtsystems verantwortlich. Zu diesen Komponenten gehören die Anwendungssteuerung, Utilities und das Kommunikationsmodul. Die Anwendungssteuerung dient dabei als Controller. Es nimmt Ereignisse und Daten(-objekte) entgegen und leitet sie weiter an die richtigen fachlichen Klassen. Das Modul Utilities besteht aus sogenannten Hilfsklassen, die Teilfunktionalitäten zur Verfügung stellen. Konkret bedeutet das, dass dieses Modul keine fachliche Funktionalität anbietet, sondern die fachlichen Klassen bei der Umsetzung ihrer jeweiligen Aufgaben unterstützt. Das Kommunikationsmodul beinhaltet die Schnittstellen und Bibliotheken, welche für die Nutzung von Webservices benötigt werden.

Die fachlichen Komponenten sind ebenfalls Teil der Anwendungslogikschicht und sind vorwiegend für die Bereitstellung von Diensten für das Gesamtsystem zuständig. Sie bestehen aus dem Lokalisierungsmodul, dem Geoinformations-Client sowie der Fahrgastinformation. Das Lokalisierungsmodul ist dafür verantwortlich, dass die Positionsdaten des aktuellen Standorts mit Hife einer externen GPS-Maus ermittelt und bereitgestellt werden. Der Geoinformations-Client nutzt die Dienste von externen Karten-Servern und stellt das benötigte Kartenmaterial und Routeninformationen zur Verfügung. Das Modul Fahrgastinformation besteht aus der fachlichen Anwendungslogik zur Ermittlung von optimierten Fahrplanverbindungen und den dafür notwendigen Schnittstellen zum Fahrgastinformationsdienst Geofox. Da es verschiedene Ansätze zur Realisierung dieser Schnittstellen gibt, wird im folgenden Kapitel 5.3.2 noch detaillierter darauf eingegangen.

5.3.2 Schnittstellen zum Fahrgastinformationsdienst

In diesem Kapitel werden drei verschiedene Möglichkeiten vorgestellt, die als Schnittstellen zum Fahrgastinformationsdienst fungieren und von denen zwei implementiert und evaluiert wurden. Alle Möglichkeiten beziehen sich dabei auf das bereits vorhandene und etablierte Fahrgastinfomationssystem Geofox der HBT GmbH, welches auch vom Hamburger Verkehrsverbund seit 1993 eingesetzt wird.

Das Geofox Webservice Interface (auch Geofox Soap Interface) ist dabei die effektivste und zuverlässigste Möglichkeit. Da diese aber zu Beginn der Arbeit noch nicht verfügbar war, wurden mögliche Alternativen auf ihre Verwendbarkeit überprüft. Zu diesen Alternativen gehören der SMS-Fahrplan und der Geofox Webclient. Alle drei Möglichkeiten werden nachfolgend im Detail vorgestellt und erläutert.

5.3.2.1 SMS-Fahrplan

Die erste Möglichkeit verfolgt die Nutzung des SMS-Fahrplan Dienstes, der vom Hamburger Verkehrverbund (HVV) in Zusammenarbeit mit Geofox zur Verfügung gestellt wurde. Dieser Dienst erlaubt es, eine optimierte Fahrplanverbindung direkt vom mobilen Engerät per SMS anzufordern. Dafür sind die Nummern von vier Netzbetreibern (T-Online,Vodafone,E-Plus,O2) in der Rubrik SMS-Fahrplan auf der HVV-Webseite[16] vermerkt und können zum Netztarif des jeweiligen Mobilfunkanbieters angewählt werden. Die Antwort per SMS des Hamburger Verkehrsverbundes ist in diesem Fall kostenlos, allerdings muss beachtet werden, dass Besonderheiten bei der Eingabe der Start- und Zieldaten berücksichtigt werden müssen. Diese Besonderheiten sind auf der HVV-Webseite (SMS-Fahrplan) anschaulich erklärt. Nachfolgend wird ein Beispiel einer SMS-Anfrage und eines Ergebnisses dargestellt:

SMS-Anfrage: berliner tor 7!jungfernstieg!an1500 (von Berliner Tor 7 zum Jungfernstieg,Ankunft um 15 Uhr)

SMS-Ergebnis:13.01.2009#Ab:14:53#Lohmühlenstraße#U1#An:14:59#Jungfernstieg (am 13.01.2009,Abfahrt um 14:53 Uhr von der Haltestelle Lohmühlenstraße mit der U-Bahn U1, Ankunft um 14:59 Uhr an der Haltestelle Jungfernstieg)

Wie man an Hand dieses Beispiels sehen kann, sind die SMS-Anfrage und das Ergebnis relativ einfach zu interpretieren und somit könnte das Ergebnis extrahiert und weiterverwendet werden. Damit diese Funktion in die J2ME-Anwendung des mobilen Endgerätes integriert werden kann, muss das optionale Paket JSR-205 Wireless Messaging API 2.0 vorinstalliert sein. Allerdings wurde bei der Überprüfung der Spezifikationen (vgl.[JSR205 (2004)] und [Ortiz (2002)]) folgende Schwierigkeiten festgestellt:

Bewertung

Das Senden der SMS-Anfrage aus der Anwendung heraus ist ohne weiteres möglich aber das Empfangen des Ergebnisses kann nur von den Basisfunktionen des mobilen Endgerätes erfolgen und somit landet die Ergebnis-SMS im Standard Post(SMS)-Eingang. Das hat den Grund, dass die SMS nur über speziell vergebene Ports verschickt und emfangen werden können. In der J2ME-Anwendung kann der Standard-Port für SMS, über den das Ergebnis eintrifft, allerdings nicht belegt werden und somit kann das Ergebnis innerhalb der Anwendung nicht empfangen werden und es besteht auch keine andere Möglichkeit auf diese SMS zuzugreifen. Aus diesem Grund ist der SMS-Fahrplan als Schnittstelle zum Geofox Fahrgastinformationsdienst völlig ungeeignet wird als möglicher Ansatz nicht weiter verfolgt.

[16] http://www.hvv.de/fahrplaene-strecken/sms-fahrplan/

5.3.2.2 Geofox Webclient

Die zweite Alternative besteht darin einen HTTP-Client zu implementieren, der auf die bestehende Webapplikation des Geofox Fahrgastinformationssystems[17] zugreifen kann. Dafür ist es erforderlich, die Formulardaten automatisiert einzugeben und somit einen HTTP-Request-Response-Zyklus zu simulieren und die entsprechende Ergebnis-HTML-Seite zu parsen und die benötigten Informationen zu extrahieren.

In diesem Abschnitt wird die Vorgehensweise bei der Problemlösung dieser Alternative in einem fachlichen Rahmen aufgezeigt, ohne auf spezielle Implementierungsdetails wie Klassen, Objekte und Methoden einzugehen. Da der technische Aufwand und die benötigte Rechenkapazität im Bezug auf ein mobiles Endgerät relativ hoch ist, wird die Java-basierte Implementation auf dem zentralen Server vorgenommen. Ein weiterer Grund ist der Wartungsaufwand und die Änderbarkeit, die bei einer mobilen Anwendung wesentlich höher ausfallen würde.

Um sich über die Geofox Webapplikation optimierte Fahrplanverbindungen berechnen zu lassen, kann man die grundlegenden Eingabedaten in einer URL als Attribute mit angeben. In der Abbildung 5.4 wird ein Beispiel angezeigt, in der der Start- und Zielort fett markiert sind. Diese URL kann dann mit Hilfe von vorhandenen Komponenten der Java Network API wie `HttpURLConnection` automatisiert abgeschickt werden.

```
http://www.geofox.de/base/showPersonalSchedule.do?search=search&
INIT=false&startType=H&startName=lohmühlenstraße&destType=H&
destName=mönckebergstraße&destSpirit=false&departureOrArrival=1&
startTime=22%3A10&startDate=20.12.2008&search=Suche+starten&
numberOfRoutes=0&LANGUAGE=de_DE&LANGUAGE=de_DE&
useStationPosition=true
```

Abbildung 5.4: Beispiel- URL einer automatisierten Anfrage

Das Ergebnis der Anfrage wird dann als HTML- Code zurückgegeben und muss geparsed werden, um die benötigten Informationen zu extrahieren. Ein Ausschnitt ist in Bild 5.5 zu sehen.

[17] www.geofox.de/base

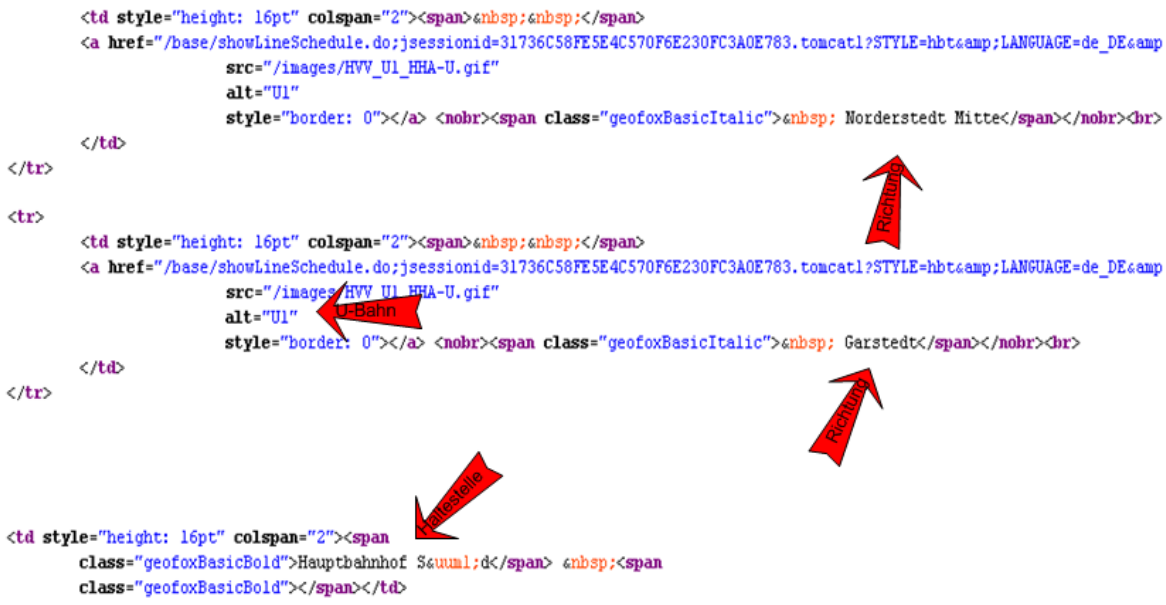


Abbildung 5.5: Ergebnis HTML- Code (Ausschnitt)

Wie man in diesem Ausschnitt sehen kann, ist es sehr aufwendig die benötigten Daten zu extrahieren und richtig zuzuordnen, vor allem wenn man die Tatsache berücksichtigt, dass das Ergebnis im Normalfall aus etwa 600 bis 800 Zeilen HTML- Code besteht. Um die benötigten Informationen besser interpretieren zu können, wird als nächstes nur der darzustellende Text aus dem HTML- Code extrahiert und alle übrigen Elemente, wie Metainformationen, Tags und Hyperlinks, entfernt. Dafür wird die frei verfügbare Bibliothek HTMLParser 1.4 [Oswald (2006)] verwendet, die bei der groben Filterung der überflüssigen HTML-Elemente behilfreich ist. Als letztes wird eine dynamische Anwendungslogik entwickelt, die den herausgefilterten Text korrekt interpretieren und den Attributen (Ankufts- und Abfahrtszeiten, Verkehrsmittel, Haltestellen, etc.) richtig zuordnen kann. Somit können optimierte Fahrplanverbindungen über diese selbstimplementierte Schnittstelle zum Geofox Fahrgastinformationsdienst ermittelt werden.

Allerdings bringt diese Schnittstelle nicht unerhebliche Schwierigkeiten mit sich, so dass bei der kleinsten Änderung der Webapplikation die Anwendungslogik überprüft und gegebenfalls angepasst werden muss. Ausnahmesituationen wie z.B. einer Mehrdeutigkeit der Start- oder Zielhaltestelle würden die Komplexität dazu noch um einiges erhöhen. Aus zeitlichen Gründen wird diese Situation nicht implementiert, sondern nur durch eine Fehlermeldung darauf aufmerksam gemacht. Ein weiterer wichtiger Punkt ist, dass das Ergebnis hauptsächlich von der dynamischen Darstellung der Ergebnis- Webseite (HTML- Code) abhängig ist. Da aber nicht davon ausgegangen werden kann, dass alle möglichen Darstellungen bekannt

sind, kann die Lieferung von zuverlässigen Fahrplaninformationen nicht garantiert werden. Das bedeutet, dass diese alternative Schnittstelle den nichtfunktionalen Anforderungen auf Zuverlässigkeit und Korrektheit nicht entspricht.

Als letztes wird auf rechtliche Bestimmungen hinsichtlich der Benutzungshinweise eingegangen, die auf der Geofox Webseite[18] zu finden sind. Darin heißt es „Die Inhalte dieser Web-Seiten dürfen ohne die vorherige schriftliche Zustimmung der HBT GmbH nicht verändert und nicht für öffentliche oder kommerzielle Zwecke vervielfältigt, vorgeführt, verbreitet oder anderweitig verwertet werden. Insbesondere sind automatisierte Zugriffe mit dem Ziel, systematisch große Mengen der Fahrplan-Inhalte herauszuziehen, nicht gestattet. Ausnahmen, z.B. zu Forschungszwecken, sind vorab mit HBT GmbH abzustimmen, um Störungen des Auskunftbetriebes zu vermeiden."

Bewertung

Wie in diesem Kapitel deutlich wurde, bringt das Parsen der Geofox Webseite einen sehr hohen Aufwand mit sich und bietet auch nicht die gewünschte Zuverlässigkeit. Dazu kommt, dass auf Grund der Benutzungshinweise und der damit verbundenen Rahmenbedingungen, diese Möglichkeit der Fahrplanermittlung nur zu Forschungszwecken eingesetzt werden darf und eine Erlaubnis der HBT GmbH eingeholt werden müßte. Auf Grund der Tatsache, dass sich kurz nach erfolgreichen Tests dieser Schnittstelle eine weitere Möglichkeit (Geofox Webservice Interface, Kapitel 5.3.2.3) ergeben hat, die wesentlich erfolgversprechender war, wird aus den genannten Gründen von der Weiterentwicklung und dem Einsatz der Geofox Webclient Schnittstelle abgesehen.

5.3.2.3 Geofox Webservice Interface

Das Geofox Webservice Interface stellt verschiedene Dienste als Webservices zur Ermittlung von optimierten Fahrplanverbindungen zur Verfügung. Die Kommunikation mit diesen Diensten erfolgt dabei über das standartisierte Netzwerkprotokoll SOAP über HTTP. Durch die klar definierten Methoden und Strukturen des Webservice Interfaces, kann der Aufwand für die Ermittlung von optimierten Fahrplanverbindungen gering gehalten werden und darüber hinaus können auch alle Randbedingungen wie z.B. „Zwischenhaltestelle" oder „nur Bus" auf einfache Art und Weise realisiert werden. Eine regelmäßige Aktualisierung und Wartung des Geofox Fahrgastinformationsdienstes sorgt dafür, dass er auch bei Änderungen z.B. der Linienfahrpläne, immer auf dem neusten Stand gehalten wird.

[18] www.geofox.de/base/ Stand 20.04.2009

Der Hamburger Verkehrsverbund ist Lizenznehmer des Geofox Fahrgastinformationsdienstes und nutzt das Geofox Webservice Interface schon seit Jahren für die eigene Webseite[19]. Dort wird unter anderem das am häufigsten genutzte Produkt von Geofox angeboten, der persönliche Fahrplan. Dieser liefert dort jeden Monat über 3 Millionen Verbindungsempfehlungen. Auf Grund des hohen Bedienkomforts, der Zuverlässigkeit und Verfügbarkeit (vgl.GeofoxServer (2008)und HBT (2008)) , die das Geofox Webservice Interface schon seit Jahren unter Beweis stellt, ist es für das Navigationssystem NavMobile die optimale Schnittstelle zum Geofox Fahrgastinformationssystem.

Allerdings ist zu beachten, dass diese Schnittstelle nicht allgemein zugänglich ist. Auf Anfrage bei Vertretern des Hamburger Verkehrverbundes wurde einer Nutzung des Geofox Webservice Interface im Rahmen dieser Bachelorarbeit unter folgender Einschränkung zugestimmt. Der Zugang zu dieser Schnittstelle kann nur von einer festen IP-Adresse, die zuvor serverseitig freigegeben wurde, erfolgen. Da aber in der Systemarchitektur ohnehin ein zentraler Server vorgesehen war, stellt diese Einschränkung kein Problem dar.

Anwendung

Damit diese Schnittstellen genutzt werden können, werden Webservice Client-Stubs benötigt. Für deren Generierung bietet sich das Webservice Framework Apache Axis 1.4 (siehe Kapitel 2.5.1) für Java an. Dieses kann aus den Spezifikationen des Webservices, welche als WSDL Dokument beschrieben sind, automatisch die notwendigen Client-Klassen generieren. Mit Hilfe dieser Klassen und den zur Verfügung gestellten einführenden Dokumenten (vgl. [Weik (2006)] und [Weik (2007)]) kann auf die notwendigen Funktionen zugegriffen und optimierte Fahrpläne ermittelt werden.

[19] www.hvv.de

5.3.3 Verteilung der Komponenten auf die Systemarchitektur

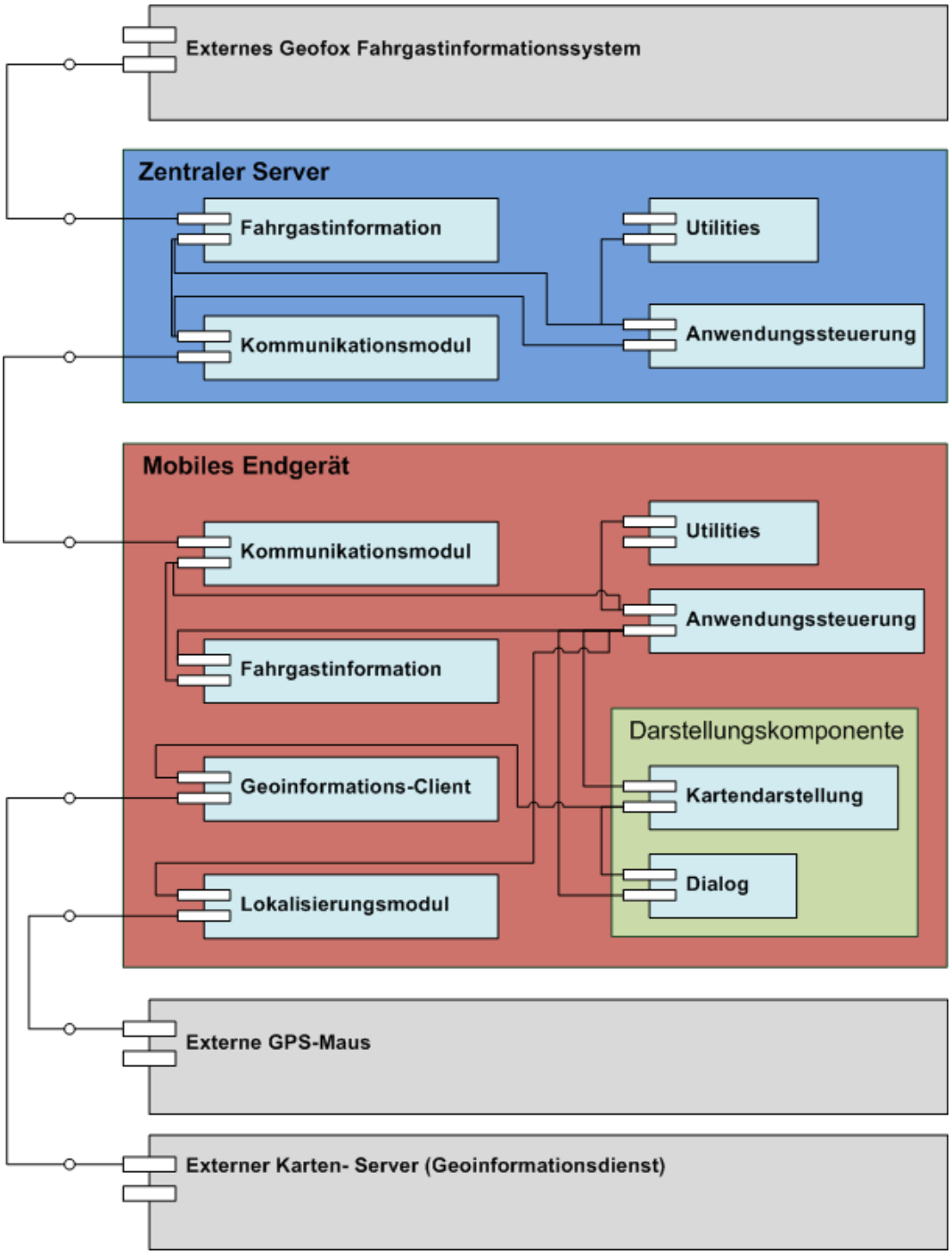


Abbildung 5.6: Verteilung der Komponenten auf das System

Wie in der Abbildung 5.6 zu sehen ist, lässt sich aus den Designentscheidungen, der Systemarchitektur und der Softwarearchitektur eine Verteilung der Komponenten auf das Gesamtsystem zusammenfügen. Genauer gesagt, werden die einzelnen Komponenten der Softwarearchitektur auf die einzelnen Elemente der Systemarchitektur, unter Berücksichtigung der Designentscheidungen, verteilt. Die speziellen Aufgaben, Beziehungen und Zuständigkeiten der einzelnen Komponenten werden nachfolgend noch detaillierter erläutert.

Dialog

Der Dialog ist eine Darstellungskomponente und wird dem mobilen Endgerät zugeordnet. Es stellt die Schnittstelle zum Benutzer dar und steuert die gesamte Kommunikation mit dem System (vgl.[Oestereich (2004)]). Es sorgt für eine angemessene Darstellung und Formatierung der Informationen. Der Dialog besteht unter anderem aus verschiedenen graphischen Elementen wie Eingabefelder, Schaltflächen und mehreren Fenstern, in denen der Dialog geführt wird. Für diese graphischen Elemente wird die High- und Low-Level API von J2ME verwendet, welche dafür unter anderem die Klassen `Form`, `TextBox` und `List` bereitstellt.

Der Dialog verfügt allerdings über kein graphisches User Interface, das Kartenmaterial darstellen kann. Aus diesem Grund arbeitet der Dialog eng mit dem Modul Kartendarstellung zusammen. Direkte Benutzereingaben über den Dialog, wie die Navigation durch verschiedene Eingabefelder eines Formulars, werden unmittelbar verarbeitet. Für diese Benutzerinteraktionen stellt J2ME zum Beispiel die Klasse `Command` mit dem zu implementierenden Interface `CommandListener` zur Verfügung. Mit `Command`-Objekten kann die gewünschte Benutzerinteraktion definiert werden, während der `CommandListener` für die Entgegennahme und Verarbeitung dieser Interaktionen zuständig ist. Andere Eingaben, die nicht direkt verarbeitet werden können, werden an die Anwendungssteuerung weitergeleitet.

Kartendarstellung

Das Modul Kartendarstellung ist ebenfalls eine Darstellungskomponente und dem mobilen Endgerät zuzuordnen. Der Aufgabenbereich dieses Moduls besteht aus dem Darstellen des Kartenmaterials sowie der Annotation der Routen und zusätzlichen Informationen, wie POI's. Zur visuellen Aufbereitung der darzustellenden Informationen wird ein graphisches User Interface, welches für diese Anforderungen konzeptioniert wurde, verwendet. Dieses wird von der J2MEMap API zur Verfügung gestellt, welches unter anderem die Klassen `MapDisplay`, `Marker` und `Track` beinhaltet. Die Klasse `MapDisplay` ist dabei für die Darstellung des Kartenmaterials verantwortlich. Die Klassen `Marker` und `Track` werden für die Annotation von Routen und POI's benötigt.

Um die korrekte Funktionalität dieses Moduls zu gewährleisten, werden verschiedene Komponenten der Anwendungslogikschicht genutzt. Wenn das Display Kartenmaterial anzeigen soll, wird dieses von der fachlichen Komponente Geoinformations-Client angefordert und nachgeladen. Bei der Darstellung von zusätzliche Informationen, POI's oder Routeninformationen, interagiert dieses Modul mit der Infrastruktur-Komponente Anwendungssteuerung, welche dann für die Bereitstellung der benötigten Daten verantwortlich ist.

Anwendungssteuerung

Das Modul Anwendungssteuerung dient auf dem mobilen Endgerät sowie dem zentralen Server als Controller bzw. Koordinator. Es hat die Aufgabe, Ereignisse und Daten entgegen zu nehmen, auszuwerten und sie an die richtigen fachlichen Klassen weiter zu leiten. Die Manipulation von Daten ist dabei nicht Aufgabe der Anwendungssteuerung.

Auf dem mobilen Endgerät können Ereignisse oder Daten allerdings auch von den Darstellungskomponenten Dialog oder Kartendarstellung der Präsentationsschicht kommen, weil die Darstellungskomponenten für die Bereitstellung ihrer Funktionalität auf andere Komponenten angewiesen sind (z.B. Position anzeigen->Lokalisierungsmodul). Die Kommunikation der Anwendungssteuerung auf dem zentralen Server erfolgt nur innerhalb der Anwendungslogikschicht, da keine Darstellungskomponenten vorhanden sind.

Darüber hinaus kann dieses Modul auf dem mobilen Endgerät als Anwendungsfallsteuerung fungieren. Dabei ist es für die Kontrolle und Koordination der Interaktionen und Kommunikationen zwischen den fachlichen Klassen und Komponenten, entsprechend der im Anwendungsfall festgelegten Möglichkeiten, verantwortlich. Das sorgt bei den entsprechenden Klassen für eine Reduzierung der Abhängigkeiten und eine hohe Kohäsion.

Kommunikationsmodul

Das Kommunikationsmodul stellt die notwendigen Schnittstellen und Bibliotheken (J2ME Webservice API (JSR-172), AXIS 1.4) zur Nutzung von Webservices zur Verfügung. Dafür werden auf dem zentralen Server die Server-Skeletons erstellt, um einen Zugang zu den jeweiligen Webservices mit der jeweiligen Funktionalität zu gewährleisten. Auf dem mobilen Endgerät werden die für Webservices notwendigen Client-Stubs erstellt, die dann eine Verbindung zu den Server-Skeletons des zentralen Servers herstellen und deren bereitgestellte Funktionen nutzen können. Die Kommunikation erfolgt mit den standardisierten Protokollen SOAP über HTTP, bei welchen die Daten im XML-Format repräsentiert werden.

Lokalisierungsmodul

Dieses Modul ist auf dem mobilen Endgerät dafür verantwortlich, die Positionsdaten des aktuellen Standorts bereit zu stellen. Die Positionsdaten werden zum Beispiel der Kartendarstellung indirekt (über die Anwendungssteuerung) zur Verfügung gestellt, um den eigenen Standort auf der Karte anzuzeigen. Dafür ruft das Lokalisierungsmodul die Daten, die intern an Hand von verschiedenen Satellitensignalen berechnet werden, der externen GPS-Maus in regelmäßigen Abständen ab. Diese Daten enthalten unter anderem die benötigten geographischen Koordinaten nach WGS84 (siehe Kapitel 2.2.1.1). Darüber hinaus ist dieses Modul für die Interpretation und Weiterleitung der Statusmeldungen der GPS-Maus (verfügbar, kein Satellitensignal oder außer Betrieb) zuständig.

Damit das Lokalisierungsmodul eine Datenverbindung zur externen GPS-Maus herstellen kann, wird das Location API (JSR-179) eingesetzt, welches als optionales Paket für J2ME bereitgestellt wird. Die Kommunikation mit der externen GPS-Maus wird dadurch sehr vereinfacht, weil es die spezifischen Funktionen des Lokalisierungsverfahrens kapselt und durch Schnittstellen zur Verfügung stellt (vgl.[JSR179 (2003)]). Um die Kriterien für die Auswahl einer Methode zur Positionsbestimmung zu setzen, wird zum Beispiel die Klasse `Criteria` verwendet. Mit dieser lassen sich Anforderungen, wie maximaler Stromverbrauch oder Positionsgenauigkeit definieren. Mit der Klasse `LocationProvider` kann man hingegen die Verbindung zur externen GPS-Maus herstellen und dafür einen festen Timeout festlegen, nachdem der Verbindungsversuch abgebrochen wird.

Geoinformations-Client

Dieses Modul ist dem mobilen Endgerät zugeordnet und enthält die Schnittstellen zu den verschiedenen Karten-Servern, wie zum Beispiel Google, MSN, Yahoo oder Ask. Dadurch wird zum Beispiel Kartenmaterial, wenn es von dem Modul Kartendarstellung angefordert wird, automatisiert von dem jeweiligen Karten-Server nachgeladen. Diese Funktionalität ist in der verwendeten J2MEMap API vollständig und transparent gekapselt.

Fahrgastinformation

Die Fahrgastinformation ist ein Modul des mobilen Endgerätes und des zentralen Servers. Trotz dieser Gemeinsamkeit, kann die zu erfüllende Funktionalität allerdings deutlich voneinander abgegrenzt werden.

Das Modul Fahrgastinformation auf dem mobilen Endgerät ist dafür verantwortlich, aus den Eingabedaten des Dialogs und den Positionsdaten des Lokalisierungsmoduls eine Fahrplananfrage zu formulieren. Dafür werden nur Java Basisklassen verwendet, damit der Payload

für die Kommunikation mit dem zentralen Server so weit wie möglich reduziert wird. Die Kommunikation mit dem Webservice des zentralen Servers erfolgt über den Client-Stub.
Ebenso muss das Modul Fahrgastinformation den komprimierten Ergebnis-Request des zentralen Servers in ein einfaches Objektmodell zur Repräsentation von Fahrplanverbindungen transformieren. Dafür werden die Klassen `Schedule` und `Trip` bereitgestellt.

Auf dem zentralen Server hat das Modul Fahrgastinformation die Aufgabe, die Fahrplananfragen des mobilen Endgerätes zu verarbeiten. Dafür werden sie von den Server-Skeletons des Kommunikationsmoduls entgegengenommen und an das Modul Fahrgastinformation weitergeleitet. Die einfachen Repräsentationen von Fahrplananfragen werden dann in eine komplexe Objektstruktur der Geofox-Schnittstelle transformiert (z.B. `GRRequestLight` oder `StartDestLight`). Das ist nötig, weil dieses Modul gleichzeitig der Geofox-Client für den Fahrgastinformationsdienst ist und somit alle notwendigen Schnittstellen und fachlichen Klassen beinhaltet, die für die Kommunikation mit diesem Dienst benötigt werden.
Nach der erfolgreichen Ermittlung von optimierten Fahrplanverbindungen werden die verwendeten komplexen Strukturen der Geofox-Schnittstelle in einen einfachen komprimierten Ergebnis-Request transformiert und unter Nutzung der Server-Skeletons des Kommunikationsmoduls an das mobile Endgerät geschickt.

Eine zusätzliche Aufgabe hat das Modul Fahrgastinformation auf dem zentralen Server. Denn das Geofox Fahrgastinformationssystem stellt die Haltestellenpositionen in Gauss-Krüger-Koordinaten (siehe Kapitel [2.2.1.2]) bereit, während die J2MEMap API zur Darstellung dieser Haltestellenpositionen die geographischen Koordinaten nach WGS84 benötigt. Aus diesem Grund müssen die Gauss-Krüger-Koordinaten zuerst in geographische Koordinaten umgewandelt werden, bevor sie dem Ergebnis-Request hinzugefügt und dem mobilen Endgerät geschickt werden können.
Ebenso müssen die geographischen Koordinaten, die in der Fahrplananfrage enthalten sind und den eigenen Standort markieren, erst in Gauss-Krüger-Koordinaten umgewandelt werden, weil der Geofox Fahrgastinformationsdienst als Startpunkt für die Verbindungsanfrage nur Koordinaten in diesem Format akzeptiert.
Für diese Transformation der entsprechenden Koordinaten werden Funktionen des Utilities-Moduls in Anspruch genommen.

Utilities

In diesem Modul werden auf dem mobilen Endgerät sowie auf dem zentralen Server Teilfunktionalitäten durch Hilfsklassen zur Verfügung gestellt. So beinhaltet dieses Modul auf dem mobilen Endgerät zum Beispiel die Klasse `StringTokenizer`, welche zwar standardmäßig in der Java Bibliothek aber nicht in J2ME enthalten ist. Auf dem zentralen Server

enthält es unter anderem die Klasse `CoordTransformation`, welche Gauss-Krüger-Koordinaten in geographische Koordinaten und umgekehrt transformiert.

5.4 Realisierung der Anwendungssoftware

Dieses Kapitel beschreibt die Realisierung des Navigationssystems NavMobile. Auf Grund der Komplexität der Anwendung wird dabei nur auf die wichtigsten Implementierungsdetails eingegangen. Nachfolgend werden zuerst die technischen Voraussetzungen erläutert.

5.4.1 Technische Voraussetzungen

In diesem Kapitel wird die Hardware und Software vorgestellt, die für die Realisierung der Anwendungssoftware verwendet wird.

5.4.1.1 Hardware

- Mobiltelefon Nokia N93 (Bild 5.7 links) mit folgenden Eigenschaften:
  - Datenübertragung über GPRS/UMTS/WLAN möglich (Mobilfunkkarte für UMTS und ohne Volumenbegrenzung wird von der HAW-Hamburg zu Testzwecken zur Verfügung gestellt)
  - Bluetooth Funktechnik (kann auf externem GPS-Empfänger zugreifen)
  - unterstützt javafähige Anwendungen (J2ME CLDC Version 1.1, MIDP 2.0)
  - bereits vorhandene optionale Pakete JSR-82 (Bluetooth API), JSR-172 (Web Service API), JSR-179 (Location API)
- Holux GPSlim236 (Bild 5.7 rechts)
  - externer GPS Empfänger (GPS-Maus) mit Bluetooth Schnittstelle
  - Akku mit mindestens 10 Stunden Betriebszeit
  - unterstützt NMEA 0183 Protokoll für Lieferung von geographischen Koordinaten nach WGS84
- stationärer PC der HAW Hamburg als zentraler Server

  - AMD Athlon XP2600+ 2.08 Ghz
  - Festplattenspeicher 80 GB, 1GB RAM

  Dieser Server stellt den Adapter (Adapter-Pattern, vgl. [Erich Gamma (1996)]) zwischen dem mobilen Endgerät und dem Fahrgastinformationsdienst Geofox dar. Darüber hinaus hat er eine feste IP-Adresse und ist über das Internet erreichbar. Diese IP-Adresse wurde den Vertretern des Hamburger Verkehrsverbund mitgeteilt, um den Fahrgastinformationsdienst Geofox von diesem stätionären PC aus nutzen zu können. Mit diesem PC der HAW Hamburg wird auch die Implementation der Server-Anwendung umgesetzt.

- privates Notebook, Toshiba Satellit M 100 - 152
  - Intel Core Duo T2300 / 1.66 GHz (Dual-Core)
  - Festplattenspeicher 80 GB, 1 GB RAM
  - 14.1 Zoll TFT Display (TruBrite)
  - wird hauptsächlich für die Implementation der mobilen Client-Anwendung verwendet

Abbildung 5.7: Nokia N93 und GPS-Maus Holox GPSlim236

5.4.1.2 Software

Als Laufzeitumgebung für das mobile Endgerät wird die auf dem N93 integrierte Java 2 Micro Edition (CLDC 1.1, MIDP 2.0) verwendet. Zur Implementation der mobilen Client-Anwendung wird die Entwicklungsumgebung Eclipse 3.3.2 mit dem Plugin EclipseME 1.7.9 für J2ME genutzt. Um die mobile Anwendung zu simulieren wird außerdem das Sun Java Wireless Toolkit (WTK) 2.5.2 als Emulator eingesetzt. Zusätzlich werden die auf dem N93 bereits vorhandenen optionalen Pakete JSR-172 (Web Service API) und JSR-179 (Location API)

verwendet. Als letztes Softwareelement der mobilen Client-Anwendung wird die J2MEMap API (Version 0.9 Beta) zur Implentation der Kartendarstellung eingesetzt, wie schon in den Designentscheidungen in Kapitel 5.1 ausführlich erläutert wurde.

Auf dem zentralen Server wird als Laufzeitumgebung Java 1.6 (JRE) und zur Implementation der Server-Anwendung das Java 1.6 SDK verwendet. Zusätzlich wird die auf dem PC der HAW Hamburg bereits vorhandene Entwicklungsumgebung Eclipse 3.2 genutzt. Als Webserver wird Apache Tomcat 5.5 eingesetzt, der sich in Verbindung mit Eclipse sehr gut für die Entwicklung von Server-Anwendungen eignet. Für die Implementation der Webservices wird die SOAP-Engine Apache Axis 1.4 verwendet.

5.4.2 Realisierung der mobilen Client-Anwendung

Im folgenden Kapitel wird auf die Realisierung und Implementierung der mobilen Client-Anwendung eingegangen. Dafür wird zuerst in Kapitel 5.4.3.1 der funktionale Umfang der mobilen Client-Anwendung festgelegt.

5.4.2.1 Funktionaler Umfang der mobilen Client-Anwendung

Im Hinblick auf die zur Verfügung stehende Zeit kann der funktionale Umfang der mobilen Client-Anwendung nicht vollständig den gestellten Anforderungen aus Kapitel 4 entsprechen. Dabei wurde folgende Anforderung aus zeitlichen Gründen ausgelassen: Darstellung des Fussweges von der Endhaltestelle zum Ziel.

Der Benutzer soll mit der mobilen Anwendung NavMobile seinen eigenen aktuellen Standort mit Hilfe einer externen GPS-Maus ermitteln können. Darüber hinaus kann er sich diesen Standort auf einer interaktiven Kartendarstellung anzeigen lassen, wobei der eigene Standort auch in regelmäßigen Abständen aktualisiert wird. Desweiteren kann der Benutzer über ein Eingabeformular folgende Angaben zum Ziel machen:

- die Zielart auswählen, welche in Straße, Haltestelle oder Besondere Stätte gegliedert ist
- den Namen der ausgewählten Zielart eingeben, z.B. Hamburger Michel (Besondere Stätte)

- den Ort, wobei der Gesamtbereich[20] Hamburg vorgegeben wird-> z.B. Ahrensburg ist auch möglich
- die Zeit, welche direkt eingegeben oder per Menü (jetzt, in 10, 20 oder 30 Minuten) ausgewählt werden kann
- das Datum, wobei das aktuelle vorgegeben wird

Sobald der Benutzer die Zieldaten eingegeben hat, kann er die Fahrplansuche starten. Die Zieldaten werden dann über eine Kommunikationsverbindung mit dem zentralen Server übertragen. Bei einer Eindeutigkeit des Zieles wird das ermittelte Ergebnis als strukturierter Fahrplan auf dem Display dargestellt. Wenn das Ziel mehrdeutig ist, werden die verschiedenen Möglichkeiten zur Auswahl auf dem Display angezeigt. Von der Darstellung des Fahrplans kann dann auf die interaktive Kartendarstellung gewechselt werden. In die Kartendarstellung sind folgende Funktionen integriert:

- Anzeige und regelmäßige Aktualisierung des eigenen Standorts
- Anzeige des Fussweges zur nächsten Haltestelle, wobei der Fussweg im Bezug auf die eigene Position nicht aktualisiert wird
- Kennzeichnung aller beteiligter Haltestellen der Fahrplanverbindung als POI's
- Darstellung der Informationen zu den POI's, wie zu verwendende Verkehrsmittel, Abfahrts- und Ankunftszeiten in einer kurzen Form auf dem Display

Im Folgenden wird erläutert, wie die genannte Funktionalität auf Basis der technischen Vorausetzungen realisiert ist.

5.4.2.2 Graphische Benutzeroberfläche

Das Diagramm aus Abbildung 5.8 zeigt die Elemente der graphischen Oberfläche auf der mobilen Client-Anwendung. Die Darstellung der Elemente erfolgt dabei hauptsächlich durch Komponenten (z.B. `Form`) der High-Level-API von J2ME. Allerdings wird für die Darstellung des Kartenmaterials die J2MEMap API Klasse `MapDisplay` genutzt, welche Funktionen der Low-Level-API (`Canvas`) verwendet.

[20] der Gesamtbereich umfasst das Bundesland Hamburg sowie die Landkreise Harburg, Lüneburg, Stade, Herzogtum Lauenburg, Segeberg, Stormarn und Pinneberg

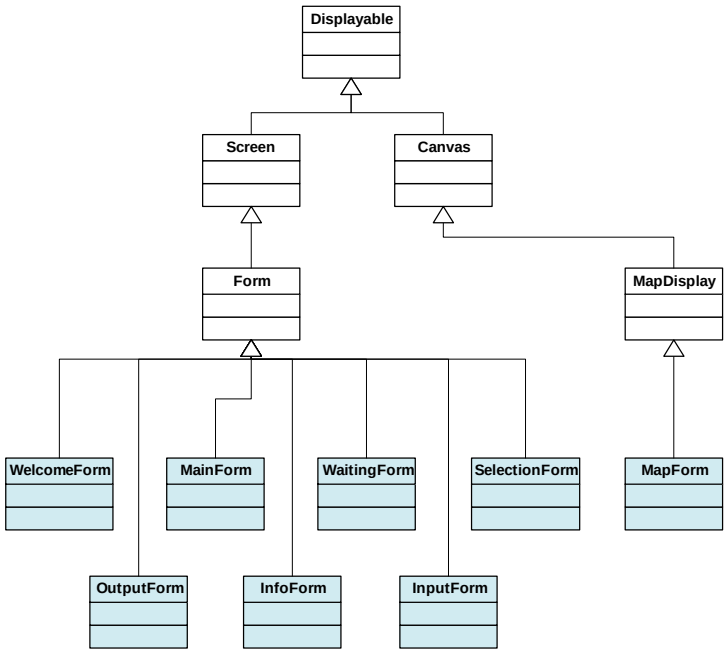


Abbildung 5.8: Abbildung der graphischen Klassen

Die farblich gekennzeichneten Klassen in Abbildung 5.8 stellen die Klassen der graphischen Oberläche dar. Diese sind alle indirekt von der abstrakten Klasse `Displayable` abgeleitet, welche die grundlegenden Eigenschaften für die Anzeige auf einem mobilen Endgerät definiert. Um diese Klassen darstellen zu können, wird die Klasse `Display` benötigt. Diese repräsentiert das physische Abbild des Bildschirms und genau eine Instanz dieser Klasse wird beim Start der Anwendung zur Verfügung gestellt. Somit ist gewährleistet, dass nur eine graphische Klasse gleichzeitig auf dem Bildschirm dargestellt werden kann.

Der größte Teil der graphischen Klassen ist direkt von `Form` abgeleitet. `Form` bündelt verschiedene Komponenten der High-Level-API in einem gemeinsamen Formulardialog. Diese Komponenten sind zum Beispiel eine `Textbox`, `List`, `Alert`, `StringItem` oder `ImageItem`, mit denen relativ einfach die benötigten Benutzeröberflächen für J2ME-Anwendungen zusammengestellt und implementiert werden können.

Die einzige Ausnahme bildet die Klasse `MapForm`, welche direkt von der Klasse `MapDisplay` abgeleitet ist. Diese Basisklasse der J2MEMap API benötigt wiederum für die korrekte Funktionalität die Low-Level-API (`Canvas`), welche flexiblere Fähigkeiten hinsichtlich der Darstellung von visuellen Elementen und elementaren Benutzerinteraktionen (z.B. Tastenbelegung) zur Verfügung stellt.

Um die Benutzerinteraktion zu ermöglichen, enthalten alle graphischen Klassen Objekte der

Klasse `Command` und implementieren das dazugehörige Interface `CommandListener`. Mit den Objekten der Klasse `Command` können interaktive Funktionen (z.B. Beenden, Fahrplan suchen) für die graphische Oberfläche definiert werden. Die zu implementierende Methode `commandAction(Command cmd, Displayable d)` des Interfaces `CommandListener` sorgt hingegen dafür, dass die Interaktionen des Benutzers ereignisbasiert abgefangen und verarbeitet werden können. Detailliertere Informationen zur Erstellung von graphischen Oberflächen mit J2ME sind im [Schmatz (2004)] zu finden.

Nachfolgend werden die graphischen Klassen und deren Aufgaben kurz erläutert, die in Kapitel 5.4.2.3 als exemplarischer Durchlauf der graphischen Oberfläche in Form von Screenshots des mobilen Endgerätes gezeigt werden.

WelcomeForm: Diese graphische Klasse wird gleich nach dem Start der Anwendung initialisiert und enthält die Aufforderung zur Inbetriebnahme der externen GPS-Maus. Nach der erfolgreichen Herstellung einer Datenverbindung und dem Erhalt von gültigen Positionsdaten erfolgt eine automatische Weiterleitung zur `MainForm`.

MainForm: Mit Hilfe der `MainForm` werden die ermittelten geographischen Koordinaten in einem Textfeld dargestellt und regelmäßig aktualisiert. Außerdem stellt es ein Auswahlmenü bereit, mit welchem ein Wechsel zur Kartendarstellung (`MapForm`) oder zur Eingabe der Zieldaten (`InputForm`) möglich ist.

InputForm: Die `InputForm` stellt ein Formular zur Eingabe der Zieldaten für den zu ermittelnden Fahrplan bereit. Desweiteren kann der Benutzer die Aktion „Fahrplan suchen" über das Menü initiieren, worauf der automatische Wechsel zur `WaitingForm` folgt.

WaitingForm: Die `WaitingForm` wird während der Ermittlung der optimierten Fahrplanverbindung mit dem „Bitte warten" Hinweis dargestellt. Der darauf folgende automatische Wechsel zu einer anderen graphischen Klasse wird von drei möglichen Szenarien bestimmt:

- Auftreten eines Fehlers (Kommunikation, falsche Eingabe,etc.)-> Wechsel zurück zur `InputForm`
- Fahrplanverbindung wurde korrekt ermittelt -> Wechsel zur `OutputForm`
- Zieleingabe ist mehrdeutig -> Wechsel zur `SelectionForm`

SelectionForm: Diese graphische Klasse stellt die Auswahl der möglichen Ziele dar. Mit der Bestätigung des Benutzers für ein bestimmtes Ziel wird erneut „Fahrplan suchen" initiiert und automatisch zur `WaitingForm` gewechselt.

OutputForm: Mit Hilfe der `OutputForm` wird die optimierte Fahrplanverbindung in strukturierten Textfeldern dargestellt. Darüber hinaus wird dem Benutzer über das Menü die Aktion „Weg zeigen" zur Auswahl gestellt. Konkret bedeutet das, dass zur Kartendarstellung (`MapForm`) gewechselt und der Fussweg zur nächsten Haltestelle sowie zusätzlichen Information und alle beteiligten POI's dargestellt werden sollen.

MapForm: Die `MapForm` enthält die Schnittstellen zur J2MEMap API und ist für die Initialisierung der Kartendarstellung verantwortlich. Auf dieser Karte wird die eigene aktuelle Position angezeigt. Sobald eine Fahrplanverbindung ermittelt wurde, können der Fussweg, die Haltestellen und zusätzliche Informationen zu den POI's dargestellt werden. Außerdem ist über das Menü ein Wechsel zur `OutputForm` möglich, um sich die Fahrplanverbindung detailliert anzeigen zu lassen. Darüber hinaus ist auch ein Wechsel zur `InputForm`, `MainForm` und `InfoForm` möglich.

InfoForm: Die `InfoForm` ist nur von der `MapForm` aus erreichbar. Sie enthält eine Kontexthilfe über Tastenkürzel für spezielle Funktionen der `MapForm` (Zoom in, Zoom out, nächster Wegpunkt,etc.) kurz beschreibt.

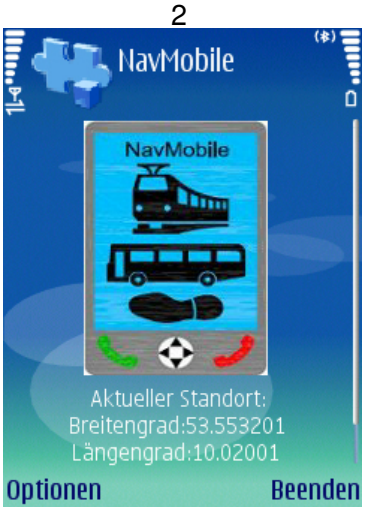
5.4.2.3 Screenshots der mobilen Client-Anwendung

In diesem Abschnitt wird ein vollständiger Durchlauf der mobilen Client-Anwendung NavMobile in Form von Screenshots des Mobiltelefons Nokia N93 dargestellt.
Hinweis zu den Screenshots: Ein heller Balken, der nicht über die gesamte Breite des Displays geht, stellt ein Popup-Menü dar (gerätespezifisch). Der ausgewählte Eintrag dieses Popup-Menüs wird als Beschriftung dieses hellen Balkens angezeigt.

1

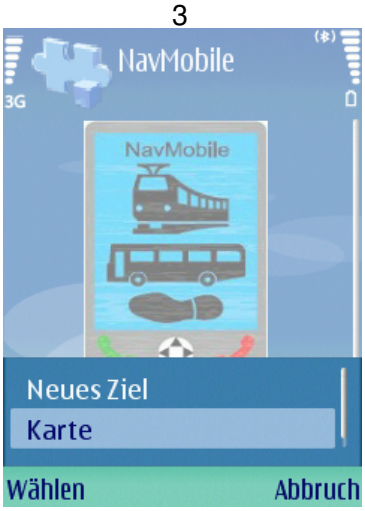
Nach dem Start wird mit dem Lokalisierungsprozess durch Betätigung der *OK*-Taste begonnen.

2



Im Hauptmenü wird der Benutzer über seinen aktuellen Standort informiert.

3

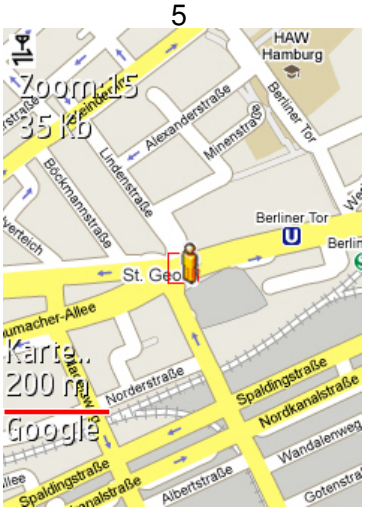


Vom Hauptmenü aus kann zum Eingabeformular des Zieles oder zur Kartendarstellung gewechselt werden.

4

Eine Internetverbindung muss hergestellt werden, um das Kartenmaterial herunterzuladen.

5

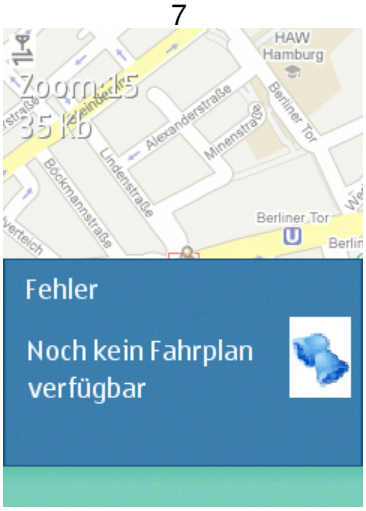


Der aktuelle Standort wird in Form eines Person-Icons auf der Karte dargestellt.

6

Das Aufrufen von *Fahrplan* oder *Ziel-Standort* ist zu diesem Zeitpunkt nicht möglich. Bei einer Auswahl folgt...

7



..eine Fehlermeldung. Die beiden Menüpunkte werden erst nach der Fahrplanermittlung freigegeben.

8

Hier wird über die Belegung der Tastenfunktionen informiert. Die Scala stellt den roten Balken auf der Karte sowie die reale Größe des Kartenausschnitts (z.B. 200 m) dar.

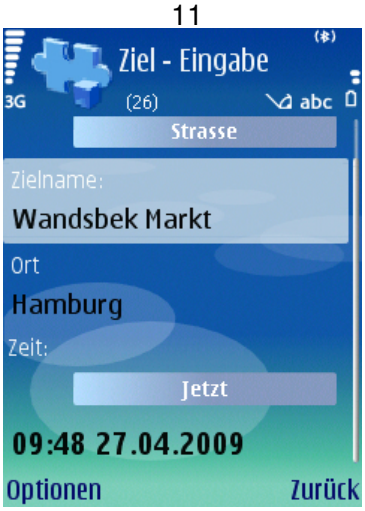
9

Mit der Auswahl von *Neues Ziel* wird zum Eingabeformular des Zieles gewechselt.

10

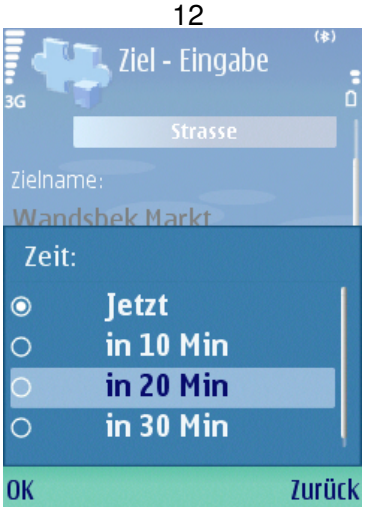
In dem Eingabeformular des Zieles kann die Zielart auf Strasse, Haltestelle oder Besondere Stätte festgelegt werden.

11



Desweiteren muß der Zielname eingegeben werden. Die Startzeit bzw. das Startdatum kann editiert werden oder...

12



es wird dieses einfache Popup-Menü für die Festlegung der Startzeit verwendet.

13

Mit der Auswahl von *Fahrplan suchen* werden die Eingabedaten bestätigt und mit der Ermittlung der Fahrplanverbindung begonnen.

14

Während der Datenübertragung wird eine Fortschrittsanzeige auf dem Display angezeigt.

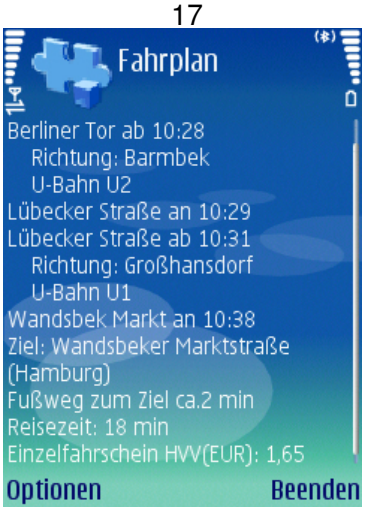
15

Da die Eingabe des Zieles nicht eindeutig war, werden entsprechende Vorschläge unterbreitet.

16

Eines der möglichen Ziele kann dann explizit ausgewählt und bestätigt werden. Die Fahrplansuche beginnt dann erneut und es folgt ein Wechsel zur Fortschrittsanzeige (Screenshot 14).

17



Das Ziel war eindeutig und die ermittelte optimierte Fahrplanverbindung wird auf dem Display dargestellt.

18

Mit der Auswahl *Weg zeigen* folgt ein Wechsel zur Kartendarstellung.

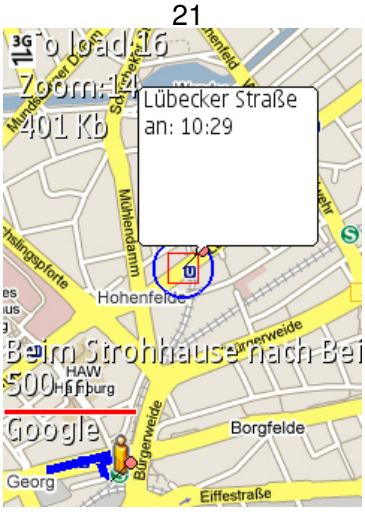
19

Der eigene aktuelle Standort und der Fussweg zur nächsten (ermittelten) Haltestelle wird angezeigt.

20

Alle beteiligten Haltestellen werden als POI's auf der Karte angezeigt und enthalten notwendige Fahrplandaten oder Informationen (z.B das Ziel).

21

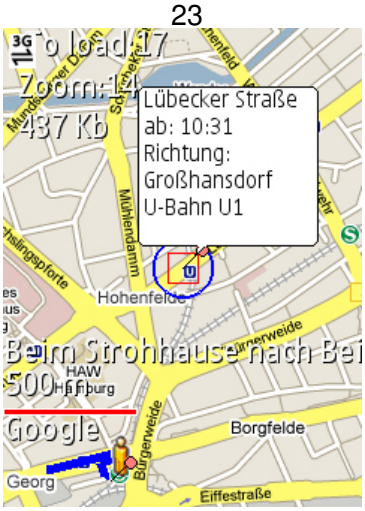


Die Informationen der POI's werden nur angezeigt, wenn sie mit dem Cursor zentriert oder mit Hilfe der Funktionstasten (7,9) selektiert werden.

22

Beim Betreten von Gebäuden (z.B. U-Bahnstation) können keine GPS-Positionsdaten ermittelt werden. Das Person-Icon verbleibt während dieser Zeit auf der letzten bekannten Position bis wieder eine GPS-Verbindung besteht.

23



Wenn an einer Haltestelle umgestiegen werden muß, werden aus Platzgründen 2 POI's (Ankunft, Abfahrt) für dieselbe Haltestelle annotiert. Diese können nur mit den Funktionstasten (7,9) selektiert werden, weil der Cursor sie nicht unterscheiden kann.

24

Die Endhaltestelle wurde erreicht und beim Verlassen der U-Bahnstation...

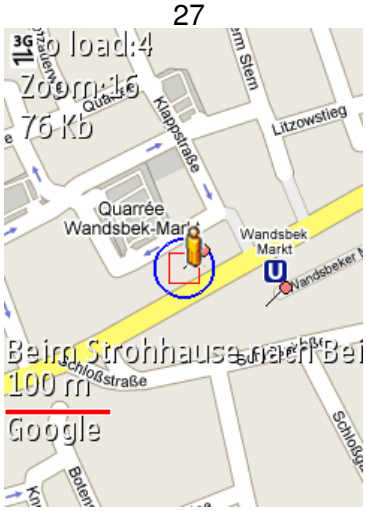
25

...ist wieder eine GPS-Verbindung verfügbar. Der aktuelle Standort wird sofort auf der Karte angezeigt.

26

Der Fussweg von der Endhaltestelle zum Ziel wurde aus zeitlichen Gründen nicht implementiert.

27



Das Ziel wurde erreicht und der vollständige Durchlauf ist damit abgeschlossen.

5.4.2.4 Klassendiagramm der mobilen Client-Anwendung

Auf Grund der Komplexität der mobilen Client-Anwendung werden in der Abbildung 5.9 nur die wichtigsten Klassen als Klassendiagramm dargestellt. Um ein besseres Verständnis für die Zuständigkeiten und Aufgaben zu erreichen, werden nach diesem Klassendiagramm die einzelnen Klassen nach ihrer Zugehörigkeit zu den jeweiligen Komponenten aus dem Kapitel 5.3.3 (Verteilung der Komponenten auf Systemarchitektur) eingeteilt, weil dort die speziellen Aufgaben schon detailliert erläutert wurden. Nach dieser Zuordnung wird noch auf wichtige Implementierungsdetails eingegangen.

Für das hier gezeigte Klassendiagramm gilt: Aus Platzgründen wurde auf die Darstellung der Methodensignatur sowie auf die vollständige Darstellung der Objekt- und Parameterstruktur verzichtet.

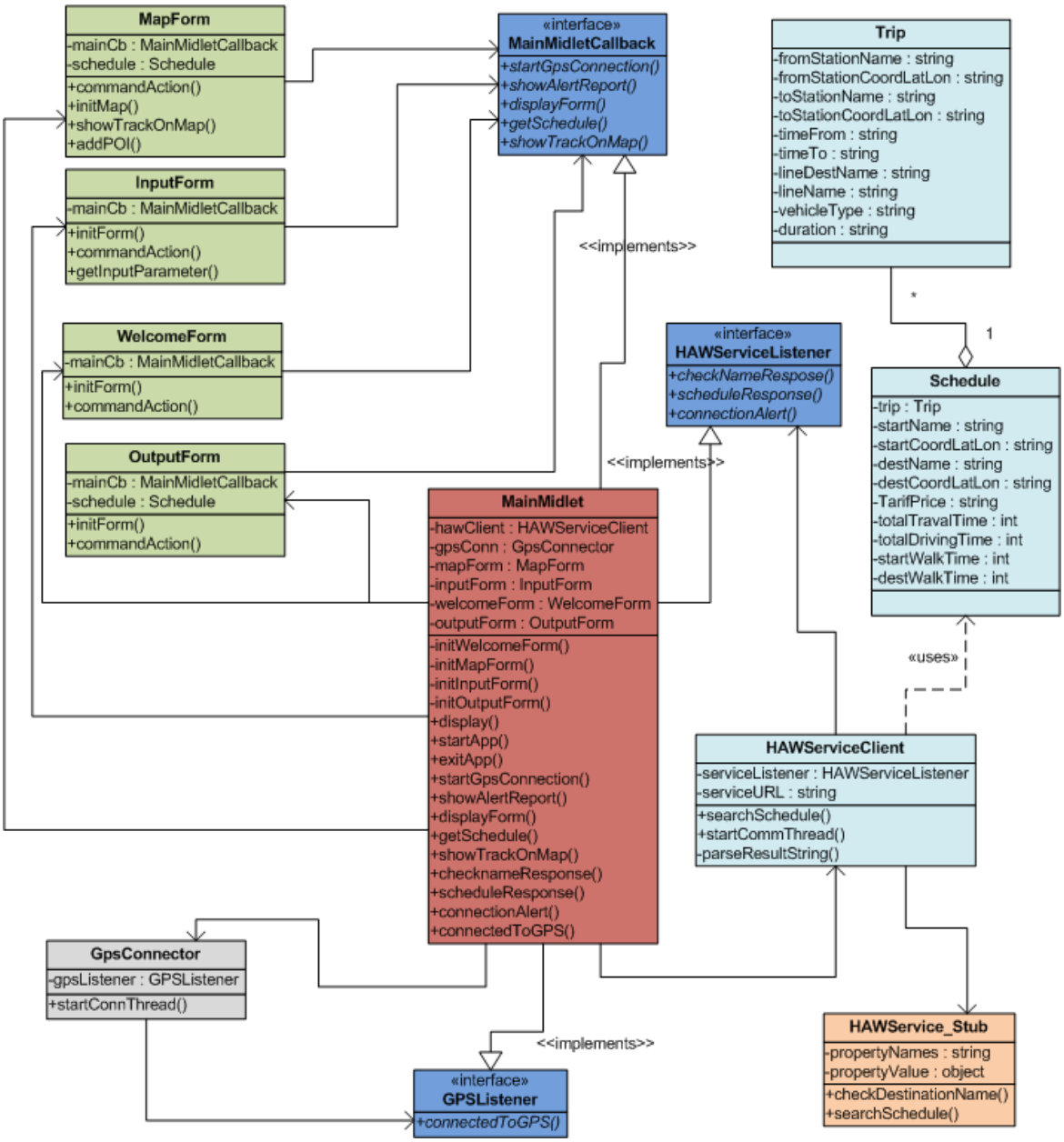


Abbildung 5.9: Klassendiagramm der Client-Anwendung

Bei der folgenden Zuordnung sind die blau gekennzeichneten Interfaces `MainMidletCallback`, `GPSListener` und `HAWServiceListener` ausgenommen, weil sie die Verbindungsglieder zwischen den einzelnen Komponenten sind und eine indirekte Nachrichtenkommunikation (Sender kennt nicht den Empfänger) ermöglichen. Diese wurden nach dem Inversion of Control Paradigma (vgl. [Martin (2003)]) und dem Ent-

wurfsmuster Observer-Pattern (vgl. [Erich Gamma (1996)]) konzipiert und implementiert, um eine lose Kopplung der beteiligten Klassen zu gewährleisten.

Zuordnung der Klassen auf Komponenten:

- Anwendungssteuerung: `MainMidlet`
- Darstellungskomponenten
  - Dialog: `WelcomeForm`, `InputForm`, `OutputForm` (nicht dargestellt: `SelectionForm`, `InfoForm`, `WaitingForm`, `MainForm`)
  - Kartendarstellung: `MapForm`
- Lokalisierungsmodul: `GpsConnector`
- Fahrgastinformation: `HAWServiceClient`, `Schedule`, `Trip`
- Kommunikationsmodul: `HAWService_Stub`
- Geoinformations-Client: J2MEMap API (nicht dargestellt)
- Utilities: z.B. `StringTokenizer` (nicht dargestellt)

Die Klasse `MainMidlet` stellt in der mobilen Client-Anwendung die Anwendungssteuerung dar. Sie koordiniert die Darstellung der verschiedenen graphischen Klassen. Darüber hinaus steuert sie den Kontroll- und Nachrichtenfluss zwischen verschiedenen Klassen. Aus diesem Grund implementiert die Klasse `MainMidlet` die Methoden der Interfaces `MainMidletCallback`, `GPSListener` und `HAWServiceListener`. Somit ist diese Klasse in der Lage Nachrichten zu empfangen ohne die Abhängigkeit zum jeweiligen Sender zu erhöhen.

Das Interface `MainMidletCallback` wurde dafür konzipiert, die Benutzerinteraktionen, die nicht innerhalb der graphischen Klassen verarbeitet werden können, an das `MainMidlet` weiter zu senden. Da nicht für jede einzelne graphische Klasse ein eigenes Interface definiert werden sollte, weil es die Anwendung nur unnötig aufblähen und die Verständlichkeit verschlechtern würde, sind alle benötigten Methoden für die verschiedenen klassenübergreifenden Benutzerinteraktionen in dem Interface `MainMidletCallback` zusammengefasst worden.
Die Methoden `displayForm()` und `showAlertReport()` werden dabei von allen graphischen Klassen benötigt, um einen Wechsel der Bildschirmanzeige zu initiieren (`displayForm()`) oder eine Fehlermeldung weiter zu leiten (`showAlertReport()`).

Die Methode `startGpsConnection()` wird von der `WelcomeForm` aufgerufen, damit das `MainMidlet` mit Hilfe der Klasse `GpsConnector` eine Datenverbindung zur externen GPS-Maus herstellen kann. Diese Datenverbindung wird in einem separaten Thread der Klasse `GpsConnector` etabliert, damit das `MainMidlet` nicht blockiert.
Die Methode `getSchedule()` wird von der `InputForm` benötigt, um die Ermittlung einer Fahrplanverbindung über den `HAWServiceClient` zu initiieren. Das Sequenzdiagramm in Abbildung 5.12 schildert detailliert die dynamischen Aspekte dieser Interaktion und beschreibt den Austausch der Nachrichten. Die Ermittlung der Fahrplanverbindungen durch die Klasse `HAWServiceClient` erfolgt dabei in einem separaten Thread. Die Klassen `Schedule` und `Trip` werden von dem `HAWServiceClient` genutzt, um die ermittelte Fahrplanverbindung in einem einfachen Objektmodell zu repräsentieren.
Die letzte Methode `showTrackOnMap()` des Interfaces `MainMidletCallback` ermöglicht es der `OutputForm` das `MainMidlet` darüber zu informieren, dass der Fussweg zur nächsten Haltestelle dargestellt werden soll. Daraufhin initiiert das `MainMidlet` die Darstellung der `MapForm` sowie die Annotation des Fussweges, der beteiligten POI's und zusätzlichen Informationen.

Das Interface `GPSListener` wird von der Klasse `GpsConnector` verwendet, um das MainMidlet in regelmäßigen Abständen mit Nachrichten über Positionsdaten und Statusinformationen zu versorgen. Ebenso wird das Interface `HAWServiceListener` von der Klasse `HAWServiceClient` dazu verwendet, Nachrichten an das `MainMidlet` zu verschicken. Diese Nachrichten können Informationen über Fahrplanverbindungen oder Fehlermeldung enthalten.

5.4.2.5 Sequenzdiagramm des Szenarios Fahrplan suchen und Ziel auswählen

Das Sequenzdiagramm in Abbildung 5.12 zeigt exemplarisch den Kontrollfluss über die Komponenten der mobilen Client-Anwendung für das Szenario *Fahrplan suchen* und *Ziel auswählen*. In diesem Szenario wird davon ausgegangen, dass das eingegebene Ziel nicht eindeutig ist und somit der Benutzer dazu aufgefordert wird, das Ziel noch einmal explizit auszuwählen.

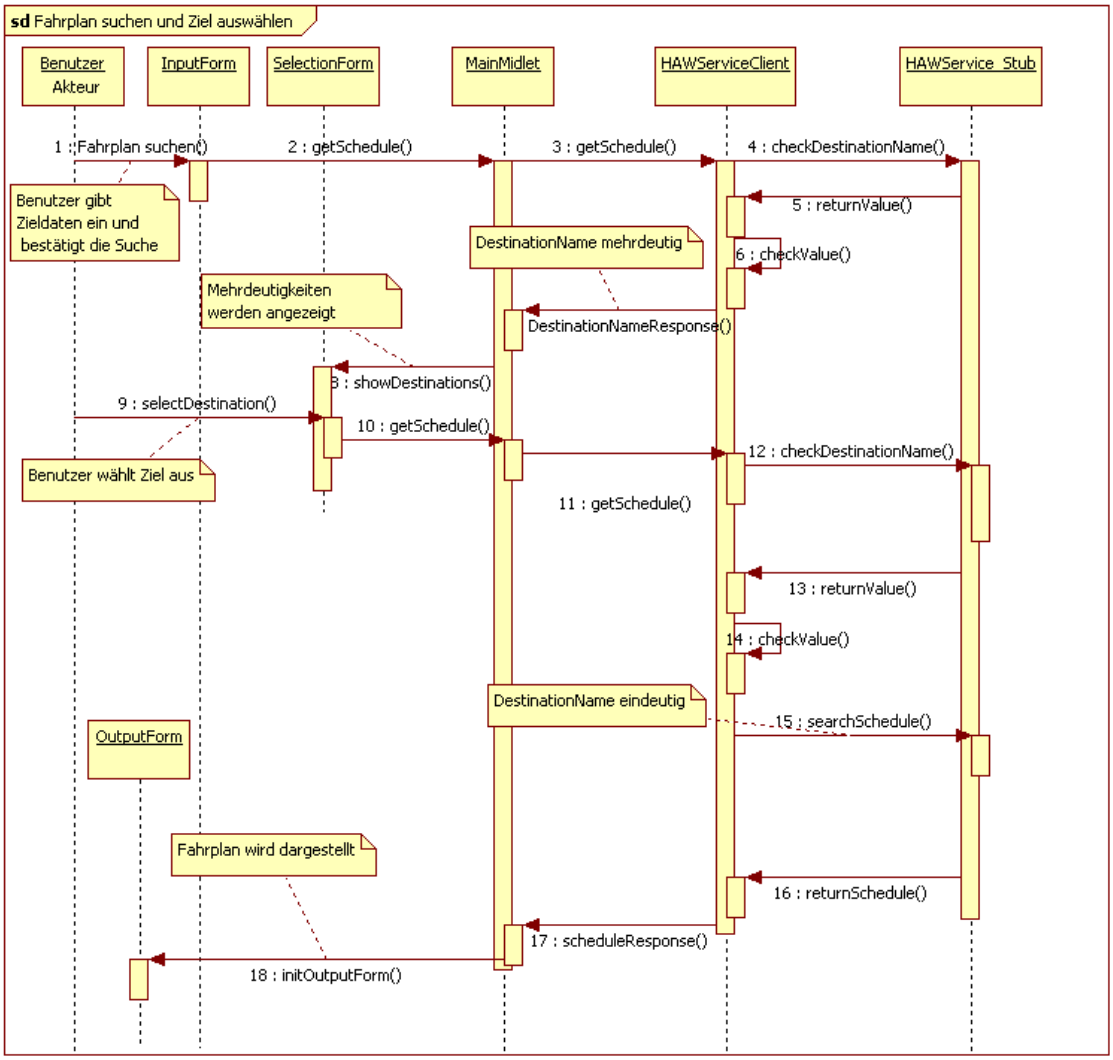


Abbildung 5.10: Sequenzdiagramm Fahrplan suchen und Ziel auswählen (Client)

Das Szenario des dargestellten Sequenzdiagramms beginnt mit der Interaktion *Fahrplan suchen* des Benutzers. Die `InputForm`, welche diese Aktion anbietet, leitet diese Fahrplananfrage mit allen eingegebenen Parametern an das `MainMidlet` weiter. Wie schon in der Beschreibung des Klassendiagramms der mobilen Client-Anwendung in Kapitel 5.4.2.4 erläutert wurde, erfolgt die gesamte Nachrichtenübermittlung zum `MainMidlet` über speziell dafür vorgesehene und implementierte Listener- und Callback Interfaces. Aus diesem Grund wird in dieser Beschreibung nicht mehr näher darauf eingegangen.

Das `MainMidlet` nimmt die Fahrplananfrage und die eingegebenen Parameter entgegen,

fügt als Startpunkt die aktuellen Positionsdaten (geographische Koordinaten) hinzu und sendet die Anfrage über die Methode `getSchedule()` an den `HAWServiceClient` weiter. Dieser startet einen separaten Thread, in dem eine logische Verbindung zum Webservice des zentralen Server etabliert wird. Konkret bedeutet das, dass der `HAWServiceClient` auf die Methoden `checkDestinationName()` und `searchSchedule()` des Webservices zugreifen kann. Mit Hilfe der Methode `checkDestinationName()` wird nun eine Überprüfung der eingegebenen Zieldaten vorgenommen und zum `HAWService_Stub` weitergeleitet. Dieser ist für die physische Verbindung zuständig und sendet die Zieldaten als XML Repräsentation über HTTP an den zentralen Server. Dann nimmt er das XML-Ergebnis entgegen, transformiert es in den Java Basistyp `String` und schickt es an den `HAWServiceClient` zurück. Dort wird dieser Ergebnis-`String` auf Eindeutigkeit des Zieles überprüft. Wenn das Ziel eindeutig ist, enthält der `String` die von dem Geofox Fahrgastinformationsdienst vorgegebene eindeutige ID dafür. Bei einer Mehrdeutigkeit sind in dem `String` die verschiedenen Auswahlmöglichkeiten beinhaltet.

Da das Ergebnis nicht eindeutig war, werden die Auswahlmöglichkeiten über das `MainMidlet` an die `SelectionForm` gesendet, wo sie dargestellt werden. Der Benutzer kann nun das richtige *Ziel auswählen* und die Fahrplananfrage wird erneut durchführt. Ebenso wird das Ziel noch einmal überprüft, um eine eindeutige ID für die Fahrplananfrage zu erhalten. Wie man auf dem Sequenzdigramm sehen kann, ist die zweite Überprüfung erfolgreich und eine eindeutige ID steht nun für die Fahrplananfrage zur Verfügung. Diese Anfrage stellt der `HAWServiceClient` mit Hilfe der Methode `searchSchedule()` über den `HAWService_Stub` an den zentralen Server. Das Ergebnis dieser Fahrplananfrage wird wieder als `String` an den `HAWServiceClient` zurückgegeben. Das Ergebnis transformiert er dann in ein einfaches Objektmodell zur Repräsentation von Fahrplananfragen mit Hilfe der Klassen `Schedule` und `Trip`, welche allerdings nicht auf dem Sequenzdiagramm dargestellt sind. Dieses einfache Objektmodell wird dann über das `MainMidlet` an die `OutputForm` übergeben, welche die optimierte Fahrplanverbindung strukturiert auf dem Display darstellt.

Das serverseitige Sequenzdiagramm dieses Szenarios wird in Kapitel 5.4.3.3 dargestellt und beschrieben.

5.4.3 Realisierung des Server-Anwendung

In diesem Kapitel wird die Realisierung und Implementierung der Server-Anwendung vorgestellt. Dafür wird zuerst der funktionale Umfang in Kapitel 5.4.3.1 festgelegt.

5.4.3.1 Funktionaler Umfang der Server-Anwendung

Die Hauptaufgabe des Servers besteht darin, Anfragen vom mobilen Endgerät entgegen zu nehmen, umzuformulieren und diese an den Fahrgastinformationsdienst Geofox weiter zu leiten. Ebenso werden die Ergebnisse von Geofox entgegengenommen, komprimiert und an das mobile Endgerät versendet. Dafür wird konkret das Adapter-Pattern (vgl. [Erich Gamma (1996)]) umgesetzt.

Im Detail stellt die Server-Anwendung einen Webservice bereit, der die Überprüfung des Zielnamens und die konkrete Fahrplansuche als Dienst anbietet. Darüber hinaus ist die Server-Anwendung dafür vorgesehen, die geographischen Koordinaten des mobilen Endgerätes in Gauss-Krüger-Koordinaten umzuwandeln. Der Grund dafür ist, dass der Geofox Fahrgastinformationsdienst die Definition eines Startpunktes in Form von Koordinaten zwar erlaubt aber das Format (Gauss-Krüger) vorschreibt.

Außerdem muß die Anwendung ebenfalls Gauss-Krüger-Koordinaten wieder in geographische Koordinaten (WGS84) umrechnen können. Denn die ermittelten Haltestellenpositionen sind nur als Gauss-Krüger-Koordinaten beim Geofox Fahrgastinformationsdienst hinterlegt. Da aber die J2MEMap API der mobilen Client-Anwendung nur geographische Koordinaten der Haltestellen annotieren kann, ist eine Umrechnung unbedingt notwendig.

5.4.3.2 Klassendiagramm der Server-Anwendung

In der Abbildung 5.11 wird das Klassendiagramm der Server-Anwendung dargestellt. Nach dieser Abbildung werden die einzelnen Klassen den jeweiligen Komponenten des Servers aus dem Kapitel 5.3.3 (Verteilung der Komponenten auf Systemarchitektur) zugeordnet, weil die dort festgelegten Aufgaben der Komponenten denen der zugehörigen Klassen weitestgehend entsprechen. Implementierungsdetails zu den Aufgaben und Beziehungen der Klassen werden anschließend erläutert.

Für das hier gezeigte Klassendiagramm gilt: Aus Platzgründen wurde auf die Darstellung der Methodensignatur sowie auf die vollständige Darstellung der Objekt- und Parameterstruktur verzichtet.

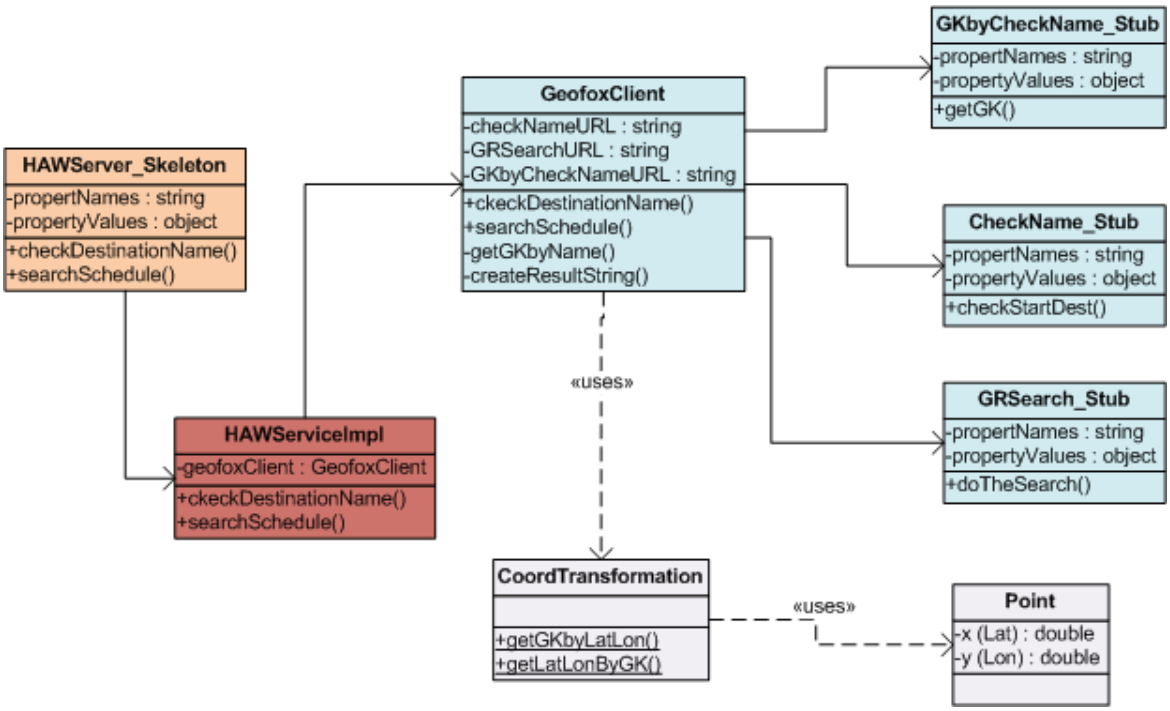


Abbildung 5.11: Klassendiagramm der Server-Anwendung

Zuordnung der Klassen auf Komponenten:

- Anwendungssteuerung: `HAWServiceImpl`
- Kommunikationsmodul: `HAWServer_Skeleton`
- Fahrgastinformation: `GeofoxClient`, `GKbyCheckName_Stub`, `CheckName_Stub`, `GRSearch_Stub`
- Utilities: `CoordTransformation`, `Point`

Die Klasse `HAWServiceImpl` repräsentiert die Anwendungssteuerung in der Server-Anwendung und dient als Bindeglied zwischen der Klasse `HAWServer_Skeleton` und dem `GeofoxClient`. Die `HAWServiceImpl` nimmt die Anfragen der Methoden `checkDestinationName()` und `searchSchedule()` der Klasse `HAWServer_Skeleton` entgegen und leitet sie an den `GeofoxClient` weiter. Die Ergebnisse werden dann über die Klasse `HAWServer_Skeleton` wieder zurück an die mobile Client-Anwendung gesendet. Obwohl der Aufgabenbereich dieser Klasse bei weitem nicht dem der Anwendungssteuerung der mobilen Client-Anwendung entspricht,

so ist sie trotzdem hinsichlich der Erweiterbarkeit ein wichtiger Teil dieser Anwendung. Denn vom Webservice Framework ist die Klasse `HAWServiceImpl` der vorgesehene Erweiterungspunkt für die Anwendungslogik.

Die Klasse `GeofoxClient` beinhaltet hingegen den größten Funktionsumfang der Server-Anwendung. Sie nutzt die Dienste des Geofox Fahrgastinformationsdienstes mit Hilfe der Stubs `GKbyCheckName_Stub`, `CheckName_Stub`, `GRSearch_Stub`. Um die Methoden dieser Stubs nutzen zu können, muß der `GeofoxClient` die einfachen Anfragen (in Form von Java Basistypen) der mobilen Client-Anwendung in die komplexen Objekte des Geofox Fahrgastinformationsdienstes umwandeln (z.B. `GRRequestLight`, `GRResponseLight`). Darauf wird allerdings nicht näher eingegangen.

Der `CheckName_Stub` ermöglicht eine Überprüfung auf Eindeutigkeit an Hand des Namens der Start- oder Zielparameter (z.B. Haltestellenname,Straßenname). Sind die Parameter mehrdeutig, werden die verschiedenen Möglichkeiten des Zieles zurückgegeben. Bei einer Eindeutigkeit wird eine spezielle ID zurückgegeben, die jeden erreichbaren Ort des Geofox Fahrgastinformationsdienstes eindeutig identifiziert. Diese ID wird mit Hilfe des `GRSearch_Stub` dazu verwendet, eine optimierte Fahrplanverbindung zu ermitteln. Der `GKbyCheckName_Stub` wird für die Ermittlung der Gauss-Krüger-Koordinaten an Hand von Straßennamen, Haltestellen oder Besonderen Stätten verwendet. Die Klasse `CoordTransformation` kann hingegen die Gauss-Krüger-Koordinaten in geographische Koordinaten (WGS84) umgewandeln und beherrscht ebenso die Umwandlung in umgekehrter Reihenfolge.

Das letzte wichtige Implementierungsdetail der Server-Anwendung stellt die Methode `createResultString()` der Klasse `GeofoxClient` dar. Diese ist dafür verantwortlich, alle ermittelten Informationen in einen strukturierten Result-String zu schreiben.

Im nachfolgenden Sequenzdiagramm werden die Zusammenhänge zwischen den einzelnen Klassen und die speziellen Funktionen der Methoden noch verdeutlicht.

5.4.3.3 Sequenzdiagramm des Szenarios Fahrplan suchen und Ziel auswählen

Das Sequenzdiagramm in Abbildung 5.12 stellt die Szenarien *Fahrplan suchen* und *Ziel auswählen* dar und verdeutlicht den Kontrollfluss über die verschiedenen Komponenten der Server-Anwendung. Dieses Sequenzdiagramm ist die serverseitige Fortführung des Diagramms aus Abbildung 5.12. Dabei wird davon ausgegangen, dass die Eingabe der Zielparameter nicht eindeutig ist. Somit muss der Benutzer das Ziel explizit auswählen, was eine weitere Fahrplananfrage auslöst. Nach dem Diagramm folgt noch eine detaillierte Beschreibung und Erläuterung.

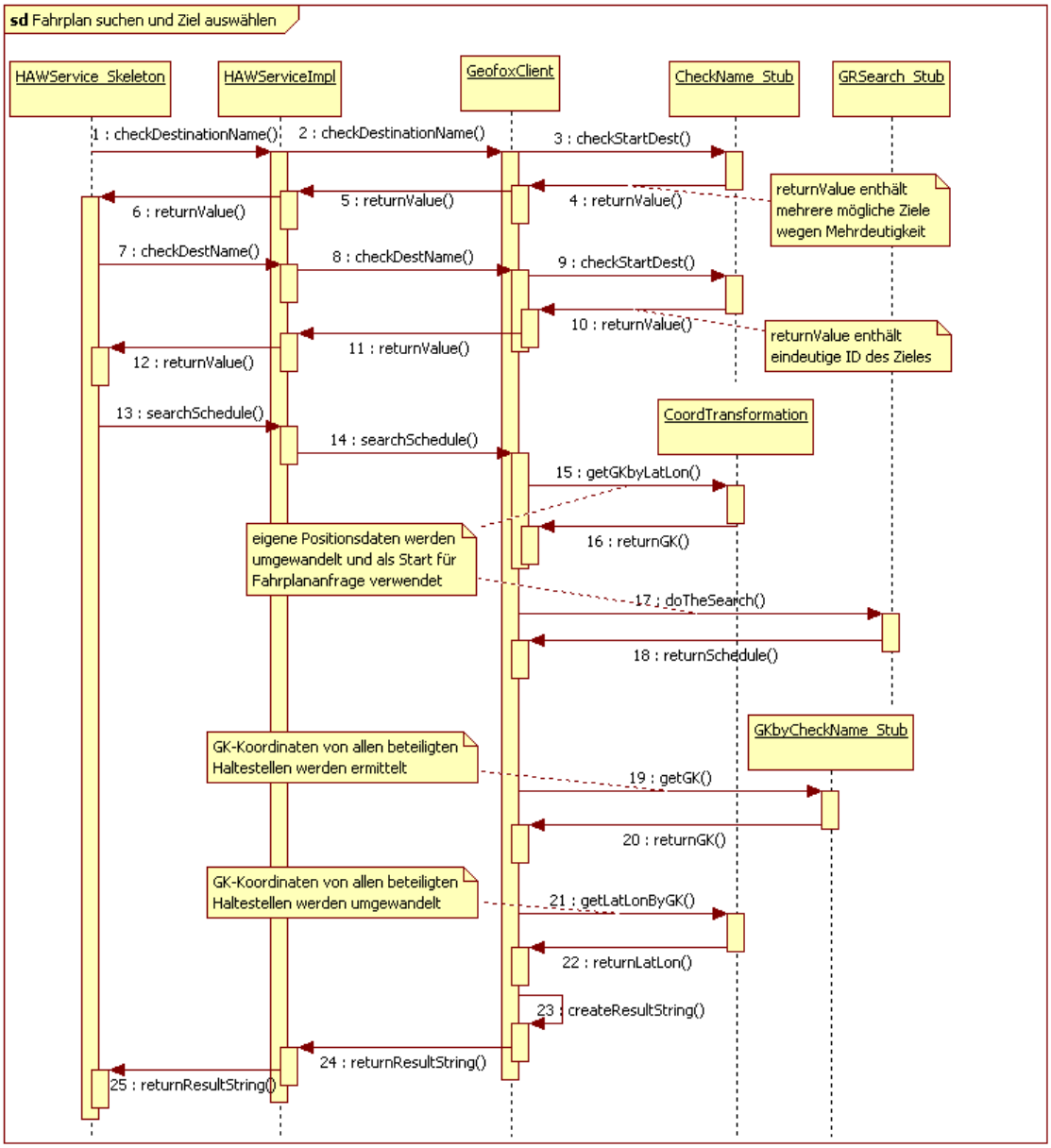


Abbildung 5.12: Sequenzdiagramm Fahrplan suchen und Ziel auswählen (Server)

Die Anfrage der mobilen Client-Anwendung auf Überprüfung der Zielparameter wird von der Klasse `HAWServer_Skeleton` empfangen. Diese wandelt die XML-Repräsentation in Java Basistypen um und sendet die Anfrage mit der Methode `checkDestinationName()` über die Klasse `HAWServiceImpl` an den `GeofoxClient`. Dieser leitet die Anfrage mit der Methode `checkStartDest()`

an den `CheckName_Stub` weiter, welcher diese Methode auf dem Server Skeleton des Geofox Fahrgastinformationsdienstes aufruft. Das Ergebnis, welches in einem komplexen Objekt repräsentiert wird, wandelt der `GeofoxClient` in einen `String` um. Dieser wird dann bis zum `HAWService_Skeleton` gesendet, in eine XML-Repräsentation umwandelt und zum mobilen Client schickt.

Bei der nächsten Anfrage ist der Ablauf identisch mit dem der ersten Anfrage, denn die unterschiedlichen Ergebnisse (mögliche Ziele oder ID) spielen für den `GeofoxClient` keine Rolle.
Schließlich wird dann die Fahrplananfrage mit der Methode `searchSchedule()` initiiert. Sobald diese Anfrage beim `GeofoxClient` angekommen ist, extrahiert er aus den übergebenen Parametern die Positionsdaten (Latitude, Longitude) und wandelt diese mit Hilfe der Klasse `CoordTransformation` in Gauss-Krüger-Koordinaten um. Mit diesen Koordinaten als Startwert wird als nächstes die konkrete Fahrplansuche über den `GRSearch_Stub` initiiert. Aus dem Ergebnis dieser Fahrplansuche werden die genauen Bezeichnungen aller beteiligter Haltestellen, einschließlich des Zieles, extrahiert und mit `GKbyCheckName_Stub` die jeweiligen Gauss-Krügger-Koordinaten ermittelt. Anschließend werden diese Koordinaten mit der Klasse `CoordTransformation` in geographische Koordinaten (WGS84) umgewandelt, weil das J2MEMap API auf dem mobilen Client nur geographische Koordinaten für die Annotation benutzen kann.

Die letzte wichtige Funktionalität ist das Zusammenstellen der ermittelten Fahrplanverbindungen, einschließlich der geographischen Koordinaten in einem strukturierten Result-String. Das Format des Result-Strings (siehe Abbildung 5.13) bezieht sich auf das selbstentwickelte einfache Objektmodell zur Repräsentation von Fahrplanverbindungen (`Schedule` und `Trip`) und ermöglicht in der Client-Anwendung eine effiziente Umwandlung in die entsprechenden Objekte.

Format des Result-Strings:
In der nachfolgenden Definition des Result-String Formates werden Platzhalter in eckigen Klammern dargestellt.
Result-String besteht aus 1 `Schedule` + 1..n `Trips`, getrennt durch „@" , die zugeordneten Attribute werden durch „$" getrennt
z.B. Result-String=`[Schedule]`@`[Trip]` (Fahrplanverbindung ohne Umsteigen) oder
Result-String=`[Schedule]`@`[Trip]`@`[Trip]`(Fahrplanverbindung mit einmal Umsteigen)
`Schedule` = [StartName] $ [StartCoordAsLonLat] $ [DestinationName] $ [DestCoordAsLonLat] $ [TotalTravelTime] $ [TotalDrivingTime] $ [StartWalkTime] $ [DestWalkTime] $ [TarifPrice]
`Trip` = [FromStationName] $ [FromStationCoordAsLonLat] $ [ToStationName] $ [ToStationCoordAsLonLat] $ [TimeFrom$TimeTo] $ [LineDestStationName] $ [LineName] $ [VehicleType] $ [Duration]

Abbildung 5.13: Format des Result-Strings

5.5 Probleme bei der Realisierung

Während der Realisierung des mobilen Navigationssystems NavMobile sind einige Schwierigkeiten und Probleme aufgetreten, die nachfolgen kurz erläutert werden.

Für die Entwicklung der mobilen Client-Anwendung wurde das Java Wireless Toolkit von Sun (WTK) verwendet. Da dieses einen Emulator enthält, konnte die Ausführung der Anwendung auf dem Entwicklungsrechner simuliert werden. Dabei ergaben sich allerdings gravierende Unterschiede zwischen der Darstellung des Emulators und der des verwendeten Nokia N93 Mobiltelefons. Aus diesem Grund wurde die Simulationsumgebung nur zum Testen von implementierter Funktionalität verwendet. Zum Testen der graphischen Oberfläche musste die Anwendung auf dem Nokia N93 installiert werden. Da der Installationsvorgang relativ viel Zeit in Anspruch nimmt, war der dafür verwendete Aufwand verhältnismässig hoch.
Neben dem Emulator wurde ebenfalls der Stub-Generator des Java Wireless Toolkit verwendet. Mit diesem konnten Client-Stubs, die für die Kommunikation mit dem Webservice (SOAP über HTTP) des zentralen Servers notwendig sind, aus einer vorgegeben WSDL-Datei automatisch generiert werden. Nachdem erste Versuche negativ verliefen, wurde festgestellt, dass der Stub-Generator des WTK's nur WSDL-Dateien akzeptiert, die im Dokument/Literal -Format kodiert vorliegen. Der Grund dafür ist, dass das Webservice API für J2ME und das darin enthaltene JAX-RPC nur eingeschränkte Funktionen für SOAP-Webservices anbietet. Aus diesem Grund muss der vom zentralen Server bereitgestellte Webservice mit einer

WSDL-Datei im kodierten Dokument/Literal-Format definiert werden. Außerdem müssen die Server-Skeletons ebenfalls mit Hilfe dieser WSDL-Datei generiert werden, damit die XML-basierten Nachrichten korrekt interpretiert werden können.
Ein weiteres Problem ist beim WTK in Verbindung mit dem verwendeten J2MEMap API aufgetreten. Dabei kann es passieren, das sich bei mehrmaliger Ausführung der Kartendarstellung, die von J2MEMap zur Verfügung gestellt wird, das MIDlet aufhängt. Der Grund dafür ist eine temporäre Datei, die das WTK in dem Verzeichnis \j2mewtk\2.5.2\appdb\DefaultColorPhone erstellt. In diesem Fall wurde natürlich das Profil DefaultColorPhone verwendet. Um den Fehler zu beheben muss die temporäre Datei einfach gelöscht werden.

Durch die Verwendung des J2MEMap API's in dieser Arbeit wurde wesentliche Funktionalität, wie die Darstellung des Kartenmaterials sowie der Routen und annotierten POI's, erheblich leichter in das mobile Navigationssystem NavMobile integriert. Dabei sind gerade zu Beginn verschiedene Schwierigkeiten aufgetreten. Der Grund dafür ist die sehr schlechte Dokumentation der jeweiligen Schnittstellen. Darüber hinaus werden Tutorials bereitgestellt, die schon längst nicht mehr vorhandene Datenstrukturen benötigen. Außerdem ist es häufig vorgekommen, dass das benötigte Kartenmaterial von einem speziellen Anbieter (z.B. Google Maps) nicht heruntergeladen werden konnte. Da sich das J2MEMap API aber noch in einem Beta-Status der Version 0.9 befindet, kann über diese Unzulänglichkeiten hinweg gesehen werden.

Für die Realisierung des mobilen Navigationssystems wurde die GPS-Technologie eingesetzt. Wie schon im Kapitel 2.2.2.1 deutlich wurde, stellt GPS keine Indoor-Fähigkeiten zur Verfügung und begrenzt den funktionalen Umfang dieser Anwendung auf die Outdoor-Navigation. Somit ist in der praktischen Anwendung keine durchgehende Verfolgung der eigenen Position auf der Kartendarstellung möglich. Stattdessen wird die letzte bekannte Position so lange angezeigt, bis die Verbindung zu den Positionssatelliten wiederhergestellt wurde und gültige Positionsdaten geliefert werden.

Ein weiteres Problem ist im Zusammenhang mit der Fussgängernavigation aufgetreten. Die Darstellung der Fusswege auf der Karte kann von den realen Fusswegen erheblich abweichen. Der Grund dafür ist, dass die Navigationsunterstützung für Fussgänger meistens auf dem Straßennetz basiert und es sich bei den zur Verfügung gestellten Routeninformationen für Fussgänger meist um adaptierte Autonavigationslösungen handelt.

Kapitel 6

Zusammenfassung und Ausblick

6.1 Zusammenfassung

Für die Entwicklung eines mobilen verteilten Systems wurden zu Beginn der Arbeit alle dafür notwendigen Grundlagen erarbeitet und erläutert. Anschließend wurden Methoden und Konzepte von vergleichbaren Arbeiten erörtert und eine Einordnung in den aktuellen Stand der Technik vorgenommen. Um die funktionalen und nichtfunktionalen Anforderungen an das zu entwickelnde System zu identifizieren, wurde eine Analyse nach den Standards des Software Engineerings durchgeführt. Für die funktionalen Anforderungen wurden dazu Anwendungsfälle bestimmt und gemäß dem UML-Standard als Use-Case-Diagramm modelliert. Die einzelnen Use-Cases wurden dann in tabellarischer Form detailliert beschrieben, um Abhängigkeiten und Zuständigkeiten aufzuzeigen. Dabei wurde insbesondere auf die Vollständigkeit und Konsistenz aller beteiligten Use-Cases Wert gelegt, damit sich keine Widersprüche und Ungenauigkeiten bei der Entwicklung des Designs ergeben können.

Im Design wurde die System- und Softwarearchitektur entwickelt und erläutert. Dabei wurden die nichtfunktionalen Anforderungen aufgegriffen und nachfolgend kurz erläutert. Für die Benutzerfreundlichkeit wurde die graphische Oberfläche der mobilen Anwendung übersichtlich und mit einer einfachen Menüstruktur entworfen. Um die Zuverlässigkeit zu gewährleisten, wurden bewährte Technologien für die Entwicklung verwendet. Darüber hinaus wurden fehlerhafte Abläufe identifiziert und dafür adäquate Fehlerdialoge entwickelt. Die Korrektheit von Ergebnissen wurde während der Entwicklung durch umfangreiche Smoke-Tests[21]

[21] Ausprobieren des Testobjektes, das vorwiegend die prinzipielle Lauf- und Testfähigkeit des Testobjekts prüft

(vgl.[Pressman (2000)]) sichergestellt. Die Portierbarkeit wird durch den Einsatz von plattformunabhängigen Programmiersprachen sowie durch standardisierte Kommunikationsverfahren gewährleistet. Um die Wartbarkeit und Erweiterbarkeit zu verbessern, wurde eine mehrschichtige Softwarearchitektur konzipiert sowie eine lose Kopplung zwischen den beteiligten Komponenten umgesetzt. Für die Effizienz und Performance wurden rechenintensive Aufgaben auf den zentralen Server ausgelagert. Darüber hinaus wurden für die Kommunikation zwischen dem mobilen Client und dem zentralen Server nur einfache Datenstrukturen, wie z.B. Java Basistypen, verwendet. Die Sicherheit und der Datenschutz wurden aus zeitlichen Gründen vernachlässigt.
Das Softwaredesign wurde in Klassen-, Sequenz- und Komponentendiagrammen erläutert und damit eine transparente Schnittstelle zur Realisierung und Implementierung vorgestellt.

Ebenfalls liegt als Ergebnis der Arbeit eine vollständige Implementierung der NavMobile Anwendung als verteiltes Multi-Tier Client-Server System vor, das die funktionalen und nichtfunktionalen Anforderungen der Analyse umsetzt. Dieses System besteht aus einem J2ME-Client, einem Apache Axis-basierten Webserver und dem dritten „Tier", der Anbindung des operativen Fahrgastinformationssystems Geofox. Unter anderem wurden dazu verwendet: Webservice Technologie, J2ME, Location API, Web Service API, J2MEMap.

6.2 Ausblick

Die Anwendungsfunktionalität des mobilen Navigationssystems NavMobile ist zu einem großen Teil von dem J2MEMap API abhängig. Auf Grund dieser Abhängigkeit und den aufgetretenen Schwierigkeiten mit dem J2MEMap API, die schon in Kapitel 5.5 erläutert wurden, wird an dieser Stelle ein alternativer Ansatz vorgestellt, über den ein funktionaler Überblick gegeben wird, ohne auf technische Details näher einzugehen.

Dieser Ansatz beinhaltet eine browserbasierte Umsetzung des mobilen Navigationssystems. Das bedeutet, dass die gesamte Anwendungsfunktionalität auf einem Server implementiert werden muss. Der Benutzer eines mobilen Endgerätes verfügt nur über ein browserbasiertes Benutzerinterface, mit dem er interagieren kann. Wenn der Benutzer eine Aktionen ausführt, wird diese sofort an den Server weitergeleitet. Der Server wertet die Aktion aus und generiert entsprechende Ergebnisse. Dafür könnte der Server zum Beispiel standortbezogenes Kartenmaterial von Geoinformationsdiensten beziehem und mit zusätzlichen Annotationen (z.B. Fußweg zu einer Haltestelle) verknüpfen. Dieses generierte Ergebnis sendet er dann an das mobile Endgerät zurück, der das Ergebnis mit Hilfe des Browsers darstellen kann. Die Aufgaben des mobilen Endgerätes beschränkt sich somit nur auf die Interaktion mit dem Benutzer, die Kommunikation mit dem Server und die Darstellung der angeforderten Ergebnisse. Durch diese Umverteilung der Funktionalität auf den Server, wird die Erweiterbarkeit,

die Änderbarkeit und Wartbarkeit erheblich verbessert. Ein weiterer Vorteil ist, dass alle mobilen Endgeräte bereits über integrierte Browser verfügen und somit unabhängig von der Plattform und dem Betriebssystem sind.

Trotzdem kann es auch bei dem browserbasierten Ansatz zu Kompatibilitätsproblemen kommen, z.B. durch unterschiedliche Displaygrößen oder die Tatsache, dass viele Geräteherstellen sich nicht immer an Standards halten und die integrierten Webbrowser nicht die vollständige Funktionalität unterstützen. Dazu kommt, dass die mobilen Endgeräte noch keine aufwendigen Grafiken darstellen können, die durch moderne Webtechnologien ermöglicht werden. Dafür ist die notwendige Rechen- und Grafikleistung der Endgeräte immer noch zu begrenzt. Zusätzlich muss beachtet werden, dass sich die gesamte Anwendung auf die Kommunikation mit dem Server stützt und somit eine ständige Internetverbindung sowie eine Datenübertragungstechnologie mit sehr hohem Datendurchsatz, wie z.B. UMTS, notwendig ist. Obwohl der browserbasierte Ansatz auch viele Nachteile mit sich bringt, wäre eine Umsetzung dieser Alternative sicherlich sehr aufschlussreich und interessant.

An dieser Stelle werden noch verschiedene Erweiterungsmöglichkeiten für das Navigationssystem NavMobile erörtert. Diese können die Akzeptanz des mobilen Navigationssystems NavMobile bei potenziellen Kunden weiter erhöhen.
Das E-Ticket ist eine Erweiterung, die es ermöglichen würde, einen Fahrschein für öffentliche Verkehrsmittel über die mobile Anwendung online anzufordern. Dabei müsste das Ticket auf dem mobilen Endgerät gespeichert und bei Bedarf (z.B. Fahrscheinkontrolle) wieder geladen und auf dem Display dargestellt werden. Darüber hinaus ist zu beachten, dass dieser Service nur in Zusammenarbeit mit einem Verkehrsverbund erarbeitet und implementiert werden kann. Außerdem müsste NavMobile das Speichern von Daten in Profilen erlauben.

Dafür könnte eine Erweiterung für Benutzerprofile implementiert werden, die nicht nur das E-Ticket speichern könnte, sondern auch häufig verwendete Ziele für die Fahrplansuche z.B. zu Hause, Arbeit, Freundin. Mit diesem Profil könnte man das Ziel auswählen ohne es jedes Mal neu eingeben zu müssen. Um die Benutzerfreundlichkeit noch weiter zu erhöhen, könnte ebenfalls eine Auto-Vervollständigen-Funktion integriert werden, bei dem die zu vervollständigen Eingaben in dem Benutzerprofil gespeichert werden.

Für Touristen wäre zudem eine ortsbezogene Suche eine äußerst sinnvolle Erweiterung. Dabei könnte eine Suche nach Geschäften, Restaurants, Sehenswürdigkeiten,etc.., an Hand des Names durchgeführt und die Position automatisch ermittelt werden. Diese könnte dann wieder als mögliches Ziel für die Fahrplansuche verwendet werden.

Literaturverzeichnis

[tomcat] *Apache Tomcat*. – URL http://tomcat.apache.org/. – 2009

[eclipse] *Eclipse IDE for Java Developers*. – URL http://www.eclipse.org/. – 2008

[HVV] *Hamburger Verkehrsverbund- Bericht 2007*. – URL http://www.hvv.de/pdf/wissenwertes/verbundbericht_2007.pdf. – 2009

[j2me] : *Java Platform Micro Edition*. – URL http://java.sun.com/javame/

[JSR179 2003] *JSR-179: Location API for Java 2 Micro Edition*. 2003. – URL http://www.forum.nokia.com/Resources_and_Information/Documentation/Java/Java_API_Specifications.xhtml#jsr179. – 2009

[JSR172 2004] *JSR-172 J2ME Web Services Specification*. 2004. – URL http://jcp.org/aboutJava/communityprocess/final/jsr172/. – 2009

[JSR205 2004] *JSR 205: Wireless Messaging API 2.0*. 2004. – URL http://jcp.org/aboutJava/communityprocess/final/jsr205/index.html. – 2009

[OpenSPIRIT 2006] *Open-SPIRIT-Studie und Pilot-Demonstration zu Intermodalen Reise-Informations-Technologien*. 2006. – URL http://www.salzburgresearch.at/research/projects_detail.php?proj=84. – 2009

[EclipseME 2008] : *EclipseME- Eclipse Mobile Tools for Java*. 2008. – URL http://eclipseme.org/

[HBT 2008] : *Geofox Projekteckdaten*. 12 2008. – URL http://www.hbt.de/geofox+M54a708de802.html

[WTK 2008] : *Sun Java Wireless Toolkit for CLDC*. 2008. – URL http://java.sun.com/products/sjwtoolkit/

[GeofoxServer 2008] : *Technische Informationen zu Geofox Server*. 12 2008. – URL `http://www.geofox.de/about/about.html`

[weTravel 2008] : *We-Travel - Free navigation, 3D maps, voice guidance and tourist information*. 2008. – URL `http://we-travel.co.cc/joomla/`

[Bauer 2002] BAUER, Manfred: *Vermessung und Ortung mit Satelliten: GPS und andere satellitegestützte Navigationssysteme*. 5. Wichmann, 2002

[Buchholz 2007] BUCHHOLZ, Clemens: Mobile Mitfahrboerse. (2007)

[Buth 2006] BUTH, Prof. Dr. B.: *Requirements Engineering*. 2006. – Vorlesungsskript Softwareengineering Teil III , Wintersemester 06/07

[Chanliau 2004] CHANLIAU, Marc: Web Services-Sicherheit und die SAML. In: *XML & Web Services Magazin Ausgabe: 1.2004* (2004). – URL `http://entwickler.com/itr/online_artikel/psecom,id,468,nodeid,69.html`

[Dapeng Wang 2004] DAPENG WANG, Thilo Frotscher Marc T.: *Java Web Services mit Apache Axis*. Software & Support Verlag, 2004

[Dietz 2005] DIETZ, Wilhelm: *Location Based Services in der Mobilkommunikation*. 2005

[Erich Gamma 1996] ERICH GAMMA, Ralph Johnson John Vlissides; Deutsche Übersetzung Dirk R.: *Entwurfsmuster- Elemente wiederverwendbarer objektorientierter Software (eng. Design Patterns)*. Bonn : Addison-Wesley, 1996

[Hanno Heeskens 2003] HANNO HEESKENS, Helge T.: *GPS und seine Anwendungen*. 2003

[Jochen Schiller 2004] JOCHEN SCHILLER, Agnés V.: *Location-Based Services*. Morgan Kaufmann Publishers, 2004

[Kahmen 2006] KAHMEN, Heribert: *Angewandte Geodäsie: Vermessungskunde*. 20. Walter de Gruyter Lehrbücher, 2006

[Landspurg] LANDSPURG, Thomas: *J2MEMap*. – URL `http://j2memap.landspurg.net`. – 2009

[Martin 2003] MARTIN, Robert C.: *Agile Software Development, Principles, Patterns, and Practices*. Pearson Education, 2003 (Alan Apt)

[Oestereich 2004] OESTEREICH, Bernd: *Objektorientierte Softwareentwicklung- Analyse und Design mit der UML 2.0*. 6. Oldenbourg verlag, 2004

[Ortiz 2002] ORTIZ, C. E.: The Wireless Messaging API. (2002). – URL `http://developers.sun.com/mobility/midp/articles/wma/`. – 2009

[Oswald 2006] OSWALD, Derrick: *HTML Parser*. 2006. – URL `http://htmlparser.sourceforge.net/`. – 2008

[Pressman 2000] PRESSMAN, Roger S.: *Software Engineering a Practitioner's Approach,*. 5. McGraw-Hill Publishing Company, 2000

[Ralf Möller 1991] RALF MÖLLER, Lothar H.: *Suchalgorithmen und Interaktionstechniken für Fahrplan-Informationssysteme*. Juli 1991. – URL `http://kogs-www.informatik.uni-hamburg.de/publikationen/pub-moeller/MoHa91a-1.pdf`

[de Ruyter und Aarts 2004] RUYTER, Boris de ; AARTS, Emile: Ambient intelligence: visualizing the future. (2004). ISBN 1-58113-867-9

[Sauter 2008] SAUTER, Martin: *Grundkurs mobile Kommunikationssysteme : Von UMTS und HSDPA, GSM und GPRS zu Wireless LAN und Bluetooth Piconetzen*. 3. Vieweg Verlag, 2008

[Schmatz 2004] SCHMATZ, Klaus-Dieter: *Java 2 Micro Edition*. dpunkt Verlag, 2004

[Schmidt 2007] SCHMIDT, Christian: *Autarke mobile Echtzeit-Fahrplannavigation*. 8 2007. – URL `http://www8.informatik.uni-erlangen.de/IMMD8/Research/MONA/document.pdf`

[Schwab 2007] SCHWAB, Tobias: *Mobile Fahrplanauskunft mit GPS und Web Services*. 3 2007. – URL `http://www8.informatik.uni-erlangen.de/IMMD8/Research/MONA/ausarbeitung.pdf`

[Stefan Bruntsch 2006] STEFAN BRUNTSCH, Karl R.: *Der Persönliche Reisebegleiter am Smartphone-Reiseinformation und Orientierung immer und überall*. 2006

[Stefan Steiniger 2006] STEFAN STEINIGER, Alistair E.: *Foundations of Location Based Services*. 2006. – URL `www.geo.unizh.ch/publications/cartouche/lbs_lecturenotes_steinigeretal2006.pdf`. – 2008

[Störrle 2007] STÖRRLE, Harald: *UML 2 erfolgreich einsetzen : Einführung und Referenz*. München : Addison-Wesley, 2007

[Tobias Hauser 2003] TOBIAS HAUSER, Ulrich M. L.: *Web Services - Die Standards - XML-RPC, SOAP, WSDL, Verzeichnisdienste, Sicherheit*. Galileo Press, 2003

[Ullenboom 2005] ULLENBOOM, Christian: *Java ist auch eine Insel, Programmieren mit der Java Standard Edition Version 5*. 5. Galileo Computing, 2005

[Weik 2006] WEIK, Dr. F.: *CheckName and GRSearch SOAP Interface*. 2006

[Weik 2007] WEIK, Dr. F.: *GEOFOX SOAP Interface*. 2007

[Weiser 1999] WEISER, Mark: *The Computer for the 21st Century- ubiquitous computing*. 1999. – URL `http://www.ubiq.com/hypertext/weiser/SciAmDraft3.html`. – 2009

[Wendholt 2007] WENDHOLT, Prof. Dr. B.: *Apache Axis*. 2007. – Vorlesungsskript Webservices / Java Webapplications, Sommersemester 07

[Zipf und Malaka 1999] ZIPF, A. ; MALAKA, R.: *Web-basierte Planung und animierte Visualisierung von 3D Besichtigungstouren im Rahmen des Touristeninformationssystems Deep Map*. B. Zagel, 1999

[Zohurian 2007] ZOHURIAN, Dariusch: *Entwicklung einer J2ME Anwendung für Mobiltelefone zur multimedialen Live Annotation von Points of Interest*. 10 2007. – URL `http://www.mindmatters.de/assets/2009/2/6/Bachelorarbeit.pdf`. – 2009

Tabellenverzeichnis

Hiermit versichere ich, dass ich die vorliegende Arbeit im Sinne der Prüfungsordnung nach §22(4) bzw.§24(4) ohne fremde Hilfe selbständig verfasst und nur die angegebenen Hilfsmittel benutzt habe.

Hamburg, 11.06.2009 Dirk Parlowski