KONZEPTION UND IMPLEMENTIERUNG EINER MOBILEN ANWENDUNG FÜR J2ME-FÄHIGE ENDGERÄTE

Diplomarbeit

vorgelegt von Yasser Al-Biladi

am Fachbereich Informatik, Mathematik und Naturwissenschaften der

Hochschule für Technik, Wirtschaft und Kultur Leipzig (FH)

Leipzig, 22. Juni 2004

Inhaltsverzeichnis

Erklärung

Hiermit versichere ich, dass ich die vorliegende Arbeit selbständig und ohne unerlaubte fremde Hilfe verfasst habe, und dass alle wörtlich oder sinngemäß aus Veröffentlichungen entnommenen Stellen dieser Arbeit unter Quellenangabe einzeln kenntlich gemacht sind.

Leipzig, den 22. Juni 2004

Yasser Al-Biladi

Vorwort

Diese Arbeit befasst sich mit der Konzeption und Implementierung eines Client-/Server-basierenden Fahrplanauskunftssystems für den städtischen Linienverkehr. Nutzern mobiler Kommunikationsgeräte wird damit ein komfortabler Zugriff auf Fahrplan-Daten ermöglicht.

Nach einer Einführung in das Thema werden dann die Technologien betrachtet und bewertet, die die Realisierung eines mobilen Informationsdienstes ermöglichen.

Danach werden die Anforderungen an die Hard- und Software zur Durchführung und Entwicklung des Fahrplanauskunftssystems festgelegt.

Weitere Kapitel beschreiben die Spezifikation und Implementierung des Prototyps.

Abschließend werden noch die erzielten Ergebnisse zusammengefasst und ausgewertet.

Kapitel 1

Einleitung

1.1 Motivation

Der weltweite Markt für Mobiltelefone wächst von Jahr zu Jahr. Das Marktforschungsinstitut Gartner teilte mit, dass im dritten Quartal 2003 weltweit 22 Prozent mehr Mobiltelefone als im dritten Quartal des Vorjahres verkauft wurden. Das sind insgesamt 132,8 Millionen verkaufte Mobiltelefone in einem Quartal.[1] Mit der Verbreitung und der steigenden Leistungsfähigkeit von mobilen Kommunikationsgeräten wächst auch der Bedarf, damit jederzeit und überall auf Informationen und Daten zuzugreifen. Der Zugriff auf Informationen sollte dabei möglichst effektiv und einfach erfolgen. Anhand dieser Arbeit werden die Möglichkeiten und die Realisierung von Informationssystemen für mobile Kommunikationsgeräte untersucht und beschrieben.

Diese Arbeit ist im Bereich mobile Informationssysteme und Softwareentwicklung einzuordnen. Es sind die Werkzeuge für die Entwicklung und die Komponenten für die Durchführung des Fahrplanauskunftssystems zu betrachten und

[1] [Hei04b]

die passenden auszuwählen. Für die Kommunikation zwischen dem Nutzer und dem Server ist eine geeignete Anwendung zu entwickeln. Diese Anwendung auf dem mobilen Gerät wird eine einfache Bedienoberfläche zur Verfügung stellen. Für den Datenaustausch mit einem Server muss eine geeignete Schnittstelle definiert werden. Die Fahrplan-Daten werden von einem http-Server abgerufen. Die praktische Durchführung erfolgt auf Emulatoren und mobilen Endgeräten und die gewonnenen Erkenntnisse können auch für andere mobile Anwendungen oder Informationsdienste genutzt werden.

1.2 Begriffe und Definitionen

In diesem Abschnitt werden grundlegende Begriffe, die für diese Arbeit von Bedeutung sind, erläutert. Dazu heißt es in [Leh03, Seite 5], dass in Verbindung mit mobilen Systemen und Anwendungen inzwischen zahlreiche Begriffe in Gebrauch sind, die trotz ihrer häufigen Verwendung noch keine präzise Definition erfahren haben. Und weiter heißt es, dass die Vielfalt der bereits existierenden Anwendungen und die beobachtbare Dynamik vermutlich eine Definition der Begriffe noch eine Weile hinauszögern wird.[2] Daher werden nur die wichtigsten Begriffe einer näheren Betrachtung unterzogen.

Kommunikation bedeutet bidirektionalen Austausch von Nachrichten zwischen Menschen und/oder Maschinen. [3]
Mobilität bedeutet Bewegung bzw. Beweglichkeit. [4]
Als **mobil** wird die Ortsunabhängigkeit der Kommunikationspartner bezeichnet.

[2] [Leh03, Seite 10]
[3] vgl. [Myr04, Seite 5]
[4] vgl. [Myr04, Seite 4]

Dabei unterscheidet man zwischen persönlicher Mobilität und Endgeräte-Mobilität.

Persönliche Mobilität bezieht sich auf den Nutzer eines Kommunikationsgerätes und die **Endgeräte-Mobilität** auf das Kommunikationsgerät, das zu jeder Zeit an jedem Ort an ein Kommunikationsnetz angeschlossen werden kann.

Als **drahtlos/wireless** wird die Kommunikationsart der Geräte bezeichnet, die Daten ohne "Kabel" mittels Funk oder Infrarot austauschen. [5]

"Unter **mobilen Endgeräten** *versteht man alle Endgeräte, die für den mobilen Einsatz konzipiert sind."* [6]

Mobile Anwendungen haben das wesentliche Merkmal, drahtlos mit anderen Systemen zu kommunizieren. Die charakteristischen Eigenschaften von Mobilen Anwendungen sind nach [Leh03, Seiten 11-13]:

Ortsunabhängigkeit: damit ist der Benutzer mit einem mobilen Endgerät jederzeit in der Lage, unabhängig von seinem Aufenthaltsort, Echtzeitinformationen abzurufen und Transaktionen durchzuführen.

Erreichbarkeit: der Nutzer ist überall und jederzeit erreichbar und kann somit überall ohne großen Zeitverlust aktuelle Informationen erhalten und diese auch nutzen.

Sofortige Verfügbarkeit: mobile Endgeräte bieten die Möglichkeit, sofort nach dem Einschalten benutzt werden zu können.

Convenience: mobile Endgeräte sind einfacher zu bedienen als herkömmliche PCs und haben eine weite Verbreitung und Akzeptanz gefunden.

Sicherheit: In Handys sind SIM- oder Smartcards integriert, die eine Authentifizierung von Personen möglich machen.

Lokalisierbarkeit: damit ist es möglich, den exakten Standort eines Mobilfun-

[5] vgl. [Leh03, Seite 9]

[6] nach [KT03, Seite 2]

knutzers zu bestimmen.

Personalisierung: Anhand der persönlichen Rufnummer ist der Mobilfunknutzer identifizierbar.

Kostengünstigkeit: Mobile Endgeräte sind im Unterschied zu herkömmlichen PCs preiswerter in der Herstellung und preiswerter als PCs im Handel erhältlich.

Besonders die Eigenschaften Ortsunabhängigkeit, Erreichbarkeit, sofortige Verfügbarkeit, Convenience und Kostengünstigkeit sind für diese Arbeit von Bedeutung. Ein schneller und einfacher Zugriff auf Fahrplandaten zu jeder Zeit und jedem Ort ist zu realisieren.

Andere Begriffe, die in diesem Zusammenhang auftauchen, sind Mobile Business und Mobile Commerce. Hier sei auf die ausführliche Literatur in [Leh03] und [KT03] verwiesen.

Kapitel 2

Grundlegende Technologien

2.1 Mobile Kommunikationsgeräte

Mobile Kommunikationsgeräte treten im geschäftlichen und privaten Alltag in vielfältiger Form in Erscheinung. Als ständiger Begleiter können sie unterwegs nützliche Informationen liefern und den Kontakt zur Außenwelt ermöglichen. In den folgenden Unterkapiteln werden die verschiedenen mobilen Kommunikationsgeräte aus technologischer Sicht eingeordnet und beschrieben.

2.1.1 PDAs

Personal Digital Assistents, auch als Handhelds, Pocket-PCs oder Organizer bezeichnet, sind akkubetriebene “Jackentaschen”-Computer mit einem Gewicht von ca. 100g - 500g. Die aktuellen Geräte verfügen über 32-Bit CPUs mit Taktraten von 16 MHz bis 400 MHz und 2 MB bis 64 MB RAM-Speicher. Sie sind mit speziellen, proprietären Betriebssystemen wie Symbian OS von Symbian, PALM OS von 3com, WINDOWS CE oder dem Nachfolger Pocket PC 2002/2003 von

Microsoft ausgestattet. Die aktuellsten und bekanntesten sind Pocket PC 2002 und PALM OS. Bei einigen Geräten ist es möglich, das Betriebssystem durch ein anderes, wie zum Beispiel Calcaria Linux7k, zu ersetzen. Für die Betriebssysteme können, mit den entsprechenden Software Development Kits(SDK)[1], Anwendungen entwickelt und nachgeladen werden. Folgende CPUs kommen in den aktuellen Geräten zum Einsatz: Intel XScale, Motorola Dragonball, TI OMAP. Mit einschiebbaren Speicherkarten (MMC (Multi Media Card)/SD- (Secure Digital Memory Card), CF-Karten (Compact Flash), Memory Sticks) ist der Speicher erweiterbar. Die PDAs kann man nach ihrer Bedienung in zwei große Gruppen einteilen: Stiftgeräte und Tastaturgeräte. Stiftgeräte werden per Touchscreen bedient und die Tastaturgeräte besitzen eine kleine Tastatur für die Eingabe. Diese zwei Hauptgruppen können nochmals in zwei Gruppen aufgeteilt werden: Geräte mit Monochrom-Displays und mit Farb-Displays. Die Displays der Stiftgeräte haben Auflösungen von 160 * 180 bis 320 * 480 Pixeln mit einer Farbtiefe bis zu 65000 Farben. Die Displays der Tastaturgeräte haben Auflösungen von bis zu 640 * 480 Pixeln. Bei einigen Geräten sind Bluetooth oder Infrarot-Schnittstellen vorhanden. Die PDAs können, je nach Gerät und Anschlussmöglichkeit, mit GSM-GPRS-Modulen oder W-LAN-Modulen ausgerüstet werden. Da bei den Anschlussmöglichkeiten kein einheitlicher Standard existiert, ist die Kompatibilität zwischen den Technologien und Geräten oft nicht gegeben.[2]

2.1.2 Handys

Mobiltelefone, im deutschsprachigen Raum auch als Handys bezeichnet, sind die bekanntesten mobilen Kommunikationsgeräte. Mobiltelefone können nicht nur zum Telefonieren genutzt werden. Mit ihnen können auch Textnachrichten

[1] vgl. [PP04]

[2] vgl. [CHI03] [Leh03, Seiten 151-153]

(Short Messaging Service) oder Multimedianachrichten (Multimedia Messaging Service) verschickt werden. Einige Handys haben eine Digitalkamera fest eingebaut oder es besteht dafür eine Erweiterung. Außerdem finden sie auch als Modem für ein PDA oder ein Notebook Verwendung. Mit Browsern kann das Handy auch zum Betrachten von WAP- Seiten (Wireless Application Protocol) oder anderen Diensten, wie I-Mode, im Internet genutzt werden. Die Geräte haben gewöhnlich eine kleine Telefon - Wähltastatur und einige Sondertasten für eine menügeführte Eingabe. Die Ausstattungen der Geräte variieren sehr stark. Es gibt einfache Handys mit Monochrom-Displays und wenig Arbeitsspeicher und Handys mit Farb-Displays und mehreren Megabyte Arbeitsspeicher. Die Displays der Geräte sind meist kleiner als die der PDAs. Als Schnittstellen zu anderen Endgeräten setzen die Hersteller auf Infrarot, Bluetooth oder Datenkabel. Als Funkschnittstelle bieten die Hersteller GMS (Global System for Mobile Communications), GPRS (General Packet Radio Service), HSCD (High Speed Circuit Switched Data), EDGE (Enhanced Data-Rates for GSM Evolution) oder UMTS (Universal Telecommunications System) an. Die Betriebssysteme der Mobiltelefone sind herstellerspezifisch. Der dominierende Betriebssystem-Anbieter ist Symbian. Andere Hersteller sind Microsoft, Palm Source und SavaJe. Nokia ist laut dem Marktforschungsinstitut Gartner mit 35,9 Prozent Marktführer unter den Mobiltelefon-Produzenten. Als Betriebssystem wird in den Nokia-Handys Symbian OS eingesetzt. Smartphones vereinen die Funktionen eines Handys und PDAs. Sie verzichten im Gegensatz zu PDAs auf große Displays und leistungsfähige Prozessoren zugunsten geringer Abmessungen und langer Akkuzeiten. Da die Funktionen der Handys und PDAs sich immer mehr angleichen, ist es schwierig, die Begriffe klar abzugrenzen. Auf [Hei04a] findet man eine große Übersicht der derzeit auf dem Markt erhältlichen Mobiltelefone und Smartphones.

2.1.3 Zusammenfassung zu den Mobilgeräten

Weitere mobile Endgeräte sind Notebooks, Sub-Notebooks, Tablet PCs und Pager. Diese kommen aber aufgrund ihrer Hardware- Ausstattung oder ihrer Größe nicht für einen sinnvollen und effektiven Einsatz einer mobilen Fahrplanauskunft in Frage. Als Zielplattformen sind PDAs und Handys aufgrund ihrer Größe und Verfügbarkeit am besten geeignet. Im weiteren Verlauf der Arbeit werden hauptsächlich Handys als mobile Endgeräte in Betracht gezogen, da sie den Massenmarkt abdecken. Die Anwendung wird aber auch auf PDAs lauffähig sein. Die Charakteristika von mobilen Endgeräten sind **eingeschränkte Displaygrößen, wenig Speicher, leistungsschwache Prozessoren, geringe Akkuleistung und eingeschränkte Eingabemöglichkeiten**(siehe Abschnitte 2.1.1 und 2.1.2).[3] Dies gilt es bei der Entwicklung von mobilen Anwendungen zu berücksichtigen.

2.2 Mobilfunknetze

2.2.1 Einführung in die Mobilfunkstandards

Mobilfunkstandards der ersten Generation (1G) sind die älteren Mobilfunknetze, die mit analoger Übertragungstechnik arbeiten. Die wichtigsten Vertreter in Deutschland waren das A-Netz, das B-Netz und das C-Netz. Der Betrieb der Netze der ersten Generation wurde in Deutschland mittlerweile eingestellt. Das A-Netz und das B-Netz boten nur einen Sprachdienst an. Das C-Netz war ein analoges Netz mit zellularem Aufbau und die Signalisierung war schon digital. Das C-Netz verfügte über einen Sprach- und einen Datendienst. Die Übertragungsgeschwindigkeit für den Datendienst betrug 2,4 kBit/s.In den Abschnitten 2.2.2 bis 2.2.6

[3] vgl. [Leh03, Seite 150]

werden die digitalen Mobilfunkstandards ab der zweiten Generation beschrieben. In Abbildung 2.1 ist die zeitliche Entwicklung der digitalen Mobilfunknetze ab der zweiten Generation dargestellt.

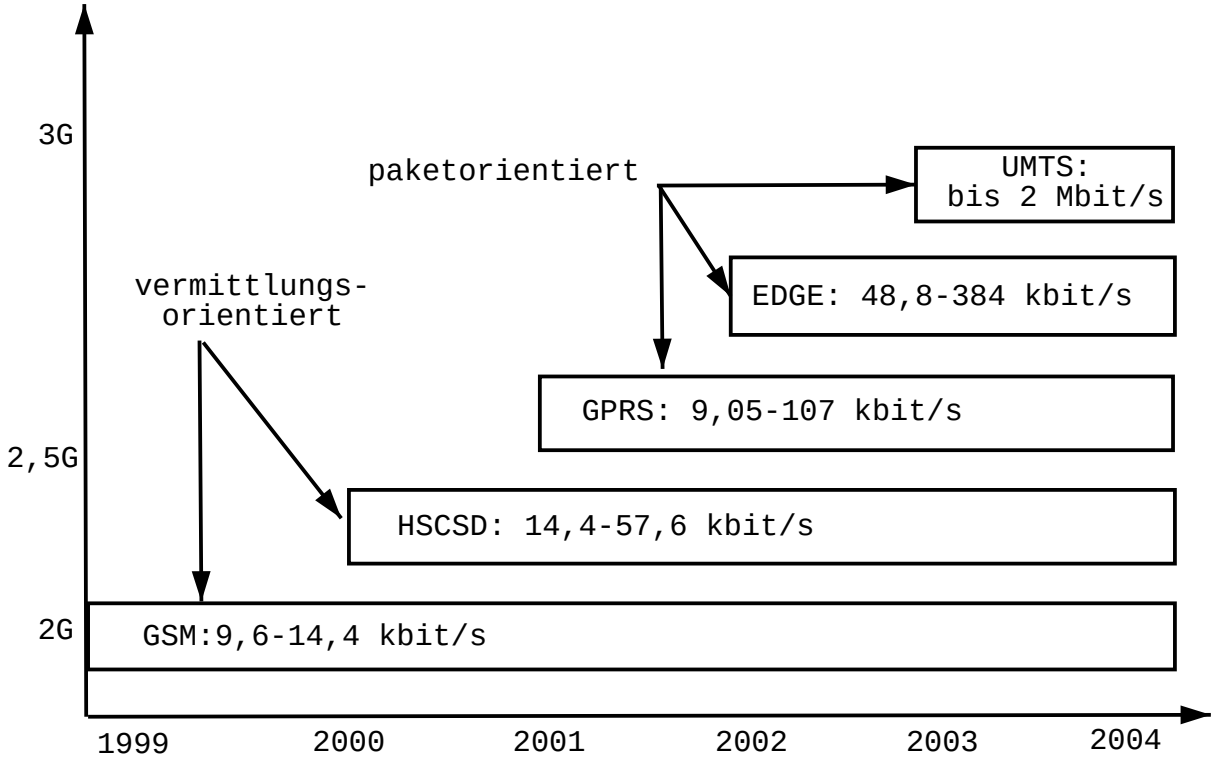

Abbildung 2.1: Zeitliche Entwicklung der digitalen Mobilfunkstandards GSM, HSCSD, GPRS und UMTS (vgl. [Leh03, Seite 41])

2.2.2 GSM

GSM ist das weltweit führende Mobilfunknetz der Welt mit 372 Netzwerken in 142 Ländern. Es ist ein zellulares Funknetz, das die zu versorgende Fläche in Funkzellen mit je einer Sende-/Empfangsstation einteilt. GSM 900 arbeitet mit gepaarten Frequenzbereichen. Für die Verbindung vom mobilen Endgerät zur Ba-

sisstation, dem Uplink, ist der Frequenzbereich zwischen 890 und 915 MHz vorgesehen. Für den Downlink, der Verbindung von der Basisstation zur Mobilstation, ist der Frequenzbereich zwischen 935 und 960 MHz vorgesehen. Die beiden Senderichtungen trennt man durch FDD (Frequency Division Duplex).[4] Die Fre-

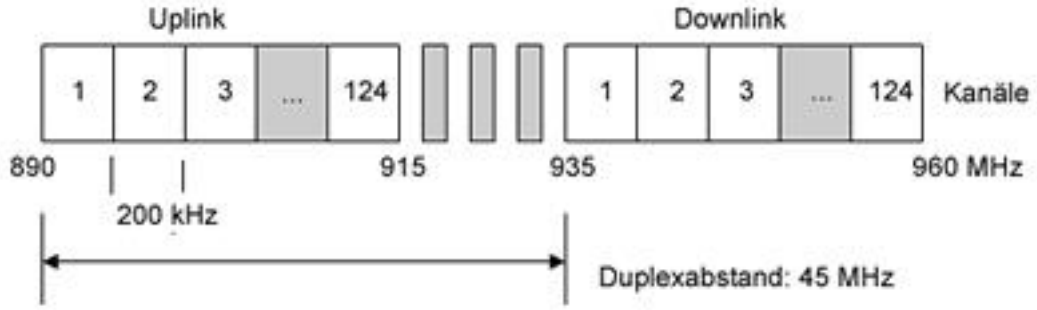

Abbildung 2.2: GSM-Frequenzbänder mit je 124 Kanälen zum Senden (Uplink) und Empfangen (Downlink) (vgl. [Leh03, Seite 32])

quenzbänder sind in Kanäle von 200kHz eingeteilt. Damit gibt es insgesamt je 124 FDM- Kanäle zum Senden und Empfangen. Abbildung 2.2 veranschaulicht dies. Der Duplexabstand beträgt 45 MHz. Daneben gibt es noch andere GSM-Systeme auf unterschiedlichen Frequenzen: GSM 400, GSM 1800/DCS 1800, GSM 1900/PCS 1900 und GSM - Rail. In Abbildung 2.3 werden diese verschiedenen GSM-Systeme verglichen. Die Dienste und Architekturen sind in allen GSM - Versionen annähernd gleich.[5] Die GSM-Architektur ist aus folgenden drei Teilsystemen aufgebaut[6]:

- Funk-Feststationssystem : Radio Subssystem (RSS)
- Vermittlungssystem : Network and Switching System (NSS)

[4] vgl. [Leh03, Seite 20]
[5] vgl. [Leh03, Seite 31]
[6] vgl. [Leh03, Seiten 35-37]

GSM-Systeme	GSM 900	GSM 1800/ DCS 1800	GSM 1900	R-GSM (Railway)	GSM 400	
Uplink [MHz]	890-915	1710-1785	1850-1910	876-915	450,4-457,6	478,8-486
Downlink [MHz]	935-960	1805-1880	1930-1990	921-960	460,4-467,6	488,8-496
Merkmale/ Einsatzgebiet	Europa, Rest der Welt	Europa, Ballungszentren	USA, Kanada	Speziell für Züge	Vorteile in der Abdeckung	

Abbildung 2.3: GSM-Systeme mit verschiedenen Uplink- und Downlink-Frequenzen (Quelle: [Leh03, Seite 33])

- Betriebs- und Wartungssystem: Operation Subsystem (OSS)

Das RSS enthält die mobilen Endgeräte (Mobile Station, MS) und die Base Station Subsystems (BSS). Die MS besteht aus einem teilnehmerunabhängigen Teil mit allen Hard- und Softwarekomponenten für die Funkschnittstelle und einem teilnehmerabhängigen Teil, dem Subscriber Identity Module (SIM). Ein GSM-Netz besteht aus vielen BSS, die wiederum aus einer Funkfeststation, der Base Tranceiver Station (BTS) und der Steuerungseinheit, dem Base Station Controller (BSC) bestehen. Das NSS stellt vermittlungstechnische Funktionen zur Verfügung und bildet ein Übergangsnetz zwischen Funk- und Festnetz. Das NSS ist aus folgenden Komponenten aufgebaut: der Mobilvermittlungsschnittstelle (Mobile Services Switching Center, MSC), der Heimatdatei (Home Location Register, HLR) und der Besucherdatei (Visitor Location register). Das MSC ist eine leistungsfähige ISDN-Vermittlungsstelle, die zusammen mit anderen MSC das “Rückgrat” des GSM-Netzes bildet. Einer MSC sind meist mehrere BSCs zugeordnet. Das OSS hat die folgenden Aufgaben: Teilnehmerverwaltung, Netz-

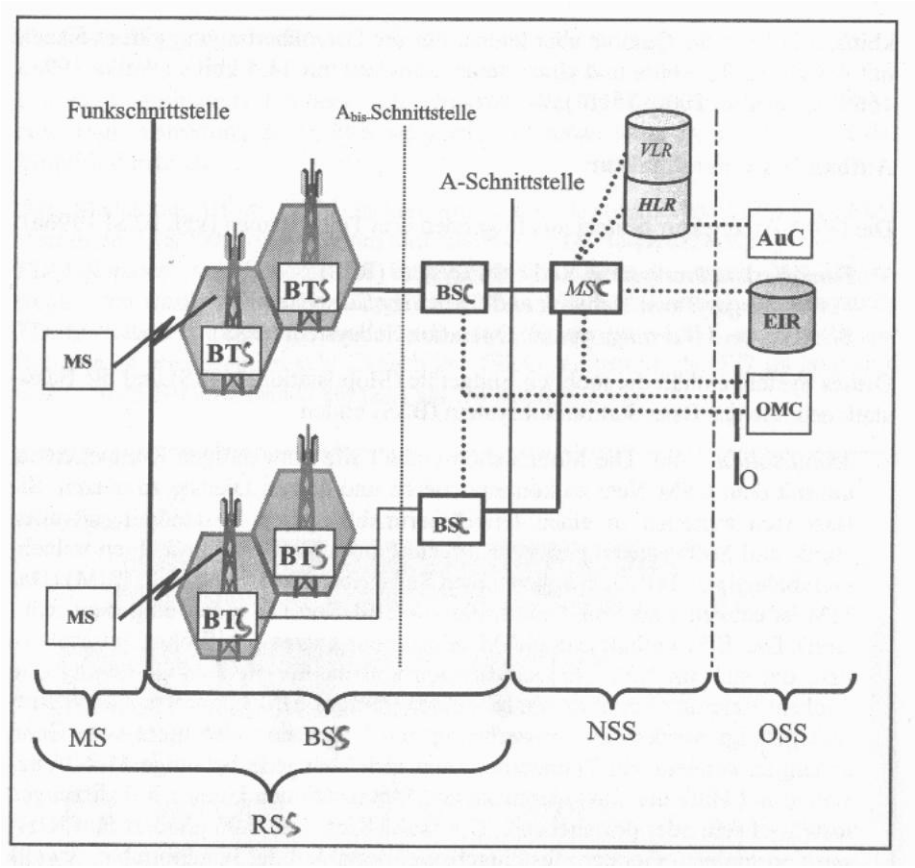

Abbildung 2.4: Funktionale Architektur von GSM mit ihren Teilsystemen Radio Subsystem(RSS), Network and Switching Subsystem (NSS), Operation Subsystem (OSS) und den dazugehörigen Komponenten(Quelle: [Leh03, Seite 36])

betrieb und -wartung und die Endgeräteverwaltung.[7] Es besteht aus dem OMC (Operation & Maintanance Centre), dem AuC (Authentication Centre) und dem EIR (Equipment Identity Register). Diese beschriebene Architektur von GSM wird in Abbildung 2.4 dargestellt. GSM stellt folgende Dienste den Teilnehmern zur Verfügung[8]: Trägerdienste (Bearer Service), Teledienste und Zusatzdienste (Supplementary Services). Trägerdienste erlauben die Zusammenarbeit

[7] vgl. [KT03]

[8] vgl. [Leh03, Seite 38]

mit ISDN, paketvermittelnden Datendiensten oder den Telefonnetzen. Die maximale Datenrate für Nichtsprachdienste beträgt 9,6 kbit/s. Teledienste bieten die Sprachübertragung, Notrufdienste und den Short Message Service (SMS). Zusatzdienste dienen der Unterstützung oder der Erweiterung. Typischerweise sind dies: Teilnehmerkennung, Rufumleitung, Rufweiterleitung oder Konferenzschaltung. In Deutschland gibt es zwei GSM900-Netzbetreiber und zwei DCS1800-Netzbetreiber.[9] GSM ist hauptsächlich auf Sprachübertragung ausgerichtet. Die auf kanalvermittelter Übertragung basierenden Datendienste haben eine maximale Bandbreite von 9,6kbit/s. Andere digitale Mobilfunksysteme der zweiten Generation sind die amerikanischen Systeme TDMA/IS-136 und CDMA/IS-95 sowie das japanische PDC.[10]

2.2.3 HSCSD

HSCSD (High Speed Circuit Switched Data) ist vollständig in GSM integriert. Änderungen ergeben sich vor allem an der Funkschnittstelle.[11] Eine Mobilstation kann mehrere Kanäle innerhalb eines 200 kHz- FDM- Kanal anfordern. Bei einer Kanalbündelung von bis zu acht Kanälen pro TDMA-Rahmen[12] erreicht man eine theoretische Bandbreite von 76,8 kbit/s. Durch den Verzicht auf einen Teil der Fehlerkorrektur kann der Datendurchsatz von einem Verkehrskanal auf 14,4 kbit/s erhöht werden. Dies führt zu theoretischen Datenraten von 115,2 kbit/s, wie aus Tabelle 2.1 ersichtlich ist. ETSI spezifizierte allerdings die Obergrenze bei vier Zeitschlitzen und daraus resultierenden Datenraten von 38,4 kbit/s bzw. 57,6 kbit/s. Die Verbindung erfolgt bei HSCSD vermittlungsorientiert, d.h. ein bzw.

[9] vgl. [Leh03, Seiten 39-40]
[10] vgl. [Leh03, Seite 31]
[11] vgl. [Leh03, Seite 41]
[12] vgl. [Leh03, Seite 21]

Tabelle 2.1: Verschiedene Datenraten bei HSCSD durch Bündelung von 1,4 und 8 Kanälen, vgl. [Leh03, Seite 42]

Kanalbündelung	1 Zeitschlitz	4 Zeitschlitze	8 Zeitschlitze
9,6 kbit/s pro Kanal	9,6 kbit/s	38,4 kbit/s	76,8 kbit/s
14,4 kbit/s pro Kanal	14,4 kbit/s	57,6 kbit/s	115,2 kbit/s

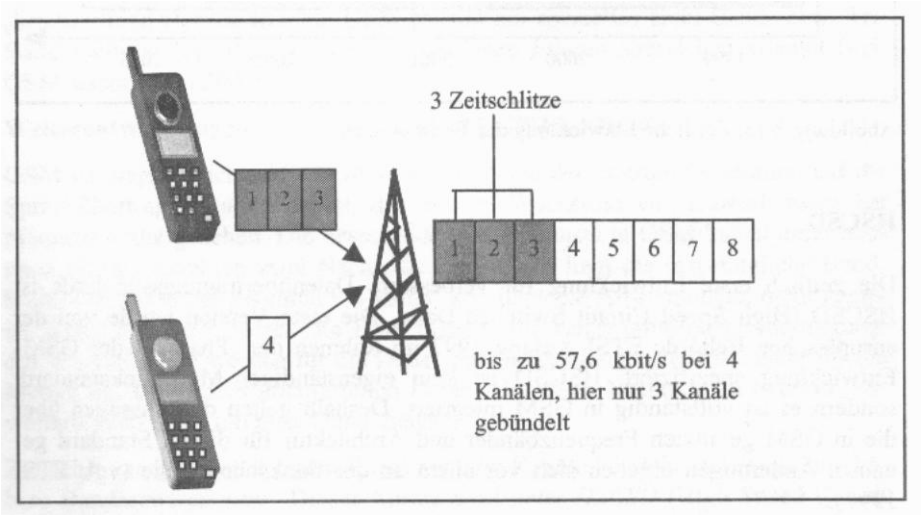

Abbildung 2.5: Funktionsweise von HSCSD - Endgerät bündelt 3 Kanäle (Quelle: [Leh03, Seite 42])

mehrere Kanäle werden exklusiv von einem Teilnehmer benutzt und während der Verbindungsdauer aufrechterhalten. Die Funktionsweise von HSCSD ist in Abbildung 2.5 dargestellt.Ein Vorteil von HSCSD sind die geringen Kosten für den Netzbetreiber bei der Implementierung der Technik. Es ist im Wesentlichen nur eine Splitting/Combining- Funktion in der MS und dem MSC nötig. In Deutschland wird HSCSD von E-Plus und Vodafone mit Datenübertragungsraten bis zu 38.4 kbit/s angeboten. Um HSCSD zu nutzen, benötigt man ein HSCSD - fähiges Mobilgerät.[13] HSCSD ist damit für Datendienste oder Internetdienste mit akzep-

[13]vgl. [Leh03, Seiten 41-44]

tablen Geschwindigkeiten geeignet.

2.2.4 GPRS

Mit dem General Packet Radio Service (GPRS) wird ein paketorientierter Datendienst in den bestehenden GSM-Netzen angeboten. Damit können die Funkkanäle gegenüber der verbindungsorientierten Datenübertragung effizienter genutzt werden. Ein weiterer Vorteil ist, dass GPRS einen schnellen Verbindungsaufbau ermöglicht und der Nutzer ständig “online” ist, ohne dauerhaft eine Funkverbindung zu belegen.[14] Der Vorteil von GPRS liegt darin, dass ein Mobilfunknutzer stets mit dem Mobilfunknetz über Signalisierungskanäle verbunden bzw. eingewählt bleibt (“always on”). Es muss keine Verbindung ständig neu aufgebaut werden.[15] Die Belegung der Zeitschlitze wird bedarfsgerecht auf die Nutzer vergeben. Die Datenströme werden in Pakete zerlegt, jedes mit eigener Zieladresse, und über verschiedene Kanäle ins Netz gesendet, die beim Empfänger dann wieder in die richtige Reihenfolge gebracht werden. Das Netz wird nur benutzt, wenn Daten zu übertragen sind, ansonsten werden die freien Kapazitäten anderen zur Verfügung gestellt. Mehrere Nutzer in einer Mobilfunkzelle teilen sich die zur Verfügung gestellten Kapazitäten.[16] Es ist keine Höchstgrenze für die Datenraten vorhergesehen. Die Datenraten werden durch die technische Realisierung beschränkt. In den GPRS - Spezifikationen gibt es vier verschiedenartige Kodierschemata, die Coding-Schemes CS-1 bis CS-4, die jeweils bestimmte Datenraten zur Verfügung stellen. Die Coding-Schemes werden in Tabelle 2.2 aufgezählt. Die verschiedenen Übertragungsraten werden durch die Anwendung unterschiedlich ausgedehnter Fehlerkorrekturverfahren erreicht. Dabei verfügt CS-4 über keiner-

[14] vgl. [www04a]

[15] vgl. [Leh03, Seite 46]

[16] vgl. [Leh03, Seite 45]

lei Fehlerkorrektur.[17] CS-3 und CS-4 sind für die Netzbetreiber komplex zu rea-

Tabelle 2.2: Coding-Schemes der GPRS-Spezifikationen mit den dazugehörigen Datenraten bei 8 Zeitschlitzen, vgl. [Leh03, Seite 47]

Coding-Scheme	Datenrate [kbit/s]	Maximale Datenrate bei 8 Zeitschlitzen [kbit/s]
CS-1	9,05	72,4
CS-2	13,40	107,2
CS-3	15,60	124,8
CS-4	21,40	171,2

lisieren und zudem kostenintensiv, da größere Änderungen an der Infrastruktur nötig sind.[18]

GPRS setzt auf die GSM-Technik auf. Dabei werden die bisherigen kanalvermittelten Dienste nicht ersetzt, sondern ergänzt. GSM und GPRS verwenden dieselben Komponenten für das BSS. Zusätzlich müssen drei neue Netzkomponenten integriert werden: der Serving GPRS Support Node (SGSN), der Gateway GPRS Support Node (GGSN) und das GPRS- Register(GR). Außerdem muss der BSC mit einer Packet Control Unit nachgerüstet werden.[19] Die Architektur von GPRS ist in Abbildung 2.6 dargestellt. Die GPRS- fähigen Endgeräte werden durch zwei Merkmale charakterisiert: durch die Geräteklasse und durch die Multislot-Klasse. Dabei gibt es drei verschiedene Geräteklassen: Mit der Geräteklasse A kann gleichzeitig GSM und GPRS genutzt werden. Geräteklasse B unterstützt GSM und GPRS, aber nicht gleichzeitig, sondern sequentiell. Bei der Geräteklasse C muss der Benutzer manuell den GSM- oder GPRS- Modus auswählen.[20] Multislot- Klassen geben eine bestimmte Kombination von Zeitschlitzen im UP-

[17] vgl. [Leh03, Seite 46]
[18] vgl. [Leh03, Seite 47]
[19] vgl. [Leh03, Seite 47]
[20] vgl. [Leh03, Seite 49]

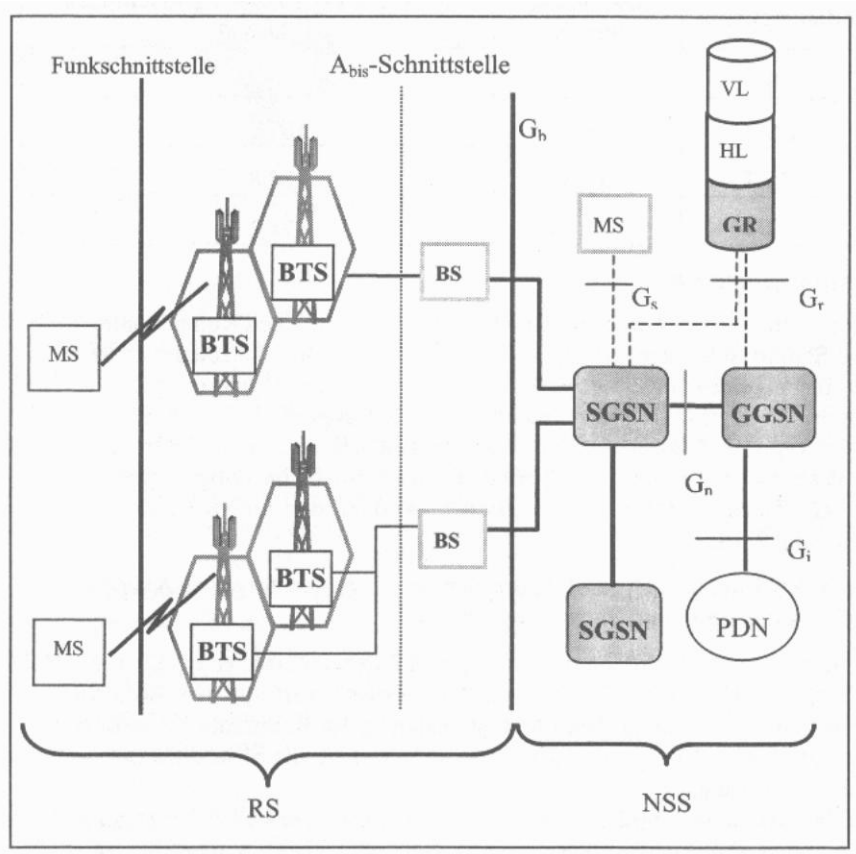

Abbildung 2.6: Architektur von GPRS mit den Teilsystemen und ihren Komponenten(Quelle: [Leh03, Seite 48])

und Downlink an, welche von dem Endgerät maximal benutzt werden kann. Insgesamt gibt es 29 Multislot-Klassen.[21]

Die Abrechnung erfolgt bei GPRS volumenabhängig, d.h. es fallen nur Gebühren für tatsächlich übertragene Daten an. In Deutschland wird GPRS von allen vier Netzbetreibern angeboten. Mit GPRS sind derzeit Datenraten von 40 - 56 kbit/s möglich.[22]

[21] vgl. [Leh03, Seite 49]

[22] vgl. [Leh03, Seite 50]

2.2.5 EDGE

Die Grundidee von EDGE (Enhaced Data Rates for Global Evolution) ist die Leistungsmerkmale eines 3G-Netzes so gut wie möglich zu realisieren, ohne ein neues Netz aufzubauen.[23] Dabei bietet EDGE die bestehenden Datendienste mit höherer Bandbreite an: das leitungsvermittelte ECSD (Enhanced CSD) und das paketvermittelte EGPRS (Enhanced GPRS). EDGE stellt zusätzliche Kapazitäten in den existierenden Frequenzen zur Verfügung. Es basiert also auf denselben 200-kHz-FDM-Kanälen der bisherigen GSM-Systeme und kann somit die Strukturen von GSM wieder verwenden. Um die Datenraten zu steigern, werden fortgeschrittenere Modulationstechniken im Vergleich zu GSM eingesetzt. Statt der bisherigen GMSK-Modulation wird die so genannte 8-PSK-Modulation verwendet.[24]Mit dieser Modulation kann man Bruttodatenraten pro Zeitschlitz auf ca. 69,2 kbit/s erhöhen. Dies ergibt bei acht Zeitschlitzen eine Bruttobitrate von ca. 554 kbit/s. Die maximale Nettodatenrate pro Zeitschlitz bei EGPRS liegt aber bei 48 kbit/s und somit für acht Zeitschlitze bei 384 kbit/s.[25] Es werden darauf die von GPRS her bekannten Verfahren angewendet. Abhängig von der Kanalqualität, existiert für ECSD und EGPRS je ein Satz verschiedener Codierungsverfahren. Die veränderten Modulationsverfahren machen Hardware-Änderungen bis auf BTS-Ebene erforderlich. Ansonsten wird für die meisten Komponenten nur ein Software-Update benötigt. Der Vorteil gegenüber GPRS ist die höhere Datenübertragungsrate.[26]

Bei GPRS und EDGE spricht man von 2.5G-Netzen. Mit diesen beiden Mobilstandards erreicht man Datenraten, die für mobile Informationsdienste

[23] vgl. [KT03]
[24] vgl. [Sch03, Seiten 82f.,341f.]
[25] vgl. [Leh03, Seite 51]
[26] vgl. [Leh03, Seiten 51-54]

akzeptabel sind.

2.2.6 UMTS

UMTS ist eine Weiterentwicklung von 2G-Mobilfunksystemen, die auf GSM beruhen.[27]Es ist ein Mobilfunkstandard der 3. Generation (3G). Mit UMTS sollen echtzeitfähige Dienste, paket- und leitungsorientierte Datenübertragung sowie unterschiedliche Datenraten bis zu 2 Mbit/s angeboten werden. Es baut auf einem klassischen Zellularnetz auf, dessen Zellen hierarchisch organisiert sind. Dabei unterscheidet man nach [KT03, Seite 46] folgende Zellkonzepte (siehe auch Abbildung 2.7):

- Makrozelle (300m - 10km): Für Nutzer mit hoher Mobilität (bis 500km/h) können mittels FDD Übertragungsraten bis 144 kbit/s bereitgestellt werden.
- Mikrozelle (100m - 300 m): Für Nutzer mit mittlerer Mobilität (bis 120km/h) können mittels TDD oder FDD Übertragungsraten bis 384 kbit/s bereitgestellt werden.
- Pikozelle (bis 100m): Für Nutzer mit geringer Mobilität (10 km/h) können mittels TDD Übertragungsraten bis 2048 kbit/s bereitgestellt werden.

Folgende Frequenzbereiche sind für UMTS definiert:
1920-1980 MHz und 2110-2170 MHz (paired), 1900-1920 MHz und 2010-2025 MHz unpaired. Für die Satellitenkomponenten sind 60 MHz in den Frequenzbändern von 1980-2010 MHz und 2170-2200 MHz vorgesehen. Zusätzlich definiert und derzeit noch anderweitig belegt sind die Frequenzbereiche 806-960 MHz, 1710-1885 MHz und 2500-2690 MHz.[28] Zwei Konzepte für die Übert-

[27]vgl. [Leh03, Seite 65]
[28]vgl. [Leh03, Seite 66]

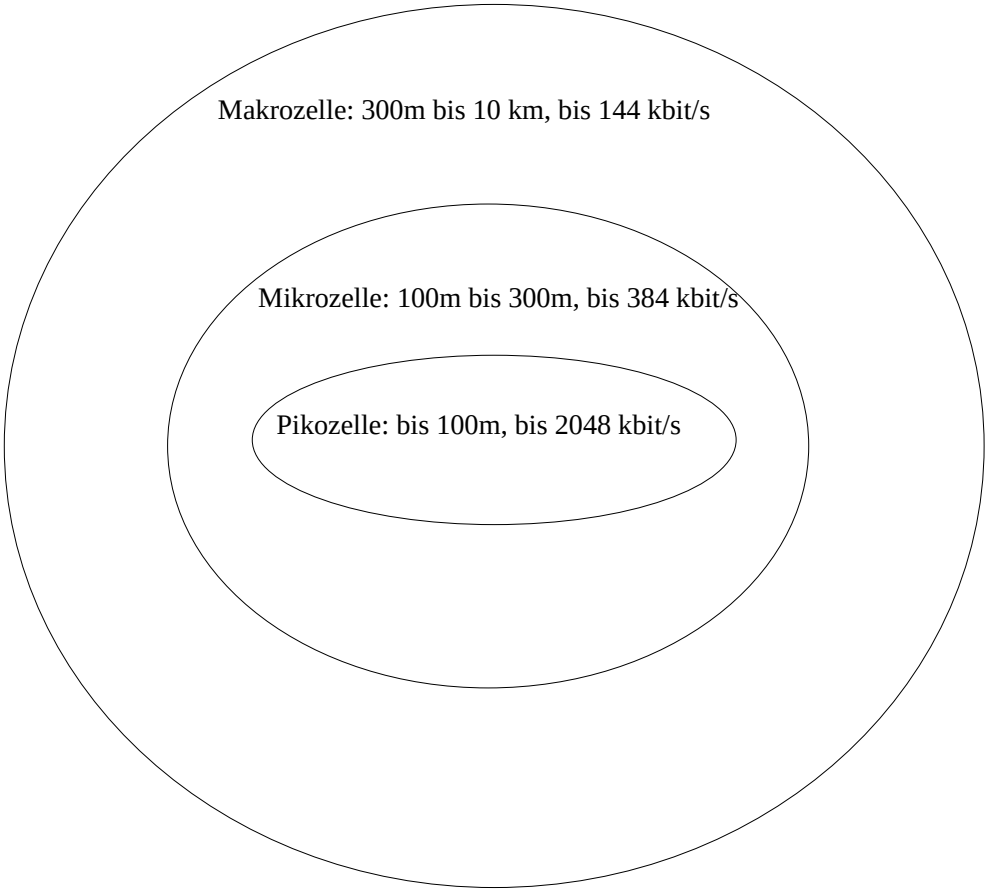

Abbildung 2.7: Zellaufbau von UMTS mit Makrozelle, Mikrozelle und Pikozelle und deren Reichweiten

ragungstechnik an der Luftschnittstelle, dem UMTS Terrestrial Radio Access (UTRA) wurden adaptiert, da man sich nicht auf einen einzigen Standard einigen konnte. Die beiden Zugriffsverfahren sind W-CDMA für das gepaarte Frequenzspektrum und eine Kombination aus TDMA (Time Division Multiple Access) und CDMA (Code Division Multiple Access) (TD-CDMA) für das ungepaarte Frequenzspektrum.[29] Die beiden entsprechenden Standards sind UTRA-FDD-Modus und UTRA-TDD-Modus (Frequency Division Duplex, Time Divi-

[29] vgl. [Leh03, Seite 67]

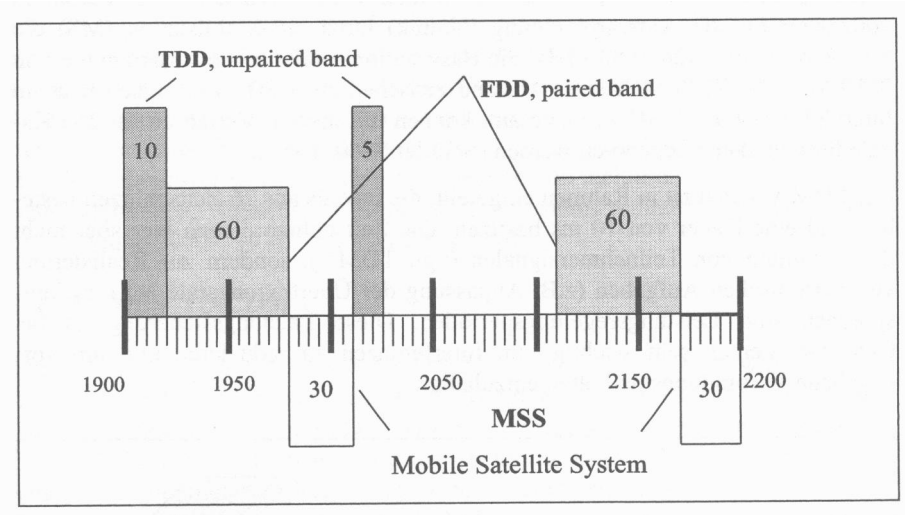

Abbildung 2.8: Frequenzbänder für UMTS mit Frequency Division Duplex(FDD) im gepaarten Spektrum und Time Division Duplex(TDD) im ungepaarten Spektrum (Quelle: [Leh03, Seite 67])

sion Duplex). Diese Standards können auch nebeneinander existieren. Eine Verbindungsübergabe zwischen den Verfahren ist ebenso möglich. Der UTRA-FDD-Modus wird im gepaarten Frequenzspektrum eingesetzt,und zwar mit FDD zur Richtungstrennung und W-CDMA (Wideband-CDMA) als Zugriffsverfahren. Die Frequenzbänder für die beiden Verfahren werden in Abbildung 2.8 illustriert. Im ungepaarten Frequenzspektrum wird der UTRA-TDD-Modus eingesetzt mit dem TDD- und dem TD-CDMA-Verfahren. Näheres zu den Technologien befindet sich in [Leh03]. Die vereinfachte UMTS- Referenzarchitektur besteht aus dem mobilen Endgerät (User Equipment, UE), dem Zugangsnetzwerk UTRAN (UMTS Terrestrial Radio Access Network) und dem Kernnetz (Core Network, CN). Abbildung 2.9 zeigt diese Komponenten. Das UE besteht aus dem mobilen Endgerät (Mobile Equipment, ME) und dem USIM (UMTS Subscriber Identity Module). Für die mobilen Endgeräte existieren vier verschiedene UMTS- Leistungsklassen mit unterschiedlichen Sendeleistungen: Leistungsklasse 1: bis 2 W,

Abbildung 2.9: UMTS-Architektur mit User Equipment (UE), UMTS Terrestrial Radio Access (UTRAN) und Core Network (CN) (Quelle: [Bra04])

Leistungsklasse 2: bis 0,5 W, Leistungsklasse 3: bis 0,25 W und die Leistungsklasse 4: bis 0,125 W). Die USIM- Karte verfügt gegenüber der GSM-SIM-Karte über eine erhöhte Speicherkapazität und eine Präzisierung der Zugriffsbedingungen. [30] Das UTRAN bildet die Schnittstelle zwischen dem UE und dem CN. Es umfasst mehrere Funksubsysteme (Radio Network Subsystem, RNS). Das RNS besteht wiederum aus einem Radio Network Controller (RNC) und mehreren Node-B-Komponenten. Die Node-B Komponenten unterstützen entweder den FDD- Modus, den TDD- Modus oder beide. Das CN besteht aus einer logischen Einheit für kanalvermittelte Dienste (Circuit Switched Domain, CSD) und einer Einheit für paketvermittelte Dienste (Packet Switched Domain). In der CSD entsprechen die Elemente im Wesentlichen denen des GSM-Netzes (MSC, VLR etc.) und in der PSD entsprechen die Elemente denen des GPRS-Netzes (SGSN, GGSN etc.). Daher können Bestandteile der existierenden Netze mit bestimmten Erweiterungen wieder verwendet werden.[31] In UMTS werden verschiedene Dienstarten unterschieden: Trägerdienste, Teledienste, Zusatzdienste und Mehrwertdienste.[32] Trägerdienste ermöglichen sowohl Sprach- als auch Datendienste.

[30] vgl. [KT03, Seite 41]
[31] vgl. [Leh03, Seite 71]
[32] vgl. [Leh03, Seiten 71-72]

Neben leitungsvermittelten werden ebenso paketvermittelte Dienste mit verschiedenen Übertragungsraten angeboten. Die Übertragungsdienste reichen von 16 kbit/s für Sprachdienste bis max. 2 Mbit/s für Datendienste. Folgende Teledienste werden unter anderen von UMTS unterstützt: Telephonie, Videotelephonie, Videoübertragung, Paging, Telefax, Kurznachrichten, Notruf. Zusatzdienste sind ähnlich wie im GSM-Netz Nummernidentifikation, Rufumleitung, Rufweiterleitung usw. Mehrwertdienste sind zum Beispiel Electronic Banking, Datenbank-Abfragen oder Verkehrsinformationen. Diese sind unabhängig von Netz und Endgerät nutzbar und können auch von Dritten angeboten werden.[33] Die Vorteile der 3G-Systeme liegen in den hohen Übertragungsraten, die neue Anwendungsfelder ermöglichen und bisherige Dienste erheblich verbessern werden. Vodafone, E-Plus, O2 und T-Online haben den Start von UMTS in Deutschland ins Jahr 2004 verschoben. Andere Anbieter die UMTS- Lizenzen ersteigert haben sind noch Quam und MobilCom.[34] Für die Nutzung werden UMTS- fähige Endgeräte benötigt.

Schon mit 2G-Systemen stehen Mobilfunkstandards mit akzeptablen Übertragungsraten zur Verfügung, die die Realisierung von mobilen Informationsdiensten ermöglichen.

2.3 Drahtlose Kommunikation

2.3.1 WLAN

Wireless LAN (WLAN) ist eine mobile elektronische Kommunikationstechnik auf lokaler Ebene. WLAN bezeichnet ein drahtloses lokales Netz nach einer der

[33] vgl. [Leh03] und [Wal00]

[34] http://www.heise.de/mobil/newsticker/meldung/39973

IEEE 802.11 Spezifikationen. Die Spezifikationen mit ihren maximalen Bandbreiten sind in Tabelle 2.3 aufgelistet. Der marktbeherrschende Standard ist IEEE 802.11b mit maximalen Datenraten von bis zu 11 MBit/s im lizenzfreien 2,4 GHz-Band. Die Sendeleistung beträgt 100 mW mit Reichweiten zwischen 30 Meter innerhalb und 300 Meter außerhalb von Gebäuden. Mit Hilfe von gerichteten Antennen können Reichweiten von einigen Kilometern erreicht werden. WLAN ermöglicht die Vernetzung im Infrastruktur-Modus und im Ad-hoc-Modus. Im Infrastruktur-Modus wird ein sternförmiges LAN um einen Funk-Zugangsknoten errichtet (Access-Point). Ad-hoc-Modus bedeutet eine Peer-to-Peer Vernetzung von Endgeräten untereinander. Von Bedeutung sind derzeit WLANs bei der Ermöglichung von Internet- oder Intranetzugängen. Typischerweise sind die WLAN-Module im PCMCIA- oder CF-Format erhältlich. Bei Notebooks oder PDAs sind die WLAN- Module häufig integriert oder werden als Erweiterung angeboten.[35] Da bei Handys die WLAN- Technologie keine allzu große Rolle spielt, wird in dieser Arbeit nicht weiter darauf eingegangen. Prinzipiell ist jedoch der Aufbau eines Informationsdienstes mittels der WLAN-Technologie möglich.

Tabelle 2.3: Übertragungsstandards der IEEE 802.11-Spezifikationen und die entsprechenden maximalen Bandbreiten, vgl. [KT03, Seite 49]

Standard	Band	Max.Bandbreite
802.11	2,4 GHz	2 MBit/s
802.11a	5 GHz	54 MBit/s
802.11b	2,4 GHz	11 MBit/s
802.11g	2,4 GHz	54 MBit/s

[35] vgl. [KT03, Seiten 49-52],

2.3.2 Bluetooth

Mit Bluetooth werden Endgeräte miteinander oder mit Peripheriegeräten vernetzt. Bluetooth ist eine Funktechnik, die zur Übertragung den Bereich zwischen 2402 und 2480 MHz nutzt. Es gibt drei Klassen mit maximalen Sendeleistungen von 1 mW, 10 mW und 100 mW und damit maximalen Reichweiten je nach Klasse von 10 Metern, 50 Metern und 100 Metern. Innerhalb dieser Werte erreicht man Datenübertragungsraten von bis zu 1 MBit/s. Als Modulation wird die Gaussian Frequency Shift Keying (GFSK) verwendet. Die Bluetooth- Netto- Datenraten betragen synchron 64 kBit/s für Sprachübertragung. Asynchron gibt es zwei Varianten. In der asymmetrischen Variante sind es 721 kBit/s in eine Richtung und in der anderen 57,6 kBit/s. In der symmetrischen Variante sind es in beide Richtungen 432,6 kBit/s. Bluetooth- Verbindungen können paket- und leitungsorientiert sein. Ein Vorteil gegenüber IrDa ist, dass Bluetooth- Geräte ohne direkten Sichtkontakt miteinander kommunizieren können. Die Spezifikation beschreibt Kurzstreckenverbindungen, die mobile und stationäre abhörgesicherte Verbindungen ad hoc aufbauen kann. Ad-hoc-Netze sind selbstorganisierende Netze mit Komponenten, die sich logisch zusammenschließen können. Die Endgeräte können direkt und spontan miteinander kommunizieren. Bei Bluetooth erklärt sich ein Gerät zum Master und bietet anderen Geräten, bis zu 7 Slaves, innerhalb der Funkreichweite eine sternförmige Vernetzung. Dieses entstehende Netz wird als Piconet bezeichnet. Mehrere Piconets bilden ein Scatternet. Dazu muss ein Gerät an mehren Piconets teilnehmen. Ein Endgerät kann in einem Piconet als Master und in einem anderen als Slave agieren.[36]Eine Reihe von Handys und PDAs sind mit Bluetooth ausgestattet. Beispielsweise wäre es möglich einen Informationsdienst zu realisieren, in dem Bluetooth zur Datenübertragung genutzt wird. Das mobile Endgerät lädt die Fahrplandaten von entsprechenden Stationen mittels Bluetooth.

[36]vgl. [KT03, Seite 52],[Ziv04]

2.3.3 IrDa

IrDA ist ein Standard zur Datenübertragung mit Hilfe von Infrarotlicht. Die Infrarotschnittstellen müssen zur Übertragung aufeinander ausgerichtet sein und eine Sichtlinie zwischen den Geräten ist erforderlich. Die Übertragung kann durch äußere Einflüsse wie Lichtquellen, Streuung oder Reflexion gestört werden. Die Reichweite zur Übertragung beträgt 1-2 Meter. Es existieren mehrere Standards. Die Übertragungsrate bei Serial Infrared (SIR), dem Standard IrDa 1.0, beträgt 115,2 kBit/s. Fast Infrared (FIR), IrDa 1.1, bietet eine Übertragungsrate von bis zu 4 MBit/s und mit Very Fast Infrared (VFIR), IrDa 1.2, sind Übertragungsraten bis zu 16 MBit/s möglich. Typische Einsatzszenarien sind 1:1 Verbindungen zwischen Endgeräten, aber auch eine Vernetzung zwischen mehreren Endgeräten ist möglich.[37] Die IrDa- Technologie kommt bei Handys, Handheld-Computern und Notebooks zum Einsatz. IrDa kann auch zum Aufbau eines Informationsdienstes genutzt werden indem ein Endgerät die Daten über die IrDa-Schnittstelle von einem anderen Endgerät abfragt.

2.4 Dienste für Mobilgeräte

In den folgenden Kapiteln werden Dienste, die für einen mobilen Informationsdienst in Frage kommen, näher betrachtet und auf ihre Tauglichkeit hin überprüft. Es gibt nach [KT03, Seite 85] zwei Möglichkeiten der Realisierung von mobilen Anwendungen: die Verwendung serverseitiger und clientseitiger Techniken. Diese werden nochmals in zwei Kategorien unterteilt: Pull- und Pushdienste. Bei Pull-Diensten geht die Datenübertragung vom Client aus. Der Client sendet eine Anfrage an einen Server (Request) und bekommt eine Antwort mit den gewünschten In-

[37] vgl. [KT03, Seite 54],[wc04]

formationen (Response). Bei Push-Diensten geht die Datenübertragung vom Server aus. Dieser sendet ohne Anforderung die Daten an den Client. Dieser Dienst kann aber vom Nutzer initiiert und kofiguriert werden.

2.4.1 SMS

SMS (Short Message Service) ist ein GSM-Dienst zum Versenden und Empfangen von Kurznachrichten. Die Nachrichten bestehen aus bis zu 140 Byte. Damit können Nachrichten mit einer Länge von bis zu 160 alphanumerischen Zeichen verschickt werden. Diese Grenze kann auch überschritten werden, indem eine längere Nachricht in mehrere Einzelnachrichten aufgeteilt wird. SMS ist ein Store-and-Forward Dienst, der bei Nichterreichbarkeit des Empfängers die Nachricht zwischenspeichert und erst bei Erreichbarkeit des Empfängers die Zustellung vornimmt. Bei SMS Point-to-Point wird eine Nachricht an einen einzelnen Empfänger gesendet und bei SMS Cell Broadcast (SMS CB) wird eine Nachricht an alle Empfänger in einer, mehreren oder allen Zellen eines Netzes versendet. Um SMS nutzen zu können ist eine Kurznachrichtenzentrale (Short Message Service Center, SMSC) erforderlich. SMSC ist eine Anwendung, die meist auf Standard-Serverplattformen wie UNIX läuft. Die Aufgaben der SMSC sind die Annahme und Weiterleitung von SMS an die MSC, Zwischenspeicherung und spätere Zustellung und der Versand von Nachrichten, die von Netzdiensten generiert werden. Der Netzbetreiber verfügt über ein oder mehrere SMSC. SMS kann für Informationsdienste verwendet werden. Dabei unterscheidet man zwischen Pull- und Pushdiensten. Bei Pull-Diensten sendet der Nutzer manuell eine SMS, um einen Dienst anzufordern und die Anwendung antwortet mit einer SMS, die die gewünschten Informationen enthält. Bei Push-Diensten werden SMS ohne Anfor-

derung an die Nutzer gesendet.[38] Auf diese Weise können Informationssysteme mit SMS verwirklicht werden. Beispiele von Informationsdiensten auf SMS-Basis sind News-Meldungen, Sportmeldungen, Staumeldungen, usw. HAFAS-SMS ist zum Beispiel eine Fahrplanauskunft mit SMS.[39] Eine Fahrplanauskunft auf SMS-Basis ist folgendermaßen zu realisieren: der Nutzer sendet eine SMS mit dem Startort (Haltestelle), der Linienbezeichnung und der Uhrzeit an einen Server. Diese Daten werden von dem Server verarbeitet und die gewünschten Informationen von einem Datenbankserver abgefragt und an das Handy per SMS zurückgeschickt. SMS sind geräteunabhängig. Die Nachteile von SMS für Informationsdienste sind die umständliche und nutzerunfreundliche Eingabe, die beschränkten Ausgabemöglichkeiten und es sind kaum Interaktionen möglich. Für einfache Anwendungen ist ein Informationsdienst auf SMS-Basis tauglich. Für eine Fahrplanauskunft, aufgrund der umständlichen Eingabe, ist es aber weniger geeignet. Außerdem kommen noch die entstehenden Kosten je gesendeter Nachricht hinzu.

2.4.2 USSD

Unstructured Supplementary Service Data (USSD) ist ebenfalls ein GSM-Dienst zum Versand von Kurznachrichten. Mit USSD ist es möglich 182 Zeichen zu je 7 Bit pro Nachricht zu verschicken. Bei USSD wird jedoch für die Übermittlung eine Verbindung aufgebaut. Damit ist unmittelbar und exakt nachvollziehbar, ob und wann die Nachricht vollständig übertragen wurde. Dadurch werden vollständige Transaktionen garantiert.[40] Ansonsten gelten die gleichen Nachteile für die Realisierung von Informationsdiensten mit USSD wie bei SMS.

[38] vgl. [KT03, Seiten 86-87]
[39] vgl. [hac04]
[40] vgl. [KT03, Seite 88]

2.4.3 MMS

Mit Multimedia Messaging Service (MMS) werden Nachrichten mit integrierten Bild-, Video und Audiodateien versendet. Dieser Dienst wird in GPRS- und UMTS-Netzen als Erweiterung und zusätzlich zum SMS angeboten. Außerdem ist der Versand an beliebige E-Mail-Adressen möglich. Zur Nutzung von MMS sind zusätzlich zum SMSC noch Multimedia Message Service Center (MMSC) nötig, die die Annahme, Weiterleitung, Zwischenspeicherung und spätere Zustellungen von MMS durchführen. Eine weitere Aufgabe der MMSC besteht darin, vor der Weiterleitung, Kompatibilitätstests mit den Endgeräten durchzuführen, da der Dienst in den Endgeräten sowie den Netzen unterschiedlich implementiert ist. Außerdem werden die SMTP- Funktionalität für den Versand ins Internet sowie optional Auslieferungs- und Lesebestätigungen bereitgestellt. Es gibt grundsätzlich keine Größenbeschränkung für MMS, nur durch Endgerät, MMSC und Gateways. Prinzipiell ist MMS unabhängig vom Trägermedium und kann damit auch per Bluetooth oder W-Lan angeboten werden.[41] Es können dieselben Informationsdienste wie mit SMS realisiert werden, aber mit völlig anderen Qualitäten bei der Ausgabe. Beispielsweise bietet MMS die Möglichkeit einen Haltestellenfahrplan als Grafik zum Nutzer zu senden. Die Nachteile der umständlichen Eingabe und der beschränkten Interaktion bleiben jedoch weiterhin bestehen. Damit ist das Erstellen einer komfortablen Fahrplanauskunft mit MMS ebenfalls weniger geeignet. Hinzu kommen noch die Inkompatibilitäten der Endgeräte und der Netze sowie die entstehenden Kosten pro Versand.

[41] vgl. [KT03, Seite 88],[Dat04],[tel04]

2.4.4 HTTP und HTML

HTTP hat auch für mobile Anwendungen eine zentrale Bedeutung. Es stellt die Grundlage für die Übertragung von Daten zwischen Client und Server bei HTML-Browsern und WML-Browsern dar. Der Client nimmt Verbindung mit einem Server auf, fordert von diesem bestimmte Daten und der Server sendet die gewünschten Daten zum Client und schließt die Verbindung wieder. Für die Regelung des Ablaufs der Übertragung wird das Transmission Control Protocol/Internet Protocol (TCP/IP) verwendet. TCP/IP ist unabhängig vom Trägermedium und es kann daher auch im mobilen Bereich genutzt werden. Da es für verkabelte Netze entworfen wurde, kann es im drahtlosen Bereich zu starken Leistungseinbußen führen. Im Vergleich zu verkabelten Netzen führt das zu einer höheren Anzahl verlorener Datenpakete. Die Übertragung der Daten im ASCII-Format ist ebenfalls nicht effizient, da diese unkomprimiert übertragen werden.[42] HTTP ohne TCP/IP kann als Protokoll für eine mobile Anwendung verwendet werden. Wenn ein mobile Client keinen TCP/IP - Stack verwendet, dann ist dennoch eine Anbindung an einen HTTP-Server möglich. Dabei wird der Unterbau des HTTP - Protokolls im Mobilfunknetz von einem Gateway durch einen TCP/IP - Stack ausgetauscht. Ein Programmierer einer Anwendung kann Verbindungen auf Basis von HTTP herstellen, ohne sich um den Unterbau des Protokolls zu kümmern.[43] Auch für den mobilen Zugriff kann die Hypertext Markup Language (HTML) genutzt werden. Dazu muss im Endgerät ein entsprechender HTML-Browser integriert sein. Es existieren Browser, z.B. Opera, die den Inhalt der HTML-Seiten auf die Display-Größen der mobilen Endgeräte anpassen.[44] Da Standard - HTML nicht für den mobilen Bereich konzipiert wurde, ist vom Aufbau eines mobilen

[42] vgl. [Tan03],[Leh03, Seiten 140-141]
[43] vgl. [SR03, Seiten 146-147]
[44] vgl. [MB04]

Informationsdienstes mittels dieser Technologien abzuraten. Ein Vorteil ist, dass die bereits existierenden WEB - Lösungen auch im mobilen Bereich eingesetzt werden können. Prinzipiell ist es jedoch möglich, damit mobile Anwendungen zu entwickeln.

2.4.5 WAP

Wireless Application Protocol (WAP) ist ein Standard zur Übertragung und Darstellung von Daten auf mobilen Endgeräten. Es wurde vom WAP-Forum, das über 200 Mitglieder der weltweiten größten Telekommunikationsunternehmen als Mitglieder hat und 2003 in der Open Mobile Alliance aufging, entwickelt.[45] Der wesentliche Zweck besteht dabei in der Bereitschaft einer einheitlichen Kommunikationsumgebung auf unterschiedlichen drahtlosen Technologien, der maximal möglichen Verlagerung von Rechenlast auf den Server und der Optimierung für die eingeschränkten Bedienungsmöglichkeiten sowie geringen Datenübertragungsraten, die auf den Endgeräten zur Verfügung stehen.[46] Als WAP bezeichnet man das eigentliche Protokoll mit seinen Schichten, das für die Datenübertragung zwischen dem WAP-Gateway und dem WAP-fähigen Endgerät zuständig ist sowie die Auszeichnungssprache WML und die Scriptsprache WMLScript. Die einzelnen Schichten sind:

1. Wireless Datagram Protocoll (WDP)
2. Wireless Transport Layer Security (WTLS)
3. Wireless Transaction Protocoll (WTP)
4. Wireless Session Protocoll (WSP)

[45] vgl. [OMA04]
[46] vgl. [KT03, Seite 89]

5. Wireless Application Enviroment (WAE)

Die Schichten sind in [Leh03, Seiten 144-145] näher erläutert. Für die mobile Datenübertragung kann auch das Transmission Control Protocol/Internet Protocol (TCP/IP) verwendet werden. WAP wird nur eingesetzt, wenn es sich um eine auf diesem Standard basierende Anfrage handelt. Daher kann auch im mobilen Bereich das im Internet gebräuchliche Web-Modell verwendet werden. Die Architekturen von TCP/IP und WAP sind an das ISO-OSI-Referenzmodell angelehnt.[47] Das WAP- Modell orientiert sich an dem Web-Modell. Ein WAP-Gateway mit der Funktionalität eines Proxy-Servers ist zwischengeschaltet, der die Protokollkonversion und die Umsetzung in binäre Darstellung vornimmt. Zwischen dem http-Server und dem WAP-Gateway wird der Internet-Protokollstapel verwendet und zwischen dem WAP- Gateway und dem Endgerät der WAP- Protokollstapel. Die Funktionsweise ist in Abbildung 2.10 dargestellt. Dadurch können die bestehenden Werkzeuge wie z.B. Web-Server, CGI-Scripts usw. ebenfalls eingesetzt werden. Das Endgerät übermittelt mittels WSP seine URL-Anfrage dem WAP-Gateway. Dieser wandelt die Anfrage in eine http-Anfrage um und schickt diese dem http-Server. Der HTTP-Server bearbeitet die Anfrage und sendet die WML-Seite via HTTP an das WAP-Gateway. Dort werden die empfangenen WML- und WMLScript-Seiten in ein platzsparendes Binärformat konvertiert und über WSP an das Endgerät übermittelt. Der WAP- Protokollstapel kann sowohl auf verbindungs- als auch auf paketorientierte Träger aufsetzen. Im Endgerät werden die WML-Seiten vom Micro - Browser, der im Wireless Application Environment (WAE) enthaltenen ist, dargestellt bzw. interpretiert. Die WAP-Inhalte werden in der Wireless Markup Language (WML) beschrieben. WML ist eine XML-basierte Auszeichnungsprache mit wenigen Sprachelementen. Sie erfüllt eine ähnliche Funktion wie HTML, ist aber mit diesem nicht kompatibel. Die Sprachelemen-

[47] vgl. [Leh03, Seiten 139-147]

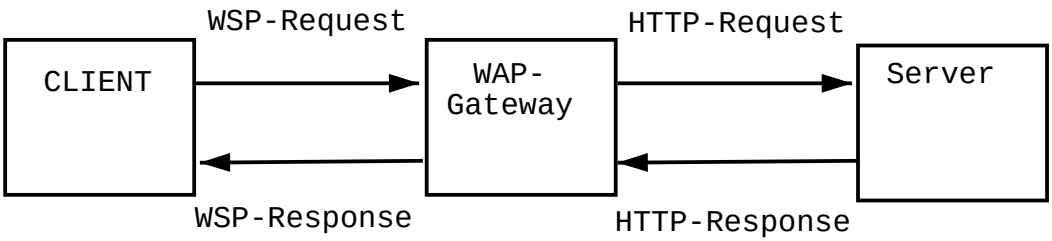

Abbildung 2.10: WAP-Modell mit Anfrage (Request) eines Clients über den WAP-Gateway und der Antwort (Response) vom Server

te ermöglichen die Formatierung von Text, Grafiken und Hyperlinks. Die WML-Seiten können mit einem Text- Editor auf einem gewöhnlichen PC geschrieben werden. Ein Deck ist das oberste Element einer WML- Datei. Ein oder mehrere zusammengehörige Cards werden zu einem Deck zusammengefasst. Cards werden nacheinander auf dem Display angezeigt und es kann zwischen ihnen navigiert werden. Abbildung 2.11 beinhaltet ein Beispiel-Script mit zwei cards. Am Anfang ist die WML-Syntax mit einer Document Type Definition (DTD) zu definieren. Diese sieht folgendermaßen aus:

```
<?xml version=’’1.0’’?>
<!DOCTYPE wml PUBLIC ‘‘-//WAPFORUM//DTD WML1.1//EN’’
‘‘http://www.wapforum.org/DTD/wml113.dtd’’>
```

Es folgt eine Beschreibung der weiteren Tags. Mit dem wml-Tag wird das Deck umschlossen, das die cards enthält. Das card-Tag eröffnet die erste Karte. Das p-tag eröffnet einen Paragraphen, um den Text “Hello Card1” darzustellen. Mit

```
 <do type=’’accept’’ label ‘‘card2’’>
            <go href =’’#card2’’>
         </do>
```

wird das Event “accept” deklariert und die zu ladende Seite “#card2” bestimmt. Durch Drücken einer Taste des Nutzers wird “card2” mit der id “card2” geladen. “card2” ist folgendermaßen definiert:

```
<card id =''card2''>
   <p>
     Hello Card2
   </p>
</card>
```

WMLScript ist ebenfalls Teil des WAE und hat wie JavaScript seinen Ursprung in ECMAScript. Es erlaubt, prozedurale Logik in WML-Seiten einzubinden und wird im WAP- Gateway in einen platzsparenden Bytecode umgewandelt.[48] WAP kann bereits im GSM-Netz mit entsprechenden WAP- Browsern auf GSM-Handys mit Datenübertragungsraten von 9600 bit/s genutzt werden. Für die Entwicklung von komfortablen Anwendungen und Informationsdiensten ist WAP geeignet. Die Anwendungslogik kann auf den Server verlagert werden. Ein Nachteil ist, dass für jede Ausführung eine Verbindung zu einem Server aufgebaut werden muss und wiederkehrende Daten jedes Mal vom Server geladen werden müssen. Des Weiteren sind die unterschiedlichen Implementierungen der Micro - Browser und die verschiedenen Displaygrößen auf den Endgeräten zu berücksichtigen. Dies erfordert möglicherweise die Entwicklung mehrerer Varianten eines WAP- Angebotes. WAP - Dienste werden von allen vier Netzbetreibern in Deutschland angeboten und auf den meisten Endgeräten sind WAP- Browser integriert. Geräte mit einem Markteintritt ab 2002 haben die WAP- Version 1.2 implementiert. WAP 2.0 ist aktuell und bereits spezifiziert, findet aber bisher keine wirkliche Beachtung.[49]

[48] vgl.[Grü04],[Pau04],[OMA04],[Leh03, Seiten 194-201]

[49] vgl. [KT03, Seite 93]

2.4.6 I-Mode

I-Mode wurde von der japanischen NTT DoCoMo ursprünglich für das dortige PDC - Netz (Personal Digital Cellular) entwickelt. Es ist ein proprietärer Standard, der in Deutschland auf GPRS basiert.[50] Bei I-Mode sind die Benutzer immer online und die Abrechnung erfolgt auf der Grundlage der übertragenen Datenmengen. Die I-Mode - Seiten werden in Compact - HTML (cHTML), auch als iHTML bezeichnet, mit einem Texteditor geschrieben. cHTML ist eine reduzierte Version von HTML und wurde ebenfalls von NTT DoCoMo entwickelt. In Abbildung 2.12 befindet sich ein Beispiel. Es folgt eine Beschreibung des Quelltexts. Dieser wird mit HTML-Tags umschlossen. Der Kopfteil besteht aus den HEAD-Tags und den Title-Tags. Der Titel in diesem Beispiel lautet “I-Mode”. Die Body-Tags umschließen den Hauptteil. Der Hauptteil beinhaltet folgende Zeilen:

```
<DIV>MENU</DIV><br>
<A ACCESSKEY=’’1’’
HREF=http://server/1.htm>Seite1</A><br>
<A ACCESSKEY=’’2’’
HREF=http://server/2.htm>Seite2</A><br>
<A ACCESSKEY=’’3’’
HREF=http://server/3.htm>Seite3</A><br>
```

Hierbei werden die Tasten “1”, “2” und “3” deklariert, die durch Betätigen des Nutzers zu den entsprechenden Seiten verweisen und diese laden. Gegenüber HTML sind einige zusätzliche Tags vorgesehen, die die speziellen Funktionen von mobilen Endgeräten unterstützen. Die cHTML- Dateien werden auf einem http-Server abgelegt und von dort abgerufen.[51] Mit I-Mode können mobile In-

[50] vgl. [KT03, Seite 65]

[51] vgl. [Leh03, Seiten 206-207],[KT03, Seite 93],[Onl04]

formationsdienste entwickelt werden und Interaktionen sind auch möglich. Allerdings sind für die Darstellung spezielle Endgeräte mit I-Mode - Browsern nötig. I-Mode - fähige Endgeräte sind in Deutschland nicht so weit verbreitet wie WAP - fähige. Die Anwendungslogik liegt auf der Serverseite. Also können damit auch keine Offline- Anwendungen ausgeführt werden. Der I-Mode - Dienst wird in Deutschland nur von E-Plus angeboten. Aufgrund der Geräteabhängigkeit und der Netzabhängigkeit wird der Aufbau eines Informationsdienstes mit I-Mode nicht empfohlen.

2.4.7 Anwendungsplattformen

Eine andere Möglichkeit zur Realisierung von Informationsdiensten ist die Entwicklung von Anwendungen, die auf dem mobilen Endgerät ausgeführt werden können. Diese offline ausführbaren Anwendungen werden auf die Endgeräte geladen und sind auch ohne kontinuierliche Netzwerkverbindung lauffähig. Die Anwendungslogik wird damit auf die Clientseite verlagert. Wenn Anwendungen auf lokale Daten und Diensten auf dem mobilen Endgerät zugreifen, werden diese besser direkt auf dem Endgerät installiert und dort ausgeführt.[52] Eine wichtige Rolle spielt die Portabilität zwischen den herstellerspezifischen Systemen und Typen von Endgeräten. Dabei bieten sich als portable Plattformen die Java 2 Micro Edition (J2ME) und das SymbianOS an. Als Kommunikationsprotokoll für die Clientapplikationen kann HTTP genutzt werden.[53] Anwendungen haben gegenüber den anderen Diensten einige Vorteile:

- Entwicklung von anwenderfreundliche Oberflächen für den Nutzer.
- Entwicklung von komplexen Anwendungen für die Endgeräte.

[52] vgl. [SR03, Seite 141]
[53] vgl. [SR03, Seite 142]

- Ausführung der Anwendungen ohne Verbindung zu einem Server.
- Es werden nur Anwendungsdaten versendet. Dazu besteht die Möglichkeit diese auf dem Client zu speichern.

Daher wird für die Fahrplanauskunft in dieser Arbeit eine Anwendung entwickelt, die auf einem mobilen Endgerät lauffähig ist.

Symbian OS

Series 60 ist eine Software-Oberfläche für Nokia - Handys. Die Series 60 basiert auf dem Symbian Betriebssystem[54] und wurde nach den Prinzipien der Open Mobile Alliance entwickelt. Zahlreiche Mobilfunkgerätehersteller haben die Series 60 lizenziert und in ihre Mobiltelefone integriert.[55] Die Bedienoberfläche, Navigationstasten und Displaygrößen bleiben im Wesentlichen gleich. Borland und Nokia bieten eine C++ -Entwicklungsumgebung für die Series 60 kostenlos an.[56] Damit wird es Entwicklern ermöglicht, Anwendungen in C++ zu entwickeln und auf verschiedene Endgeräte nachzuladen. SymbianOS bietet außerdem JAVA-Unterstützung.[57]

Java

Java ist auf zahlreichen mobilen Endgeräten integriert und man erreicht weitaus mehr Zielplattformen als mit Anwendungen für SymbianOS, das auch Java unterstützt. Damit kann man der “WORA” - Philosophie “Write Once - Run Anywhere” folgen und ein einmal geschriebenes Programm kann auf verschiedenen

[54] vgl. [Sym04a]
[55] vgl. [Sym04b]
[56] vgl. [gol04b], [gol04a]
[57] vgl. [Leh03, Seite 159]

Endgeräten ausgeführt werden. Java - fähige Endgeräte gibt es für alle Funknetze. Aufgrund der Plattformunabhängigkeit und hohen Verfügbarkeit auf unterschiedlichen Endgeräten wird die Programmierung des Prototyps, der in dieser Arbeit entwickelt wird, in Java erfolgen.

2.5 J2ME und JAVA

JAVA ist eine plattformunabhängige, objektorientierte Programmiersprache, die von SUN Microsystems entwickelt und vertrieben wird. Das erste Java Development Kit (JDK 1.0) wurde im Januar 1996 ausgeliefert. 1998 wurde dann die finale Version des JDK 1.2 herausgegeben und im Januar 1999 in JAVA 2 platform umbenannt.[58] Diese JAVA 2 platform gliedert sich in drei Bereiche: J2SE, J2EE und J2ME.[59] Abbildung 2.13 beinhaltet die drei Ausgaben mit der Ziel-Hardware. Die Standard Edition (J2SE) entspricht dem JDK 1.2 und die Enterprise Edition (J2EE) ist eine Erweiterung der J2SE um weitere Klassenbibliotheken zur Entwicklung von Serveranwendungen. Die Micro Edition (J2ME) ist speziell für Kleingeräte zugeschnitten und in Konfigurationen und Profile, die die unterschiedlichen Geräteklassen berücksichtigen, aufgeteilt. Solche Kleingeräte verfügen über ein kleineres Display (ab einer Auflösung von 96 * 54 Pixel mit 1 Bit Farbtiefe), keine Fensteroberfläche, keine Maus, wenig RAM - Arbeitsspeicher (ab 32 kB) und wenig Prozessorleistung (ab 24 MHz).[60] Unter J2ME-Konfigurationen versteht man einen für eine vorgegebene Geräteklasse universell verfügbaren minimalen Funktionsumfang. Diese Konfiguration spezifiziert die Sprachelemente, die passende Java Virtual Machine (JVM), Application Program-

[58] vgl. [Hei04c]
[59] vgl. [SUN04g]
[60] vgl. [SUN04b]

ming Interfaces (APIs) und die zugehörigen Programmbibliotheken. In J2ME gibt es die Connected Device Configuration (CDC) und die Connected Limited Device Configuration (CLDC). Profile spezifizieren weitere Eigenschaften einer Geräteklasse für eine vorgegebene J2ME-Konfiguration. Die CDC ist für Geräte mit typischerweise 32-Bit Prozessoren, 2 MBytes RAM, 2,5 MBytes ROM und einer relativ schnellen Netzanbindung konzipiert. Sie enthält eine voll ausgestatte JVM mit der Bezeichnung Classic VM(CVM). Solche Geräte sind z.B.: Set-Top-Boxen für Fernseher, Bildtelefone oder PDAs der Oberklasse. Die CDC unterstützt folgende Profile: Foundation Profile, Personal Basis Profile und das Personal Profile. In [SUN04a]sind die Profile genauer beschrieben. In Abbildung 2.14 werden die verschiedenen JVMs, Konfigurationen und Profile entsprechend zugeordnet. Die CLDC 1.0 ist für akkubetriebene Geräte mit eingeschränkter Displayfläche, Arbeitsspeicher zwischen 128 und 512 KByte sowie 16 Bit oder 32 Bit CPUs konzipiert. Mögliche Geräte wären also Mobiltelefone und Smartphones. Die Grundlage für die CLDC bildet die Kilobyte Virtual Machine (KVM). Für die KVM werden ca. 40 bis 80 KByte benötigt. Die KVM sorgt dafür, dass die Programme auf den Geräten ausgeführt werden. Der wichtigste Teil einer CLDC-Umgebung für den Programmierer ist das Mobile Interconnected Device Profile, auch Mobile Information Device Profile genannt (MIDP). Das Mobile Information Device Profile (MIDP) ist in vielen aktuellen Handys implementiert und ermöglicht die Ausführung von MIDP-Applikationen, die als Midlets bezeichnet werden. Diese Midlets begnügen sich mit wenig Speicher und Rechenleistung und den speziellen Eigenschaften von Handys.Das MIDP definiert eine API, die auf allen MIDP-fähigen Endgeräten zur Verfügung stehen muss. Dort werden Funktionen zur Eingabe, Speicherung und zur Ausgabe bereitgestellt. Das MIDP fordert für die Darstellung mindestens 96 * 54 Bildpunkten mit einer Farbtiefe von 1 Bit (monochrom) und mindestens 168 KByte ROM-Speicher. Für die MIDP-Bibliotheken

werden 128 KByte benötigt.
Für die Eingabe werden Standardtelefontastaturen oder PC-Tastaturen akzeptiert. Die Gerätehersteller bieten je nach Gerät zusätzliche programmierbare Tasten, Rädchen oder Joysticks an. Die Übernahme solcher Eingabeelemente ist in der MIDP API vorgesehen. Außerdem wird eine bidirektionale Netzwerkverbindung zur Verfügung gestellt. Die Netzwerkfunktionalitäten sind in den MIDP Network Libraries enthalten. Der Zugriff auf bestimmte Gerätefunktionen ist nur über proprietäre Bibliotheken der Hersteller möglich. Nokia und Siemens bieten zum Beispiel solche herstellerspezifischen Bibliotheken für ihre Geräte an. Es können nur Midlets, die keine gerätespezifischen APIs verwenden, auf allen MIDP-konformen Geräten ausgeführt werden. Das für die KVM vorgesehene CLDC hat nach [SR03, Seite 156])folgende Einschränkungen:

- Es existiert keine Unterstützung für Floating-Point-Datentypen und -Operationen.
- Es gibt keinen Zugriff auf in anderen Programmiersprachen geschriebene Komponenten.
- Die Klassen werden vom System in einer fest vorgegebenen Ordnung geladen.
- Eine Sichtbarkeit von Systemeigenschaften und Laufzeiteigenschaften durch Anwendungen innerhalb der virtuellen Maschine ist nicht vorhanden.
- Es sind nur eingeschränkte Unterprozesse möglich.
- Es existieren keine *finalization()* - Methode und keine schwachen Referenzen. Der Aufwand für den Garbage Collector wird dadurch vereinfacht.

- Der Aufwand der Klassenverifizierung wird auf das Entwicklungssystem als Präverifizierung verlagert. Es erfolgt dann auf dem Zielsystem nur noch eine vereinfachte Verifizierung.

Es wird in der CLDC eine Untermenge folgender Pakete der J2SE verwendet:

- *java.io* (für die Eingabe und Ausgabe durch DataStreams)
- *java.lang* (Basisklassen von Java)
- *java.util* (für Zeit, Datum, Kalender, Hashtabellen usw.)

Hinzu kommen Pakete, die sich im Namensraum *javax.microedition* befinden. Dazu zählen das "Generic Connection Framework", das ein Framework für Netzwerkverbindungen beinhaltet, das "Record Managment System (RMS)" für die persistente Speicherung von Daten und das bereits beschriebene LCDUI für die Erstellung von Benutzeroberflächen auf Mobiltelefonen. HTTP wird als einziges Protokoll von allen MIDP-Geräten unterstützt. Es erlaubt keine Übertragungen, die vom Server initiiert werden. HTTP basiert auf dem CLDC Generic Connection Framework (GCF). Bei dem Generic Connection Framework handelt sich um eine Vereinheitlichung von Netzwerkzugriffen; sie besteht nur aus einer Klasse und einer Vielzahl von Schnittstellen für Verbindungstypen. Eine Verbindung wird mit Hilfe einer URL über die Klasse *"Connector"* aus dem CLDC mit ihrer Methode *open()* hergestellt. HTTP ist das Protokoll, das für den Internet- und Netzwerkzugriff vorgesehen ist. Netzwerk umfasst hier jegliche Verbindung mit einem anderen Gerät. Wie diese Verbindung zustande kommt, ist gleichgültig. Normalerweise wird das Endgerät bei HTTP-Zugriffen eine Internetverbindung aufbauen. Im Paket *javax.microedition* ist das GCF untergebracht. Die Benutzerführung, die im Paket *"javax.microedition.lcdui"* enthalten ist, deckt einen minimal verfügbaren Funktionsumfang ab, den alle MIDP-konformen Geräte erfüllen müssen. Sie

gliedert sich in die Low-Level und High-Level-Programmierschnittstelle auf. Die Darstellung ist bildschirmorientiert und ohne eine Fensterumgebung. Scrollen ist nur vertikal möglich und die Benutzerführung erfolgt durch eine Menüsteuerung. Von der High-Level-API werden Menülisten, Textboxen, Alarmboxen, Grafiken usw. zur Verfügung gestellt. Das Low-Level API bietet auf Pixel-Ebene vollen Zugriff auf den Bildschirm eines MIDP-Gerätes. Dazu existiert nur eine Klasse Canvas, die von Displayable abgeleitet ist.

Die Anzeige und die Umsetzung der Eingaben erledigen die gerätespezifischen Implementierungen des MIDP. In Abbildung 2.15 wird ein Überblick über alle Klassen der MIDP-Benutzerschnittstellen, ihre Vererbungshierarchie und ihre Relation untereinander gegeben. Die Klassen werden in [MK03] ausführlich erläutert. Für den Zugriff auf den persistenten Speicher wird das Record Managment System (RMS) bereitgestellt. Das RMS verwaltet eine kleine Datenbank für die MIDlets. Die Midlets bekommen innerhalb des RMS einen Record Store zugewiesen, in dem es beliebig viele einzelne Einträge, begrenzt durch den vorhandenen Speicher, anlegen kann. Midlets können auf verschiedene Record-Stores zurückgreifen, die anhand ihres Namens unterschieden werden. Innerhalb einer Midlet-Suite ist der Name eindeutig. Außerhalb der Midlet-Suite sind die Record-Stores nicht sichtbar und zugreifbar. Andere Midlets können nur auf Record-Stores zugreifen wenn sie in der gleichen Midletsuits abgelegt sind. Die Objekte der Klassen können nicht direkt auf den Datenspeicher gelegt werden. Sie müssen stattdessen in einen Bytestrom umgewandelt werden. Die Records werden damit stets in Byte- Arrays gelesen und geschrieben. Daher ist für das Lesen und Schreiben eine Konversion der gewünschten Daten in dieses Format und aus diesem Format nötig. Für die Konvertierung lassen sich Klassen aus dem CLDC einsetzen. Da die einzelnen Records keine Namen haben können, bekommt jeder Record innerhalb des Record-Stores eine eigene Nummer, eine Record-ID, zugeteilt. Neue Daten können an das Ende eines Datensatzes angehangen werden. Ebenso können auch neue Daten die alten an einer vorgegebenen Stelle im Datensatz löschen. Die Record-IDs gelten nur innerhalb eines Record-Stores. Locking-Mechanismen werden vom System nicht zur Verfügung gestellt. Das Midlet muss die Zugriffe selbst koordinieren. Die RMS-API befindet sich im Paket *javax.microedition.rms*.[61] Derzeit verwenden die meisten Geräte MIDP 1.0,

[61](vgl. [MK03, Seiten 129-131])

aber MIDP 2.0 ist schon erhältlich. Das MIDP 2.0 basiert ebenfalls auf dem CLDC 1.0 und ist abwärtskompatibel zum MIDP 1.0. Es erweitert den Funktionsumfang um mehrere Bereiche. Diese sind in [SR03, Seiten 172-177] näher beschrieben.

2.6 XML

XML ist eine Markup-Sprache, in der die Struktur innerhalb eines Dokumentes festgelegt wird. Der XML-Standard wurde im Februar 1998 vom W3C verabschiedet und veröffentlicht. XML beruht auf der bereits 1986 entstandenen SGML (Standard Generalized Markup Language).Es werden Tags verwendet, um spezielle Elemente in einem Dokument zu definieren. Diese Tags legen die strukturellen Elemente des Dokuments sowie die Bedeutung dieser Elemente fest. XML definiert die Struktur und den Inhalt eines Dokuments und trennt ihn so von der Darstellung des Dokuments. Damit wird eine sinnvolle Bearbeitung durch Programme ermöglicht. Die XML-Spezifikation legt die Tags nicht selbst fest, es wird stattdessen eine Standardmethode zur Definition von Tags und Beziehungen geboten. Damit ist XML eine äußerst flexible Meta-Sprache, mit der jede Art von Dokumenten bearbeitet werden kann. Alle Daten in einem XML-Dokument sind von Tag-Elementen umgeben, die die Daten beschreiben. Die Tags sind von spitzen Klammern umschlossen und damit von den Daten des Dokuments getrennt. Dabei werden öffnende und schließende Tags unterschieden. Schließende Tags werden mit einem Slash vor dem Bezeichner gekennzeichnet. Alle Daten werden somit innerhalb des öffnenden und schließenden Tags diesem XML-Element zugeordnet. Das XML-Element kann dann über den Bezeichner im Tag identifiziert werden. XML-Dateien bestehen ausschließlich aus Text. Damit werden die Entwicklung von plattformunabhängigen Tools und der Austausch von Dateien zwischen Anwendungen vereinfacht. Ein Vorteil von XML ist die Platt-

formunabhängigkeit. XML-Dokumente können auf einem Windows-PC ebenso genutzt werden wie auf Linux/Unix-Systemen oder auf mobilen Kleincomputern wie PDAs oder Mobiltelefonen. Da XML-Dokumente strukturiert, plattformunabhängig und textbasierend sind, können sie mit vielen Programmen geöffnet und bearbeitet werden. Es ist damit möglich, Daten unabhängig von bestehenden Programmen oder Systemen zu speichern und zu organisieren, ohne dabei die Struktur der Daten zu verlieren. Es erlaubt anderen Applikationen, auf die Daten lesend oder schreibend zu zugreifen, ohne die Informationen in ein proprietäres Format umwandeln zu müssen. XML-Parser können die Daten aus einem XML-Dokument lesen und interpretieren, unabhängig von der zugrundeliegenden Struktur. Somit können komplexe Strukturen plattformunabhängig in Netzwerken transferiert werden.[62] Aus diesen Gründen wird für den Datenaustausch in der zu entwickelnden Anwendung in dieser Arbeit auf XML zurückgegriffen. Die zu entwickelnde Anwendung wird auf Anfrage des Nutzers die gewünschte Abfahrtszeit einer Bahn oder eines Busses anzeigen. Dazu müssen die Abfahrtszeiten, die in der Anwendung abgefragt werden, in einem geeigneten Format abgelegt werden. Diese benötigten Fahrplandaten werden in XML-Dokumenten abgelegt. Damit können auch andere Client-Anwendungen wie HTML-Browser, WAP-Browser oder Standalone-Clients diese Daten nutzen.[63] In Abbildung 2.16 wird eine Architektur dargestellt, in der ein Server XML-Daten verschiedenen Clients (HTML-Browser, WAP-Browser, Standalone Client, MIDP-CLIENT) über verschiedene Protokolle zur Verfügung stellt. Das CLDC bietet keine Unterstützung für XML-Parser. Um XML mit J2ME zu verarbeiten, existieren mehrere Parser von Drittanbietern.[64] Einige dieser Kandidaten sind kXml, NanoXML, TinyXML. Parser für J2ME müssen klein sein, da wenig Speicher zur Verfügung steht. Keiner der

[62] vgl. [Mic04] und [Amm03, Seite 30]

[63] vgl. [SUN04f]

[64] vgl. [MK03, Seiten 281-294]

für J2ME verfügbaren Parser ist ein validierender Parser. Für diese Arbeit wird der kXML-Parser verwendet. kXML ist Open Source und wurde speziell für die J2ME-Plattform (MIDP und CLDC) geschrieben. Dieser kann von [kXM04] bezogen werden. Es gibt grundsätzlich drei Typen von Parsern:

- Model-Parser
- Push-Parser
- Pull-Parser

Ein Model-Parser liest den gesamten Inhalt einer Datei aus und erstellt im Speicher ein Objekt, das dieses Dokument repräsentiert. Model-Parser benötigen mehr Speicher als die anderen Typen.Push-Parser "drücken" alle Ereignisse in einige wenige zentrale Behandlungsmethoden. Die Anwendung muss in diesen Behandlungsmethoden ihren internen Status ermitteln, bevor sie entscheiden kann, was mit dem eingegangenen Ereignis zu tun ist. Bei Pull-Parsern hat die Anwendung die Kontrolle über das Lesen der Daten. Die Anwendung "zieht" die Ereignisse aus dem XML-Strom. kXML ist ein Pull-Parser und basiert auf dem XMLPull-Interface, das ein gemeinsames Interface für verschiedene Pull-Parser darstellt. kXML ist in der Klasse *KXmlParser* implementiert. Mit der Methode *next()* kann das nächste XML-Ereignis im Eingabestrom angesteuert werden. Daneben gibt es noch die Methoden *nextToken(), nextTag() und require()*. Mit den Methoden *getType(),getName()* und *getText()* kann auf das aktuelle XML-Ereignis zugegeriffen werden. kXML akzeptiert beliebige Reader oder InputStreams als Eingabe. [65]

[65] vgl. [xml04],[MK03, Seiten 281-294] und [www04b]

```
<?xml version=’’1.0’’?>
<!DOCTYPE wml PUBLIC ‘‘-//WAPFORUM//DTD WML1.1//EN’’
‘‘http://www.wapforum.org/DTD/wml113.dtd’’>
<wml>
   <card>
      <p>
         <do type=’’accept’’ label ‘‘card2’’>
            <go href =’’#card2’’>
         </do>
         Hello Card1.
      </p>
    </card>
    <card id =’’card2’’>
       <p>
         Hello Card2
       </p>
    </card>
</wml>
```

Abbildung 2.11: Beispiel-Quelltext WML - die Erläuterungen zu den Tags sind im Abschnitt “WAP” beschrieben

```
<HTML>
<HEAD>
<TITLE>I-MODE</TITLE>
</HEAD>
<BODY>
<DIV>MENU</DIV><br>
<A ACCESSKEY=’’1’’
HREF=http://server/1.htm>Seite1</A><br>
<A ACCESSKEY=’’2’’
HREF=http://server/2.htm>Seite2</A><br>
<A ACCESSKEY=’’3’’
HREF=http://server/3.htm>Seite3</A><br>
</BODY>
</HTML>
```

Abbildung 2.12: Beispiel-Quelltext I-Mode(cHTML) - im Abschnitt “I-Mode” ist der Quelltext beschrieben

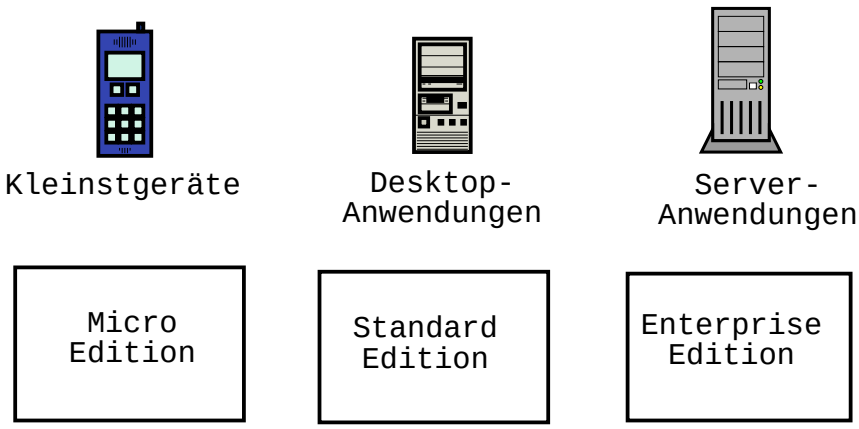

Abbildung 2.13: JAVA 2

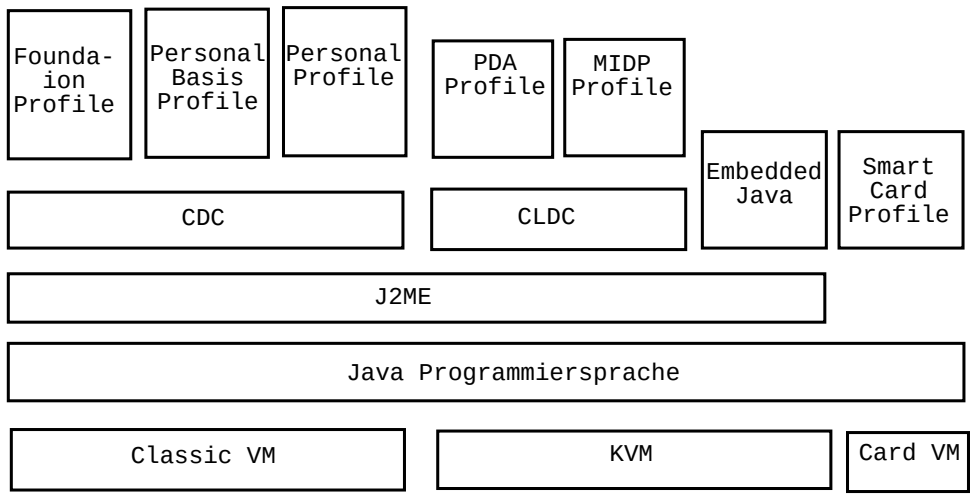

Abbildung 2.14: Profile

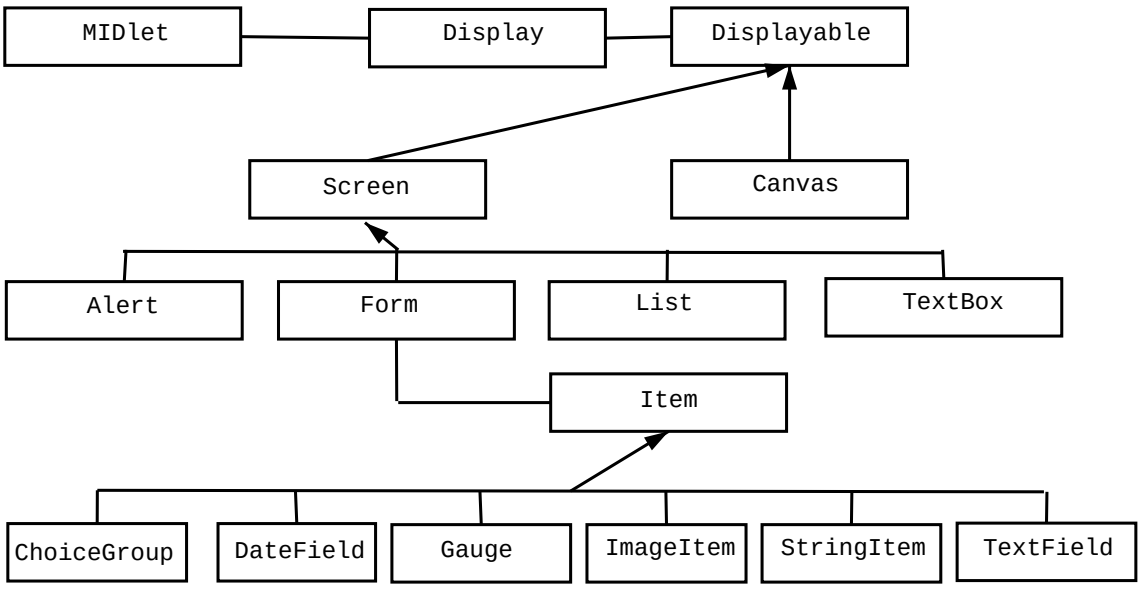

Abbildung 2.15: MIDP-Klassen der grafischen Benutzeroberfläche (vgl. [MK03, Seite 64])

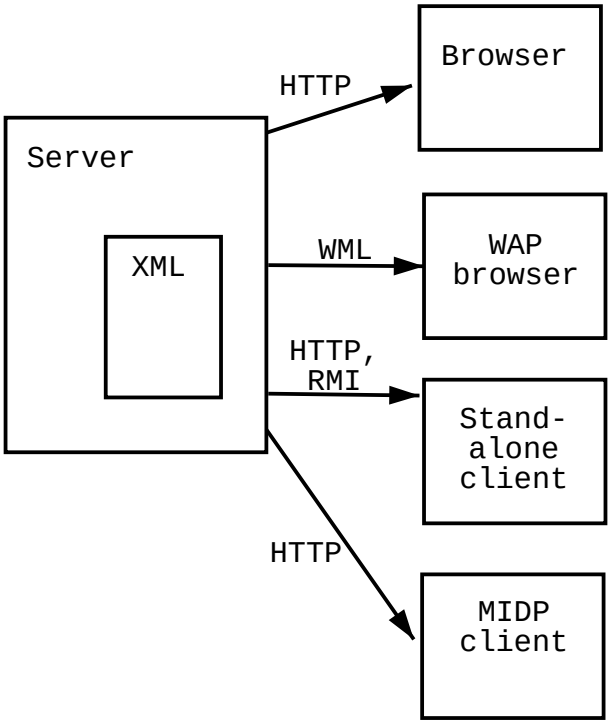

Abbildung 2.16: Architektur mit XML

Kapitel 3

Hard- und Softwarevoraussetzungen

3.1 Hard- und Softwareumgebungen für die Entwicklung

Die Entwicklung erfolgte unter Windows XP auf einem Rechner mit Intel Celeron CPU - 2,4 GHz und 256 MByte RAM, aber auch andere Betriebssysteme wie Linux oder Solaris/Sparc kommen in Frage, da dafür auch die Entwicklungswerkzeuge vorhanden sind. [1] Jedoch sind die meisten SDKs und Emulatoren für die Windows Plattform vorhanden.[2] Auf dem Rechner wurde Java 2 SDK Standard Edition für Windows in der Version 1.4.2 installiert. Diese wird benötigt, um die Programmquellen zu kompilieren und sie kann von [SUN04d] kostenlos bezogen werden. Weiterhin wurde das Java 2 Platform Micro Edition, Wireless Toolkit 2.0 (WTK) installiert, das bei [SUN04c] ebenfalls kostenlos erhältlich ist. Das WTK beinhaltet eine Emulationsumgebung, Werkzeuge, Dokumentationen und Beispiele zum Entwickeln von Midlets. Es ist keine vollständige IDE, sondern ein

[1] vgl. [Nou04] und [MK03, Seite 31]

[2] vgl. [SUN04e]

einfaches Projektverwaltungs-Werkzeug. Editieren des Quellcodes und Debuggen ist damit nicht möglich. Das WTK erlaubt das Kompilieren, Packen und die Ausführung von MIDP-Anwendungen. Für das Editieren muss ein zusätzlicher Editor verwendet werden. Zum Debuggen kann man beispielsweise das Siemens SDK hinzunehmen. Die Hardwarevoraussetzungen für das WTK sind folgende:

- 166 MHZ CPU
- 64 MB RAM
- 30 MB Festplatten-Speicher

Für das SDK, das für die Entwicklung mit dem WTK nötig ist, werden eine 166 MHZ CPU und 32 MB RAM verlangt. Die Installation erfolgt durch Ausführen der beiden exe - Dateien. Dabei muss das Java 2 SDK als erstes installiert werden. Das WTK kann auch in IDEs wie Sun ONE Studio, Eclipse IDE, Borland JBuilder und andere integriert werden.[3] Für die Midlet- Entwicklung in dieser Arbeit wird der frei erhältliche XEmacs als Editor verwendet, der von [www04c] bezogen werden kann. Dazu wird noch das Paket JDE installiert, das den Xemacs in eine vollwertige Java-Entwicklungsumgebung verwandelt. Im Folgenden werden die einzelnen Schritte zur Entwicklung eines Midlets beschrieben. Der Quelltext kann mit einem Editor geschrieben und als java-Datei abgespeichert werden. Zum Kompilieren des Quelltextes wird der *javac* - Compiler verwendet. Wird J2SDK 1.4 genutzt, muss noch als Parameter *-target 1.1* angegeben werden, da der Compiler ab 1.4 standardmäßig Bytecode erzeugt, der nicht mit der Micro Edition kompatibel ist.[4] Nach dem Kompilieren muss die erzeugte class-Datei präverifiziert werden. Dies geschieht mit *preverify*. Dabei werden den Klassendateien Verifizierungshinweise hinzufügt, die die Überprüfung auf dem Zielgerät

[3] vgl. [Tay04]
[4] vgl. [MK03]

vereinfachen. Zum Schluss werden alle Klassendateien und Resourcen wie z.B. Bilder in ein Archiv mittels *jar* gepackt. Dieses Archiv muss noch ein so genanntes Manifest mit Informationen über den Inhalt des Archivs enthalten. Dazu gehören die Version des CLDC und MIDP, die enthaltenen Midlets und die Version des Pakets. Informationen über den Inhalt der Midlet-Suite können in einer separaten Datei, der Java Application Description (JAD), erstellt werden. Die JAD- Datei kann unter anderem die URL, von der die Midlet-Suite geladen werden kann, die Dateigröße und den Dateinamen enthalten.[5] Die Schritte Kompilieren, Präverifizierung und Packen können in der KToolbar des WTK mit Project/Package/Create Package durchgeführt werden. Danach stehen die jar- Datei und die jad- Datei im bin- Verzeichnis des Projekt-Verzeichnisses zur Verfügung. Diese bilden die Midlet-Suite und können dann auch mit anderen Emulatoren oder MIDP-fähigen Geräten getestet werden. Vorher muss in der KToolbar ein neues Projekt angelegt werden. Die zu editierenden Quelltexte müssen im Verzeichnis *apps/Projektname/src* mit einem Editor bearbeitet werden. Zusätzlich kann ein Obfuscator für die Optimierung der Midlets genutzt werden. Ein Obfuscator ist eigentlich für das Erschweren von Reverseengineering gedacht. Dabei werden alle Namen der Variablen, Methoden und Klassen in kürzere umbenannt. Hier kann er verwendet werden, um Speicherplatz und Geschwindigkeit zu gewinnen. Das Laden von Klassen mit kürzeren Namen bewirkt einen Geschwindigkeitsgewinn, da weniger verglichen werden muss als mit längeren Namen und die Größe der class-Dateien wird verkleinert.[6] Der Einsatz eines Obfuscator ist im WTK vorgesehen. Erreichbar ist der Obfuscator mit Project/Package/Create Obfuscated Package .Für die Integration des Proguard-Ofuscator muss die Datei *proguard.jar* in das bin-Verzeichnis des WTK kopiert werden, die bei [Pro04]

[5] vgl. [Nou04]

[6] vgl. [int04]

erhältlich ist. Ein anderer Obfuscator, der Retroguard, steht bei [Sys04] ebenfalls zum kostenlosen Download bereit. Für die Entwicklung des Prototyps wird die Kombination aus XEMACS, dem J2SDK 1.4.2, dem WTK 2.0, kXML2 und dem Proguard verwendet. Die kxml2.zip muss für die Nutzung in das lib-Verzeichnis des Projektverzeichnisses kopiert werden. Zusätzlich werden noch zum Testen ein Nokia-Emulator und ein Siemens-Emulator installiert. Das Siemens SDK kann auch in Borland JBuilder und SUN ONE STUDIO integriert werden. Die SDKs von Siemens und NOKIA sind nach einer Registrierung kostenlos bei [Mob04] und [Nok04] erhältlich. Auf [SUN04e] befinden sich weitere Links zu anderen SDKs mit Emulatoren, die von verschiedenen Mobiltelefonherstellern bereitgestellt werden.

3.2 Hard- und Softwareumgebungen für den Betrieb der Anwendung

Für den Betrieb sind J2ME-fähige Geräte mit einer Verbindung zu einem Mobilfunknetz notwendig. Auf diesen Geräten muss die Anwendung dann geladen und gestartet werden. Zum Testen des Prototyps, der in dieser Arbeit entwickelt wird, kommt ein Siemens MC 60 zum Einsatz. Auf Serverseite wird der Apache-WebServer unter Windows XP genutzt, von dem die Anwendung und die Daten geladen werden können. Andere Betriebssysteme oder HTTP-Server können jedoch auch verwendet werden. Diese sind für die Nutzung von jad- und jar- Dateien zu konfigurieren. Beim Apache erfolgt das durch Hinzufügen der folgenden Einträge in *.htaccess*:

```
AddType text/vnd.sun.j2me.app-descriptor jad
AddType application/java-archive jar
```

oder in *mime.types* im Verzeichnis *conf/*:

```
text/vnd.sun.j2me.app-descriptor jad
application/java-archive jar
```

Den Apache-Server kann man von [Apa04] herunterladen. In dieser Arbeit wird die Version 2.0.48 verwendet. Die Installation erfolgt durch Ausführen des Installationsprogrammes. Das Midlet wird durch Angabe der URL, auf der sich die jad-Datei befindet, über die Luftschnittstelle (OTA. Over The Air) geladen. Aber auch ein Laden mittels Datenkabel oder USB- Kabel ist möglich. Bei dem Siemens MC 60 erfolgt das Installieren eines Midlets mit Menü/ Surf & Fun/Internet/Gehe zu URL, wobei die URL der jad-Datei angegeben werden muss.

Kapitel 4

Konzeption

Anwendungsgebiet des Prototyps wird eine mobile Fahrplanauskunft sein. Der Benutzer soll die Abfahrtszeiten von verschiedenen Bussen und Straßenbahnen mit einem Mobilgerät abfragen können. Durch eine menügeführte Bedienung wird die gewünschte Linie und Haltestelle abgefragt. Die Daten werden dauerhaft auf dem Mobilgerät abgelegt und können bei Bedarf wieder gelöscht werden. In diesem Kapitel wird eine Anforderungsanalyse erstellt. Hier wird beschrieben, welche Funktionalität die Anwendung besitzen wird. Um den generellen Nutzerpräferenzen entgegen zu kommen, muss eine erfolgreiche Anwendung schnell, schlank sowie intuitiv sein. “Schnell und schlank” bedeutet hierbei wenig Online-Zeit (Laden der Daten in weniger als 180 Sekunden), ein gutes Antwortzeit-Verhalten (Anzeige der Abfahrtszeit in weniger als 30 Sekunden) und ein niedriges Datenvolumen (3 KByte je Fahrplan) . Intuitiv bedeutet, dass das Ziel mit wenigen Klicks (vier Klicks bis zur Anzeige der Abfahrtszeit) erreichbar sein soll.[1] Diese Regeln müssen auf die Anwendung in dieser Arbeit übertragen werden. Dabei sind die Einschränkungen der mobilen Endgeräte zu beachten. Die Bedie-

[1] vgl. [KT03, Seite 96]

nung wird durch eine einfache und komfortable menügeführte Eingabe erfolgen. Folgende Optionen sollen dem Anwender bereitgestellt werden:

- Der Benutzer kann aus einer Liste, die vom Server bereitgestellt wird, einen beliebigen Linienfahrplan auswählen, der dann auf dem Gerät dauerhaft im Gerätespeicher abgelegt wird. Damit entfallen spätere Kommunikationskosten. Dazu müssen auf einem http-Server die verschiedenen Fahrpläne in XML-Dateien abgelegt werden. Für die Speicherung stellt J2ME den "Record Managment Store"(RMS) zur Verfügung. Damit steht ein Mechanismus für den persistenten Speicher bereit. Die Fahrplandaten müssen in Form einer Liste von der Anwendung im Record-Store gespeichert werden.

- Der Benutzer kann abgespeicherte Linienfahrpläne wieder löschen. J2ME erlaubt mit dem Zugriff auf den Record-Store auch das Entfernen von Daten.

- Der Benutzer kann aus einer Liste, die der Linienfahrplan bereitstellt, die Haltestelle auswählen. Anhand der aktuellen Uhrzeit und des Tages wird die nächste Abfahrtszeit angezeigt. Durch einen Vergleich der aktuellen Uhrzeit des Mobilgerätes mit den Fahrplan-Daten im Record-Store kann der passende Eintrag gefunden werden. Dazu müssen alle Abfahrtszeiten verglichen werden.

- Der Benutzer kann durch den Fahrplan navigieren. Frühere oder spätere Abfahrtzeiten können abgerufen werden. Durch Auswahl des Benutzers kann eine spätere oder frühere Abfahrtzeit gefunden werden, indem auf den vorherigen oder nächsten Eintrag der Fahrplandaten im Record-Store zugegriffen wird.

- Der Benutzer wird benachrichtigt wenn die Fahrplan-Gültigkeit abläuft. Da-

zu muss das aktuelle Datum mit der Fahrplan-Gültigkeit verglichen werdem. Dafür ist ein Eintrag der Gültigkeit des Fahrplans im Record-Store notwendig.

In Abbildung 4.1 ist die Funktionsweise der Client-/Server-Anwendung dargestellt. Dabei wird für die Kommunikation das HTTP-Protokoll verwendet. Ein

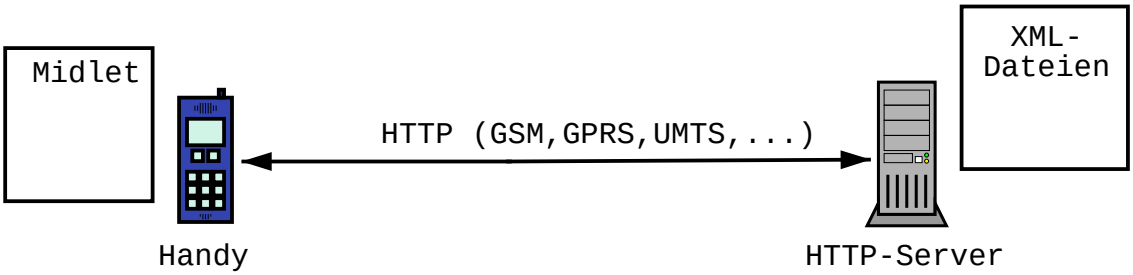

Abbildung 4.1: HTTP-Verbindung vom Mobilgerät zum Server, auf dem Handy wird die Anwendung ausgeführt und auf dem HTTP-Server sind die XML-Dateien abgelegt

Fahrplan enthält folgende Daten:

- Name der Linie
- Gültigkeit des Fahrplans
- die Haltestellen der Linie
- die Zeitentfernungen zur Starthaltestelle
- die Abfahrtzeiten für Wochentage, Samstag und Sonntag

Diese Daten müssen in XML-Dateien abgelegt werden. Der Datenaustausch zwischen Client und Server erfolgt mit XML über eine HTTP-Verbindung. In einer

XML-Datei ist die Liste der Fahrpläne abgelegt, die zur Auswahl bereitgestellt werden. Die Fahrpläne der einzelnen Linien, die abgerufen werden können, sind ebenfalls in XML-Dateien abgespeichert. Zur Speicherung der Fahrplan-Daten im persistenten Speicher muss eine geeignete Datenstruktur ausgewählt werden. Dafür werden die einzelnen Haltstellen mit der Zeit-Entfernung zur Starthaltestelle und die Abfahrtszeiten der Starthaltestelle benötigt. Daraus wird dann in der Anwendung die aktuelle Abfahrtszeit einer Haltestelle berechnet. In der Anwendung muss beachtet werden, dass drei Fahrpläne pro Linie existieren (Wochentag, Samstag, Sonntag). In Abbildung 4.2 ist die Datei *linien.xml* abgebildet. Die Abbildung 4.3 enthält eine Datei mit dem Fahrplan der Linie “10 Wahren”(*wahren10.xml*). *linien.xml* enthält die Elemente *Bahn*, in der die Elemente *richtung* und *datei* existieren. In *richtung* ist der Name der Linie festgelegt und in *datei* wird auf den Namen der dazugehörigen Datei verwiesen. In der Datei *wahren10.xml* wird der komplette Fahrplan einer Linie abgebildet. Das Element *Linie* enthält den Namen einer Linie. Im Element *Gueltig* wird die Gültigkeit eines Fahrplans festgelegt. Das Element *Stationen* enthält die Elemente *s* in der die Namen der Stationen in *hs* und deren Zeitabstand zur ersten Haltestelle in *zeit* definiert werden. Das Element *Abfahrtzeiten* beinhaltet die Abfahrtszeiten für *Wochentag*, *Samstag* und *Sonntag*. Diese werden in den Elementen *t* aufgelistet.

Abbildung 4.2: Datei *linien.xml* mit den Elementen *Bahn*, *richtung* und *datei*, das Element *datei* verweist auf die Datei eines Fahrplans

```
<?xml version="1.0"?>
<Linien>
   <Bahn>
      <richtung value="10 Wahren"/>
      <datei value="wahren10.xml"/>
   </Bahn>
   <Bahn>
      <richtung value="10 Loessnig"/>
      <datei value="loessnig10.xml"/>
   </Bahn>
   <Bahn>
      <richtung value="16 Loessnig"/>
      <datei value="loessnig16.xml"/>
   </Bahn>
   <Bahn>
      <richtung value="16 Messegelaende"/>
      <datei value="messegelaende16.xml"/>
   </Bahn>
</Linien>
```

Abbildung 4.3: Fahrplan einer Linie - Beschreibung der einzelnen Elemente im Text

```
<?xml version="1.0"?>
<Fahrplan>
   <Linie>10 Wahren</Linie>
   <Gueltig>23.08.2004</Gueltig>
   <Stationen>
   <s><hs>Loessnig</hs><zeit>0</zeit></s>
   <s><hs>Moritzhof</hs><zeit>1</zeit></s>
   ...
   </Stationen>
    <Abfahrtzeiten>
    <Wochentag>
     <t>0500</t><t>0515</t><t>0530</t><t>0545</t>
     <t>0600</t><t>0609</t><t>0619</t><t>0629</t>
     ...
     <t>2100</t>
    </Wochentag>
    <Samstag>
     <t>0600</t>
     ...
    </Samstag>
    <Sonntag>
     <t>0600</t>
     ...
   </Sonntag>
   </Abfahrtzeiten>

</Fahrplan>
```

Kapitel 5

Entwicklung

Die Anwendung besteht aus den Klassen *Diplom, Readthread, ReadTimetable-Thread und FirstScreen*. Die Klasse *FirstScreen* dient der Anzeige am Anfang der Anwendung. In der Klasse *ReadThread* wird eine HTTP-Verbindung zu einem Server aufgebaut und die Datei *linien.xml* wird geladen. Es wird ein Thread für die Verbindung gestartet um Deadlocks zu vermeiden. Die Datei *linien.xml* wird in dieser Klasse geparst und die Attribute der Elemente *richtung* werden auf dem Mobilgerät in Form einer Liste angezeigt. In ihr befinden sich die Linien-Namen und die dazugehörige XML-Datei. Die Attribute der Elemente *datei* werden in dem String-Array *str* für die weitere Verarbeitung abgelegt. In dem String-Array *str* befinden sich dann die Dateinamen, der auf dem Server abgespeicherten Fahrpläne. In der Klasse *ReadTimetableThread* wird eine Verbindung als Thread zum HTTP-Server aufgebaut um einen Fahrplan in Form einer XML-Datei zu laden. Dabei wird die Verbindung zu *"http://Server/"+str[list.getSelectedIndex()])* aufgebaut. Mit dem Parser werden die Fahrplandaten der XML-Datei ausgelesen. Die aus der XML-Datei ausgelesenen Elemente liegen als String-Datentyp vor. Diese werden der Methode *WriteRS* in der Klasse *Diplom* übergeben. Die zu übergeben-

den Datentypen sind:

- *String linie*: der Name der Linie
- *String gueltig*: Gueltigkeit des Fahrplans
- *String [] Stationen*: die Haltestellen der Linie
- *String [] Zeiten*: die Zeit-Entfernungen der Haltestellen zur Starthaltestelle
- *int z*: die Anzahl der Haltestellen
- *String [] AbfahrtzeitenW*: die Abfahrtszeiten des Fahrplan an Wochentagen
- *int w*: die Anzahl der Abfahrtszeiten an Wochentagen
- *String [] AbfahrtzeitenSa*: die Abfahrtszeiten des Fahrplans am Samstag
- *int v*: die Anzahl der Abfahrtszeiten des Fahrplans am Samstag
- *String[] AbfahrtzeitenSo*: die Abfahrtszeiten des Fahrplans am Sonntag
- *int u*: die Anzahl der Abfahrtszeiten am Sonntag

In der Klasse *Diplom* sind die Methoden *Screen2(), Screen3(), Screen4(), Screen5(), Screen6(),IntgetBytes(int z),IntsetBytes(byte [] data) und WriteRS(String linie, String gueltig, String [] Stationen, String [] Zeiten, int z, String []AbfahrtzeitenW,int w, String []AbfahrtzeitenSa, int v, String []AbfahrtzeitenSo, int u)* implementiert. Für die Darstellung der Bildschirmausgaben werden nur die Unterklassen *List* und *Alert* der Klasse *Screen* verwendet. Die Anwendung reagiert auf Kommandos, die an den registrierten *CommandListener* abgesetzt werden. Dazu zählen Kommandos vom Typ *List.SELECT_COMMAND*, die abgesetzt werden, sobald ein neues Listenelement selektiert wird. In der Methode

Screen2() werden die Listenelemente “Fahrplan” und “Lade neuen Fahrplan” sowie das Kommando “Beenden” zur Auswahl dargestellt. Wird das erste Listenelement selektiert, dann wird die Methode *Screen4()* aufgerufen Bei der Auswahl des zweiten Listenelements erfolgt der Aufruf der Methode *Screen3()*. In der Methode *Screen3()* wird eine neue Instanz der Klasse *ReadThread* erzeugt und der Thread zum Laden und Parsen der Datei *linien.xml* wird gestartet. Dort werden die Linien, die in der XML-Datei zur Verfügung stehen, in einer Liste zur Auswahl angezeigt. Außerdem wird ein Kommando *cmdBack* erzeugt, um damit wieder zu *Screen2()* zu gelangen. Wird eine der Linien selektiert, so wird eine Instanz der Klasse *ReadTimetableThread* erzeugt und der in dem String-Array *str* abgelegte Dateiname wird über eine HTTP-Verbindung geladen. Die Daten werden geparst und, wie bereits beschrieben, der Methode *WriteRS()* übergeben. Die Methode *WriteRS()* dient der Speicherung der Daten im Record-Store. Falls der Datei-Name im Record-Store schon existiert, werden keine Daten geschrieben.*WriteRS* schreibt die Daten in folgende IDs des Record-Stores:

- ID1: Name der Linie linie
- ID2: Gültigkeit des Fahrplans gueltig
- ID3: Anzahl der Stationen z
- ID4 bis ID4+(2*z)-1: Die Haltestellen und deren Entfernung zur Starthaltestelle
- ID4+(2*z): Wochentag
- ID4+(2*z)+1: Anzahl der Abfahrtszeiten für einen Wochentag w
- ID4+(2*z)+2 bis ID4+(2*z)+2+w-1: Die Abfahrtszeiten eines Wochentags
- ID4+(2*z)+2+w:Samstag

- ID4+(2*z)+3+w:Anzahl der Abfahrtszeiten am Samstag v
- ID4+(2*z)+4+w bis ID4+(2*z)+4+w+v-1: Die Abfahrtszeiten am Samstag
- ID4+(2*z)+4+w+v: Sonntag
- ID4+(2*z)+5+w+v: Anzahl der Abfahrtszeiten am Sonntag u
- ID4+(2*z)+6+w+v bis ID4+(2*z)+6+w+v+u-1: Die Abfahrtszeiten am Sonntag

Für jeden Fahrplan wird ein neuer Record-Store angelegt. Da in den Record-Store nur Byte-Arrays geschrieben werden können, werden der Name der Linie, die Gültigkeit des Fahrplans, die Namen der Haltestellen sowie die Strings *Wochentag, Samstag und Sonntag* mit der String-Methode *getBytes()* in Byte-Arrays konvertiert und in den Record-Store abgelegt. Die Anzahl der Stationen, die Anzahl der Abfahrtzeiten und die Abfahrtzeiten selbst werden mit der Methode *Integer.parseInt(int)* aus Strings in Integer-Werte konvertiert. Danach erfolgt die Konvertierung aus Integer-Werten in Byte-Arrays mit der implementierten Methode *byte[] IntgetBytes (int z)*. Für die Rückkonvertierung in Integer-Werte aus den Byte-Arrays wurde die Methode *int IntsetBytes (byte [] data)* implementiert. In der Methode *byte[] IntgetBytes (int z)* wird die Kombination aus *DataOutputStreams* und *ByteArrayOutputStreams* verwendet. Mit *DataOutputStreams* können einfache Typen wie *ints, longs* oder *Strings* in einen *OutputStream* geschrieben werden. Mit *ByteArrayOutputStream* kann dann aus den geschriebenen Daten ein Byte-Array erzeugt werden. Um ein Byte-Array zu erzeugen, werden einfach die beiden Streams hintereinander geschaltet. Mit der Methode *toByteArray()* kann dann am Ende das fertige Byte-Array von dem *ByteArrayOutputStream* gelesen werden. Zum Rekonstruieren der Integer-Werte aus dem

Byte-Array werden in der Methode *IntsetBytes(byte [] data)* analog ein *ByteArrayInputStream* und ein *DataInputStream* genutzt.[1] Die Abfahrtzeiten werden in Stunde*100 + Minuten im Record-Store abgelegt. In der Methode *Screen4()* werden mit *str1 = RecordStore.listRecordStores();* die Namen aller Record-Stores, die in der aktuellen Midlet-Suite verfügbar sind, in das String-Array *str1* geschrieben. Danach werden die einzelnen Record-Stores geöffnet und die erste ID mit dem Namen der Linie ausgelesen. Diese werden in Form einer Liste auf dem Mobilgerät zur Anzeige gebracht. Weiterhin werden die Kommandos “Zurück”, “Select” und “Löschen” hinzugefügt. Wählt der Nutzer das Kommando “Zurück”, dann wird wieder die Methode *Screen2()* aufgerufen. Mit “Löschen” wird der selektierte Record-Store der Midlet-Suite gelöscht. Wird mit “Select” der Name der Linie gewählt, dann wird die Methode *Screen5()* gestartet. Die Methode *Screen5()* dient dazu, den selektierten Record-Store zu öffnen. Dies geschieht mit *rs=RecordStore.openRecordStore(str1[li],true);*. *li* ist das selektierte Element aus der Liste: *li=list.getSelectedIndex();*. Hier wird die Gültigkeit des Fahrplans überprüft. Dazu wird der aktuelle Tag, der Monat und das Jahr mit dem Datum in der ID2 verglichen. Danach wird ID3 mit der Anzahl der Haltestellen gelesen und die einzelnen Haltestellen werden in dem Feld *str2* abgelegt. Die dazugehörigen Zeitentfernungen zur Starthaltestelle werden in dem Feld *tx* gespeichert. Außerdem werden die Haltestellen in einer Liste zur Auswahl gestellt. Mit dem Kommando “Zurück” gelangt man wieder zu *Screen4()*. Wird der Name einer Haltestelle selektiert, dann wird die Methode *Screen6()* aufgerufen. Diese Methode zeigt die nächste Abfahrtszeit an. Dazu wird die aktuelle Stunde und Minute abgefragt und es wird überprüft, ob der aktuelle Tag ein Sonntag, Samstag oder Wochentag ist. Die aktuelle Stunde und Minute wird in den Variablen *th* und *tm* aufgenommen: *th=dat.get(Calendar.HOUR_OF_DAY); und tm=dat.get(Calendar.MINUTE);* Da-

[1] vgl. [MK03, Seite 141]

nach wird die Zeitentfernung von der aktuellen Zeit subtrahiert. Die neue Zeit wird in *t* abgelegt. *t* ist *th*100+tm*. Anhand dieser Zeit wird die Abfahrtszeit ermittelt. Dabei werden alle Abfahrtszeiten des entsprechenden Tages (Wochentag, Samstag, Sonntag) gelesen und mit *t* verglichen. Sobald eine gleiche oder größere Abfahrtszeit gefunden wurde, wird diese in der Variablen *n* abgelegt. Die Abfahrtszeit erhält man durch Addition der Zeitentfernung zur ermittelten Abfahrtszeit *n*. Die ID der ermittelten Abfahrtszeit wird in *az* abgelegt, um damit dann durch die Abfahrtszeiten zu navigieren. In *azstart* wird die erste Abfahrtszeit gespeichert und in *azl* wird die Anzahl der Abfahrtszeiten abgelegt. Die Kommandos “Später”, “Früher” und “Zurück” werden hinzugefügt. Wird das Kommando “Später” aufgerufen, so wird die Abfahrtszeit mit der ID *az=az+1* angezeigt. Bei Betätigung des Kommandos “Früher” wird die Abfahrtszeit mit der ID *az=az-1* ausgegeben. Bei Erreichen von *azl* bzw. *azstart* wird der Variablen *az* der Wert von *azstart* bzw. *azl* zugewiesen. Das Kommando “Zurück” ruft die Methode *Screen5()* zur Anzeige der Haltestellen wieder auf. In Abbildung 5.1 sind die verschiedenen Methoden der Klasse *Diplom* dargestellt. Methode *Screen2()* kann, je nach Auswahl des Nutzers, die Methode *Screen4()* oder die Methode *Screen3()* aufrufen. Von *Screen4()* gelangt man zu den Methoden *Screen5()* und *Screen6()*. Die Funktionen der Methoden sind in der Abbildung 5.1 beschrieben. Die Quelltexte der Klasse *Diplom* und der XML-Dateien befinden sich auf der beiliegenden CD-ROM.

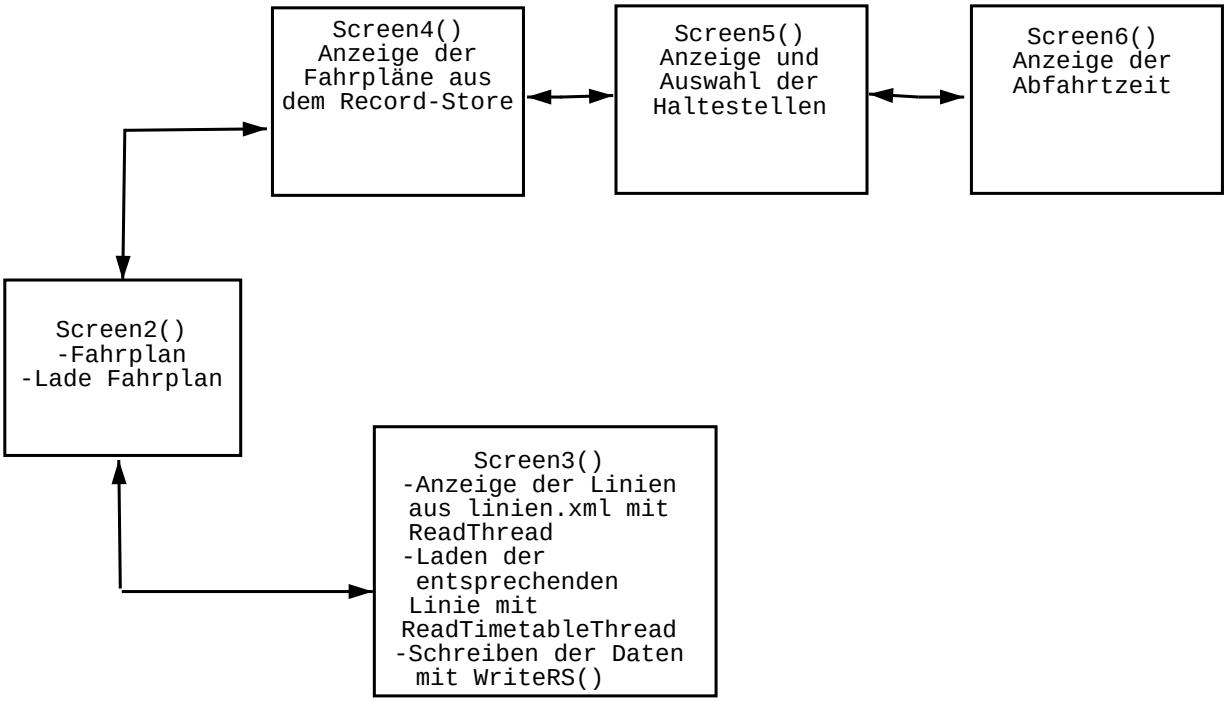

Abbildung 5.1: Methoden der Klasse *Diplom* und ihre Aufgaben

Kapitel 6

Implementierung

Nachdem das Programm gestartet wurde, erscheint der Bildschirm mit den Listenelementen “Fahrplan” und “Lade Fahrplan” (siehe Abbildung 6.1). Wählt man das Kommando “Beenden”, so wird das Programm beendet. Wenn “Lade neuen Fahrplan” selektiert wird, dann wird eine Verbindung zum Internet aufgebaut und die Datei linien.xml wird geladen. Danach werden die einzelnen Fahrpläne aus der Datei *linien.xml* in einer Liste zur Auswahl angezeigt. Wird ein Fahrplan selektiert, dann wird er von dem http-Server geladen und im Record-Store abgelegt. Mit “Zurück” gelangt man wieder zum Anfangsbildschirm. Mit der Auswahl von “Fahrplan” werden alle im Record-Store abgelegten Fahrpläne zur Auswahl angezeigt. Nachdem der Nutzer einen Fahrplan ausgewählt hat, werden die Haltestellen einer Linie angezeigt. Wählt man eine Haltestelle, wird die entsprechende Abfahrtszeit der Haltestelle angezeigt.

Der Nutzer kann mit “Später” oder “Früher” durch den Fahrplan der Haltestelle navigieren. Mit dem Kommando “Zurück” gelangt man jeweils zum vorherigen Bildschirm. Die Abbildungen 6.1, 6.2, 6.3, 6.4, 6.5 und 6.6 enthalten Screenshots der entwickelten Anwendung ausgeführt auf dem Emulator des WTK.

Abbildung 6.1: Anfangsbildschirm der Anwendung - der Nutzer hat die Möglichkeit einen gespeicherten Fahrplan zu wählen oder einen neuen Fahrplan vom Server zu laden

Abbildung 6.2: In diesem Bildschirm werden alle Fahrpläne angezeigt, die auf dem Server abgelegt sind. Selektiert man einen, wird dieser geladen und im Record-Store abgespeichert. In diesem Beispiel werden nur vier Fahrpläne bereitgestellt.

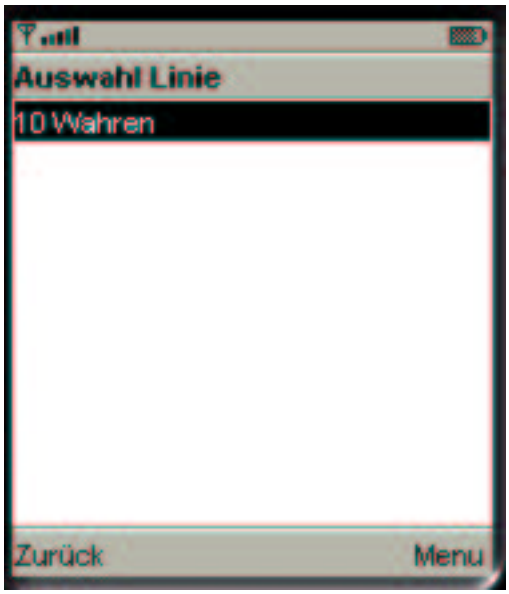

Abbildung 6.3: Hier werden alle Fahrpläne des Record-Stores angezeigt. Es befindet sich nur ein Farhplan “10 Wahren” im Record-Store.

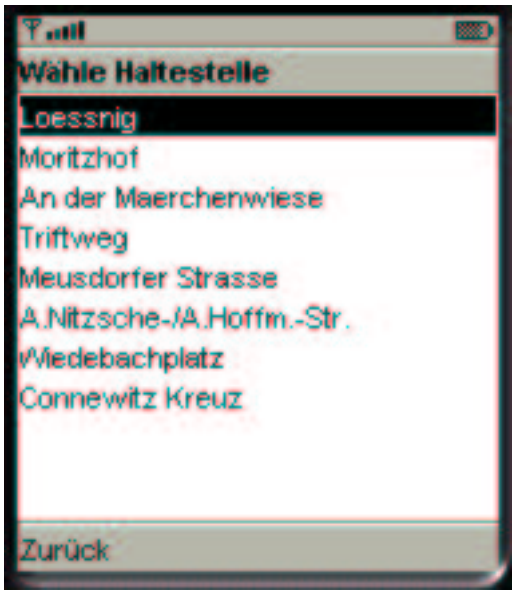

Abbildung 6.4: Nach Auswahl des Linien-Fahrplans werden alle Haltestellen angezeigt. Der Nutzer kann die gewünschte Haltestelle wählen. Der Nutzer wählt in diesem Beispiel die Haltestelle “Meusdorfer Strasse”.

Abbildung 6.5: Nach Auswahl der Haltestelle erscheint die Abfahrtszeit. Hier wird die Abfahrtszeit “06:06” angezeigt.

Abbildung 6.6: Nach Anzeige der Abfahrtszeit kann mit “Später” und “Früher” durch den Fahrplan navigiert werden. In diesem Fall wurde “Später” gewählt und die nächste Abfahrtszeit wird angezeigt.

Kapitel 7

Ergebnisse der praktischen Tests

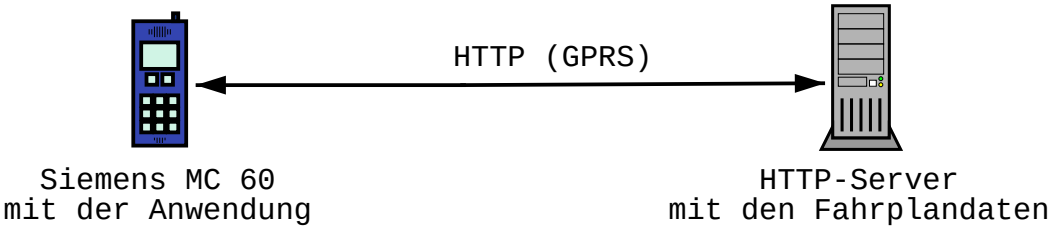

Abbildung 7.1: Testkonfiguration mit dem Siemens MC 60 und einem HTTP-Server. Für die Übertragung kommt das HTTP-Protokoll mit GPRS zum Einsatz.

Anwendungen laufen auf den Emulatoren schneller als auf den Endgeräten. Die in dieser Arbeit entwickelte Anwendung war im Siemens-Emulator (MC 60, C 55, SL 55) und im Nokia-Emulator (Nokia 7210) lauffähig. Für die Messungen wurde das GPRS-Handy Siemens MC 60 genutzt. Die Testkonfiguration ist in Abbildung 7.1 illustriert. Das MC 60 hat 2 MByte Speicherplatz .Diese 2 MByte teilen sich allerdings Java-Anwendungen, Bilder, MMS und Klingeltöne.[1], Für

[1] vgl. [wp04]

die Anwendung und die Fahrplandaten sind die 2 MByte ausreichend. Ein Fahrplan belegt ca. 3 KByte, abhängig von der Anzahl der Abfahrtzeiten. Die Anwendung hat eine Größe von 26 KByte. Das Laden der Anwendung dauerte mit dem MC 60 über eine GPRS-Verbindung in mehreren Tests weniger als 20 Sekunden. Die Dauer ist abhängig von der GPRS-Verbindung und der HTTP-Verbindung zum HTTP-Server. Weiterhin wurde das Antwort-Zeitverhalten der Anwendung untersucht. Die Zeit des Ladens eines Fahrplans auf das Mobilgerät bildet sich aus dem Laden der XML-Datei, dem Parsen der XML-Datei und dem Schreiben der Daten in den Record-Store. Die in den Tests verwendete XML-Datei hatte eine Größe von 1826 Bytes. Die Zeit-Messungen wurden mit der Methode System.current.TimeMillis() im Programm durchgeführt. Die Ergebnisse werden auf dem Mobilgerät angezeigt und werden in den Tabellen 7.1 und 7.2 aufgelistet. In Tabelle 7.1 wurde die Datei in 5 Versuchen geladen und im Record-Store abgespeichert. Die Mittelwerte betragen für das Laden der XML-Datei vom HTTP-Server 4460,6 ms, für das Parsen 16201,8 ms und für das Schreiben in den Record-Store 17862,6 ms. Die verschiedenen Messergebnisse sind auf die Architektur der Hard- und Software des Mobilgerätes zurückzuführen. Außerdem müssen parallel zur Ausführung der Anwendung noch andere Aufgaben vom Mobilgerät ausgeführt werden, die von äußerlichen Einflüssen abhängig sind. In Tabelle 7.2 wurde die Abfrage der Abfahrtszeit gemessen. Dabei wurde berücksichtigt, an welcher Position sich die Einträge im Record-Store befinden. Die Dauer des Ladens der XML-Datei ist abhängig von ihrer Größe und der http-Verbindung zum Server. Die Zeit des Parsens ist ebenfalls abhängig von der Größe der XML-Datei. Die Dauer des Schreibens in den Record-Store ist abhängig von der Anzahl der zu schreibenden Elemente.

Die Dauer einer Suche nach der Abfahrtzeit im Record-Store wurde auch gemessen. Sie ist abhängig von der Anzahl der Abfahrtzeiten-Einträge im Record-

Tabelle 7.1: Laden eines Fahrplans und Speicherung im Record-Store

Versuch	Laden der XML-Datei vom HTTP-Server	Parsen	Schreiben in Record-Store
1	4264 ms	16019 ms	18723 ms
2	4361 ms	16647 ms	17504 ms
3	4024 ms	17034 ms	17606 ms
4	5358 ms	15553 ms	17934 ms
5	4296 ms	15756 ms	17546 ms

Store und an welcher Position im Record-Store sich der Eintrag befindet.

Tabelle 7.2: Suchen der Abfahrtzeit im Record-Store. Hierbei wurden die Positionen der Einträge im Record-Store berücksichtigt.

Abfahrtzeit	Zeit
an 1. Position	1661 ms
an 10. Position	5727 ms
an 20. Position	9807 ms
an 30. Position	13393 ms
an 40. Position	16577 ms
an 50. Position	19198 ms
an 60. Position	21363 ms
an 70. Position	23006 ms
an 80. Position	24261 ms
an 90. Position	25031 ms

Kapitel 8

Zusammenfassung

8.1 Zusammenfassung

In dieser Arbeit konnte gezeigt werden, wie eine mobile Client-/Server Anwendung für mobile Endgeräte entwickelt werden kann. Dabei wurden verschiedene Technologien betrachtet und eingesetzt. Beispielmäßig wurde ein Prototyp für eine mobile Fahrplanauskunft entwickelt. Der Prototyp wurde in Java in Verbindung mit XML entwickelt. Die Fahrplan-Daten sind in XML-Dateien abgelegt. Als Server wurde der Apache WEB-Server eingesetzt. Die größten Zeitkosten der Anwendung entstanden durch das Parsen einer XML-Datei und den Zugriffen auf den Record-Store des Mobilgerätes. Die Datenübertragung über das Netzwerk lief mit angemessenen Geschwindigkeiten. (vgl. Kapitel Ergebnisse der praktischen Tests) Von der Programmierung von Anwendungen mit Record-Store-Zugriffen auf größere Datenmengen(größer als 3 KByte), als sie bei einer Fahrplanauskunft auftreten, ist abzuraten, da die Zugriffe auf die Daten im Record-Store zu langsam sind. Ein Vorteil dieser Anwendung ist, dass die Anwendungsdaten auf dem Endgerät gespeichert werden können. Damit entfallen Kommunikationsko-

sten. Ein Nachteil ist das langsame Antwort-Zeit-Verhalten bei der Abfrage der Abfahrtszeit. Aber mit der rasanten Entwicklung der Mobilfunktechnologie ist dies in Zukunft sicher zu bewältigen. Eine andere Möglichkeit um die langsamen Record-Store-Zugriffe zu umgehen, wäre die Anwendungdaten mit in die Anwendung zu packen. Damit würden auch die Zugriffe auf einen HTTP-Server entfallen und die Nutzer müssten sich nur noch die aktuelle Version auf das Mobilgerät installieren. Die Messungen wurden mit dem Siemens MC 60 durchgeführt, das als kostengünstiges Einsteigergerät zählt. Die gewonnenen Erkenntnisse sind auch zum Aufbau anderer Informationsdienste nutzbar. Die genutzten Entwicklungswerkzeuge sind im Internet kostenlos zu beziehen. Es wurden die passenden Entwicklungswerkzeuge ausgewählt und ihre Handhabung in dieser Arbeit beschrieben. Teile des Quellcodes sind auch für andere mobile Anwendungen wiederverwendbar.

8.2 Ausblick

Mit der zukünftigen Verfügbarkeit von UMTS und den damit einhergehenden höheren Bandbreiten werden umfangreichere Informationsdienste zu realisieren sein. Größere Datenmengen können schneller übertragen werden (theoretisch sind Datentransferraten von bis zu 2 MBit/s möglich). Die Leistungsfähigkeit der Mobilgeräte wird auch in Zukunft steigen. Mobiletelefone werden zu multimedialen Universalgeräten. Sie werden in Zukunft zum Kommunizieren, Steuern und zum Bezug von Informationen genutzt werden können. Die Nachfrage nach innovativen Anwendungen und Informationsdiensten wird damit steigen. Welche Technologien sich durchsetzen werden, ist noch nicht entschieden, da sich der Markt noch entwickelt. Folgende Erweiterungsfunktionen sind für eine zukünftige mobile Fahrplanauskunft denkbar:

- Routen-, Zeitplaner von A nach B
- Anzeige von Umsteigemöglichkeiten
- Preisauskunft und Bezahlmöglichkeit
- Lokalisierbarkeit um nächste Haltestelle anzuzeigen
- Anzeige eines Stadtplans in Form einer Grafik
- Anzeige des Liniennetzes in Form einer Grafik

8.3 Inhalt der CD

Auf der beiliegenden CD befinden sich die für die Entwicklung und den Betrieb der Anwendung notwendigen Daten und Programme:

- Java-Quelltext der Anwendung
- Beispiele eines Fahrplans (XML-Dateien)
- J2ME Wireless ToolKit 2.1
- Proguard-Obfuscator 2.1
- Retroguard-Obfuscator 1.1.14
- kXml-Parser

Tabellenverzeichnis

Abkürzungsverzeichnis

3G	3. Generation
CDC	Connected Device Configuration
CF	Compact Flash Card
CLDC	Connected Limited Device Configuration
GPRS	General Packet Radio Service
GSM	Global System for Mobile communications
HSCSD	High Speed Circuit Switched Data
HTTP	Hypertext Transfer Protocol
J2EE	Java 2 Enterprise Edition
J2ME	Java 2 Micro Edition
J2SE	Java 2 Standard Edition
JDK	Java Development Kit
JVM	Java Virtual Machine
KVM	K Virtual Machine
MIDP	Mobile Information Device Profile
MMC	Multi Media Card
PDA	Personal Digital Assistends
SD	Secure Digital Memory Card
SDK	Software Development Kit
SGML	Standard Generalized Markup Language

UMTS	Universal Mobile Telecommunications System
WAP	Wireless Application Protocol
XML	Extensible Markup Language

Abbildungsverzeichnis

Literaturverzeichnis

[Amm03] Dirk Ammelburger. *XML mit Java*. Markt und Technik, erste Auflage, 2003. ISBN 3-8272-6327-1.

[Apa04] Apache. *Apache HTTP Server Project*. http://httpd.apache.org/download.cgi, Stand:01.05.2004.

[Bra04] TU Braunschweig. *Vorlesung Mobilkommunikation*. http://www.ibr.cs.tu-bs.de/lehre/ss03/mk/pdf-2x2/K04-Drahtlose_Telekommunikationssysteme-p2.pdf, Stand:01.05.2004.

[CHI03] CHIP. *Handhelds im Härtetest*. PROFI-LINE, 2003.

[Dat04] Datenfunk. *MMS*. http://www.dafu.de, Stand:01.05.2004.

[gol04a] golem. *Borland: C++ Mobile Edition für mobile Applikationen*. http://www.golem.de/0211/22660.html, Stand:01.05.2004.

[gol04b] golem. *Borland: Kostenlose Entwicklungsumgebung für Series 60*. http://www.golem.de/0304/25255.html, Stand:01.05.2004.

[Grü04] Stefan Gründhammer. *Eine kurze Einführung in die Technologie von WAP*. http://www.aspheute.com/artikel/20000605.htm, Stand:01.05.2004.

[hac04] hacon. *hacon-sms*.
http://www.hacon.de/hafas/sms.shtml, Stand:01.05.2004.

[Hei04a] Heise. *Handy Galerie*.
http://www.heise.de/mobil/handygalerie, Stand:01.05.2004.

[Hei04b] Heise. *Nokia verliert Marktanteile auf dem weltweiten Handy-Markt*.
http://www.heise.de/newsticker/result.xhtml?url=/newsticker/data/anw-08.12.03-002/default.shtml&words=Gartner, Stand:01.05.2004.

[Hei04c] Heise.de. *Internet Timeline @ix*.
http://www.heise.de/ix/raven/Web/xml/timeline/default.phtml?what=java, Stand:01.05.2004.

[int04] integis. *Spieleprogrammierung unter J2ME*.
http//www.integis.ch/documents/j2me_gamedev-5-1.html, Stand:01.05.2004.

[KT03] Key Pousttchi Klaus Turowski. *Mobile Commerce, Grundlagen und Techniken*. Springer Verlag, erste Auflage, 2003. ISBN 3-540-00535-1.

[kXM04] kXML. *kXML*.
http://kxml.enhydra.org, Stand:01.05.2004.

[Leh03] Franz Lehner. *Mobile und drahtlose Informationssysteme*. Springer Verlag, erste Auflage, 2003. ISBN 3-540-43981-1.

[MB04] Mobile-Business. *Opera Reformats Existing HTML Pages for Mobile Devices*.
http://mobilebusinessadviser.com/doc/11308, Stand:01.05.2004.

[Mic04] Microsoft. *XML Kurzeinführung*. http://www.microsoft.com/germany/ms/officesystem/officeundxml/xmlkurzeinfuehrung.htm, Stand:01.05.2004.

[MK03] Stefan Haustein Michael Kroll. *J2ME Developer's Guide*. Markt und Technik, erste Auflage, 2003. ISBN 3-8272-6509-6.

[Mob04] Siemens Mobile. *Developer Portal*. http://www.siemens-mobile.com/developer, Stand:02.01.2004.

[Myr04] Dr. Thomas Myrach. *Grundlagen der Mobilkommunikation*. http://www.rwth-aachen.de/wt/lehre/lv/ss2002/EBmobile/pdf/mc01_grundlagen.pdf, Stand:01.05.2004.

[Nok04] Nokia. *Nokia 7210 SDK 1.0*. http://www.forum.nokia.com/main/1,6566,030,00.html?fsrParam=2-3-/main/1,6566,030,00.html&fileID=2703, Stand:01.05.2004.

[Nou04] Dana Nourie. *MIDP Setup Explained*. http://developers.sun.com/techtopics/mobility/midp/articles/setup/, Stand:01.05.2004.

[OMA04] OMA. *Open Mobile Alliance*. http://www.openmobilealliance.org, Stand:01.05.2004.

[Onl04] Onlinekosten.de. *i-mode und WAP*. http://www.onlinekosten.de/mobilfunk/imode_wap, Stand:01.05.2004.

[Pau04] W. Pauly. *WAP und WML*. http://www.htw-saarland.de/fb/gis/people/wpauly/vortraege_internet/ws99_00/WAP_und_WML/ausarb/index.html, Stand:01.05.2004.

[PP04] Palmtop-Pro. *SDKs für mobile Computer*. http://www.kefk.net/pda/PDF/Development/palmtopsdks.pdf, Stand:01.05.2004.

[Pro04] Proguard. *Proguard*. http//proguard.sourceforge.net, Stand:01.05.2004.

[Sch03] Jochen Schiller. *Mobilkommunikation*. Addison Wesley, zweite Auflage, 2003. ISBN 3-8273-7060-4.

[SR03] Gerd Siegmund Stephan Rupp. *Telefonie 2.0 - Java in der Telekommunikation. Grundlagen, Entwicklung, Anwendung*. Dpunkt Verlag, erste Auflage, 2003. ISBN 3-898-64244-5.

[SUN04a] SUN. *CDC Technology*. http://java.sun.com/products/cdc, Stand:01.05.2004.

[SUN04b] SUN. *Java 2 Platform, Micro Edition*. http://java.sun.com/j2me, Stand:01.05.2004.

[SUN04c] SUN. *Java 2 Platform Micro Edition*. http://java.sun.com/products/j2mewtoolkit/, Stand:01.05.2004.

[SUN04d] SUN. *Java 2 Platform, Standard Edition*. http://java.sun.com/j2se/1.4.1/download.html/, Stand:01.05.2004.

[SUN04e] SUN. *MIDP Emulators*. http://developers.sun.com/techtopics/mobility/midp/articles/emulators/, Stand:01.05.2004.

[SUN04f] SUN. *Parsing XML in J2ME*. http://developers.sun.com/techtopics/mobility/midp/articles/parsingxml/, Stand:01.05.2004.

[SUN04g] SUN. *The Source for Java Technology*. http://java.sun.com, Stand:01.05.2004.

[Sym04a] Symbian. *Symbian OS - the mobile operating system*. http://www.symbian.com, Stand:01.05.2004.

[Sym04b] Symbian. *Symbian OS phones*. http://www.symbian.com/phones/index.html, Stand:01.05.2004.

[Sys04] Retrologic Systems. *Retroguard*. http//www.retrologic.com/download.html, Stand:01.05.2004.

[Tan03] Andrew S. Tanenbaum. *Computernetzwerke*. Pearson Studium, vierte Auflage, 2003. ISBN 3-8273-7046-9.

[Tay04] Michael Taylor. *J2ME IDE Comparison*. http//www.microjava.com/articles/J2ME_IDE_Comparison.pdf, Stand:01.05.2004.

[tel04] teletarif.ch. *MMS-Technik*. http://www.teletarif.ch/i/mms-technik.html, Stand:01.05.2004.

[Wal00] Bernhard Walke. *Mobilfunknetze und ihre Protokolle, 2 Bde*. Teubner, zweite Auflage, 2000. ISBN 3-519-26430-7.

[wc04] www.tu chemnitz.de. *Infrarot*. http://www.tu-chemnitz.de/informatik/RA/kompendium/vortr_2001/notebook/irda.htm, Stand:01.05.2004.

[wp04] www.telekom presse.at. *MC 60*.
http://www.telekom-presse.at/channel_mobile/tests_handys_siemens.html,
Stand:01.05.2004.

[www04a] www.boeschatt.at. *Mobilfunk*.
http://www.boeschatt.at/Mobil/mobilfunk_html.php,
Stand:01.05.2004.

[www04b] www.wirelessdevnet.com. *Deliver Mobile Services Using XML and SOAP*.
http://www.wirelessdevnet.com/columns/feb2001/editor14.html,
Stand:01.05.2004.

[www04c] www.xemacs.org. *XEmacs*.
http//www.xemacs.org, Stand:30.01.2004.

[xml04] xmlpull.org. *Common API for XML Pull Parsing*.
http://xmlpull.org, Stand:01.05.2004.

[Ziv04] Dusan Zivadinovic. *Privatfunk*.
http://www.heise.de/mobil/artikel/2003/02/26/privatfunk,
Stand:01.05.2004.