Thema:

Entwicklung eines Museumsführers für multimediafähige Mobiltelefone

Diplomarbeit

Arbeitsgruppe Wirtschaftsinformatik
- Managementinformationssysteme -

Themensteller: Prof. Dr. Hans-Knud Arndt
Betreuer: Dipl.-Kfm. Henner Graubitz

vorgelegt von: Sebastian König

Abgabetermin: 22. Juli 2009

Inhaltsverzeichnis

Verzeichnis der Abkürzungen und Akronyme

AMS	Application Management Software
API	Application Programming Interface
BCC	Bluetooth Control Center
CDC	Connected Device Configuration
CLDC	Connected Limited Device Configuration
CMS	Content Management System
COD	Class of Device
DOM	Document Object Model
DTD	Document Type Definition
EDGE	Enhanced Data Rates for GSM Evolution
GCF	Generic Connection Framework
GPRS	General Packet Radio Service
GSM	Global System for Mobile Communications
HCI	Host Controller Interface
HSDPA	High Speed Downlink Packet Access
HTTP	Hypertext Transfer Protocol
ID	Identifikationsnummer
IrDA	Infrared Data Association
J2EE	Java 2 Enterprise Edition
J2ME	Java 2 Micro Edition
J2SE	Java 2 Standard Edition
JABWT	Java API for Bluetooth Wireless Technology
JAD	Java Application Descriptor
JAR	Java Archive
JCP	Java Community Process
JSR	Java Specification Request
JVM	Java Virtual Machine
KVM	Kilobyte Virtual Machine
L2CAP	Logical Link Control and Adaption Protocol
LC	Baseband Link Controller
LCDUI	Liquid Crystal Display User Interface
LMP	Link Manager Protocol
MIDP	Mobile Information Device Profile
MMAPI	Mobile Media API
OBEX	Object Exchange Protocol
PDA	Personal Digital Assistant
RFCOMM	Radio Frequency Communication Protocol
SAX	Simple API for XML
SDP	Service Discovery Protocol
SIM	Subscriber Identity Module
SMS	Short Message Service
UMTS	Universal Mobile Telecommunications System
URL	Uniform Resource Locator
UUID	Universal Unique Identifier
WLAN	Wireless Local Area Network
WPAN	Wireless Personal Area Network
WWW	World Wide Web
XML	Extensible Markup Language

Abbildungsverzeichnis

Quellcodeverzeichnis

Tabellenverzeichnis

1 Einleitung

1.1 Motivation

In unserer heutigen Informationsgesellschaft erfreuen sich Museen und Ausstellungshäuser nach wie vor ungebrochener Beliebtheit und sind ein Teil des kulturellen Lebens. Laut einer statistischen Erhebung des Instituts für Museumsforschung wurden im Jahr 2007 107 Mio. Museumsbesuche registriert, was einen Anstieg der Besuchszahlen um 4,5 % im Vergleich zum Vorjahr bedeutet.[1]

Museen sind Orte des Wissens, der Wissensvermittlung und der Wissensbewahrung. Sie bieten Besuchern unter anderem die Möglichkeit, Geschichte durch historische Gegenstände oder Ausgrabungsfunde zu erleben, Kunst in Form von Gemälden zu betrachten und die Biologie von Mensch, Tier und Pflanzen anhand von Präparaten zu entdecken.

Die traditionelle Art der Wissensdarbietung in Museen sind Schautafeln. Auf diesen erhält der Besucher in Text und Bild Erklärungen über die betrachteten Exponate. Mittlerweile greifen die Einrichtungen jedoch in zunehmendem Maße auf elektronische Hilfsmittel zur Informationspräsentation zurück. Als Hilfsmittel werden sowohl aus dem Bereich der Unterhaltungselektronik stammende, als auch speziell für Museen entwickelte Geräte, eingesetzt. Die Einrichtungen können damit ihr Informationsangebot um multimediale Inhalte erweitern und den Besucher bei der Informationsaufnahme sowie bei der Begehung der Ausstellungsräume unterstützen.

Der Einführung solcher elektronischen Hilfsmittel stehen jedoch hohe Investitionskosten gegenüber, wodurch die Realisierung für den Großteil der Einrichtungen aufgrund der angespannten Haushaltslage der öffentlichen Kassen nicht durchführbar ist.

Ziel dieser Arbeit ist es daher, ein System zu entwickeln, mit dem die hohen Investitionskosten vermieden werden und dem Besucher die Möglichkeit bietet, Textinformationen sowie multimediale Inhalte zu einem Exponat betrachten zu können.

1.2 Aufgabenstellung

Die im Rahmen der vorliegenden Arbeit untersuchte Themenstellung befasst sich mit der Entwicklung und Implementierung einer Anwendung für Mobiltelefone, die es Besuchern eines Museums oder einer Ausstellung ermöglichen soll, die Räumlichkeiten

[1] Institut für Museumsforschung (2008), S. 11.

individuell zu besichtigen und Informationen zu einzelnen Exponaten auf den Mobiltelefonen darzustellen. Neben reiner Textdarstellung sollen auch die Multimediafähigkeiten aktueller Mobiltelefone genutzt werden, um Bild-, Audio- und Videoinhalte präsentieren zu können.

Die Übermittlung der darzustellenden Informationen soll über eine im Mobiltelefon integrierte Netzwerkschnittstelle erfolgen. Die Bereitstellung der Informationen wird durch einen zentralen Datenserver vorgenommen, den die Mobiltelefonanwendung über die Netzwerkschnittstelle kontaktieren und somit die erforderlichen Daten abrufen kann. Die Implementierung des Servers sowie Einzelheiten der Datenhaltung auf dem Server sind nicht Gegenstand dieser Arbeit. Ein entsprechender Server, mit dem die Mobiltelefonanwendung kommuniziert, wird demnach für den Betrieb vorausgesetzt. Die Kommunikation und der Datenaustausch zwischen der Mobiltelefonanwendung und dem Datenserver sollen mittels der Extensible Markup Language erfolgen.

Um eine hohe Portabilität der Mobiltelefonanwendung zu gewährleisten, soll die Anwendung in einer Programmiersprache implementiert werden, die von der Mehrheit der Mobiltelefonbetriebssysteme unterstützt wird sowie keine zusätzlichen Änderungen am Quelltext und eine erneute Kompilierung der Anwendung notwendig macht.

1.3 Gliederung der Arbeit

Die vorliegende Arbeit ist in sechs Kapitel untergliedert. Im ersten Kapitel erfolgt eine kurze Einführung in die Thematik. Die Aufgabenstellung sowie die Zielsetzung werden genannt und der Aufbau der Arbeit wird beschrieben.

Das zweite Kapitel erläutert, wie Museumsbesucher die Erkundung der Ausstellungsinhalte vornehmen können und welche Anstrengungen seitens der Einrichtungen unternommen werden, um das Lernerlebnis durch die Bereitstellung elektronischer Hilfsmittel zu verbessern.

In Kapitel 3 werden die Grundlagen für die Entwicklung der Anwendung des Museumsführers erläutert. Dabei wird zunächst gezeigt, dass sich Mobiltelefone aufgrund ihrer Eigenschaften und Funktionen als Basis für die zu entwickelnde Anwendung nutzen lassen. Nach Auswahl einer geeigneten Programmiersprache werden die Besonderheiten und Einschränkungen bei der Anwendungsentwicklung für Mobiltelefone aufgezeigt und die für die Implementierung grundlegenden Technologien beschrieben.

Kapitel 4 befasst sich mit der Entwicklung und Umsetzung der Museumsführeranwendung, wobei zunächst die an die Anwendung gestellten Anforderungen vorgestellt werden und anschließend eine detaillierte Beschreibung der Infrastruktur für die Übertragung von Informationen auf das Mobiltelefon erfolgt. Die Implementierung wird anhand der für die Übertragung und Verarbeitung von Informationen wichtigen Schritte erläutert.

In Kapitel 5 erfolgt abschließend eine Demonstration der entwickelten Anwendung am Beispiel der Meisterhäuser Dessau.

Kapitel 6 fasst die durch die Arbeit gewonnenen Erkenntnisse zusammen und zeigt mögliche Verbesserungsvorschläge für weitere Untersuchungs- und Entwicklungsmaßnahmen der Anwendung auf.

2 Formen der Museumsführung

Besucher von Museen und Ausstellungen haben im Allgemeinen zwei Möglichkeiten, die Ausstellungsräume zu besichtigen und Informationen über Exponate zu erhalten. Die Museumsführung kann durch die Person des Museumsführers geleitet oder individuell durch den Besucher gestaltet werden.

Bei geführten Museumsbesichtigungen wird eine Gruppe von Besuchern von einem Angestellten der Einrichtung durch die Ausstellungsräume geleitet und über ausgewählte Exponate informiert. Merkmale geleiteter Touren sind, dass der Museumsführer sich mit der Thematik intensiv beschäftigt hat und sein Wissen gezielt sowie interaktiv an die Besucher weitergibt, d.h. die Führung sowohl didaktisch als auch methodisch aufbereitet und auf Nachfragen reagiert. Besonders für Interessierte, die keine oder nur geringe Vorkenntnisse besitzen, bietet diese Art der Besichtigung eine gute Möglichkeit, einen ersten Eindruck und einen Überblick über die Ausstellungsinhalte zu erhalten. Zusätzlich können die Besucher Unklarheiten sofort beseitigen und dem Museumsführer weiterführende spezifische Fragen zur Thematik stellen. Je nach persönlichem Empfinden können die geschilderten Merkmale jedoch nachteilig aufgefasst werden. Wenn sich Besucher für die geleitete Besichtigung entscheiden, sind sie an die Führung gebunden und ein vorzeitiger Abbruch ist selten möglich. Die Besucher sind in ihrer Entscheidungsfreiheit bezüglich der Auswahl der Informationen dahingehend eingeschränkt, dass der Museumsführer die Auswahl für sie bereits getroffen hat. Ein geführter Museumsrundgang nimmt weiterhin keine Rücksicht auf spezielle Interessen einzelner Personen oder auf besondere, nicht durch die Führung abgedeckte Inhalte. Für ausländische Museumsbesucher ist zudem die Frage nach den angebotenen Sprachen durch den Museumsführer relevant. Besonders in kleinen Museen und Ausstellungen werden die Führungen in der Landessprache gehalten. Die Sprachbegabtheit des Museumsführers entscheidet dann über weitere Sprachen. Ausländischen Besuchern kann dadurch der Inhalt der Führung verborgen bleiben. Des Weiteren können Wartezeiten für die Besucher entstehen, da geleitete Führungen zu festgelegten Terminen durchgeführt werden.

Als Alternative zur geleiteten Museumsführung ist der individuelle Rundgang der Besucher zu sehen, bei dem die Reihenfolge der Begehung der Ausstellungsräume selbstständig gewählt wird. Die Besucher entscheiden eigenständig und nach persönlicher Interessenlage, welche Exponate betrachtet werden und zu welchen Exponaten nähere Informationen erwünscht sind. Die maximale Dauer des Aufenthalts innerhalb der Ausstellungsräume ist lediglich durch die Öffnungszeiten beschränkt. Der Museumsbesuch kann jederzeit beendet werden. Bei der individuellen Museumsbesichtigung ist der Besucher auf die von der Einrichtung zur Verfügung

gestellten Informationsquellen angewiesen. Fragen zu stellen, wie bei den geleiteten Führungen, ist nicht unmittelbar möglich. Art und Umfang sowie die Gestaltung der Informationsquellen im Einzelnen ist abhängig von der jeweiligen Einrichtung. Typisch sind Schautafeln, Broschüren oder Handzettel. Diese können in verschiedenen Sprachen verfügbar sein.

Die Art der Informationsvermittlung in Museen und Ausstellungen hat sich im letzten Jahrzehnt grundlegend gewandelt. Neben der klassischen textbasierten Darbietung von Informationen werden die Ausstellungen zunehmend um multimediale Inhalte angereichert, mit denen Besucher zum Teil sogar interagieren können. Die multimedialen Inhalte können in Form von audiovisuellen Medien (z.B. Videosequenzen) oder rein auditiv (z.B. gesprochene Kommentare) vorliegen. Mit dem Einzug technischer Geräte aus den Bereichen der Unterhaltungselektronik sowie der Büro- und Telekommunikation wird zunehmend eine Erlebniswelt zum Anfassen und Ausprobieren geschaffen. Die zum Abspielen der multimedialen Inhalte verwendeten technischen Hilfsmittel reichen von stationär installierten Hörstationen, interaktiven Multimediaterminals, Einzelbildschirmen und Medienräumen bis hin zu tragbaren Geräten wie AudioGuide-Systemen.[2] Im Folgenden wird der Fokus dieser Arbeit aufgrund der Aufgabenstellung (vgl. Kapitel 1.2) auf AudioGuide-Systemen liegen.

Unter dem Begriff AudioGuide wird ein elektronisches Gerät verstanden, mit dem ein Besucher die Ausstellungsräume durchläuft und Informationen über Exponate in Form von Audiokommentaren abrufen kann. AudioGuides und stationäre Systeme unterstützen den Besucher bei der individuellen Besichtigung und bieten zusätzliche Informationen zu einzelnen Exponaten an. AudioGuides haben gegenüber stationären Systemen jedoch den Vorteil, dass diese dem Besucher während der gesamten Aufenthaltsdauer zur Verfügung stehen und somit keine Wartezeiten für den Geräte- und Informationszugriff entstehen.

Ursprünglich hatten AudioGuides die Form eines Telefonhörers. Durch Eingabe einer Nummer über das integrierte Ziffernfeld werden die Informationen zu einem Exponat selektiert und die Wiedergabe über eine andere Taste gestartet (vgl. Abbildung 2-1). Die auf den AudioGuides enthaltenen gesprochenen Texte liegen meist in verschiedenen Sprachen vor, aus denen der Besucher wählen kann. Des Weiteren besitzen die Geräte ein kleines Display zur Anzeige von Text, wie beispielsweise Informationen zum aktuell laufenden Audiokommentar und dessen verbleibender Laufzeit.

[2] Vgl. Iglhaut (2006), S. 68.

Abbildung 2-1: AudioGuide der Firma Audioguide Ltd.[3]

Mittlerweile offerieren einige Hersteller softwarebasierte AudioGuides, die auf einem Personal Digital Assistant (PDA) basieren und neben auditiven Informationen auch multimediale Inhalte wie Videosequenzen und Bilder bereithalten können. Ein Beispiel dafür ist der momentan im Bauhaus Dessau eingesetzte und in Abbildung 2-2 dargestellte AudioGuide. Die Bedienung der Software und der Start der multimedialen Elemente erfolgt über das berührungsempfindliche Display des Gerätes. Die Tonwiedergabe ist entweder über den integrierten Gerätelautsprecher oder über zur Verfügung gestellte Kopfhörer möglich.

Abbildung 2-2: Beispiel eines softwarebasierten AudioGuides auf einem PDA.[4]

[3] Die Abbildung zeigt ein AudioGuide-Gerät der Firma AudioGuide Ltd. und wurde entnommen aus AudioGuide Ltd. (2009).

[4] Eigene Fotografie, aufgenommen mit freundlicher Genehmigung des Bauhaus Dessau.

AudioGuides kombinieren die Vorzüge einer geleiteten Führung mit denen einer individuellen Besichtigung. In Tabelle 2-1 erfolgt eine Gegenüberstellung der Vor- und Nachteile, die AudioGuides im Vergleich zu einer geleiteten Führung aufweisen.

AudioGuides	
Vorteile	Nachteile
- ständige Verfügbarkeit von Informationen	- hohe Anschaffungskosten
- Bereitstellung von umfangreicheren Informationen als auf Schautafeln möglich	- Zusatzkosten für Wartung und Pflege der Geräte
- keine baulichen Veränderungen in den Ausstellungsräumen notwendig	- Übersendung der Geräte zum Hersteller bei Aktualisierung der Software notwendig
- beliebig häufige Wiederholung der Informationen möglich	- Verwaltung der Geräteausgabe
- Rundgänge erfolgen individuell und können jederzeit begonnen oder beendet werden	- keine Möglichkeit, inhaltliche Fragen zu stellen
- Unterstützung unterschiedlicher Sprachen	
- einfache und intuitive Bedienung der Geräte	
- Kosteneinsparung für freiwerdendes Führungspersonal	

Tabelle 2-1: Vor- und Nachteile von AudioGuides im Vergleich zu geleiteten Führungen.[5]

Zusammenfassend lässt sich festhalten, dass AudioGuides Besuchern einen individuellen Rundgang durch ein Museum oder eine Ausstellung erlauben und sie bei der Erkundung der Ausstellungsinhalte unterstützen. AudioGuides können die Besucher darüber hinaus mit mehr Informationen zu einzelnen Exponaten versorgen, als dies durch klassische Schautafeln möglich ist. Die Einrichtungen entscheiden selbst, in welchem Umfang und in welcher Art die Informationen zur Verfügung gestellt und aufbereitet werden.

Um AudioGuides in Museen und Ausstellungen in ausreichender Menge einzusetzen, müssen die Einrichtungen zunächst hohe Investitionskosten für die Geräte tätigen sowie zusätzlich anfallende Kosten für Wartung und Pflege der Geräte einplanen. Museen sind gemeinnützige Einrichtungen, die dem Non-Profit-Sektor zuzuordnen sind, im Gegensatz zu Wirtschaftsunternehmen, die profitabel arbeiten müssen. Eine Finanzierung erfolgt daher durch die erzielten Einnahmen wie Eintrittsgelder, aber hauptsächlich über öffentliche Gelder von Staat, Bundesland und Kommunen.[6]

[5] Tabelleninhalt wurde nach eigenen Überlegungen erstellt.
[6] Vgl. Bristot (2007), S. 30 f.

Inwiefern eine derartige Institution die benötigten Gelder für die Investition in AudioGuides zur Verfügung stellen kann, hängt somit meist von der Finanzlage des entsprechenden öffentlichen Haushalts des Trägers der Einrichtung ab. Eine Reinvestition aus erzielten Gewinnen ist nicht der Regelfall.

Die Themenstellung der vorliegenden Arbeit setzt an dieser Problematik an. Eine Möglichkeit, die hohen Anschaffungskosten der AudioGuides zu umgehen, ist die Nutzung von Geräten, die durch die Besucher selbst in das Museum mitgebracht werden. Die Geräte müssen die Wiedergabe von multimedialen Inhalten unterstützen sowie die Möglichkeit zur Installation von Software bieten. Eine Geräteklasse, die diese Kriterien erfüllt, sind handelsübliche Mobiltelefone. Mit der Verwendung dieser Geräte können seitens der Institutionen sowohl die Anschaffungskosten als auch die Kosten für Wartung und Pflege eingespart werden.

3 Technologische Grundlagen

3.1 Multimediafähige Mobiltelefone

Damit Mobiltelefone als Basis für einen Museumsführer verwendet werden können, müssen sie gewissen Anforderungen genügen. Aus der in Kapitel 1.2 beschriebenen Aufgabenstellung leiten sich folgende Anforderungen an ein Mobiltelefon ab:

- Unterstützung einer nachträglichen Installation von Softwareanwendungen,
- Darstellung von Textinformationen sowie Unterstützung diverser Multimediaformate wie Bild-, Audio- und Videoinhalte,
- Zwischenspeicherung von Daten auf dem Gerät zur Verarbeitung und Darstellung der Informationen sowie
- Verfügbarkeit einer integrierten Netzwerkschnittstelle für die Kommunikation und den Austausch von Daten zwischen der Mobiltelefonanwendung und einer Serveranwendung.

Bereits das weltweit erste kommerziell erhältliche tragbare Mobiltelefon der Firma Motorola aus dem Jahr 1983 verfügte über ein einzeiliges Display zur Anzeige der gewählten Rufnummer.[7] Mobiltelefone jener Zeit konnten ausschließlich zum Telefonieren genutzt werden. Sie waren in ihren Abmessungen sehr groß und bis zu 800 Gramm schwer. Mit der zunehmenden Miniaturisierung im Bereich der Halbleiter- und Computerchipindustrie sowie der damit einhergehenden Steigerung der Rechenleistung entwickelten sich Mobiletelefone zu immer kleineren, leichteren und leistungsstärkeren Geräten. Sinkende Gerätepreise und Verbindungsentgelte für Telefonie machten Mobiltelefone in den folgenden Jahren für weite Teile der Bevölkerung attraktiv. Erst die Integration einer neuen Kommunikationstechnologie, der Short Message Services (SMS) im Jahre 1992[8], führte dazu, dass Mobiltelefone zum Massenprodukt wurden. Seither haben weitere Kommunikationstechnologien und Dienste in Mobiltelefone Einzug gehalten, die eine Veränderung der Geräteeigenschaften bedingten.

Um Kommunikationsdienste, wie beispielsweise SMS, in der Praxis nutzen zu können, wurden die einzeiligen Displays durch mehrzeilige ersetzt. Bei der Entwicklung von Mobiltelefonen ging der Trend in den vergangen Jahren zu Multifunktionsgeräten, die immer mehr Funktionen in sich vereinten. Die Hersteller integrierten in zunehmendem Maße Geräte der elektronischen Unterhaltungsindustrie aber auch Funktionalitäten, die ursprünglich PCs und PDAs vorbehalten waren, wodurch sich neuartige

[7] Vgl. Motorola (2009).
[8] Vgl. Bitkom (2007).

Anwendungsmöglichkeiten der Mobiltelefone eröffneten. Aktuelle Geräte eignen sich unter anderem als portables UKW-Radio, MP3-Player für die Wiedergabe diverser Audio- und Musikformate, als Diktiergerät sowie als mobiler Massenspeicher für jegliche Art von Daten durch eine integrierte Speicherkarte. Eine eingebaute Digitalkamera ermöglicht die Aufnahme von Bildern und Videosequenzen, die anschließend in einem integrierten Multimedia-Player auf den farbigen Displays der Geräte wiedergegeben werden können. Über integrierte Netzwerkschnittstellen können Nutzer Netzwerke zwischen mehreren Geräten aufbauen und Daten austauschen. Kontaktdaten, Aufgaben und Termine werden innerhalb des Mobiltelefons verwaltet und gepflegt, wobei eine Synchronisation und ein Austausch der Daten bei Bedarf mit Desktopapplikationen oder anderen Geräten erfolgen kann. Mit Einführung des mobilen Internets auf Mobiltelefonen wurde auch der Kommunikationsdienst E-Mail in den Funktionsumfang aufgenommen und durch einen installierten Webbrowser ist der Zugriff auf das weltweite Datennetz möglich.

Die Fülle an integrierten Funktionalitäten in Mobiltelefonen lässt die Grenzen zwischen den Geräteklassen Mobiltelefon und PDA zunehmend verschwimmen. Den hybriden Charakter eines solchen neuen Gerätes unterstreicht die Bildung einer neuen Geräteklasse, den Smartphones.

Für den durchschnittlichen Nutzer ist eine Unterscheidung in Mobiltelefon und Smartphone meist nicht ohne Weiteres möglich. Smartphones sind Abkömmlinge von Mobiltelefonen, was sich in den Abmessungen der Geräte widerspiegelt und bieten neben Mobiltelefonfunktionen auch umfangreichere Organizerfunktionen. Ohne eine eingelegte SIM-Karte (Subscriber Identity Module) können diese allerdings, genau wie Mobiltelefone, nicht benutzt werden. Mobiltelefone sind in der Unterstützung von nachträglich installierten Anwendungen eingeschränkt. Die diesbezüglichen Einschränkungen werden in Kapitel 3.2 geschildert. Smartphones hingegen ermöglichen die nachträgliche Installation von Zusatzprogrammen, die in Anzahl und Leistungsfähigkeit meist denen der für Mobiltelefone verfügbaren Anwendungen überlegen sind.[9]

Sowohl Mobiltelefone als auch Smartphones erfüllen die eingangs beschriebenen Anforderungen, um die Software einer Museumsführer-Anwendung auf diesen Geräten betreiben zu können.

[9] Vgl. Opitz, Lüders (2006), S. 60 f.

3.2 Betriebssysteme und Programmiersprachen für Mobiltelefone

Das Betriebssystem des Mobiltelefons ist die primäre Schnittstelle zwischen dem Gerät und dem Benutzer. Auf diesem laufen die Anwendungen, die den Funktionsumfang des Mobiltelefons bestimmen. Das Betriebssystem ist somit für die Bereitstellung und Verwaltung der Hardwareressourcen wie Prozessor, Speicher, Ein- und Ausgabegeräte, etc. zuständig und steuert sowohl den Ablauf als auch die Abfolge der auf dem Gerät ausgeführten Prozesse und Anwendungen. Durch Abstraktion der zugrunde liegenden Hardware erhält zum einen der Nutzer eine Schnittstelle zur einfacheren Bedienung des Gerätes und zum anderen erhalten Anwendungsentwickler über standardisierte Programmierschnittstellen Zugriff auf die zur Verfügung stehenden Geräteressourcen.[10]

Die auf Mobiltelefonen zum Einsatz kommenden Betriebssysteme werden auch als eingebettete Systeme bezeichnet. Diese Systeme weisen Eigenschaften von Echtzeitsystemen auf, wie beispielsweise die Bereitstellung von Ressourcen für einen Prozess oder die Verfügbarkeit von Berechnungsergebnissen innerhalb eines festgelegten Zeitintervalls. Darüber hinaus sind eingebettete Systeme in besonderem Maße an die Eigenschaften der Geräte angepasst wie:

- geringe Gehäuseabmessungen,
- niedrige Rechenleistungen und Ressourcenverfügbarkeit sowie
- schonende Verwendung der Energieressourcen, da die Hauptenergieversorgung dieser Geräte durch Batterien gewährleistet wird.[11]

Die angebotenen Funktionalitäten der auf Mobiltelefonen und Smartphones eingesetzten Betriebssysteme unterscheiden sich voneinander. Mobiltelefone verwenden herstellereigene proprietäre, Smartphones hingegen geräteübergreifende Betriebssysteme. Letztere können in Abhängigkeit der Speicherverfügbarkeit in den Geräten um zusätzliche Softwareanwendungen erweitert werden. Um die Portabilität einer Anwendung zu erhöhen, ist die Wahl einer geeigneten Programmiersprache entscheidend. Nicht alle Programmiersprachen werden durch die verschiedenen auf dem Markt befindlichen Betriebssysteme gleichermaßen unterstützt. Wie bereits in Kapitel 3.1 beschrieben, können Mobiltelefone nur begrenzt um zusätzliche Programme erweitert werden. Meist handelt es sich dabei um Java-Anwendungen, deren Ausführung einer Java-Laufzeitumgebung bedarf.[12] Anwendungen einer anderen Programmiersprache können nicht auf dem Gerät installiert und ausgeführt werden. Auf Smartphones ist die Programmiersprachenunterstützung hingegen breiter gefächert. Je

[10] Vgl. Richter (2004), S.11 ff.
[11] Vgl. Tanenbaum (2002), S. 30 ff.
[12] Vgl. Opitz, Lüders (2006), S. 60 f.

nach konkretem Betriebssystem werden ein oder mehrere Programmiersprachen unterstützt.

Einen Überblick über die aktuelle Marktsituation geben Marktstudien der Firma Gartner Inc. Die Studien untersuchten für das Jahr 2008 sowohl die weltweiten Mobiltelefonverkäufe als auch die Marktanteile der Betriebssysteme, die auf Smartphones eingesetzt werden. Insgesamt wurden im Jahre 2008 weltweit 1,22 Mrd. Mobiltelefone an Endkunden verkauft. Dies entspricht einem Marktwachstum von ca. 6 % im Vergleich zum vorangegangenen Jahr.[13] Der Anteil an Smartphones lag bei ungefähr 11,4 % bzw. 139,3 Mio. Geräten. Die auf Smartphones eingesetzten Betriebssysteme veranschaulicht Abbildung 3-1.

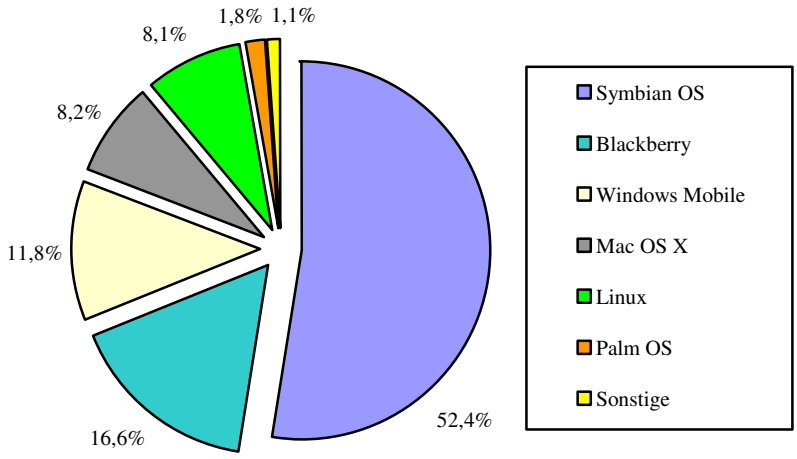

Abbildung 3-1: Smartphone-Betriebssysteme und deren weltweiter Marktanteil für das Jahr 2008.[14]

Mit einem Marktanteil von 52,4 % ist das *Symbian OS* des gleichnamigen Softwareunternehmens derzeit Marktführer im Segment der Betriebssysteme für Smartphones. Auf dem zweiten Platz befindet sich das *Blackberry* Betriebssystem der Firma Research In Motion mit einem Marktanteil von 16,6 %. Den dritten Platz belegt das Betriebssystem *Windows Mobile* der Firma Microsoft mit einem Marktanteil von 11,8 %. Seit dem Verkaufsstart des Apple iPhone am 29. Juni 2007[15] in den Vereinigten Staaten von Amerika besitzt das auf den Geräten eingesetzte *Mac OS X* Betriebssystem (iPhone OS) bereits einen Marktanteil von 8,2 % und befindet sich damit auf dem vierten Platz. Bei den *Linux*-basierenden Smartphone-Betriebssystemen existieren mehrere Ansätze nebeneinander, die insgesamt einen Marktanteil von 8,1 % aufweisen. Das *Palm OS* sowie die unter *Sonstige* zusammengefassten Betriebssysteme werden im

[13] Vgl. Gartner Inc. 1 (2009).
[14] Die Abbildung wurde basierend auf Daten einer Marktanalyse von Gartner Inc. 2 (2009) erstellt.
[15] Vgl. Apple Inc. (2007).

Folgenden nicht näher betrachtet, da diese mit einem Marktanteil von insgesamt 2,9 % nur eine untergeordnete Rolle einnehmen.[16]

In Abhängigkeit der auf Smartphones verwendeten Betriebssysteme können Anwendungsentwickler zwischen verschiedenen Programmiersprachen wählen. Kostenpflichtige Programmiersprachen und integrierte Programmierumgebungen sind ebenso verfügbar wie kostenlose. Die von Seiten der Betriebssystemanbieter offiziell unterstützten und in diesem Bereich verbreiteten Programmiersprachen sind in Tabelle 3-1 dargestellt.

	Smartphone-Betriebssystem				
Programmiersprache	Symbian OS	Blackberry	Mac OS X	Windows Mobile	Linux
Java	Ja	Ja	-	Ja	Teilweise
C/C++	Ja	-	-	-	Teilweise
Microsoft .Net	Ja	-	-	Ja	-
Cocoa Touch	-	-	Ja	-	-
Python	Ja	-	-	Ja	Teilweise

Tabelle 3-1: Verfügbarkeit von Programmiersprachen auf Smartphone-Betriebssystemen.[17]

Zu den vom *Symbian OS* offiziell unterstützten Programmiersprachen gehören Java, C/C++, das Microsoft .Net Framework sowie Python. Die Firma Research In Motion bietet mit dem *Blackberry* ein im besonderen Maße für den geschäftlichen Einsatz konzipiertes und in die Infrastrukturlandschaft des Unternehmens integrierbares Betriebssystem an, das Java-Applikationen unterstützt. Das auf dem Apple iPhone eingesetzte Betriebssystem ist eine speziell auf das Gerät angepasste Version des ansonsten auf Apple-Computern zum Einsatz kommenden Betriebssystems *Mac OS X*. Mit dem proprietären Cocoa Touch Framework haben Entwickler die Möglichkeit, in der Programmiersprache Objective-C multitouchfähige Applikationen für das iPhone zu implementieren. Eine Unterstützung weiterer Programmiersprachen ist derzeit nicht vorgesehen. Microsoft unterstützt auf dem *Windows Mobile* Betriebssystem die Ausführung von Applikationen, welche wahlweise in Java, Python oder mittels des Microsoft .Net Frameworks entwickelt wurden. Bei den *Linux*-basierten Betriebssystemen ist das Spektrum der Unterstützung von Programmiersprachen sehr weit gefächert. Zu den populärsten Open-Source Betriebssystemen gehören Openmoko (Openmoko Inc.), LiMo (LiMo-Foundation) sowie das Android-Betriebssystem (Open Handset Alliance). Je nach Offenheit der Systeme kann die fehlende

[16] Vgl. Gartner Inc. 2 (2009).
[17] Eigene Darstellung, basierend auf Angaben der jeweiligen Betriebssystemhersteller.

Programmiersprachenunterstützung entweder über Softwaremodule nachträglich installiert werden, wie zum Beispiel beim Openmoko-Betriebssystem, oder die Systeme bleiben auf einzelne Sprachen beschränkt. Die Android-Plattform sieht lediglich eine Unterstützung der Programmiersprache Java vor, während das Betriebssystem der LiMo-Foundation mit der aktuellen R1 Plattform die Sprachen C und C++ unterstützt.

Die Aufgabenstellung (vgl. Kapitel 1.2) beinhaltet die Entwicklung der Museumsführer-Anwendung in einer Programmiersprache, die durch die Mehrzahl der Geräte unterstützt wird. Laut der Studie von Gartner Inc. verfügen 88,6 % der weltweit im Jahr 2008 verkauften Mobiltelefone über eine integrierte Java-Laufzeitumgebung. Die restlichen 11,4 % stellen Smartphones dar, die Java zu mehr als Zweidritteln unterstützen. Damit ist die Programmiersprache Java für die Bearbeitung der Problemstellung geeignet und wird für die Entwicklung der Museumsführer-Anwendung auf Basis von Mobiltelefonen verwendet.[18]

3.3 Java

3.3.1 Plattformübersicht

Der Begriff Java wird meist für verschiedene Komponenten der von der Firma Sun Microsystems Inc. entwickelten Java-Technologie synonym verwendet. Zu den Komponenten von Java zählen die objektorientierte Programmiersprache Java, die zur Ausführung von Java-Anwendungen benötigte Java Virtual Machine (JVM) sowie die Java-Plattform, welche die bei der Programmierung von Anwendungen verwendbaren Java-Klassenbibliotheken definiert. Java eignet sich sowohl für die Entwicklung von Web-basierten Anwendungen als auch für die Implementierung von Programmen, welche auf einer Vielzahl von unterschiedlichen Gerätetypen und Betriebssystemen lauffähig sind. Die Internetfähigkeit und Plattformunabhängigkeit von Java-Anwendungen basiert auf der Portierung der JVM auf die einzelnen Geräte, Betriebssysteme sowie Internet-Browser. Mittlerweile stehen JVMs für alle gängigen Desktop-Betriebssysteme und darüber hinaus für Betriebssysteme mobiler Endgeräte, wie beispielsweise PDAs, Mobiltelefone und Chipkarten, zur Verfügung. Während in den ersten Java-Versionen die Anzahl an zur Verfügung stehenden Java-Klassenbibliotheken noch recht überschaubar blieb, kamen mit jeder nachfolgenden Version zahlreiche weitere hinzu.

[18] Im weiteren Verlauf der Arbeit wird der Begriff Mobiltelefon synonym für klassische Mobiltelefone und Smartphones verwendet.

Um eine bessere Abgrenzung der Klassenbibliotheken für die einzelnen Anwendungsbereiche und Zielgeräte zu erreichen sowie Java weiterhin effektiv bei der Entwicklung von Anwendungen einsetzen zu können, wurde die Java-Plattform restrukturiert und untergliedert. Die daraus resultierende Aufteilung der Java-Plattformen in Java 2 Enterprise Edition (J2EE), Java 2 Standard Edition (J2SE), Java 2 Micro Edition (J2ME) und Java Card verdeutlicht Abbildung 3-2. Jede der vier Plattformen eignet sich für einen spezifischen Anwendungsbereich bzw. eine bestimmte Geräteklasse:

- *J2EE*: Entwicklung von Server-Applikationen im betrieblichen Umfeld,
- *J2SE*: Entwicklung von Client-Applikationen für PCs und Workstations,
- *J2ME*: Entwicklung von Applikationen für mobile Endgeräte wie PDAs, Mobiltelefone und andere Kleinstcomputer sowie
- *Java Card*: Entwicklung von Anwendungen auf Chipkarten.[19]

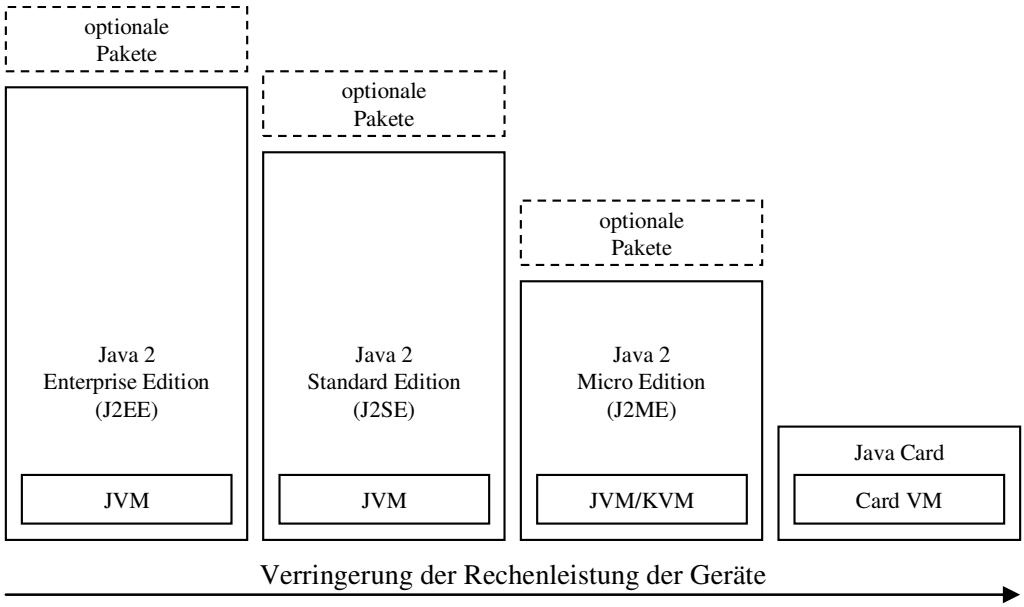

Abbildung 3-2: Untergliederung der Java-Plattformen.[20]

Die einzelnen Java-Plattformen definieren jeweils die bei der Programmierung zur Verfügung stehenden Klassenbibliotheken. Klassen, die eine bestimmte Funktionalität erfüllen, z.B. Netzwerkzugriff, Dateizugriff oder Grafikausgabe, werden in Java zu Paketen zusammengefasst. Der Plattform können optional weitere Pakete hinzugefügt werden. Die Mächtigkeit und verfügbare Anzahl an Klassenbibliotheken der einzelnen Java-Plattformen sowie die zum Einsatz kommende JVM ist abhängig von der

[19] Vgl. Abts (2000), S. 6 f.
[20] Abbildung in Anlehnung an Sun Microsystems Inc. (2009).

vorhandenen Rechenleistung und den Ressourcen der Endgeräte, für die die Java-Plattform konzipiert wurde. Je geringer die Rechenleistung der Endgeräte, desto geringer sind der Umfang der verfügbaren Klassenbibliotheken und die Funktionalität der JVM. Die J2EE umfasst als einzige Plattform die Gesamtheit der Java-Spezifikationen. Alle anderen Plattformen leiten sich aus der nächst größeren Plattform ab (vgl. Abbildung 3-2 - linker Nachbar). Die reine Reduzierung des Plattformumfangs bei der J2ME und der Java Card-Plattform war aber nicht ausreichend, um diese auf den hardwareseitig eingeschränkten Geräten verwenden zu können. Aus diesem Grund wurden in diesem Bereich einige Klassenbibliotheken modifiziert oder neu entwickelt, um der Leistungsfähigkeit der Geräte gerecht zu werden. Während für die J2EE und die J2SE eine vollwertige JVM verwendet werden kann, kommt bei der J2ME geräteabhängig eine vollwertige JVM oder eine speziell auf die Gegebenheiten von mobilen Endgeräten angepasste virtuelle Maschine, die Kilobyte Virtual Machine (KVM), zum Einsatz. Auch für die Java Card-Plattform wurde eine neue virtuelle Maschine entworfen, die sog. Card VM.

Im Rahmen dieser Arbeit wird die J2ME-Plattform verwendet und daher in den folgenden Kapiteln näher betrachtet. Es werden dabei auch Besonderheiten und Einschränkungen bei der Verwendung dieser Plattform und bei der Entwicklung von Applikationen für mobile Endgeräte aufgezeigt.

3.3.2 Java Community Process

Java wird durch eine internationale Entwicklergemeinschaft gestützt, welche aus Privatpersonen und namhaften Unternehmen der Software- und Unterhaltungselektronikindustrie besteht. Im Jahre 1998 führte die Firma Sun Microsystems Inc. den Java Community Process (JCP) ein, der eine Zusammenarbeit von Sun Microsystems und der Java-Entwicklergemeinschaft bei der Wartung und Pflege vorhandener Java-Spezifikationen gewährleistet sowie die Weiterentwicklung der Java-Technologie vorantreiben sollte. Während für Privatpersonen die Mitgliedschaft am JCP kostenlos ist, wird für Organisationen und Unternehmen eine jährliche Verwaltungsgebühr erhoben.

Das Program Management Office, eine Gruppe, die innerhalb der Firma Sun Microsystem eingerichtet wurde, überwacht und steuert den JCP.[21] Die Entwicklergemeinschaft unterbreitet dem Program Management Office Vorschläge bezüglich der Veränderung oder Neuentwicklung einer Spezifikation. Diese Vorschläge

[21] Vgl. Java Community Process (2009).

werden als Java Specification Requests (JSR) bezeichnet. Nachdem ein Vorschlag angenommen wurde, wird eine Expertengruppe aus Mitgliedern des JCP gebildet. Diese erarbeiten einen ersten Spezifikationsvorschlag, welcher der Java-Entwicklergemeinschaft zur Begutachtung und Abstimmung vorgelegt wird. Mögliche Änderungen seitens der Entwicklergemeinschaft werden eingearbeitet und in einer finalen Spezifikation zusammengeführt. Die in den Spezifikationen festgelegten Standards sind die Grundlagen für Erweiterungen und Änderungen der Programmiersprache Java sowie der Java-Plattformen. Derzeit existieren insgesamt 225 JSR für die verschiedenen Plattformen, wobei allein auf die J2ME-Plattform 83 Spezifikationen entfallen. Die JSRs der J2ME-Plattform decken unter anderem die Bereiche Laufzeitumgebung, Anwendungslebenszyklus, Netzwerk, Multimedia etc. ab.

3.3.3 Java 2 Micro Edition

Die J2ME-Plattform besteht aus eine Untermenge der in der J2SE vorhandenen Klassenbibliotheken und wurde speziell für die Entwicklung von Anwendungen auf eingebetteten Systemen und mobilen Endgeräten konzipiert. Die Architektur der J2ME basiert auf mehreren Schichten, deren Grundlage die Konfigurationen bilden. Konfigurationen legen die an die Geräte gestellten Mindesthardwareanforderungen in Bezug auf Rechenleistung, Speicherressourcen sowie Netzwerkfunktionalität fest, definieren die Klassenbibliotheken, die den Kern für die Anwendungsentwicklung bilden, und beinhalten die zur Ausführung der Anwendungen zu verwendende JVM. Profile setzen auf den Konfigurationen auf und erweitern diese beispielsweise um Programmierschnittstellen für die Entwicklung grafischer Benutzeroberflächen und Möglichkeiten der dauerhaften Datenspeicherung auf den Geräten. Während Konfigurationen minimale Hardwareanforderungen und Programmierschnittstellen für eine Geräteklasse festlegen, beziehen sich Profile verstärkt auf die Eigenschaften und Funktionalitäten eines gewissen Gerätetyps.[22] Aufgrund der Vielfalt an Geräteklassen mit unterschiedlichen Funktionalitäten, Rechenleistungen und Speicherressourcen im Bereich der eingebetteten und mobilen Endgeräte wurde die J2ME in zwei Bereiche aufgeteilt:

- die Connected Device Configuration (CDC) und
- die Connected Limited Device Configuration (CLDC).

Die Schichtenarchitektur sowie die Aufspaltung der J2ME in zwei Bereiche bietet den Vorteil, dass je nach Leistungsfähigkeit des Endgerätes, für welches eine Anwendung

[22] Vgl. Kroll, Haustein (2003), S. 25.

entwickelt wird, nach einer Art Baukastenprinzip die passende Basis sowie zusätzliche Komponenten und Funktionalitäten ausgewählt werden können. Darüber hinaus ist es möglich, auf die sich rasant entwickelnde Rechenleistungs- und Funktionszunahme der Geräte zu reagieren, indem Teile der Plattform einfach und flexibel ausgetauscht oder um zusätzliche Funktionen erweitert werden können (vgl. Abbildung 3-3).

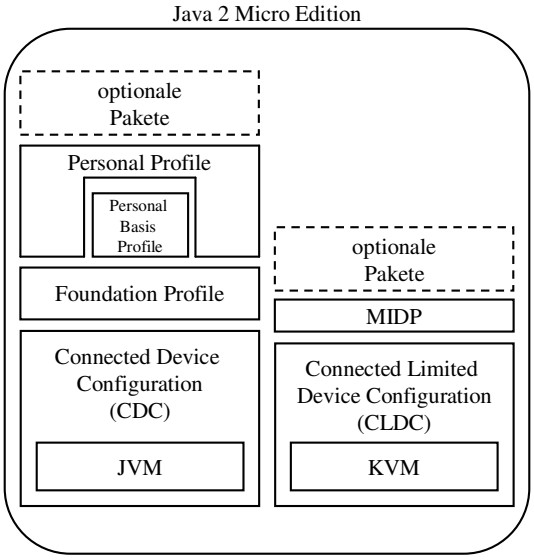

Abbildung 3-3: Darstellung der Schichtenarchitektur der J2ME-Plattform.[23]

Die CDC wird bei der Programmierung von Anwendungen auf leistungsfähigen PDAs, Settop-Boxen etc. eingesetzt. Diese Geräte verfügen über eine ausreichende Hardwareausstattung, um eine vollwertige JVM ausführen zu können. Für die Erweiterung der CDC stehen das Foundation Profile, das Personal Basis Profile und das Personal Profile zur Verfügung. Für leistungsschwächere Geräte, wie beispielsweise Einsteiger-PDAs, Mobiltelefone und Pager, wird die CLDC verwendet. Darüber hinaus kommt auf diesen Geräten durch deren verminderte Rechenleistung und Speicherressourcen eine angepasste virtuelle Maschine, die KVM, zum Einsatz. Für die CLDC existiert derweil lediglich das Mobile Information Device Profile (MIDP).[24] Zusätzlich zu den genannten Profilen kann die J2ME-Plattform durch weitere optionale Pakete erweitert werden (vgl. Abbildung 3-3).

3.3.4 Connected Limited Device Configuration

Die Basis bei der Entwicklung von Java-Anwendungen für Einsteiger-PDAs, Mobiltelefone, Pager und andere mobile Endgeräte vergleichbarer Rechenleistung und

[23] Abbildung in Anlehnung an Sun Microsystems Inc. (2009).
[24] Vgl. Kroll, Haustein (2003), S. 25.

Speicherressourcen stellt die CLDC der J2ME-Plattform dar. Diese definiert die Mindestanforderungen, über die ein Gerät dieser Geräteklasse verfügen muss, um die J2ME-Plattform verwenden zu können sowie die Kern-Klassenbibliotheken, Klassenbibliotheken für Netzwerkfunktionalität und Ein- und Ausgabeoperationen. Ferner werden der Leistungsumfang der virtuellen Maschine und der Sprachumfang der Programmiersprache Java festgelegt. Als virtuelle Maschine wird die KVM verwendet, die in besonderem Maße an die Hardwarevoraussetzungen der Geräte angepasst und in ihrer Funktionalität im Vergleich zur JVM stark eingeschränkt ist. Die Voraussetzungen sind unter anderem ein 16-Bit- oder 32-Bit-Prozessor, ein geringer Energieverbrauch, da die Energieversorgung durch Batterien erfolgt, sowie eine drahtlose Netzwerkverbindung. Die Anforderungen sind so allgemein wie möglich gehalten, um eine breite Abdeckung an Geräten zu erreichen, die unter der CLDC eingeordnet werden können.[25] Die Spezifikation der CLDC liegt derzeit in zwei Versionen vor, die CLDC 1.0 sowie CLDC 1.1. Die Ursprungsversion CLDC 1.0 ist in der JSR-30 spezifiziert. Die CLDC 1.0 sieht mindestens 128 KByte nicht-flüchtigen Speicher für Klassenbibliotheken und die KVM sowie mindestens 32 KByte flüchtigen Speicher vor, der zur Programmlaufzeit für Variablen- und Objektdefinitionen verwendet werden kann.[26] Die durch die CLDC definierten Klassenbibliotheken können in zwei Bereiche untergliedert werden. Der erste Bereich ist die aus den J2SE-Klassenbibliotheken abgeleitete Untermenge `java.lang.*`, `java.util.*` und `java.io.*`, welche die Systemklassen, Datentypen und Fehlerklassen, Zeit- und Kalenderklassen, etc. definieren.[27] Der zweite Bereich sind die CLDC-spezifischen Klassenbibliotheken `javax.microedition.io.*`, durch die das Generic Connection Framework (GCF) eingeführt wird, welches eine einheitliche Möglichkeit des Zugriffs auf Netzwerkfunktionalitäten sowie auf Ein- und Ausgaben bietet.[28] Bedingt durch die funktionalen Einschränkungen der KVM und der Geräte ergeben sich Besonderheiten bei der Verwendung der CLDC. Aufgrund des Fehlens entsprechender Rechenelektronik in der Mehrzahl der Geräte, bietet die CLDC 1.0 keine Unterstützung von Fließkommadatentypen. Eine softwareseitige Berechnung von Fließkommazahlen ist möglich, jedoch würde dies die vorhandene Rechenleistung und die Speicherressourcen der Geräte übersteigen.[29] Damit vor der Ausführung einer Java-Anwendung sichergestellt werden kann, dass die bei der Kompilierung aus dem Quelltext erzeugten Bytecodes die virtuelle Maschine nicht beeinträchtigen oder Speicherbereiche verletzen, müssen diese verifiziert werden. Bei vollwertigen JVMs

[25] Vgl. JSR-30 (2000), Kapitel 1, S. 1.
[26] Vgl. JSR-30 (2000), Kapitel 2, S. 3.
[27] Vgl. JSR-30 (2000), Kapitel 6, S. 2 ff.
[28] Vgl. JSR-30 (2000), Kapitel 6, S. 8 ff.
[29] Vgl. JSR-30 (2000), Kapitel 5, S. 1.

erfolgt diese Überprüfung vor jedem Ausführen einer Anwendung. Da dieser Prozess zu rechen- und ressourcenintensiv für mobile Endgeräte auf CLDC-Basis ist, wird der Verifikationsprozess für die KVM in zwei Phasen unterteilt. Die erste Phase erfolgt auf dem Entwicklungsrechner, wo im Anschluss an die Kompilierung des Quelltextes, die erzeugten Bytecodes für den Verifikationsprozess vorverarbeitet werden. Das Ergebnis dieser Prä-Verifikation sind sog. Stack Maps, welche an die Bytecode enthaltenden Klassen-Dateien angehängt werden und der effizienteren Verifikation auf den mobilen Endgeräten dienen. Stack Maps sind spezielle Attribute, die unter anderem die Datentypen lokaler Variablen und die zur Laufzeit der Anwendung erzeugten Objektinstanzen sowie deren Speicherbedarf abbilden. Die zweite Phase erfolgt bei der Ausführung der Anwendung auf dem mobilen Endgerät. Zur Laufzeit überprüft die KVM auf Grundlage der aus den Klassen-Dateien ausgelesenen Stack Maps die Richtigkeit des Bytecodes. Die Anwendung wird nur durch die KVM ausgeführt, sofern die Überprüfung erfolgreich abgeschlossen wurde.[30]

Durch die Weiterentwicklung der mobilen Endgeräte in Bezug auf Rechenleistung und Speicherverfügbarkeit, wurden mit Einführung der CLDC 1.1 einige Restriktionen der CLDC 1.0 aufgehoben bzw. gelockert. Die in dem JSR-139 spezifizierte CLDC 1.1 sieht mindestens 160 KByte nicht-flüchtigen Speicher statt der bisherigen 128 KByte vor. Des Weiteren wurden zahlreiche Klassen und Datentypen hinzugefügt oder überarbeitet. Die CLDC 1.1 bietet nun beispielsweise eine Unterstützung von Fließkommadatentypen an. Die Verifikation des Bytecodes kann wahlweise mit einer der beiden vorgestellten Verifikationsmethoden durchgeführt werden.[31] Mit der Wartungsversion CLDC 1.1.1, die ebenfalls in der JSR-139 enthalten ist, wurden das Sicherheitsmodell der CLDC 1.1 erweitert, zusätzliche trigonometrische Funktionen aufgenommen sowie weitere Klassen aus der J2SE Version 1.3.1 hinzugefügt.[32]

Die durch das mobile Endgerät unterstützten Konfigurationen können über die Systemeigenschaften zur Laufzeit der Anwendung abgefragt werden. Eine derartige Abfrage wird durch das folgende Quellcodefragment illustriert:

```
String config = System.getProperty("microedition.configuration");
```

Quellcode 3-1: Abfrage der Systemeigenschaften nach unterstützten Konfigurationen.

Der Rückgabewert der Abfrage enthält sowohl den Namen als auch die Versionsnummer der unterstützten Konfiguration. Bei einer Unterstützung der CLDC in der Version 1.1 ist der Rückgabewert der Funktion gleich `"CLDC-1.1"`.

[30] Vgl. JSR-30 (2000), Kapitel 5, S. 5 f.
[31] Vgl. JSR-139 (2007), Kapitel 1, S. 3.
[32] Vgl. JSR-139 (2007), Kapitel 1, S. 4.

3.3.5 Mobile Information Device Profile

Wie bereits in Kapitel 3.3.3 beschrieben, setzt das MIDP auf der CLDC auf und erweitert diese um weitere Klassenbibliotheken und Funktionen. Im Gegensatz zum CLDC geht das MIDP näher auf die spezifischen Eigenschaften von Mobiltelefonen und Pagern ein. In der Spezifikation JSR-37 wird die Version 1.0a des MIDP beschrieben, die als Grundlage für die Weiterentwicklung der Versionen 2.0 und 2.1 dient, welche im JSR-118 spezifiziert sind. Für die Verwendung der MIDP-Versionen wird eine der in den JSR-30 oder JSR-139 spezifizierten Konfigurationen vorausgesetzt. Die MIDP-Spezifikationen legen weitere Mindestanforderungen an die Hardware und Eigenschaften mobiler Endgeräte fest, die zusätzlich zu den in der CLDC aufgeführten Anforderungen gelten. Die wichtigsten Gemeinsamkeiten und Unterschiede der verschiedenen MIDP-Versionen bezüglich der Mindesthardwareanforderungen an mobile Endgeräte sind in Tabelle 3-2 gegenübergestellt.

	MIDP 1.0a	MIDP 2.0/2.1
Bildschirmabmessungen	96 x 54 Pixel	
Eingabegeräte	mindestens eine der folgenden Eingabemöglichkeiten: - Zifferntastatur (ITU-T Telefontastatur)[33] - Buchstabentastatur - berührungsempfindlicher Bildschirm	
Speicher	nicht-flüchtiger Speicher für die MIDP-Implementierung: 128 KByte 256 KByte nicht-flüchtiger Speicher für Anwendungsdaten: 8 KByte flüchtiger Speicher für die Java-Laufzeitumgebung: 32 KByte 128 KByte	
Netzwerk	drahtlose Sende-/Empfangseinheit mit limitierter Bandbreite	
Tonwiedergabe	Nein	Ja (Hardwarechip oder Softwarealgorithmus)

Tabelle 3-2: Gegenüberstellung der Mindesthardwareanforderungen der MIDP-Versionen.[34]

In Tabelle 3-2 sind die MIDP-Versionen 2.0 und 2.1 zusammengefasst, da sich die Hardwareanforderungen zwischen den Versionsschritten nicht verändert haben. Die

[33] Die International Telecommunication Union definiert im weltweit verwendeten Standard ITU-T E.161 die Anordnung und Belegung der Tasten von Telefontastaturen. Optional ist eine Belegung der Zifferntasten mit Buchstaben und Symbolen vorgesehen. Beispielsweise sind die Buchstaben A, B und C auf der Zifferntaste 2 hinterlegt. Vgl. International Telecommunication Union (2001), S. 1.

[34] Vgl. JSR-37 (2000), S. 21 f. und JSR-118 (2006), S. 16.

MIDP-Version 2.1 stellt eine Überarbeitung der Version 2.0 dar, in der zahlreiche Änderungen an der Spezifikation vorgenommen wurden.

Die durch die MIDP-Versionen definierten Mindestanforderungen gleichen sich in einigen Spezifikationspunkten, wie beispielsweise den Bildschirmeigenschaften, den unterstützten Eingabegeräten sowie der Netzwerkanbindung der Geräte. Unterschiede ergeben sich bei den benötigten Speicherressourcen für die MIDP-Implementierung und der Java-Laufzeitumgebung sowie in Bezug auf die fehlende Unterstützung von Tonwiedergaben der MIDP-Version 1.0a. Neben den genannten hardwareseitigen Anforderungen werden durch die MIDP-Versionen, zusätzlich zu den in der CLDC definierten Klassen, weitere Klassenbibliotheken und Fehlerklassen eingeführt sowie der Lebenszyklus der Anwendung festgelegt. So wurden beispielsweise zur Ausführung wiederholbarer oder in der Zukunft auszuführender Aufgaben während der Laufzeit der Anwendung die Klassen Timer und TimerTask aufgenommen. Dem GCF wurde ferner ein Interface für das Hypertext Transfer Protocol (HTTP) hinzugefügt. Darüber hinaus werden durch die MIDP-Spezifikation Klassen eingeführt, die es den Anwendungen erlauben, auf den internen Gerätespeicher zuzugreifen und Daten persistent in sog. Record Stores abzulegen. Für die Entwicklung grafischer Benutzeroberflächen wird das Paket Liquid Crystal Display User Interface (LCDUI) bereitgestellt, das nicht mit den in der J2SE verwendeten grafischen Schnittstellenbibliotheken Abstract Windowing Toolkit und Swing vergleichbar ist. Das Paket LCDUI ist in einen High-Level und einen Low-Level Bereich für die Programmierung von grafischen Benutzerschnittstellen unterteilt. Der High-Level Bereich stellt vorgefertigte Formulare und einfache Elemente wie Dialoge und Listen bereit. Die Anordnung und das Erscheinungsbild dieser Komponenten werden durch gerätespezifische Designprofile vorgegeben und können durch den Entwickler nur geringfügig beeinflusst werden. Der Low-Level Bereich ist beispielsweise für die Entwicklung von Spielen vorgesehen. Entwickler haben dabei uneingeschränkten Zugriff auf das Gerätedisplay und sind in der Lage, auf Tastatureingaben zu warten und diese auszuwerten. Eine Unterstützung von Formular- oder Dialogelementen ist bei dieser Art der grafischen Benutzeroberfläche nicht vorgesehen. Die MIDP-Spezifikation enthält ferner eine Beschreibung der Application Management Software (AMS), welche die Installation, Aktualisierung, Ausführung und Deinstallation einer MIDP-Anwendung auf einem mobilen Endgerät verwaltet. Seit der MIDP-Version 2.0 ist die Übertragung einer Anwendung über eine drahtlose Netzwerkverbindung und deren Installation vorgesehen.

Die unterstützten MIDP-Spezifikationen können über die Systemeigenschaften zur Laufzeit der Anwendung abgefragt werden. Ein Gerät kann mehrere MIDP-Versionen

unterstützen, wobei eine Abwärtskompatibilität zwischen den einzelnen Versionen besteht. Das folgende Quellcodefragment realisiert eine derartige Abfrage:

```
String profile = System.getProperty("microedition.profiles");
```

Quellcode 3-2: Abfrage der Systemeigenschaften nach unterstützten Profilen.

Bei einem Gerät, welches die MIDP-Versionen 1.0 und 2.0 unterstützt, liefert die Funktion den Rückgabewert `"MIDP-1.0 MIDP-2.0"`.

3.3.6 MIDlet-Programmierung

Die auf Basis der CLDC und des MIDP entwickelten Anwendungen für mobile Endgeräte werden MIDlets genannt. Die zu einem MIDlet gehörenden Klassendateien werden gemeinsam mit den verwendeten externen Klassenbibliotheken und anderen Ressourcen wie Bild- und Audioinhalten in einer Java-Archivdatei (JAR-Datei) verwaltet. Die JAR-Dateien tragen in Anlehnung an die in diesen enthaltenen MIDlets die Bezeichnung MIDlet-Suiten. In einer MIDlet-Suite können Entwickler eine beliebige Anzahl an MIDlets vereinen und so gemeinsam auf Mobiltelefone oder andere mobile Endgeräte übertragen. Die MIDlets einer MIDlet-Suite können miteinander interagieren, Klassendateien gemeinsam verwenden und sowohl die gerätespezifischen als auch die innerhalb der JAR-Datei enthaltenen Ressourcen gemeinschaftlich nutzen. Eine MIDlet-Suite enthält ferner eine Manifest-Datei, in der anwendungsspezifische Metainformationen abgelegt sind. Die Metainformationen werden bei der Installation und Ausführung einer MIDlet-Suite ausgewertet und beinhalten obligatorische Angaben zum Namen, der Version und dem Hersteller der Anwendung. Eine optionale Komponente der MIDlet-Suite ist der Java Application Descriptor (JAD). Der JAD ist eine Datei außerhalb der JAR-Datei, die ebenso wie die Manifest-Datei anwendungsspezifische Metainformationen enthält. Zusätzlich zu den obligatorischen Angaben aus der Manifest-Datei muss die JAD-Datei auch über die Bezugsadresse der JAR-Datei sowie deren Dateigröße Auskunft geben und sollte Angaben über die unterstützten Konfigurations- sowie MIDP-Versionen beinhalten. Wenn der Nutzer eine Anwendung von einer entfernten Netzwerkressource auf das Gerät herunterladen möchte, wird zunächst die JAD-Datei auf das Gerät übertragen und von der AMS analysiert. Ergibt die Analyse, dass keine Inkompatibilitäten zwischen der Anwendung und dem Gerät existieren, wird die eigentliche Anwendung in Form der JAR-Datei nachgeladen. Andernfalls, z.B. weil das Gerät nicht die erforderliche MIDP-Version unterstützt, wird der Lade- und Installationsprozess abgebrochen. Für die Erstellung und Korrektheit der Metainformationen in der JAR-Datei und JAD-Datei ist der Entwickler

verantwortlich. Entwicklungsumgebungen wie die Netbeans IDE, das Sun Wireless Toolkit oder Eclipse unterstützen den Entwickler jedoch dabei oder führen die nötigen Schritte automatisch aus.

Im Gegensatz zu anderen Java-Anwendungen werden MIDlets nicht über eine `main`-Methode gestartet. Der Lebenszyklus eines MIDlets ist in dem Paket `javax.microedition.midlet.MIDlet` enthalten und umfasst die möglichen Zustände: *Aktiv*, *Pausiert* und *Zerstört*. Die AMS überwacht den Lebenszyklus eines MIDlets und steuert die Überführung des MIDlets von einem Zustand in einen anderen über den Aufruf entsprechender Methoden. Der Lebenszyklus sowie die Zusammenhänge der Zustände eines MIDlets sind in Abbildung 3-4 dargestellt.

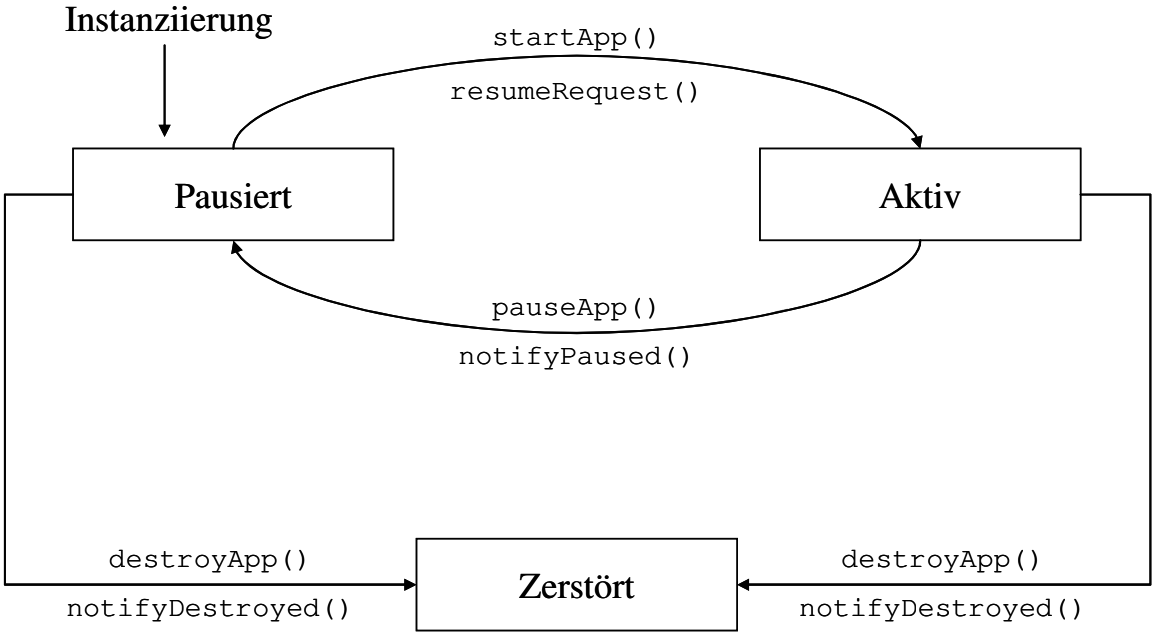

Abbildung 3-4: Der Lebenszyklus eines MIDlets.[35]

Startet der Nutzer ein auf dem Gerät installiertes MIDlet, erstellt die AMS zunächst eine neue Instanz des MIDlets und versetzt dieses in den Zustand *Pausiert*. Anschließend wird die Methode `startApp()` aufgerufen, um die eigentliche Anwendung auszuführen, wodurch der Zustand des MIDlets vom Zustand *Pausiert* in den Zustand *Aktiv* übergeht. Systemereignisse, wie beispielsweise eingehende Anrufe oder der Empfang einer SMS, die während des Zustands *Aktiv* eintreten, veranlassen die AMS, die Ausführung des MIDlets zu unterbrechen. Dazu wird die Methode `pauseApp()` aufgerufen und der Zustand des MIDlets wechselt nach *Pausiert*. Es sollten dabei alle nicht benötigten Systemressourcen freigegeben werden. Nach Beendigung des Systemereignisses kann die AMS die Ausführung der Anwendung erneut veranlassen. Nachdem die Methode

[35] Abbildung in Anlehnung an Breymann, Mosemann (2006), S. 42.

`startApp()` abgearbeitet wurde, befindet sich das MIDlet wiederum im Zustand *Pausiert*. Zur Beendigung der Anwendung wird die Methode `destroyApp()` verwendet, woraufhin alle vom MIDlet belegten Ressourcen freigegeben und die Anwendungsdaten gesichert werden. Nicht nur die AMS kann die Änderung des Zustandes eines MIDlets veranlassen. Auch das MIDlet selbst kann über die Methoden `resumeRequest()`, `notifyPaused()` und `notifyDestroyed()` die AMS darüber in Kenntnis setzen, dass eine Zustandsänderung vorgenommen werden soll.[36]

Der Quellcode 3-3 demonstriert den Aufbau eines MIDlets am Beispiel eines HelloWorld-MIDlets, das die aus der `javax.microedition.midlet.MIDLET` abgeleiteten abstrakten Methoden `startApp()`, `pauseApp()` und `destroyApp()` beinhaltet. Das MIDlet verwendet ein einfaches Formular der High-Level LCDUI, auf dem eine Textausgabe erfolgt. Darüber hinaus enthält das Formular eine Schaltfläche, über die der Nutzer die Anwendung beenden kann. Der `CommandListener` fängt das Ereignis der Betätigung der `Beenden`-Schaltfläche ab und übergibt es der Methode `CommandAction()` zur Auswertung. Durch Aufruf der Methode `notifyDestroyed()` wird der AMS mitgeteilt, dass das MIDlet beendet werden kann.

```
import javax.microedition.midlet.MIDlet;

//Import von : CommandListener, Command, Display, Displayable, Form
import javax.microedition.lcdui.*;

public class HelloWorld extends MIDlet implements CommandListener {
     public HelloWorld () {  }

     private Display _Display = null;
     private Form myForm = null;
     private Command myCommand = null;

     public void startApp () {
         if (_Display == null) {
             myForm = new Form(null);
             myCommand = new Command("Beenden", Command.EXIT, 0);

             _Display = Display.getDisplay(this);
             _Display.setCurrent(myForm);

             myForm.setTitle("Hello World");
             myForm.append("Dies ist eine Beispielanwendung!");
             myForm.addCommand(myCommand);
             myForm.setCommandListener(this);
         }
     }

     public void pauseApp() {  }
     public void destroyApp(boolean unconditional) {  }

     public void commandAction(Command c, Displayable d) {
         if (c == myCommand) {notifyDestroyed();}
     }
}
```

Quellcode 3-3: Quellcode des HelloWorld-MIDlets.

[36] Vgl. Breymann, Mosemann (2006), S. 41 ff.

Die Abbildung 3-5 zeigt die Ausführung des HelloWorld-MIDlets innerhalb des Sun Wireless Toolkit Emulators, der ein generisches Mobiltelefon auf dem Entwickler-PC emuliert.

Abbildung 3-5: Ausführung des HelloWorld-MIDlets im Sun Wireless Toolkit Emulator.[37]

3.3.7 Sicherheitsmodell

Wie bereits in den vorhergehenden Kapiteln erörtert, ergeben sich durch die eingeschränkte Verfügbarkeit der Hardwareressourcen in Mobiltelefonen und anderen mobilen Endgeräten zahlreiche Besonderheiten beim Umgang und der Entwicklung von Anwendungen auf Basis der J2ME-Plattform. Auch das Sicherheitskonzept unterscheidet sich von dem anderer Java-Plattformen. Im Gegensatz zum Sicherheitsmodell der J2SE-Plattform, bei der die Möglichkeit besteht, mit einem hohen Detaillierungsgrad die Zugriffsrechte und Privilegien einzelner Anwendungen auf kritische Systemfunktionen festzulegen, wurde das Sicherheitskonzept der J2ME sehr viel einfacher konzipiert. Das Sicherheitsmodell der J2ME-Plattform basiert auf den Bereichen der Low-Level-Security, der Application-Level-Security und der End-To-End-Security.

[37] Eigene Darstellung.

Bevor eine Anwendung in der virtuellen Maschine ausgeführt werden kann, prüft die virtuelle Maschine, ob das vorliegende Programm den semantischen Anforderungen der Programmiersprache Java gerecht wird und dass innerhalb des Programms kein Code enthalten ist, der das Gastsystem und die virtuelle Maschine schädigen kann. Diese Überprüfung ist Teil der Low-Level-Security und wird durch die in Kapitel 3.3.4 beschriebene Verifikation des Bytecodes erreicht. Damit ist stets sichergestellt, dass es sich bei der Anwendung um ein ordnungsgemäßes Java-Programm handelt. Ein in den Quellcode eingebauter Schadcode wird durch die Verifikation des Bytecodes jedoch nicht entdeckt. Mit der Application-Level-Security, welche die Low-Level-Security voraussetzt, wird diese Sicherheitslücke geschlossen. Dazu wird jede Anwendung als potenziell gefährlich eingestuft und deshalb in einem besonders geschützten Bereich der virtuellen Maschine, der Sandbox, ausgeführt. Die Sandbox gewährleistet, dass eine auszuführende Anwendung keine Klassenbibliotheken, Systemressourcen oder andere Gerätekomponenten und Funktionen nutzen kann, die nicht durch die Laufzeitumgebung und durch das Gerät explizit freigegeben wurden. Verwendbare Klassenbibliotheken werden durch die CLDC, das MIDP sowie durch herstellerspezifische Pakete festgelegt. Zusätzliche Klassenbibliotheken und Funktionalitäten, die nicht Teil der genannten Quellen sind, können im Nachhinein nicht eingebunden werden. Eine Ausnahme sind die in der JAR-Datei enthaltenen Klassenbibliotheken, sofern diese keine Systemklassen repräsentieren. Um die Integrität der Systemklassenbibliotheken wie `java.*`, `javax.microedition.*` etc. zu gewährleisten, sind diese schreibgeschützt. Des Weiteren muss sichergestellt werden, dass der gesamte Prozess des Herunterladens, der Installation und der Verwaltung von Java-Anwendungen, welcher durch die AMS (vgl. Kapitel 3.3.5) überwacht wird, nicht durch den Programmierer manipuliert oder umgangen werden kann. Die End-To-End-Security ist für die Sicherung des Kommunikationsweges zwischen dem mobilen Endgerät und einem entfernten Dienst verantwortlich. Damit wird sichergestellt, dass die gesendeten Daten zwischen den zwei Parteien unverändert und geschützt durch das für die Kommunikation genutzte Netzwerk gelangen.[38]

Die MIDP-Spezifikation in der Version 2.0 fügt dem Sicherheitsmodell das Konzept der vertrauenswürdigen und nicht-vertrauenswürdigen MIDlet-Suiten hinzu. Eine MIDlet-Suite wird als vertrauenswürdig eingestuft, wenn die Herkunft sowie Integrität der JAR-Datei durch ein mobiles Gerät nachvollzogen werden kann. Dies wird durch die Verwendung von digitalen Signaturen und Zertifikaten erreicht, mit denen die MIDlet-Suiten versehen werden. Der Zugriff auf einzelne Klassenbibliotheken wird über Sicherheitsdomänen gesteuert. Für die zu einer Sicherheitsdomäne zugeordneten

[38] Vgl. JSR-139 (2007), S. 13 ff.

Klassenbibliotheken ist festgelegt, ob einer innerhalb derselben Sicherheitsdomäne agierenden MIDlet-Suite der Zugriff gestattet ist oder explizit durch den Nutzer gestattet werden muss. Vertrauenswürdige MIDlet-Suiten besitzen im Vergleich zu nicht-vertrauenswürdigen MIDlet-Suiten mehr Privilegien und Rechte. Sowohl vertrauenswürdige als auch nicht-vertrauenswürdige MIDlet-Suiten haben, ohne explizite Autorisation durch den Nutzer, Zugriff auf die Programmierschnittstellen der folgenden Klassenbibliotheken:

- `javax.microedition.rms.*` - persistente Datenspeicherung in Record Stores,
- `javax.microedition.midlet.*` - Lebenszyklus des MIDlets,
- `javax.microedition.lcdui.*` - grafische Benutzerschnittstelle sowie
- `javax.microedition.media.*` - Audio- und Multimediawiedergabe.

Eine explizite Nutzerautorisation für beide Arten von MIDlet-Suiten ist beispielsweise für die Klassenbibliothek `javax.microedition.io.*` erforderlich, die das GCF enthält. Ohne diese explizite Bestätigung könnten Anwendungen vom Nutzer unbemerkt und ungewollt Netzwerkverbindungen, wie z.B. Internetverbindungen, aufbauen, wodurch dem Nutzer unter Umständen Kosten für den Datentransfer entstehen.[39]

3.4 Drahtlose Datenübertragungstechnologien für Mobiltelefone

3.4.1 Datennetze

Mobiltelefone können auf ein breites Spektrum an Technologien zur drahtlosen Übertragung von Daten zurückgreifen und integrieren Schnittstellen für:

- den mobilen Zugriff auf das World Wide Web (WWW),
- den Zugriff auf drahtlose lokale Netzwerke sowie
- den Aufbau von Direktverbindungen über Bluetooth oder Infrarot.

Der weltweit am weitesten verbreitete und im europäischen Raum verwendete Mobilfunk-Standard Global System for Mobile Communications (GSM) stellt die Basistechnologie für mobile Sprach- und Datenübertragung auf Mobiltelefonen dar. Mittels GSM können Daten mit einer maximalen Übertragungsgeschwindigkeit von bis zu 9,6 KBit/s auf ein Mobiltelefon übertragen werden.[40] Durch Erweiterungen des GSM-Standards um Protokolle wie GPRS (General Packet Radio Service) und EDGE (Enhanced Data Rates for GSM Evolution) konnte die Datenübertragungsrate zunächst

[39] Vgl. JSR-118 (2006), S. 30 ff.
[40] Vgl. Schiller (2000), S. 141 ff.

auf bis zu 384 KBit/s angehoben werden.[41] Die nächste Generation der mobilen Datenübertragungstechnologie wird durch UMTS (Universal Mobile Telecommunications System) repräsentiert. Durch die Erweiterungen des UMTS-Standards um das HSDPA-Protokoll (High Speed Downlink Packet Access) können Übertragungsgeschwindigkeiten von derzeit bis zu 7,2 MBit/s angeboten werden.[42] Die letztendlich erzielten Übertragungsgeschwindigkeiten bei UMTS und HSDPA hängen stark davon ab, wie weit die einzelnen Netzbetreiber den Ausbau der Infrastruktur vorangetrieben haben. UMTS und HSDPA werden derzeit nur in Ballungsgebieten angeboten, wohingegen GSM, GPRS und EDGE in Deutschland flächendeckend zur Verfügung stehen.[43] Der Nachteil der Nutzung dieser Datenübertragungstechnologien sind die zum Teil hohen Nutzungsentgelte für anfallendes Datenaufkommen. Mittlerweile bieten einige Telekommunikationsanbieter entsprechende Daten-Flatrates an, jedoch sind auch diese recht kostenintensiv und nicht sehr verbreitet.

Die Netzwerktechnologie Wireless Local Area Network (WLAN) wird vorrangig zur drahtlosen Vernetzung von PDAs und PCs eingesetzt. Das Netzwerk kann infrastrukturbasiert sein, d.h. eine Basisstation dient als Vermittler des Netzwerkverkehrs, oder die Geräte vernetzen sich direkt miteinander ohne einen Vermittler. Die Reichweite einer WLAN-Funkzelle beträgt bei optimalen Bedingungen bis zu 100 Meter und kann Daten laut dem Standard IEEE 802.11g mit bis zu maximal 54 MBit/s übertragen. Der Vorteil der Technik besteht darin, mit etablierter und kostengünstiger Hardware ein lokales Funknetzwerk aufzubauen, welches kostenlos nutzbar ist sowie über eine ausreichende Datentransferrate verfügt. [44] Die standardmäßige Integration von WLAN in Mobiltelefonen scheiterte bislang am hohen Energiebedarf der Sende- und Empfangseinheit. Seit kurzem bieten einige Mobiltelefonhersteller zwar erste Geräte mit eingebauter WLAN-Funktionalität an, jedoch ist die Laufzeit dieser Mobiltelefone bei aktiviertem WLAN stark eingeschränkt.

Bluetooth und Infrarot (IrDA) benötigen im Gegensatz zu WLAN weit weniger Energie für den Betrieb, verfügen dadurch aber auch über eine bedeutend geringere Reichweite. Mit Bluetooth und IrDA können Mobiletelefone untereinander ein sog. Wireless Personal Area Network (WPAN) aufbauen, dessen Reichweite bei Bluetooth auf etwa zehn Meter und bei IrDA auf ungefähr einen Meter begrenzt ist. Während IrDA nur reine Punkt-zu-Punkt-Verbindungen zwischen zwei Geräten ermöglicht, können Bluetooth-Geräte auch Verbindungen zu mehreren Geräten gleichzeitig

[41] Vgl. Schiller (2000), S. 177 ff. und S. 192 f.
[42] Vgl. T-Mobile 1 (2009).
[43] Vgl. T-Mobile 2 (2009).
[44] Vgl. Schiller (2000), S. 245 ff.

erzeugen. Obwohl Mobiltelefone derzeit meist noch beide Schnittstellen integrieren, wird IrDA von Bluetooth abgelöst. Gründe dafür sind unter anderem die höhere Reichweite, die Möglichkeit Verbindungen zu mehreren Geräten gleichzeitig aufzubauen sowie die nicht benötigte Sichtverbindung zwischen den Geräten.[45]

Nachdem die verfügbaren drahtlosen Datenübertragungstechnologien für Mobiltelefone mit deren Eigenschaften sowie Vor- und Nachteilen genannt wurden, bleibt zu entscheiden, welche der vorgestellten Technologien sich für die Datenübertragung beim zu entwickelnden Museumsführer eignen. Damit dem Museumsbesucher keine zusätzlichen Kosten bei der Übertragung von Daten via GSM oder UMTS auf das Mobiltelefon entstehen und da nicht sichergestellt werden kann, dass die WLAN-Technologie auf den Mobiltelefonen vorhanden ist, scheiden diese Übertragungswege aus. IrDA sollte aufgrund der im Vergleich zu Bluetooth bestehenden Nachteile ebenfalls nicht verwendet werden. Für die Entscheidung zu Gunsten von Bluetooth spricht, dass es sich als Standard für den Datenaustausch auf Mobiltelefonen etabliert hat. Darüber hinaus kann die Bluetooth-Funktionalität der Mobiltelefone über vorhandene J2ME-Programmierschnittstellen genutzt werden.

Im folgenden Kapitel werden der Aufbau und die Beschaffenheit von Bluetooth sowie dessen Verwendung mit der J2ME-Plattform thematisiert.

3.4.2 Bluetooth

Bluetooth ist ein von der Bluetooth Special Interest Group definierter und auf Funkwellen basierender Kommunikationsstandard, der zur Vernetzung von Mobiltelefonen, PDAs und anderen Geräten wie Druckern und PCs verwendet wird. Mit Bluetooth können sowohl Daten- als auch Sprachinformationen zwischen Geräten ausgetauscht werden. Die in Mobiltelefonen und PDAs verwendete Bluetooth-Hardware besitzt eine Reichweite von bis zu zehn Metern und kann Daten mit bis zu drei MBit/s übertragen. Die Bluetooth-Signale sind in der Lage, feste Objekte zu durchdringen und benötigen daher im Gegensatz zu IrDA keine Sichtverbindung, um eine Verbindung zwischen Geräten aufzubauen.

Die Kommunikation von Bluetooth-Geräten beruht auf dem Aufbau von Piconetzwerken, die aus mindestens zwei, den gleichen physischen Funkkanal verwendenden, Geräten gebildet werden. Eines der Geräte übernimmt die zentrale Verwaltung und Steuerung der Kommunikation mit den weiteren

[45] Vgl. Breymann, Mosemann (2006), S. 263 f.

Kommunikationsteilnehmern und wird als Master bezeichnet. Alle anderen Geräte werden Slaves genannt. Im einfachsten Fall besteht ein Piconetzwerk aus genau einem Master- und einem Slave-Gerät, die untereinander Daten austauschen (vgl. Abbildung 3-6 - Einzel-Slave-Betrieb). Besteht das Piconetzwerk hingegen aus einem Master und mehreren Slaves, wird diese Betriebsart als Multi-Slave-Betrieb bezeichnet (vgl. Abbildung 3-6). Die Anzahl an aktiven Slaves ist auf maximal sieben Geräte je Piconetzwerk beschränkt, wobei weitere inaktive Slaves dem Piconetzwerk angehören können. Ist ein Gerät Bestandteil von zwei oder mehr Piconetzwerken, so wird ein Scatternet aufgespannt. Ein Gerät kann jeweils nur in einem Piconetzwerk die Masterrolle übernehmen, innerhalb der anderen Piconetzwerke muss es die Rolle eines Slaves einnehmen. Der Zusammenhang zwischen Piconetzwerken und einem Scatternet ist in Abbildung 3-6 (Scatternet-Betrieb) visualisiert.

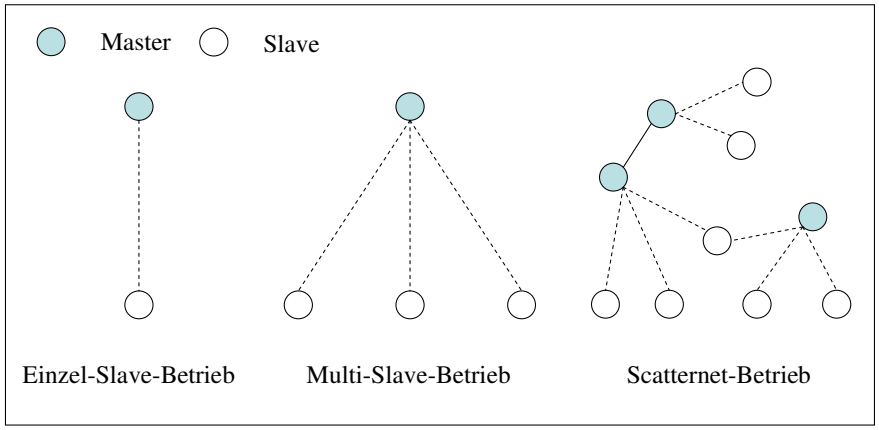

Abbildung 3-6: Topologie der Bluetooth-Kommunikation.[46]

Die Schichtenarchitektur des Bluetooth-Protokollstacks ist in Abbildung 3-7 schematisch dargestellt und untergliedert sich in die Bereiche Bluetooth Controller und Bluetooth Host. Der Bluetooth Controller umfasst die Protokollschichten Bluetooth Radio, Baseband Link Controller (LC), Link Manager Protocol (LMP) und den zur Hardware gehörenden Komponenten des Host Controller Interface (HCI). Die konkrete Ausgestaltung dieses Bereichs ist von den Bluetooth-Geräteherstellern abhängig. Diese müssen in einem Zertifizierungsprozess nachweisen, dass die Geräte die Bluetooth-Spezifikation in vollem Umfang unterstützen. Über dem Bluetooth Controller befindet sich der Bluetooth Host. Zu diesem werden die Softwareumsetzung des HCI sowie Protokolle wie das Service Discovery Protocol (SDP), das Logical Link Control and Adaption Protocol (L2CAP), das Radio Frequency Communication Protocol (RFCOMM) und das Object Exchange Protocol (OBEX) gezählt.[47] Abbildung 3-7 zeigt nicht alle verfügbaren Protokolle des Bluetooth Host. Neben den genannten existieren

[46] Vgl. Bluetooth 1 (2009).
[47] Vgl. Breymann, Mosemann (2006), S. 265 ff.

weitere, die für den Datenaustausch über die Bluetooth-Schnittstelle jedoch nicht näher betrachtet werden müssen.

Die unterste Schicht des Bluetooth Controllers ist das Bluetooth Radio. Diese ist für die physische Übermittlung und den Empfang der im 2,4 GHz-Frequenzband verwendeten Funkwellen zuständig. Der in der Protokollhierarchie übergeordnete LC steuert sowohl den Aufbau von physischen Verbindungen zwischen zwei Geräten im Bluetooth Radio als auch den Medienzugriff der höheren Protokollebenen durch den Aufbau von logischen Verbindungen zur Datenübertragung. Diese Schicht ist weiterhin für die Zusammenstellung von Datenpaketen (Framing) verantwortlich. Die Übertragung der Datenpakete kann sowohl synchron als auch asynchron erfolgen. Darüber hinaus können Suchläufe nach im Umkreis befindlichen Bluetooth-Geräten ausgelöst werden.

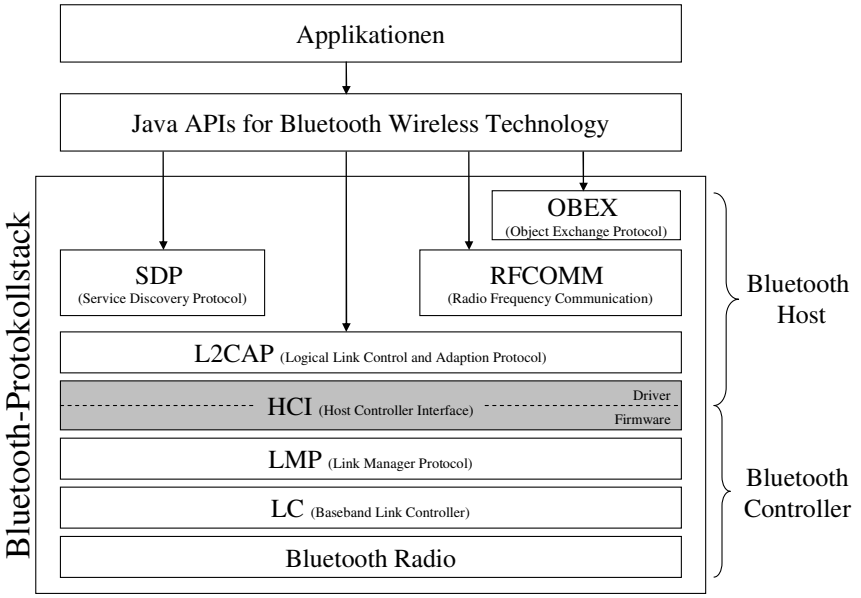

Abbildung 3-7: Aufbau des Bluetooth-Protokollstacks.[48]

Das LMP dient der Aushandlung der Verbindungsmodalitäten zwischen den Geräten sowie der Erstellung, Verwaltung und Freigabe logischer Verbindungen. Zur Sicherung einer logischen Verbindung kann das LMP Maßnahmen wie Authentifizierungs- und Verschlüsselungsprozeduren aktivieren und verwalten. Das Host Controller Interface fungiert als Schnittstelle zwischen dem LC und dem LMP und erlaubt den Zugriff auf Statusinformationen sowie Steuerregister. Der HCI ist in eine Hardwarekomponente (HCI Firmware) zur Ansteuerung der Bluetooth-Hardware und eine Softwarekomponente (HCI Driver) zum Daten- und Befehlsaustausch mit der HCI Firmware unterteilt (vgl. Abbildung 3-7). Über dem HCI befindet sich das L2CAP, welches den höheren Protokollen verbindungsorientierte und verbindungslose Dienste

[48] Abbildung in Anlehnung an Breymann, Mosemann (2006), S. 267.

zur Verfügung stellt. Die logischen Verbindungen der L2CAP-Schicht werden durch die Erstellung von logischen Kanälen weiter untergliedert. Jeder Kanal innerhalb der logischen Verbindung kann über eine eindeutige Kanalnummer angesprochen werden. Abgehende Daten der höheren Protokolle werden in Datenpakete mit einer Länge von bis zu 64 KByte segmentiert, wohingegen ankommende Datenpakete für die höheren Protokolle wieder zusammengesetzt werden. L2CAP wird von Protokollen, wie beispielsweise SDP und RFCOMM, für die Kommunikation verwendet. Wie bereits erwähnt, können benachbarte Bluetooth-Geräte mittels der Gerätesuche im LC ermittelt werden. Für die Abfrage der von diesen Geräten angebotenen Dienste wird SDP eingesetzt. Jedes Bluetooth-Gerät verwaltet intern eine Datenbank mit den Diensten, die anderen Geräten zur Verfügung gestellt werden. Das SDP sendet dem entfernten Gerät eine Anfrage, was die Abfrage der Dienstdatenbank des entfernten Gerätes zur Folge hat und erhält als Antwort eine Liste der verfügbaren Dienste des entfernten Gerätes. Jeder Dienst hat eine universelle Identifikationsnummer, den sog. Universal Unique Identifier (UUID). Das Transportprotokoll RFCOMM emuliert serielle Verbindungen zwischen Bluetooth-Geräten und dient der seriellen Übertragung von Daten. Das OBEX Protokoll nutzt die Funktionalitäten von RFCOMM und ermöglicht die Definition und den Austausch von Datenobjekten zwischen verbundenen Bluetooth-Geräten. Ein Datenobjekt kann beispielsweise eine elektronische Visitenkarte oder ein elektronischer Terminkalendereintrag sein, der zum Abgleich auf ein anderes Gerät übertragen wird.[49]

Für die Programmierung von Bluetooth-Anwendungen auf Basis der J2ME-Plattform definiert die Spezifikation JSR-82 die Klassenbibliotheken, mit denen der Zugriff auf den Bluetooth-Protokollstack erfolgt (vgl. Abbildung 3-7). Die JSR-82 ist ein optionales Paket für die J2ME-Plattform. Durch Abfrage der Systemeigenschaften zur Laufzeit der Anwendung kann ermittelt werden, ob und in welcher Version eine Unterstützung von Bluetooth auf einem Gerät enthalten ist. Eine derartige Abfrage wird durch das folgende Quellcodefragment illustriert:

```
String bt = System.getProperty("bluetooth.api.version");
```

Quellcode 3-4: Abfrage der Systemeigenschaften auf Bluetooth-Unterstützung.

Der Rückgabewert der Abfrage enthält die Versionsnummer der unterstützten JSR-82 Spezifikation. Beispielsweise ist für eine Unterstützung der JSR-82 in der Version 1.1.1 der Rückgabewert der Funktion `"1.1.1"`. Liegt keine Unterstützung von Bluetooth vor, so liefert die Funktion `"null"` zurück.

[49] Vgl. Bluetooth 2 (2009)

Die Protokolle, die angesprochen werden können, sind L2CAP, SDP, RFCOMM und OBEX. Die Java Application Programming Interfaces (API) for Bluetooth Wireless Technology (JABWT) liegen auf gleicher Ebene mit dem MIDP in der J2ME-Plattformhierachie und setzen auf der CLDC auf. Durch die Verwendung des GCF (vgl. Kapitel 3.3.4) können Verbindungen zu anderen Bluetooth-Geräten hergestellt werden. Neben der CLDC setzt die Spezifikation eine für die Konfiguration des Bluetooth-Gerätes vorhandene Verwaltungssoftware, das Bluetooth Control Center (BCC), voraus. Das BCC ist eine geräteabhängige und vom Bluetooth-Gerätehersteller mitgelieferte Software, die dem Nutzer Grundeinstellungen für das Bluetooth-Gerät erlaubt, wie beispielsweise die Aktivierung und Deaktivierung der Bluetooth-Hardware, die Vergabe eines Gerätenamens und Grundeinstellungen zur Sicherheit für die Kommunikation zwischen Bluetooth-Geräten.[50] Über entsprechende Schnittstellen können die Bluetooth-Eigenschaften unter J2ME abgefragt werden.

Wie bereits erwähnt, sind für den Datentransport zwischen zwei Geräten das paketbasierte L2CAP, das streambasierte RFCOMM und das objektbasierte OBEX Protokoll potenziell verwendbar. Da OBEX in der JSR-82 als optionaler Bestandteil beschrieben wird, ist nicht sichergestellt, dass die erforderlichen J2ME-Klassenbibliotheken für OBEX auf jedem Gerät eingebunden sind und verwendet werden können. RFCOMM hingegen ist ein fester Bestandteil der Spezifikation und kann im Vergleich zur Datenübertragung mit dem L2CAP-Protokoll aufgrund der streambasierten Übertragung einfacher verwendet werden.

3.5 Datenhaltung

Nahezu jede Anwendung ist in irgendeiner Weise darauf angewiesen, Daten während der Laufzeit des Programms persistent auf einem Speichermedium abzulegen. Dabei ist es irrelevant, ob es sich bei diesen Daten um Nutzereingaben, Konfigurationsdaten, Programmeinstellungen oder Spielstände handelt.[51] Entscheidend ist, dass diese Daten beim erneuten Start der Anwendung z.B. nach Beendigung des Programms durch den Nutzer bzw. nach dem Ausschalten des Gerätes weiterhin vorhanden sind. Die in Mobiltelefonen zum Einsatz kommende Speicherhardware variiert zwischen Flash-Speichermedien und batteriegepufferten Speichern.[52] Die genannten Speichermedien bilden den internen Speicher des Mobiltelefons, auf welchem unter anderem Kontaktdaten und empfangene Textnachrichten abgelegt werden. Eine

[50] Vgl. Breymann, Mosemann (2006), S. 270.
[51] Vgl. Topley (2002), S. 210.
[52] Vgl. Kroll, Haustein (2003), S. 129.

Möglichkeit, Daten unter der J2ME-Plattform im nicht-flüchtigen Speicher der Geräte zu sichern, stellen Record Stores dar. Ein Record Store ist eine einfache Datenbank, in der Datensätze gesichert werden können. Über das Record Management System wird dann sowohl die Zugriffskontrolle als auch die Verwaltung des internen Speichers automatisch vorgenommen. Ferner stehen dem Anwendungsentwickler Programmierschnittstellen für die Nutzung dieser Technologie zur Verfügung.

Wie bereits in Kapitel 3.3.2 erwähnt, kann eine MIDlet-Suite mehrere MIDlets umfassen. Jedes MIDlet einer MIDlet-Suite kann eine beliebige Anzahl an Record Stores erstellen. Beim Erzeugen wird einem Record Store ein Name zugewiesen, der aus einer Unicode-Zeichenkette von einem bis 32 Zeichen bestehen kann. Der Name muss innerhalb der MIDlet-Suite eindeutig sein, wobei zwischen Groß- und Kleinschreibung unterschieden wird. Alle MIDlets einer MIDlet-Suite besitzen uneingeschränkten Lese- und Schreibzugriff auf die innerhalb der gleichen MIDlet-Suite erstellten Record Stores und können nicht weiter benötigte Record Stores entfernen. In Abhängigkeit davon, welche MIDP-Spezifikation bei der Entwicklung der Anwendung genutzt wird, ergeben sich Unterschiede beim Zugriff auf Record Stores verschiedener MIDlet-Suiten. Für Anwendungen, die auf der MIDP 1.0a Spezifikation beruhen, ist der Zugriff lediglich auf Record Stores innerhalb der gleichen MIDlet-Suite gestattet.[53] Dieser Sachverhalt wird in Abbildung 3-8 in einer schematischen Darstellung verdeutlicht. Die der MIDlet-Suite 1 zugeordneten Record Stores können nur durch das MIDlet 1 genutzt werden. Auf den Record Store der MIDlet-Suite 2 hat das MIDlet der MIDlet-Suite 1 keinen Zugriff.

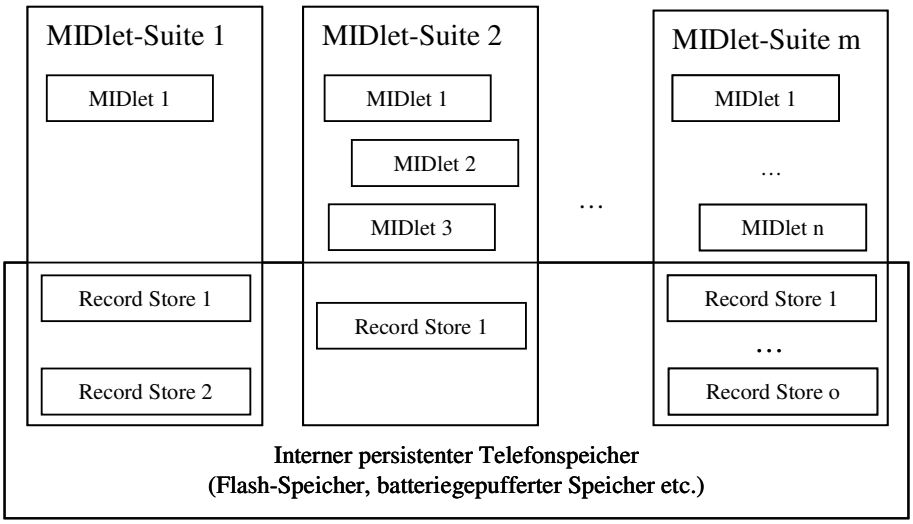

Abbildung 3-8: Zusammenhang zwischen MIDlet-Suiten und Record Stores nach MIDP 1.0a.[54]

[53] Vgl. JSR-37 (2000), S. 37.
[54] Abbildung in Anlehnung an Knudsen (2001), S 78.

Mit Einführung der MIDP 2.0 Spezifikation ist diese Restriktion aufgehoben worden. MIDlets einer MIDlet-Suite können, sofern die entsprechenden Lese- und Schreibzugriffsrechte bei der Erstellung erteilt wurden, auf Record Stores anderer MIDlet-Suiten zugreifen.[55] Damit auf einen Record Store einer anderen MIDlet-Suite referenziert werden kann, wird der Name des Record Stores in Verbindung mit dem Namen der MIDlet-Suite verwendet. Wird eine MIDlet-Suite durch den Nutzer vom Mobiltelefon entfernt, so werden gleichzeitig alle Record Stores, die von dieser MIDlet-Suite angelegt wurden, ebenfalls gelöscht. Dieser Mechanismus wird automatisch ausgeführt und ist in allen MIDP-Spezifikationen vorgeschrieben.

Die Klasse `RecordStore` aus dem Paket `javax.microedition.rms` stellt Methoden bereit, mit deren Hilfe Datensätze in einem Record Store hinzugefügt, verändert oder entfernt werden können. Ferner stehen Methoden zur Verfügung, über die Meta-Informationen über einen Record Store abgefragt werden können, wie beispielsweise die Anzahl der vorhandenen Datensätze sowie Statusinformationen über den freien und bereits belegten Speicherplatz.[56]

Die einzelnen Datensätze in einem Record Store werden als Byte-Array realisiert und können entweder aus einem leeren Datensatz bestehen oder beliebige Daten unterschiedlicher Länge aufnehmen. Die Größe der in den Record Stores ablegbaren Daten wird durch die Größe des internen Telefonspeichers limitiert. Wird ein Datensatz hinzugefügt, so wird diesem eine eindeutige ganzzahlige Identifikationsnummer (ID) zugewiesen, über die ein späterer Zugriff auf den Datensatz möglich ist. Während des gesamten Lebenszyklus eines Record Stores wird eine Datensatz-ID jeweils nur einmalig vergeben. Die Datensatz-IDs werden als Primärschlüssel für die Datensätze verwendet und vom Record Store mittels einer Collection verwaltet. Die Struktur der Collection kann intern vom Record Store über einen Vektor oder eine vergleichbare Datenstruktur realisiert werden.[57] Der eigentliche Zugriff auf abgelegte Datensätze kann sowohl durch explizite Angabe der Datensatz-ID als auch durch die Verwendung einer Enumeration erfolgen. Ein wesentlicher Vorteil der Enumeration besteht darin, dass über jeden in der Aufzählung erfassten Datensatz iteriert wird, ohne dessen ID kennen zu müssen. Zusätzlich besteht die Möglichkeit, durch Angabe von Filterkriterien gewisse Datensätze auszuwählen und diese zu sortieren. Abbildung 3-9 verdeutlicht die getroffenen Aussagen nochmals in graphischer Form.

[55] Vgl. JSR-118 (2006), S. 481.
[56] Vgl. Kroll, Haustein (2003), S. 130.
[57] Vgl. Breymann, Mosemann (2006), S. 138.

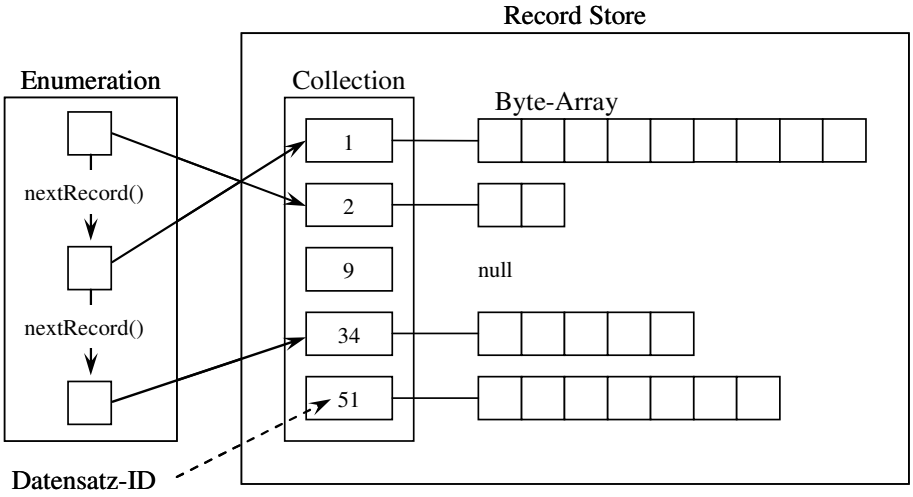

Abbildung 3-9: Aufbau der Struktur eines RecordStore-Objektes mit Enumeration.[58]

Wie bereits in Kapitel 3.1 angedeutet, ist die Mehrzahl heutiger Mobiltelefone mit integrierten Speicherkartenlesegeräten ausgestattet, über die externe Speichermedien in Form von Speicherkarten eingebunden werden können. Die Speicherkapazität des Mobiltelefons lässt sich je nach Speichermedium zwischen einigen Megabyte bis hin zu mehreren Gigabyte erweitern. Speicherkarten sind im Gegensatz zu Record Stores mit einem Dateisystem versehen, auf dem beliebige Binärdaten als Datei abgelegt werden können. Der Zugriff auf die im Mobiltelefon vorhandenen Dateisysteme erfolgt über die in der JSR-75 spezifizierten Programmierschnittstellen aus dem Paket `javax.microedition.io.file`. Dieses Paket ist optional, d.h. Mobiltelefonhersteller müssen es zusätzlich zur Java-Plattform lizenzieren. [59]

Durch Abfrage der Systemeigenschaften kann festgestellt werden, ob auf einem Gerät die Programmierschnittstellen des Dateisystemzugriffs verfügbar sind. Das folgende Quellcodefragment veranschaulicht die Abfrage der Systemeigenschaft:

```
String fc = System.getProperty("microedition.io.file.FileConnection.version");
```

Quellcode 3-5: Abfrage der Systemeigenschaften auf FileConnection-Unterstützung.

Wird bei der Abfrage die Versionsnummer `"1.0"` zurückgegeben, so ist das Paket auf dem Gerät verfügbar, andernfalls liefert die Abfrage `"null"` zurück.[60] Die Klasse `FileConnection` bietet neben den Methoden zum Lesen und Schreiben von Dateien weitere Routinen, um Dateien oder Verzeichnisse auf dem Dateisystem anzulegen oder zu löschen. Da die Möglichkeit besteht, dass der Nutzer während des laufenden Betriebs

[58] Abbildung in Anlehnung an Breymann, Mosemann (2006), S. 139.
[59] Vgl. JSR-75 (2004).
[60] Vgl. Breymann, Mosemann (2006), S. 149.

eine Speicherkarte in das Mobiltelefon einsteckt oder entfernt, werden Methoden bereitgestellt, die diese Ereignisse anzeigen. Des Weiteren können Informationen zum Speicherstatus der Speicherkarten abgefragt sowie Attribute von Dateien und Verzeichnissen ausgelesen werden.

Da auch das Mobiltelefonbetriebssystem systemrelevante Daten auf den externen Speichermedien auslagern kann, ist es notwendig, diese Dateien und Verzeichnisse vor dem Lese- und Schreibzugriff durch ein MIDlet zu schützen. Je nach Mobiltelefonhersteller kann der Schutz unterschiedlich gestaltet sein. Eine Möglichkeit besteht darin, einem nicht vertrauenswürdigen MIDlet generell keine Schreibrechte auf die externen Speichermedien zu gewähren oder diese nur in gewissen Verzeichnissen zu gestatten. Wie bereits in Kapitel 3.3.7 beschrieben, können Entwickler von Mobiltelefonanwendungen die erstellten Anwendungen durch eine Zertifizierung als vertrauenswürdig einstufen lassen. Dadurch werden die MIDlets in einer anderen Sicherheits-Domäne ausgeführt und erhalten mehr Rechte als nicht zertifizierte. Versucht ein MIDlet lesend oder schreibend auf das externe Speichermedium zuzugreifen, so wird der Benutzer durch eine Meldung darüber informiert und kann den Zugriff des MIDlets entweder gestatten oder verwehren.[61]

3.6 Mobile Media API

Für die Wiedergabe von multimedialem Inhalt wird das optionale Paket Mobile Media API (MMAPI), welches in der JSR-135 spezifiziert ist, verwendet. Dieses Paket stellt eine Obermenge der in der MIDP 2.0 eingeführten Schnittstellen zur Generierung von Tönen (vgl. Kapitel 3.3.5) dar und ermöglicht die Aufnahme und Wiedergabe diverser Multimediaformate wie Musik- und Videoinhalte. Für den Einsatz der MMAPI wird eine Fehlerklasse vorausgesetzt, welche mit der CLDC 1.1 eingeführt wurde. Ob auf einem Gerät die MMAPI verfügbar ist, kann durch Abfrage der Systemeigenschaften ermittelt werden. Das folgende Quellcodefragment veranschaulicht die Abfrage der Systemeigenschaft:

```
String mmapi = System.getProperty("microedition.media.version");
```

Quellcode 3-6: Abfrage der Systemeigenschaften auf MMAPI-Unterstützung.

Wird bei der Abfrage die Versionsnummer `"1.1"` oder `"1.2"` zurückgegeben, so ist das Paket auf dem Gerät verfügbar, andernfalls liefert die Abfrage `"null"` zurück.[62]

[61] Vgl. Breymann, Mosemann (2006), S. 155.
[62] Vgl. JSR-135 (2006).

Die zentrale Komponente bei der Aufnahme und Wiedergabe von Multimediaformaten in J2ME ist der im Paket `javax.microedition.media` enthaltene `Manager`. Dieser wird dazu verwendet, ein `Player`-Objekt zu instanziieren und mit einem `Datasource`-Objekt zu verknüpfen. Bei der Instanziierung des `Player`-Objektes wird dem `Manager` die Pfadangabe zur Ressource übergeben. Die Pfadangabe wird überprüft und anschließend als `Datasource`-Objekt an das `Player`-Objekt weitergereicht. Als Datenquelle können beispielsweise eine lokal gespeicherte Datei, ein eingebautes Mikrofon oder eine Kamera, aber auch multimedialer Inhalt von einem Server im Internet genutzt werden. Das `Datasource`-Objekt verbirgt den Ursprung und den Transport der Ressource vor dem `Player`-Objekt und stellt diesem einen Strom von Byte-Informationen bereit, der verarbeitet und dem Nutzer präsentiert wird. Für die Steuerung des `Player`-Objektes werden `Controls` verwendet. Mit diesen können z.B. die Lautstärke, die Abspieldauer oder die Wiedergabegeschwindigkeit der Inhalte gesteuert werden. Aufgrund der Fülle an verschiedenen Multimediaformaten ist davon auszugehen, dass ein Gerät nicht alle unterstützen wird. Der `Manager` stellt Methoden zur Abfrage der vom Gerät angebotenen Multimediaformate bereit und gibt Auskunft, aus welchen Quellen diese bezogen werden können.

Ein `Player`-Objekt kann sich in einem von fünf möglichen Zuständen befinden (vgl. Abbildung 3-10). Nachdem ein `Player`-Objekt instanziiert und mit einer Datenquelle verknüpft wurde, befindet sich dieser im Zustand `UNREALIZED`. Durch den Aufruf der Methode `Player.realize()` wird der `Player` in den Zustand `REALIZED` überführt, in dem Informationen über den abzuspielenden Multimediainhalt gesammelt werden.

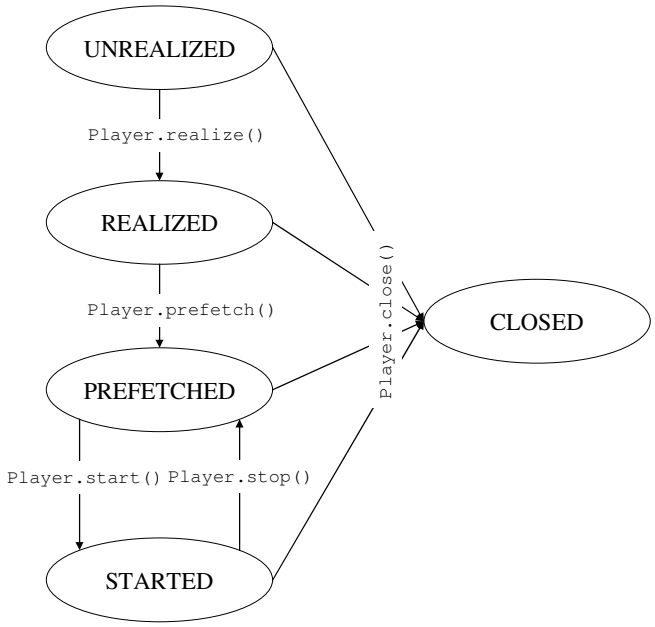

Abbildung 3-10: Zustände eines Player-Objektes.

Mit Aufruf der Methode `Player.prefetch()` wird das `Player`-Objekt veranlasst, Systemressourcen zu reservieren, die Inhalte zu laden und einer Vorverarbeitung zu unterziehen. Im Anschluss kann die Wiedergabe mit der Methode `Player.start()` begonnen werden, die zum Zustand `STARTED` führt. Ist der Abspielvorgang beendet oder durch die Methode `Player.stop()` unterbrochen worden, wird der `Player` wieder in den Zustand `PREFETCHED` versetzt. Wird der `Player` nicht mehr benötigt, so werden durch den Aufruf der Methode `Player.close()` alle belegten Systemressourcen freigegeben.[63]

3.7 XML

Die Extensible Markup Language (XML) ist eine durch das World Wide Web Consortium standardisierte Sprache zur Beschreibung und Strukturierung von Daten in einem Textformat. Ursprünglich für den Informationsaustausch und die strukturierte Speicherung von Daten im WWW entwickelt, wird XML mittlerweile unter anderem als Dokumentenformat in Officeanwendungen oder für die Speicherung von Programmeinstellungen verwendet. Genau genommen ist XML eine Metasprache, mit deren Hilfe sich Sprachen für verschiedenste Anwendungsbereiche entwickeln lassen und ist somit universell einsetzbar. Beispiele derer mit XML entwickelten Sprachen sind die zur Abbildung mathematischer Sachverhalte verwendete Mathematical Markup Language oder das zur Erstellung von Vektorgrafiken genutzte Scalable Vector Graphics.[64] Die nachfolgend aufgeführten Eigenschaften trugen zur Verbreitung und Akzeptanz von XML bei und führten dazu, dass sich XML als Standardformat für den Datenaustausch zwischen Computersystemen etablierte:

- XML basiert auf einem Textformat und ist dadurch plattformunabhängig.
- Nutzdaten können mit Metainformationen angereichert werden.[65]
- XML ist ein offener und lizenzfreier Standard.[66]
- XML strukturiert Daten hierarchisch in einer Baumstruktur.[67]
- XML trennt strikt zwischen den Daten und der Datenpräsentation.[68]
- XML kann zur Abbildung beliebig komplexer Sachverhalte genutzt werden und unterliegt keiner kapazitiven Beschränkung der Dokumentengröße.[69]

---

[63] Vgl. Breymann, Mosemann (2006), S. 169 ff.
[64] Vgl. Wittenbrink, Köhler (2003), S. 16 ff.
[65] Vgl. Wöhr (2004), S.50.
[66] Vgl. Wittenbrink, Köhler (2003), S. 39.
[67] Vgl. Wittenbrink, Köhler (2003), S. 32 f.
[68] Vgl. Wittenbrink, Köhler (2003), S. 45.
[69] Vgl. Wittenbrink, Köhler (2003), S. 25.

- XML-Dokumente können sowohl von Menschen als auch Computersystemen gelesen bzw. verarbeitet werden.
- XML-Dokumente können in andere Formate transformiert werden.
- Es existieren zahlreiche, auch frei verfügbare Standardwerkzeuge für die Validitätsprüfung und Verarbeitung der XML-Dokumente.[70]

Ein XML-Dokument besteht aus Elementen (Tags) und Attributen, mit denen den Elementen Eigenschaften zugewiesen werden können. Elemente können leer sein, weitere Elemente beinhalten, wodurch eine hierarchische Struktur aufgebaut wird, oder Inhalt in Form von Text umschließen. Optional kann ein XML-Dokument über Verarbeitungsanweisungen, Kommentare und eine Dokumenttyp-Definition (DTD) verfügen. Verarbeitungsanweisungen stellen Anweisungen für verarbeitende Anwendungen dar, die den Umgang mit dem XML-Dokument beschreiben. Zu diesen gehören die XML-Deklaration, welche ein Dokument als XML-Dokument kennzeichnet, und Referenzierungen auf Stylesheets, welche zur Überführung eines XML-Dokumentes in ein anderes Format verwendet werden. Eine Verarbeitungsanweisung in XML wird mit den Zeichen `"<?"` eingeleitet und mit `"?>"` geschlossen. Verfügt ein XML-Dokument über eine XML-Deklaration, so muss diese zu Beginn des XML-Dokumentes aufgeführt sein und zwingend über die Versionsangabe der verwendeten XML-Spezifikation verfügen.[71] Zusätzlich kann die im Dokument verwendete Zeichensatzkodierung durch das Attribut `encoding` benannt werden. Der zur Verarbeitung und Validitätsprüfung verwendeten Software (XML-Parser) kann durch das Attribut `standalone` mitgeteilt werden, ob das Dokument eine externe DTD verwendet.[72] Eine DTD beschreibt das für eine Dokumentenklasse zulässige Vokabular und die hierarchische Struktur all derjenigen Elemente und Attribute, die durch das definierte Vokabular vorgegeben werden. Des Weiteren können in der DTD Elemente und Attribute als obligatorisch oder optional gekennzeichnet und Vorgabewerte für Attribute angegeben werden. Eine extern definierte DTD wird durch eine Referenz auf die DTD in das Dokument eingebunden. Alternativ ist es möglich, die Deklaration auch innerhalb des Dokumentes vorzunehmen. Wird innerhalb des Dokumentes keine DTD oder eine Referenz auf eine solche angegeben, kann der XML-Parser beim Einlesen des Dokumentes lediglich feststellen, ob es den formalen Regeln des XML-Standards genügt.

[70] Vgl. Wittenbrink, Köhler (2003), S. 18 f.
[71] Vgl. Wittenbrink, Köhler (2003), S. 89 ff.
[72] Vgl. Breymann, Mosemann (2006), S. 380.

Ein Dokument, das die folgenden Regeln erfüllt, wird als wohlgeformt bezeichnet:

- Innerhalb des Dokumentes existiert genau ein Wurzelelement.
- Alle XML-Elemente müssen mit einem End-Tag geschlossen werden.
- Alle XML-Elemente müssen korrekt geschachtelt sein.
- Attribute sind als Name-Wert-Paare zu definieren.
- Der Attributwert ist mit Anführungszeichen oder Hochkomma zu umschließen.[73]

Jedes XML-Dokument muss die Anforderung der Wohlgeformtheit erfüllen, um verarbeitet werden zu können. Entspricht die Struktur eines Dokumentes mit der durch eine DTD vorgegebenen Struktur überein, so ist das XML-Dokument eine gültige Instanz der durch die DTD definierten Dokumentenklasse. Die Validierung (Gültigkeitsprüfung) kann ebenfalls durch einen XML-Parser vorgenommen werden, wenn es sich bei diesem um einen validierenden Parser handelt.[74]

Um die getroffenen Aussagen noch einmal zu verdeutlichen, sei auf das in Abbildung 3-11 dargestellte Beispiel verwiesen. In der ersten Zeile des Beispieldokumentes befindet sich die XML-Deklaration, die das Dokument als XML-Dokument nach der Spezifikationsversion `"1.0"` kennzeichnet. Weiterhin wird festgelegt, dass die Inhalte in der `"UTF-8"`-Zeichenkodierung vorliegen und eine Referenzierung auf eine externe DTD erfolgt (`standalone`).

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!-- XML-Deklaration. -->

<!-- Referenz auf ein XML-Stylesheet -->
<?xml-stylesheet href="ausstellung.xsl"?>

<!-- Referenz auf eine externe DTD-Deklaration -->
<!DOCTYPE ausstellung SYSTEM "ausstellung.dtd">

<!-- Wurzelelement -->
<ausstellung>
        <thema>Die Wikinger</thema>

        <institution id="0101">
                <name>Historisches Museum der Pfalz</name>

                <anschrift>
                        <strasse>Domplatz</strasse>
                        <plz>67346</plz>
                        <ort>Speyer</ort>
                </anschrift>
        </institution>

        <zeitraum>
                <beginn>14.12.2008</beginn>
                <ende>12.07.2009</ende>
        </zeitraum>
</ausstellung>
```

Abbildung 3-11: Beispiel eines XML-Dokumentes.

[73] Vgl. Wöhr (2004), S. 54.
[74] Vgl. Wöhr (2004), S. 58 ff.

Die Zeichenketten `"<!--"` und `"-->"` umschließen Kommentare, die für Beschreibungen und Bemerkungen genutzt werden können. Für die Transformierung des Dokumenteninhaltes in ein anderes Format ist eine Referenz auf ein externes XML-Stylesheet angegeben. Durch die Referenzierung auf die externe DTD `"ausstellung.dtd"` wird das Dokument als Instanz der in der DTD definierten Dokumentklasse ausgewiesen. Ein XML-Parser kann überprüfen, ob das hier gezeigte Dokument tatsächlich der Struktur dieser Dokumentenklasse folgt und es als gültig erklären. Der eigentliche Inhalt des Dokumentes befindet sich in der Dokumentinstanz, welche mit dem Wurzelelement `"ausstellung"` eingeleitet wird. Ausgehend vom Wurzelelement wird eine Baumstruktur gebildet, in der alle Unterelemente und Attribute sowie Text enthalten sind. In XML wird zwischen öffnenden Elementen (Start-Tag) und schließenden Elementen (End-Tag) unterschieden. Start-Tags werden mit `"<"` und End-Tags mit `"</"` eingeleitet. Abgeschlossen werden sowohl Start-Tags als auch End-Tags mit `">"`.

Neben der bereits erwähnten Überprüfung auf Wohlgeformtheit und Validität bieten XML-Parser Schnittstellen für die Verarbeitung von XML-Dokumenten an, mit denen der Zugriff auf Strukturelemente und den Inhalt eines Dokumentes erfolgen kann. Die Aufbereitung der Dokumentenbestandteile und die Art, wie diese einer verarbeitenden Anwendung zur Verfügung gestellt werden, sind demnach abhängig vom XML-Parser. Parser unterscheidet man in solche, die die Simple API for XML (SAX) verwenden, und solche, die auf dem Document Object Model (DOM) beruhen. SAX-Parser lesen XML-Dokumente vollständig und sequentiell als Datenstrom ein. Dabei werden eintretende Ereignisse wie der Start des Dokumentes, der Beginn oder das Ende eines Elementes durch den Aufruf von Callback-Routinen einer Anwendung angezeigt. Die Anwendung übernimmt daraufhin die Verarbeitung des Ereignisses. Ist das Parsen eines Dokumentes abgeschlossen, so stellt dies gleichzeitig das Ende der Verarbeitung dar. DOM-Parser hingegen bilden zunächst die durch das XML-Dokument beschriebene Baumstruktur vollständig im Hauptspeicher ab. Erst im Anschluss daran kann mit der Verarbeitung des Dokumentes begonnen werden. Im Gegensatz zu SAX-Parsern ist es möglich, durch den Baum zu navigieren, gezielt auf einzelne Elemente und Daten des Baumes sowohl lesend als auch schreibend zuzugreifen sowie die Struktur des Baumes zu verändern. Der Nachteil ist jedoch der im Vergleich zu SAX-Parsern hohe Verbrauch an Speicherressourcen.[75]

Um XML unter der J2ME-Plattform verwenden zu können, war es bisher notwendig, auf XML-Parser und Klassenbibliotheken von Drittanbietern zurückzugreifen.[76]

[75] Vgl. Wöhr (2004), S. 68 f.
[76] Vgl. Kroll, Haustein (2003), S. 281 ff.

ASXMLP, Xparse-J, und kXML 2 sind nur einige Beispiele von XML-Parsern, die dafür von Drittanbietern zur Verfügung stehen. Alle genannten XML-Parser sind nicht-validierende Parser, d.h. es ist nicht möglich, ein XML-Dokument gegen eine DTD zu prüfen. Begründet ist dies durch den rechen- und speicherintensiven Validierungsprozess. Weitere Unterschiede ergeben sich auch in der Art, wie die Parser mit den XML-Dokumenten umgehen. Man unterscheidet zwischen Push-Parser (SAX-basiert), Model-Parser (DOM-basiert) und Pull-Parser.[77] Die Arbeitsweise von SAX und DOM wurde bereits dargestellt. Pull-Parser kombinieren die Vorteile von SAX mit denen von DOM, indem sie ein XML-Dokument schrittweise einlesen und dadurch weitaus weniger Speicherressourcen benötigen.[78] ASXMLP und Xparse sind Push-Parser, wobei ASXMLP darüber hinaus auch als Model-Parser verwendet werden kann. Der kXML 2-Parser hingegen ist ein Vertreter der Pull-Parser.

Mittlerweile ist die XML-Funktionalität auch durch das JCP mit der Spezifikation JSR-280 in die J2ME-Plattform aufgenommen worden. Durch die Spezifikation werden die optionalen Pakete Core Package und Events Package beschrieben, welche auf der CLDC 1.1 basieren. Die Größe des Core Package beträgt bei einer vollständigen Einbindung der Pakete 175 KByte.[79]

[77] Vgl. Sun Developer Network (2002).
[78] Vgl. kXML (2009).
[79] Vgl. JSR-280 (2007), S. 1 ff.

4 Lösungsansatz und Implementierung

4.1 Anforderungen an die Museumsführer-Anwendung

Die Anforderungen an die Museumsführer-Anwendung lassen sich aus dem Ziel ableiten, Museumsbesucher bei der individuellen Besichtigung von Ausstellungsinhalten durch die Bereitstellung und Darstellung von Informationen auf ihren Mobiltelefonen zu unterstützen. Die Anforderungen spiegeln dabei den Funktions- und Leistungsumfang der Anwendung wider.

Wie die traditionelle Informationsdarbietung auf Schautafeln, soll auch die Museumsführer-Anwendung dem Besucher Beschreibungen zu einem Exponat in Textform anbieten können. Des Weiteren sollen die multimedialen Fähigkeiten der Mobiltelefone zur Darstellung von Bild-, Audio- und Videoinhalten genutzt werden, um die Texte aufzulockern und bevorzugt audiovisuell lernende Menschen bei der Informationsaufnahme zu unterstützen.

Um eine Überforderung des Besuchers durch eine Fülle an Inhalten zu vermeiden, soll eine Selektion der gewünschten Inhalte sowie eine Einschränkung des Umfangs der angebotenen Informationen möglich sein. Museen und Ausstellungen müssen das Informationsbedürfnis verschiedenster Altersgruppen, vom Kind bis zum Rentner, befriedigen können. Um dem Lernvermögen und Lernbedürfnis der verschiedenen Altersgruppen Rechnung zu tragen, werden die Ausstellungsinhalte mit Hilfe der Anwendung in einer entsprechenden altersgerechten Form vermittelt. Darüber hinaus sind die Informationen in mehreren Sprachen zur Verfügung zu stellen.

Damit der Museumsführer von allen angesprochenen Altersgruppen verwendet werden kann, soll die Anwendung intuitiv und einfach bedienbar sein und über eine grafische Benutzeroberfläche verfügen. Die Navigation innerhalb der Anwendung wird über vorgegebene Menüpunkte realisiert, aus denen der Besucher wählen kann. Über die Mobiltelefontastatur vorzunehmende Eingaben sollen sich auf die Verwendung von Ziffern beschränken.

Die Anwendung ist nicht speziell für eine konkrete Einrichtung zu konzipieren. Vielmehr soll mit dem verfolgten Ansatz eine universelle Nutzung in verschiedenen Einrichtungen ermöglicht werden. Dazu ist es notwendig, durch den Besucher ausgewählte Ausstellungsinhalte dynamisch zur Laufzeit der Anwendung von einem Server zu beziehen und temporär im lokalen Speicher des Mobiltelefons für die Verarbeitung und Darstellung abzulegen. Über die Bluetooth-Schnittstelle des Mobiltelefons soll die Datenübertragung sowie die Kommunikation mit dem Server drahtlos erfolgen, wobei XML als Datenaustauschformat für die verwendeten

Nachrichten dient. Aufgrund der Tatsache, dass alle Informationen auf dem Server zum Abruf vorgehalten werden, kann die Anwendung als solche die genannten Anforderungen der Inhaltsselektion, der Einschränkung des Informationsumfanges, der altersgerechten Aufbereitung sowie der Mehrsprachigkeit nur in Verbindung mit dem Server erfüllen. Über entsprechende Einstellungsoptionen in der Anwendung kann der Besucher die vom Server anzufordernden Inhalte festlegen sowie den Detailgrad der Informationen nach seinen Vorstellungen anpassen.

Um dem Besucher die einfache Möglichkeit zu geben, zu einem späteren Zeitpunkt der Besichtigung die bereits betrachteten Exponatinformationen erneut aufzurufen, wird eine Chronik bereitgestellt, die in Form einer Listenansicht die auf dem Mobiltelefon zwischengespeicherten Daten aufführt und durch die Auswahl eines Listeneintrags die entsprechenden Informationen zum Exponat anzeigt.

Ferner wird gefordert, dass die Anwendung auf der Mehrzahl der auf dem Markt erhältlichen Mobiltelefone ausgeführt werden kann. Dies soll durch die Verwendung der Programmiersprache Java und der J2ME-Plattform erreicht werden.

Die Anwendung des Museumsführers ist als Prototyp zu realisieren, der die grundlegenden Funktionalitäten für eine spätere Verwendung beinhaltet.

4.2 Entwurf und Spezifikation

4.2.1 Aufbau der Infrastruktur

Eine der wesentlichen Anforderungen an den zu entwickelnden Museumsführer ist die Übertragung von Informationen über eine Netzwerkschnittstelle, die das Mobiltelefon bereitstellt. Eine Beschreibung der in Mobiltelefonen integrierten Netzwerkschnittstellen ist dem Kapitel 3.4.2 zu entnehmen. Bluetooth ist derzeit die einzige Technologie, die in Mobiltelefonen weit verbreitet ist und eine für den Besucher kostenlose Datenübertragung bereitstellt.

Um Bluetooth als Übertragungstechnologie nutzen zu können, muss innerhalb des Museums oder der Ausstellung eine entsprechende Infrastruktur vorhanden sein, die einen Datenaustausch zwischen der Mobiltelefonanwendung und einem, die Informationen bereitstellenden, Infrastrukturelement erlaubt. Im Folgenden wird der Aufbau einer möglichen Infrastruktur vorgestellt. Der Aufbau als solcher ist als Vorschlag zu verstehen, der für die Entwicklung und Implementierung des Museumsführers zugrunde gelegt wurde.

Das Gesamtsystem setzt sich aus drei Teilen zusammen:

- einem zentralen Datenbankserver,
- einem oder mehreren Bluetooth-Zugangspunkten sowie
- der Anwendung des Museumsführers auf dem Mobiltelefon.

Abbildung 4-1 zeigt den vereinfachten Aufbau der vorgeschlagenen Infrastruktur. Die Datenhaltung von Informationen sowie deren Bereitstellung für den Museumsführer erfolgt durch einen zentralen Datenbankserver, welcher zusätzlich über ein Content Management System (CMS) verfügt. Das CMS ermöglicht dem Personal der Ausstellung einen einfachen Zugriff auf die Datenbasis für Verwaltungs-, Wartungs- und Pflegearbeiten. Dafür sind keinerlei Programmierkenntnisse seitens des Personals erforderlich. Darüber hinaus erlaubt es der vorgestellte Infrastrukturansatz, Änderungen an der Datenbasis unmittelbar und in der gesamten Ausstellung verfügbar sowie abrufbar zu machen. Die Einrichtungen können somit flexibel und schnell auf sich ändernde Ausstellungsinhalte reagieren und mit geringem Aufwand die Bereitstellung aktueller Informationen gewährleisten.

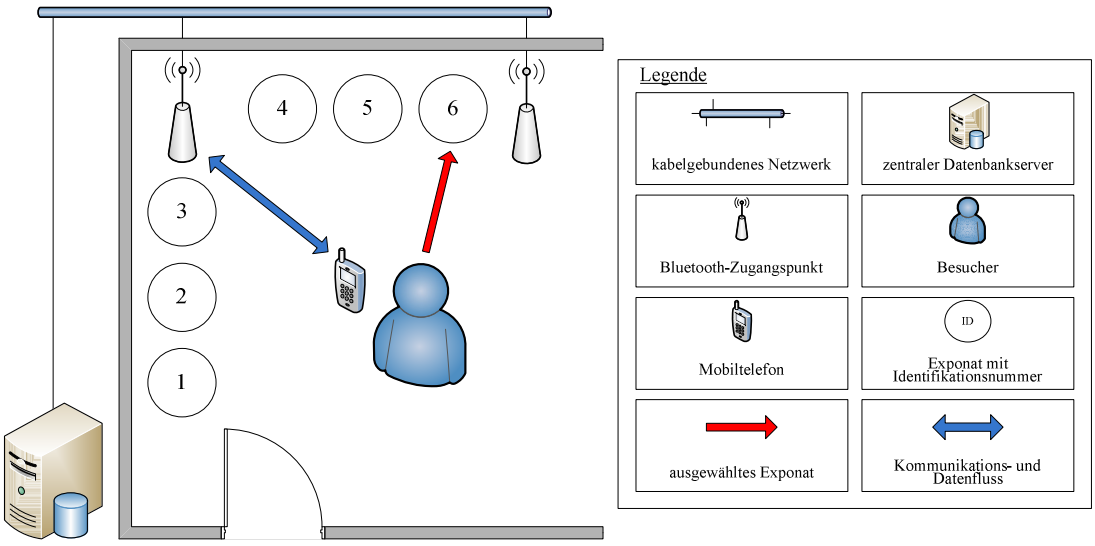

Abbildung 4-1: Vereinfachter Aufbau der Infrastruktur.

Nachdem der Besucher ein Exponat ausgewählt hat (vgl. Abbildung 4-1 - roter Pfeil), löst dieser über die Eingabe der entsprechenden Exponatnummer die Informationsabfrage aus. Alle Exponate innerhalb der Ausstellung sind mit einer eindeutigen Nummer versehen. Nach Eingabe der Nummer versucht die Anwendung, eine Verbindung zum Datenbankserver aufzubauen und die Anfrage zu senden. Die Kommunikation zwischen der Mobiltelefonanwendung und dem Server erfolgt nicht direkt, sondern über Bluetooth-Zugangspunkte, die innerhalb der Ausstellungsräume

installiert sind und zwischen den beiden Teilnehmern vermitteln. Ein Bluetooth-Zugangspunkt kann beispielsweise ein preisgünstiger Mini-PC mit Bluetooth-Schnittstelle sein, der über ein kabelgebundenes Netzwerk mit dem Server verbunden ist. Bevor also die Anfrage an den Server gesendet werden kann, muss der Museumsführer die in der Umgebung befindlichen Bluetooth-Zugangspunkte ermitteln und einen für die Kommunikation auswählen (vgl. Abbildung 4-1 - blauer Pfeil). Bewegt sich der Besucher aus dem Sendebereich des ausgewählten Zugangspunktes heraus, wird zunächst versucht, die beim ersten Suchlauf weiteren gefundenen Zugangspunkte zu kontaktieren. Sollte dies misslingen, startet automatisch ein neuer Suchlauf.

Nach dem Erhalt einer Informationsanfrage beginnt der Server mit der Zusammenstellung der angeforderten Informationen aus der Datenbank. Anschließend werden die Informationen an den Bluetooth-Zugangspunkt übermittelt, der diese wiederum an das Mobiltelefon überträgt. Aufgrund der Fülle an existierenden Multimediadatenformaten ist sicherzustellen, dass diese auch vom Mobiltelefon unterstützt und wiedergegeben werden können. Dazu ist es von Vorteil, wenn der Server Kenntnis über die vom Mobiltelefon unterstützten multimedialen Inhalte besitzt und auch nur diese bereitstellt. Liegt ein multimedialer Inhalt nicht in einem durch das Mobiltelefon unterstützten Format in der Datenbank vor, muss der Server durch den Einsatz von Konvertierungsprogrammen den Inhalt in ein entsprechendes Format überführen. Darüber hinaus könnten dem Server weitere grundlegende Eigenschaften des Gerätes, wie die Bildschirmauflösung etc., mitgeteilt werden, um eine geeignete Darstellung der Informationen zu gewährleisten.

Zur Veranschaulichung des Kommunikationsflusses sei auf das Sequenzdiagramm in Abbildung 4-2 verwiesen. Dieses stellt die bei der Anforderung von Informationen zu einem Exponat zwischen den Infrastrukturelementen ausgetauschten Nachrichten und Daten dar. Ausgehend vom Museumsführer wird eine Informationsanfrage mit der Nummer des Exponates an den Bluetooth-Zugangspunkt gesendet. Diese Nachricht wird an den Datenbankserver weitergereicht, der daraufhin die gewünschten Informationen aus der Datenbank abruft. Dies geschieht unter Berücksichtigung der Eigenschaften des Mobiltelefons. Die Daten werden gegebenenfalls in ein anderes Multimediaformat konvertiert und anschließend über den Bluetooth-Zugangspunkt auf das Mobiltelefon übertragen, wo diese für die weitere Verarbeitung und Darstellung zwischengespeichert werden.

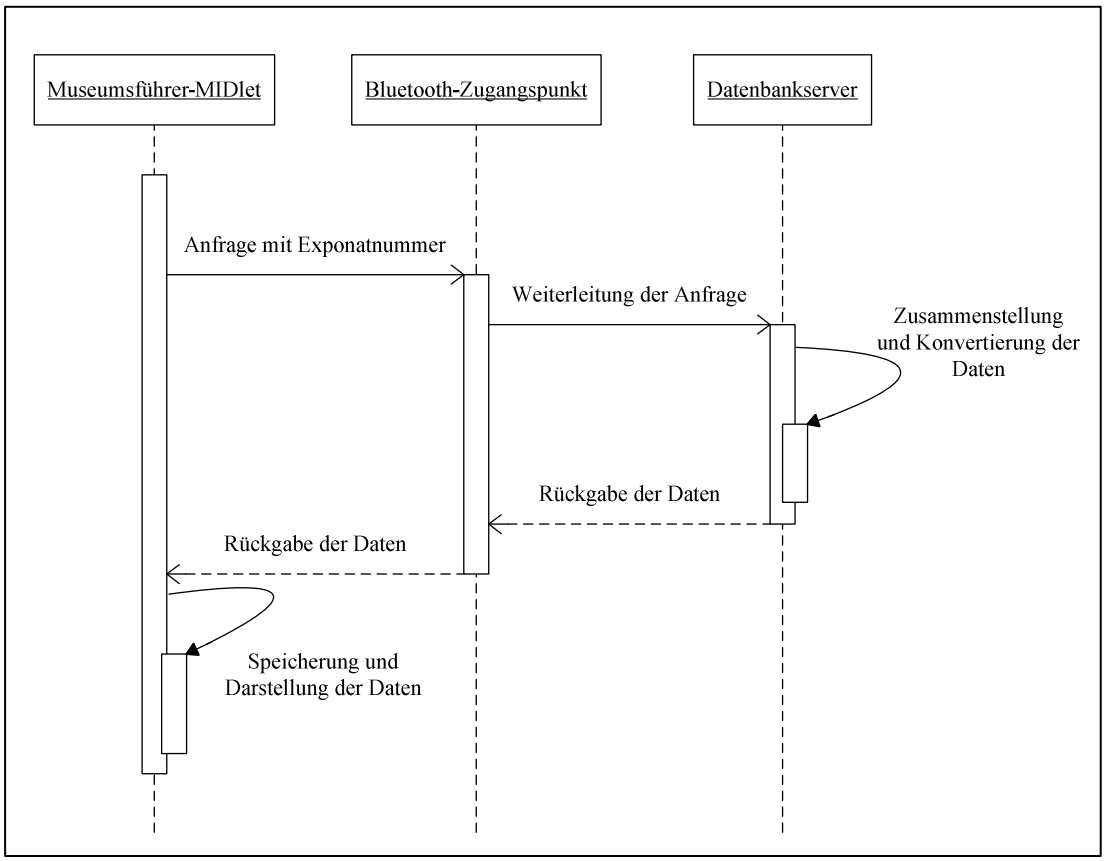

Abbildung 4-2: Sequenzdiagramm des Kommunikationsflusses.

Die Entwicklung des Datenservers sowie der Bluetooth-Zugangspunkte sind nicht Gegenstand dieser Arbeit. Die genannten Infrastrukturbestandteile wurden lediglich zum besseren Verständnis des Gesamtsystems und der Anwendung des Museumsführers erwähnt.

4.2.2 Anwendungsbestandteile

Auf Basis der in Kapitel 4.1 formulierten Anforderungen an die zu entwickelnde Museumsführeranwendung sowie dem in Kapitel 4.2.1 beschriebenen Infrastrukturaufbau des Gesamtsystems können sechs Hauptfunktionseinheiten identifiziert werden, die für die Implementierung der Anwendung benötigt werden.

Die Hauptbestandteile des MIDlets sowie deren Abhängigkeitsbeziehungen untereinander sind in dem in Abbildung 4-3 gezeigten Komponentendiagramm dargestellt.

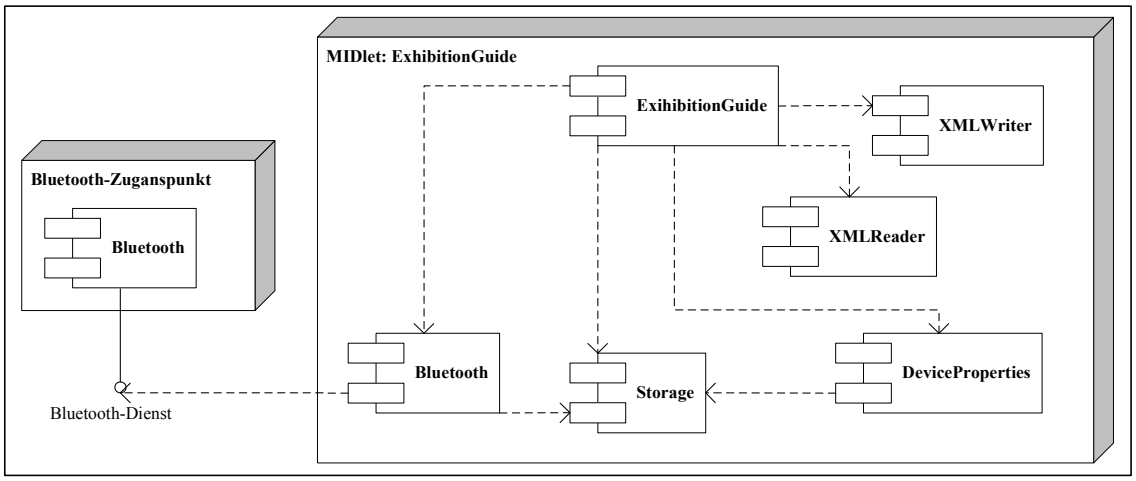

Abbildung 4-3: Komponentendiagramm der Museumsführeranwendung.

Die nachstehende Auflistung enthält die Komponenten mit ihren jeweiligen Funktionen:

- ExhibitionGuide: Programmablauf, grafische Benutzeroberfläche, Darstellung der Exponatinformationen,
- Bluetooth: Zugriff und Verwaltung auf die Bluetooth-Schnittstelle des Mobiltelefons,
- Storage: Zugriff auf den internen Mobiltelefonspeicher,
- XMLWriter: Erstellung von XML-Nachrichten,
- XMLReader: Auswertung der erhaltenen XML-Nachrichten sowie
- DeviceProperties: Abfrage der Geräteeigenschaften.

Um zu verdeutlichen, dass die Anwendung des Museumsführers auf einen durch den Bluetooth-Zugangspunkt bereitgestellten Dienst zur Datenübertragung und Kommunikation mit dem Datenbankserver angewiesen ist, wurde auch die Bluetooth-Komponente des Zugangspunktes abgebildet. Diese wird jedoch im weiteren Verlauf der Arbeit nicht näher betrachtet.

4.3 Implementierung des Museumsführers

4.3.1 Technologieauswahl

Die Verwendung der J2ME-Plattform als Basis für die Implementierung des Museumsführer-MIDlets, machte es notwendig, aus den vorhandenen Konfigurationen und Profilen eine Auswahl zu treffen. Die Auswahl konnte nicht willkürlich erfolgen, da daraus unmittelbare Auswirkungen auf die Nutzung anderer Bestandteile der J2ME-Plattform resultieren.

Die Forderung, multimediale Inhalte mit der Anwendung wiedergeben zu können (vgl. Kapitel 4.1), setzt unter der J2ME-Plattform die Verwendung der MMAPI voraus, die auf Bestandteilen der CLDC 1.1 sowie MIDP 2.0 aufbaut (vgl. Kapitel 3.6). Aufgrund dieser Abhängigkeit wurden diese Konfigurations- und Profilversionen für die Implementierung des Museumsführer-MIDlets ausgewählt und verwendet.

Zur Realisierung einer einfach zu bedienenden grafischen Benutzeroberfläche werden die mit der MIDP 2.0 bereitgestellten High-Level-Grafikelemente, wie Formulare, Auswahlboxen, Eingabefelder etc., verwendet. Bei der Datenhaltung fiel die Wahl auf Record Stores, da der Zugriff auf externe Speichermedien unter der J2ME-Plattform den in Kapitel 3.3.7 beschriebenen Sicherheitsrestriktionen unterliegt und somit ohne eine Zertifizierung des MIDlets hätte nicht praktikabel eingesetzt werden können. Bluetooth stellt momentan die einzig sinnvolle Möglichkeit dar, Daten zwischen den Geräten zu übertragen. Es ist auf die speziellen Anforderungen der Mobiltelefone, wie den geringen Energiebedarf zugeschnitten, überträgt Daten ohne zusätzliche Kosten für den Benutzer und hat sich als Standard etabliert. XML wird aus den in Kapitel 3.7 genannten Gründen für den Datenaustausch und die Kommunikation zwischen dem Museumsführer-MIDlet und dem Datenbankserver genutzt. Zur Implementierung der XML-Funktionalität wurde auf den kXML 2-Parser zurückgegriffen, da dieser mit einer Paketgröße von neun KByte im Vergleich zur JSR-280 deutlich kleiner ist und die geschilderten Vorteile eines Pull-Parsers aufweist.

Tabelle 4-1 zeigt nochmals in einer Übersicht die zur Verfügung stehenden und in Kapitel 3 vorgestellten Technologien. Die bei der Implementierung des Museumsführer-MIDlets verwendeten Technologien sind hellgrau hinterlegt.

<table>
<tr><th></th><th colspan="4">Technologie</th></tr>
<tr><td>Konfiguration (CLDC)</td><td colspan="2">CLDC 1.0</td><td colspan="2">CLDC 1.1</td><td>CLDC 1.1.1</td></tr>
<tr><td>Profil (MIDP)</td><td colspan="2">MIDP 1.0</td><td colspan="2">MIDP 2.0</td><td>MIDP 2.1</td></tr>
<tr><td>Grafikprogrammierung</td><td colspan="3">High-Level</td><td colspan="2">Low-Level</td></tr>
<tr><td>Datenhaltung</td><td colspan="3">Record Stores</td><td colspan="2">externe Speichermedien</td></tr>
<tr><td>Multimediaunterstützung</td><td colspan="5">Mobile Media API</td></tr>
<tr><td>Datenübertragung</td><td>GSM, u.a.</td><td>WLAN</td><td>Bluetooth</td><td>IrDA</td></tr>
<tr><td>Austauschformat</td><td colspan="4">XML</td></tr>
<tr><td>XML-Parser</td><td colspan="2">ASXMLP</td><td colspan="2">Xparse-J</td><td>kXML2</td></tr>
</table>

Tabelle 4-1: Auswahl der verfügbaren Technologien für die Implementierung.

Als Entwicklungsumgebung kam Netbeans IDE 6.5 in Verbindung mit dem Sun Java Wireless Toolkit 2.5.2 zum Einsatz.

4.3.2 Allgemeiner Ablauf

Aus den in Kapitel 4.2.2 beschriebenen Komponenten wurden entsprechende Klassen abgeleitet und implementiert. Die erstellten Klassen sowie deren Abhängigkeiten zueinander sind in Abbildung 4-4 dargestellt. Um die Komplexität der Darstellung des Klassendiagramms gering zu halten, werden die angebotenen Attribute und Methoden nicht gezeigt.

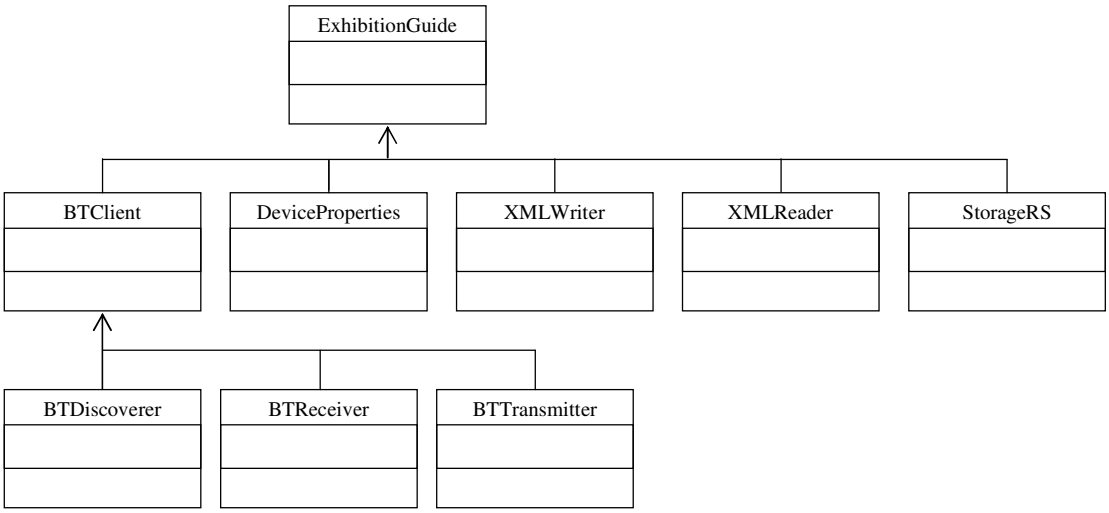

Abbildung 4-4: Klassendiagramm des Museumsführer-MIDlets.

Neben den abgebildeten Klassen im Klassendiagramm existieren weitere, die im Verlauf der Beschreibung der Implementierung erläutert werden, sofern es für das Verständnis der Abläufe erforderlich ist.

Im Folgenden wird die Implementierung des MIDlets am Ablauf der für die Informationsanforderung nötigen Schritte erläutert. Zunächst wird gezeigt, wie die Eigenschaften des Mobiltelefons ausgelesen und in einer XML-Nachricht zur Übertragung an den Datenserver strukturiert werden.

Damit sich ein Besucher Informationen zu einem Exponat anzeigen lassen kann, sind die folgenden Schritte seitens der Anwendung durchzuführen:

- Verarbeitung der durch den Besucher eingegebenen Exponatnummer,
- Erstellen einer XML-Nachricht zur Informationsanforderung,
- Suche nach im Umkreis befindlichen Bluetooth-Zugangspunkten,
- Verbindungsaufbau zu einem Zugangspunkt und Übertragung der XML-Anfragenachricht an den Server,
- Empfang der Daten vom Server,

- Speicherung der Daten im lokalen Speicher des Mobiltelefons zur weiteren Verarbeitung sowie
- Darstellung der erhaltenen Informationen.

4.3.3 Abfrage der Geräteeigenschaften

Damit der Server die Informationen entsprechend den Eigenschaften und Funktionalitäten des Mobiltelefons bereitstellen kann (vgl. Kapitel 4.2.1), müssen diese ausgelesen werden. Dies übernimmt die entwickelte Klasse `DeviceProperties`.

Die für den Server relevanten Daten über das Mobiltelefon sind:

- Eigenschaften des Bildschirms (Farbbildschirm, Auflösung etc.)
- Geräteeinstellungen (Sprache, Zeichencodierung etc.),
- Angaben zur Java-Plattform (Konfigurationen, Profile),
- Bluetooth-Adresse und Name des Gerätes sowie
- Angaben über die unterstützten Multimediainhalte.

Das Auslesen einiger dieser Eigenschaften wurde bereits im Verlauf des Kapitels 3 gezeigt (Funktion: `System.getProperty()`).

Um Einzelheiten des Bildschirms abfragen zu können, muss eine Instanz der Klasse `Display` erzeugt werden, indem die Methode `getDisplay()` aufgerufen wird (vgl. Quellcode 4-1). Die Methode erwartet als Übergabeparameter die MIDlet-Instanz.

```
import javax.microedition.lcdui.Display;
import javax.microedition.lcdui.Canvas;

    [...]
    //Instanz von Display erzeugen
    Display loc_Display = Display.getDisplay(this);

    //Displayeigenschaften auslesen
    Boolean loc_DisplayColored = loc_Display.isColor();
    int loc_DisplayColorDepth = loc_Display.numColors();
    int loc_DisplayWidth = loc_Canvas.getWidth();
    int loc_DisplayHeight = loc_Canvas.getHeight();
```

Quellcode 4-1: Abfrage der Bildschirmeigenschaften.

Das `Display`-Objekt enthält Informationen darüber, ob der Bildschirm Farben anzeigen kann und in welcher Farbtiefe. Um die Auflösung auszulesen, muss die Low-Level API der LCDUI verwendet werden. Dazu wird temporär ein `Canvas`-Objekt erzeugt, das normalerweise für den direkten Zugriff auf den Bildschirm sowie dessen Inhalte gedacht ist. Mit Hilfe des `Canvas`-Objektes erhält man dann über die Methoden

`getWidth()` und `getHeight()` die benötigten Informationen zur Auflösung des Bildschirms.

Für die Abfrage von Eigenschaften der integrierten Bluetooth-Schnittstelle benötigt man die Klasse `LocalDevice` der Bluetooth-API. `LocalDevice` bietet Zugriff auf die lokale Bluetooth-Schnittstelle und dient dem Auslesen von statischen Informationen des Gerätes. Eine Instanz der Klasse `LocalDevice` wird durch die Methode `getLocalDevice()` erzeugt (vgl. Quellcode 4-2). Die Bluetooth-Adresse sowie den Namen des Bluetooth-Gerätes erhält man über die korrespondierenden Methodenaufrufe `getBluetoothAddress()` und `getFriendlyName()`.

```
import javax.bluetooth.LocalDevice;

    [...]
    //Instanz von LocalDevice erzeugen
    LocalDevice loc_BTDevice = LocalDevice.getLocalDevice();

    //Abfrage der Bluetooth-Eigenschaften
    String loc_BTAddress = loc_BTDevice.getBluetoothAddress();
    String loc_BTName = loc_BTDevice.getFriendlyName();
```

Quellcode 4-2: Abfrage der Bluetooth-Eigenschaften.

Mit Hilfe des im Kapitel 3.6 vorgestellten `Manager`-Objektes der MMAPI werden die vom Mobiltelefon unterstützten Multimediainhalte abgefragt. Dies erfolgt durch die Verwendung der Methode `getSupportedContentTypes()` des `Manager`-Objektes, die ein String-Array zurück gibt, in dem die entsprechenden Multimediainhalte enthalten sind (vgl. Quellcode 4-3).

```
import javax.microedition.media.Manager;
import java.util.Vector;

    [...]
    //Vector der Media-Objekte
    Vector loc_Media = new Vector();

    //Abfrage der unterstützten Media-Content-Types über Manager-Objekt der MMAPI
    String[] loc_ContentTypes = Manager.getSupportedContentTypes(null);

    for (int i = 0; i < loc_ContentTypes.length; i++) {
        //anhand des unterstützten Content die unterstützten Protokolle erfragen
        String[] loc_Protocols = Manager.getSupportedProtocols(loc_ContentTypes[i]);

        for (int j = 0; j < loc_Protocols.length; j++) {
              //neues Media-Objekt mit Abfrageergebnissen erzeugen
              Media loc_MediaContent = new Media(
              loc_ContentTypes[i].substring(0, loc_ContentTypes[i].indexOf("/")),
              loc_ContentTypes[i].substring(loc_ContentTypes[i].indexOf("/")    +    1,
              loc_ContentTypes[i].length()),loc_Protocols[j]
              );

              loc_Media.addElement(loc_MediaContent);
        }
    }
}
```

Quellcode 4-3: Abfrage der unterstützten Multimediainhalte.

Das String-Array beinhaltet die ausgelesenen Multimediainhalte nach dem Schema: Medieninhalt/Medientyp (Beispiel: `"audio/mp3"`). Da Mobiltelefone das Abspielen von Multimediainhalten aus verschiedenen Quellen unterstützen, wie beispielsweise über das HTTP-Protokoll oder den lokalen Speicher des Gerätes, erfolgt im Anschluss an die Ermittlung der Multimediainhalte die Abfrage nach den unterstützten Quellen. Dazu wird die Methode `getSupportedProtocols()` des `Manager`-Objektes genutzt, die als Übergabeparameter ein zuvor ermitteltes Multimediaformat erwartet und zu diesem die Quellprotokollangabe ermittelt.

Die ausgelesenen Informationen werden zur besseren Strukturierung und späteren Weiterverarbeitung in einem Objekt der Klasse `Media` gesichert. Dazu wird für jedes im String-Array enthaltene Element dessen Zeichenkette so manipuliert, dass der Medieninhalt vom Medientyp getrennt und beide Angaben zusammen mit der Quellprotokollangabe in einem `Media`-Objekt abgelegt werden. Ist das `Media`-Objekt mit Daten gefüllt, wird es einem `Vector`-Objekt hinzugefügt. Nachdem alle Elemente des String-Arrays abgearbeitet sind, enthält der `Vector` sämtliche `Media`-Objekte und somit alle vom Mobiltelefon unterstützten Multimediainhalte.

Der Aufbau der Klasse `Media` mit deren Attributen und Methoden ist in Abbildung 4-5 dargestellt. Die von der Klasse verwendeten Attribute beziehen sich auf den Medieninhalt (`m_Content`), auf den Medientyp (`m_Type`) sowie auf die Quellprotokollangabe (`m_Protocol`). Des Weiteren stellt die Klasse Methoden zum Lesen und Schreiben der Attribute bereit.

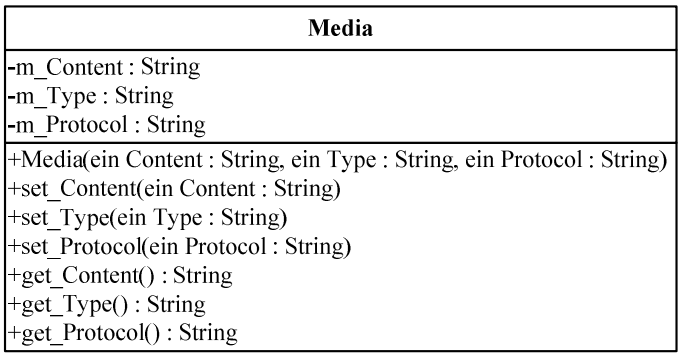

Abbildung 4-5: Klassendiagramm der Media-Klasse.

Nach dem Auslesen aller relevanten Eigenschaften des Mobiltelefons, werden die Daten an die Klasse `XMLWriter` übergeben und daraus die in Abbildung 4-6 in Ausschnitten zu sehende XML-Nachricht für den Server erstellt. Die in der XML-Deklaration verwendete Zeichensatzcodierung entspricht der auf dem Mobiltelefon verwendeten. Die restlichen Informationen innerhalb der XML-Nachricht entsprechen den zuvor ausgelesenen Eigenschaften des Mobiltelefons.

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="yes"?>

<device>
     <display color="true" colordepth="65536" width="176" height="176"/>

     <systemproperties platform="sonyericssonw810i/r4ea031" locale="de">
         <profile type="midp-1.0 midp-2.0"/>
         <configuration type="cldc-1.1"/>
     </systemproperties>

     <bluetooth address="001963da9559" name="sk-w810i"/>

     <media mmapiversion="1.1">
         [...]
         <element content="audio" type="mp3" protocol="http"/>
         <element content="audio" type="mp3" protocol="file"/>
         <element content="audio" type="wav" protocol="http"/>
         <element content="audio" type="wav" protocol="file"/>
         [...]
         <element content="video" type="3gpp" protocol="http"/>
         <element content="video" type="3gpp" protocol="file"/>
         <element content="video" type="mpeg" protocol="http"/>
         <element content="video" type="mpeg" protocol="file"/>
         [...]
         <element content="image" type="gif" protocol="http"/>
         <element content="image" type="gif" protocol="file"/>
     </media>
</device>
```

Abbildung 4-6: Ausschnitt der XML-Nachricht mit den Eigenschaften des Mobiltelefons.

Die Übertragung der XML-Nachricht an den Server soll an dieser Stelle nicht näher erläutert werden. Eine Beschreibung des Informationsaustausches über die Bluetooth-Schnittstelle ist in Kapitel 4.3.6 zu finden, wo näher auf das Versenden einer Anfrage nach Exponatinformationen eingegangen wird.

4.3.4 Anforderung von Exponatinformationen

Um Informationen zu einem Exponat zu erhalten, muss der Besucher die am Ausstellungsstück angebrachte Exponatnummer über die Tastatur des Mobiltelefons in eine Eingabemaske eingeben. Die Realisierung einer Benutzeroberfläche inklusive der Ereignisbehandlung eines Tastendrucks wurde in Kapitel 3.3.6 gezeigt. Die Benutzeroberfläche besteht aus den High-Level-Elementen der LCDUI, wie Formen, Listen, Textfeldern und Texteingabefeldern.

Im Weiteren werden einige Sachverhalte vorausgesetzt. Es wird angenommen, dass die in Kapitel 4.3.3 beschriebenen Geräteeigenschaften des Mobiltelefons auf dem Server vorhanden sind und der Besucher die Anwendungseinstellungen an seine persönlichen Bedürfnisse angepasst hat. Die Einstellungen umfassen die gewünschten Inhalte bezüglich Text, Bild, Audio und Video, die Spracheinstellungen, eine Angabe des Vorwissens zum Ausstellungsthema, um die Informationen im Detailgrad zu variieren, sowie eine Zielgruppenangabe zur altersgerechten Aufbereitung der Informationen.

Hat der Besucher die Exponatnummer eingegeben, wird die Nummer zusammen mit den Benutzereinstellungen an die Klasse `XMLWriter` übergeben. Die vom `XMLWriter` erzeugte XML-Anfragenachricht ist in Abbildung 4-7 dargestellt.

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="yes"?>

<request>
    <content id="1" type="15" language="de" preknowledge="1" audience="3"/>
</request>
```

Abbildung 4-7: XML-Anfragenachricht nach Exponatinformationen.

Die XML-Nachricht enthält eine Anfrage nach dem Exponat mit der Nummer `"1"` in deutscher Sprache. Die Bedeutung der Angaben zum Vorwissen und zur Zielgruppe können von der Einrichtung beliebig festgelegt werden. In diesem Beispiel steht die Eins für geringes Vorwissen, wodurch die Exponatinformationen eher einen Überblick über die Thematik geben sollten. Die Altersgruppe ist mit der Drei für die Gruppe der Erwachsenen belegt.

Das Attribut `"type"` bezieht sich auf die vom Besucher gewünschten Inhalte. Da dem Besucher in den Einstellungen zu den Inhalten eine Mehrfachauswahl angeboten wird, muss in geeigneter Weise kodiert werden, welche Inhalte der Besucher zulassen und betrachten möchte. Die in den Einstellungen getroffene Auswahl ist über Konstanten realisiert, deren Wertebelegung wie folgt festgelegt ist:

- `"1"`, wenn Texte zugelassen sind,
- `"2"`, wenn Bilder zugelassen sind,
- `"4"`, wenn Audio zugelassen ist sowie
- `"8"`, wenn Videos zugelassen sind.

Durch eine bitweise Oder-Verknüpfung der vier Konstanten ergeben sich die gewünschten Inhalte, die dem Besucher angeboten werden. Sind alle Inhalte zugelassen, entspricht das Ergebnis der Oder-Verknüpfung `"15"`, wie in Abbildung 4-7 gezeigt. Entscheidet sich der Besucher lediglich Text- und Videoinhalte betrachten zu wollen, ergibt sich der Wert `"9"`.

Nachdem die XML-Anfragenachricht erstellt wurde, gilt es einen im Umkreis befindlichen Zugangspunkt für die Übertragung der Informationen an den Server zu suchen. Aus den gefundenen Zugangspunkten wird ein beliebiger ausgewählt und für die Datenübertragung verwendet.

4.3.5 Suche nach Bluetooth-Zugangspunkten

Die Suche nach umliegenden Bluetooth-Geräten sowie deren angebotenen Diensten erfolgt durch die Verwendung des SDP (vgl. Kapitel 3.4.2) und wird in der Klasse `BTDiscoverer` realisiert. Dazu müssen die Klassen `DiscoveryAgent` und `DiscoveryListener` aus der Bluetooth-API in die Klasse `BTDiscoverer` eingebunden werden. Die Klasse `DiscoveryAgent` stellt unter anderem die Methoden `startInquiry()` zur Auslösung einer Gerätesuche und `searchServices()` zur Suche der angebotenen Dienste bereit. Die Rückmeldung der durch diese Methoden ausgelösten Ereignisse, wie die Beendigung der Gerätesuche (`inquiryCompleted()`) oder die beendete Dienstsuche (`serviceSearchCompleted()`), erfolgt über die eingebundenen CallBack-Methoden des `DiscoveryListener`-Interfaces.

Darüber hinaus werden weitere Klassen der Bluetooth-API benötigt:

- LocalDevice: vgl. Kapitel 4.3.3,
- RemoteDevice: zur Abfrage statischer Informationen entfernter Bluetooth-Geräte sowie
- UUID: zur Identifizierung der Bluetooth-Dienste entfernter Geräte.

Um einen Gerätesuchlauf zu starten, muss zunächst eine Instanz des lokalen Bluetooth-Gerätes über die Methode `getLocalDevice()` der Klasse `LocalDevice` erzeugt werden (vgl. Quellcode 4-4). Danach wird mit der Methode `getDiscoveryAgent()` eine Instanz des lokalen `DiscoveryAgents` erstellt. Der Suchlauf wird anschließend mit der Methode `startInquiry()` eingeleitet. Als Übergabeparameter werden der Suchmodus sowie das `DiscoveryListener`-Objekt, das die CallBack-Methoden bereitstellt, verwendet. Der Parameter `DiscoveryAgent.GIAC` besagt, dass ein kompletter Suchlauf ausgeführt werden soll.

```
//Zugriff auf den Discovery Agent über Abfrage des LocalDevices
LocalDevice loc_LocalDevice = LocalDevice.getLocalDevice();
DiscoveryAgent loc_discoveryAgent = loc_LocalDevice.getDiscoveryAgent();

//Start des Suchlaufes nach umliegenden Bluetooth-Geräten
loc_discoveryAgent.startInquiry(DiscoveryAgent.GIAC, this);
```

Quellcode 4-4: Start eines Suchlaufs nach Bluetooth-Geräten.

Können bei der Gerätesuche Bluetooth-Geräte identifiziert werden, ruft der `DiscoveryListener` die CallBack-Methode `deviceDiscovered()` für jedes gefundene Gerät auf (vgl. Quellcode 4-5). Die Übergabeparameter der CallBack-Methode umfassen ein `RemoteDevice`-Objekt und den Gerätetyp (`COD = Class of Device`). Für die im Anschluss an die Gerätesuche durchzuführende Dienstsuche werden die gefundenen Geräte in einem `Vector`-Objekt zwischengespeichert. Da bei dem Suchlauf

auch die in der Nähe befindlichen Bluetooth-Schnittstellen der Mobiltelefone anderer Besucher erkannt werden, erfolgt eine Filterung der Suchergebnisse mit Hilfe des Gerätetyps. Die Filterung basiert auf der `MajorDeviceClass`. Bei Bluetooth-Schnittstellen von Laptops, PCs und Servern entspricht die `MajorDeviceClass` dem Wert `"256"`. Die `MajorDeviceClass` von Mobiltelefonen ist hingegen auf den Wert `"512"` festgelegt.

```
public void deviceDiscovered(RemoteDevice btDevice, DeviceClass cod) {
      //Einschränkung der gefundenen Geräte auf PCs (MajorDeviceClass = 256)
      //COD = Class of Device
      if (cod.getMajorDeviceClass() == 256) {
           vector_remoteDevices.addElement(btDevice);
      }
}
```

Quellcode 4-5: Filterung der gefundenen Bluetooth-Geräte.

Nach Beendigung des Geräte-Suchlaufs schließt sich die Suche nach den angebotenen Diensten der gefundenen Geräte an. Dazu wird die UUID des durch den Bluetooth-Zugangspunkt offerierten Dienstes benötigt. Für jeden gefundenen Zugangspunkt wird nacheinander die Dienstsuche mit der Methode `searchServices`() des `DiscoveryAgents` durchgeführt. Als Übergabeparameter erwartet die Methode die Dienst-UUID, das `RemoteDevice`-Objekt des Zugangspunktes sowie den `DiscoveryListener`.

Bietet ein Zugangspunkt den gesuchten Dienst an, wird die CallBack-Methode `servicesDiscovered()` des `DiscoveryListeners` aufgerufen. Um zu einen späteren Zeitpunkt auf den Dienst zugreifen zu können, wird dessen Adresse, der sog. Uniform Resource Locator (URL), benötigt, der aus den Dienst-Eigenschaften ausgelesen wird. Die ermittelten Dienst-URLs werden in einem weiteren `Vector` zwischengespeichert. Als Protokoll für die Datenübertragung wurde RFCOMM ausgewählt (vgl. Kapitel 3.4.2). Um zu verhindern, dass andere Protokolle außer RFCOMM in den `Vector` der Dienst-URLs aufgenommen werden, erfolgt eine Filterung nach diesem Protokoll. Das Filtern besteht in der Suche nach der Zeichenkette `"btspp://"` innerhalb der Dienst-URLs, da alle RFCOMM nutzenden Dienste damit beginnen.

Die Verwendung der Dienst-URLs beim Aufbau einer Verbindung zu einem Bluetooth-Zugangspunkt wird im folgenden Kapitel beschrieben.

4.3.6 Verbindungsaufbau und Datenübertragung

Die Voraussetzung zum Datenaustausch über die Bluetooth-Schnittstelle ist eine Verbindung zu einem beim Suchlauf identifizierten Zugangspunkt. Die Suche nach in der Nähe befindlichen Zugangspunkten wurde in Kapitel 4.3.5 erläutert und liefert als Ergebnis einen `Vector` mit Dienst-URLs. Der `Vector` wird an die implementierte Klasse `BTClient` übergeben, die den Verbindungsaufbau in der Methode `OpenConnection()` mittels des in Kapitel 3.3.4 beschriebenen GCF realisiert. Der Zugriff auf das GCF erfolgt durch die Einbindung der Klasse `Connector`.

Aufgrund der in Kapitel 4.2.1 beschriebenen Struktur des Gesamtsystems ist es unerheblich, welcher der identifizierten Zugangspunkte für die Datenübertragung zum Server ausgewählt wird. Daher enthält die Methode `OpenConnection()` eine Zählschleife, in der die Iteration über alle im `Vector` enthaltenen Dienst-URLs erfolgt (vgl. Quellcode 4-6).

```
import javax.microedition.io.Connector;
import javax.microedition.io.StreamConnection;
import java.io.DataOutputStream;
import java.io.DataInputStream;

StreamConnection glb_StreamConnection = null;
DataInputStream glb_DataInputStream = null;
DataOutputStream glb_DataOutputStream = null;

[...]

public boolean OpenConnection() {
   boolean loc_ConnectionEstablished = false;

   //Versuch einen der gefundenen Bluetooth-Zugangspunkte zu kontaktieren
   for (int i = 0; i < vector_ServiceURLs.size(); i++) {
       try {
           if (glb_StreamConnection == null) {
                glb_StreamConnection = (StreamConnection) Connector.open(
                       vector_ServiceURLs.elementAt(i).toString(),
                       Connector.READ_WRITE);
           }

           if ((glb_StreamConnection != null) & !loc_ConnectionEstablished) {
                loc_ConnectionEstablished = true;

                glb_DataInputStream = glb_StreamConnection.openDataInputStream();
                glb_DataOutputStream = glb_StreamConnection.openDataOutputStream();
           }
       } catch (Exception e) {System.err.println(e.getMessage());}
   }
}
```

Quellcode 4-6: Aufbau einer Verbindung zu einem Bluetooth-Zugangspunkt.

Bei jedem Iterationsschritt wird eine Dienst-URL an die Methode `open()` des GCF übergeben, die daraufhin versucht den Zugangspunkt zu kontaktieren. Neben der Dienst-URL erwartet das GCF eine Konstante als Übergabeparameter, die den Operationsmodus bestimmt, d.h. ob auf die Verbindung lesend und/oder schreibend zugegriffen werden kann. Konnte das GCF erfolgreich einen Kontakt zu einem

Zugangspunkt herstellen, liefert die Methode `open()` die Verbindung vom Typ `Connection` zurück. Damit besteht zunächst nur eine Verbindung zwischen dem Zugangspunkt und dem Mobiltelefon, der eigentliche Übertragungskanal ist jedoch noch nicht aufgebaut. Auf einer Verbindung können zwei Übertragungskanäle eingerichtet werden, einer für den Empfang und einer für die Übertragung von Daten. Die Klassen `DataInputStream` und `DataOutputStream` erzeugen die Übertragungskanäle und stellen Methoden für Lese- und Schreiboperationen von einfachen Datentypen auf der Verbindung (`Long`, `Char`, `Boolean` etc.) bereit, was die Handhabung der Datenübertragung vereinfacht. Um `DataInputStreams` und `DataOutputStreams` nutzen zu können, muss zunächst das von der Methode `open()` zurückgegebene `Connection`-Objekt durch Typumwandlung in eine `StreamConnection` überführt werden. Anschließend werden die Übertragungskanäle durch die Methoden `openDataInputStream()` (Datenempfang) und `openDataOutputStream()` (Datenübertragung) auf der Verbindung erzeugt und den entsprechenden Variablen zugewiesen.

Um letztendlich die generierte XML-Nachricht (vgl. Kapitel 4.3.4) zur Anforderung von Exponatinformationen an den Server zu senden, wird die XML-Nachricht zusammen mit den erstellten Übertragungskanälen an die Methode `TransmitMessage()` der Klasse `BTTransmitter` übergeben. Diese realisiert den eigentlichen Sendevorgang. Um Daten in den `DataOutputStream` zu schreiben, werden dessen Methoden `writeLong()` sowie `write()` verwendet. Mittels `writeLong()` wird die Anzahl der in der XML-Nachricht enthaltenen Zeichen als Datentyp `Long` an den Zugangspunkt übertragen. Danach erfolgt die Übergabe der XML-Nachricht an die Methode `write()`, welche die Daten in den `DataOutputStream` schreibt. Um sicherzustellen, dass alle im Zwischenspeicher der Bluetooth-Schnittstelle gehaltenen Daten versendet werden, wird abschließend die Methode `flush()` aufgerufen.

Hat der Zugangspunkt die Anfrage entgegengenommen, leitet dieser die Informationen an den Datenbankserver weiter. Der Datenbankserver wertet daraufhin die erhaltene XML-Nachricht aus und stellt die gewünschten Informationen zusammen.

4.3.7 Datenempfang und Speicherung der Informationen

Im Anschluss an die Übermittlung der XML-Anfragenachricht mittels der vorgestellten Methode `TransmitMessage()`, erfolgt der Aufruf der Methode `ReceiveMessage()` der Klasse `BTReceiver`, die auf eintreffende Nachrichten des Servers wartet. Dazu wird der Methode ebenfalls der beim Verbindungsaufbau erstellte `DataInputStream` und

`DataOutputStream` übergeben. Die Rückantwort des Servers besteht aus einer XML-Nachricht, die alle ausgewählten und verfügbaren multimedialen Inhalte als Referenzen sowie sämtliche Textinformationen zu einem Exponat beinhaltet. Eine derartige XML-Nachricht ist im Anhang dieser Arbeit angeführt (vgl. Anhang A).

Der Empfang von Daten über die Bluetooth-Schnittstelle erfolgt ähnlich wie deren Übertragung. Um dem Zugangspunkt zu signalisieren, dass das MIDlet für den Datenempfang bereit ist, wird in der Methode `ReceiveMessage()` zunächst ein `Boolean`-Datentyp mit dem Wert `"true"` in den `DataOutputStream` geschrieben. Die dazu verwendete Methode `writeBoolean()` wird vom `DataOutputStream` bereitgestellt. Nachfolgend muss auf die Antwort des Zugangspunktes gewartet werden, der, sobald die erforderlichen Daten bereitstehen, zuerst die Länge der zu übertragenden Nachricht sendet. Mittels der Methode `readLong()` des `DataInputStreams` wird daraufhin die Nachrichtenlänge ausgelesen. Diese Methode wartet solange, bis auf dem eingehenden Datenstrom ein entsprechender `Long`-Datentyp empfangen wurde. Aufgrund des begrenzten Pufferspeichers der Bluetooth-Schnittstelle werden die vom Zugangspunkt zu sendenden Daten nicht in einem Block, sondern in 512 Byte großen Blöcken segmentiert übertragen. Nach jedem übertragenen Datenblock wartet der Zugangspunkt auf eine Bestätigung des Museumsführer-MIDlets, bevor mit dem Senden weiterer Daten fortgefahren wird. Die Datenblöcke werden mit der Methode `readFully()` des `DataInputStreams` in ein Byte-Array eingelesen und zwischengespeichert. Auch diese Methode wartet solange, bis das gesamte Array mit Daten gefüllt ist. In einem `String`-Objekt werden die einzelnen Datenblöcke rekombiniert, indem zum ersten Block der zweite, dann zu beiden der dritte usw. zusammengefügt werden. So entsteht nach und nach die Antwortnachricht des Servers. Die zu Beginn des Sendevorgangs übermittelte Nachrichtenlänge dient der Ermittlung der noch zu empfangenden Daten und wird dazu verwendet, den Zwischenspeicher in Form des Byte-Arrays dynamisch an die verbleibende Größe der Datenblöcke anzupassen.

Nach Abschluss des Datenempfangs und der erfolgreichen Rekombination der Datenblöcke zu einer Zeichenkette wird die erhaltene XML-Nachricht an die Methode `get_Content()` der Klasse `XMLReader` übergeben. Die Methode übernimmt das Parsen der Nachricht und gibt als Ergebnis ein Objekt der Klasse `Content` zurück, welches die in der XML-Nachricht übermittelten Informationen enthält und für die weitere Verarbeitung verwendet wird. Das `Content`-Objekt bildet die Basis für die Organisation aller zu einem Exponat vorhandenen Informationen und beinhaltet zusätzlich Verwaltungsinformationen, wie beispielsweise die Exponatnummer und die Sprachangabe. Die eigentlichen Inhalte werden unter Zuhilfenahme der Klassen

`TextContent` und `MultimediaContent` abgebildet. Zu jedem Exponat wird genau ein Haupttext übertragen, sofern der Besucher diesen Inhaltstyp nicht in den Einstellungen ausgeschlossen hat. Der Haupttext besitzt eine eindeutige ID, enthält eine Überschrift und kann mehrere Absätze umfassen. Ein Absatz wird durch die Klasse `Stanza` realisiert, die diesem eine entsprechende Absatzüberschrift zuweist sowie den eigentlichen Text beinhaltet. Die `Stanza`-Objekte werden im `TextContent`-Objekt in einem `Vector` abgelegt. Multimediale Inhalte werden durch die Klasse `MultimediaContent` abgebildet. Neben den Verwaltungsinformationen wie der eindeutigen ID, einem Titel sowie dem Inhaltstyp besteht zusätzlich die Möglichkeit zu jedem multimedialen Inhalt einen oder mehrere erklärende Textabsätze bereitzustellen. Dies geschieht ebenfalls durch die Verwendung der bereits beschriebenen Klasse `Stanza`. Da die Anzahl der zu einem Exponat angebotenen multimedialen Inhalte nicht limitiert ist, werden im `Content`-Objekt die einzelnen `MultimediaContent`-Objekte in einem `Vector` verwaltet. Der Aufbau der Klasse `Content` sowie der anderen zur Abbildung der Exponatinformationen benötigten Klassen ist im Klassendiagramm in Abbildung 4-8 dargestellt. Die Abbildung zeigt den Aufbau der Klassen aufgrund deren Komplexität nur in Auszügen. Ferner wurde auf die Darstellung der innerhalb der Klassen implementierten Methoden zum Lesen und Schreiben der objektinternen Variablen verzichtet.

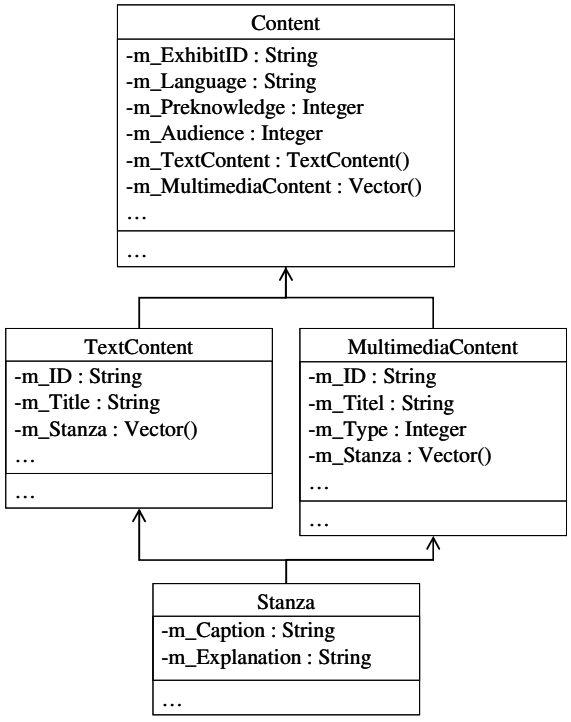

Abbildung 4-8: Klassendiagramm der Klassen Content, TextContent, MultimediaContent und Stanza.

Die in der XML-Antwortnachricht enthaltenen Informationen werden im lokalen Speicher des Mobiltelefons zwischengespeichert. Damit entfällt eine erneute

Übertragung der Daten vom Server, falls nochmals auf die Informationen zurückgegriffen werden soll. Der Zwischenspeicher wird mit den in Kapitel 3.5 besprochenen Record Stores realisiert. Der Zugriff auf die Record Stores ist in der Klasse `StorageRS` implementiert. Wie bereits beschrieben, sind die Datensätze eines Record Stores als Byte-Arrays realisiert. Der zur Ablage im Record Store erzeugte Datensatz beinhaltet die Verwaltungsinformationen aus dem `Content`-Objekt, gefolgt von der XML-Antwortnachricht. Durch das Voranstellen der Verwaltungsinformationen vereinfacht sich die spätere Suche nach Datensätzen innerhalb des Record Stores, da somit gezielt nach einem Datensatz gesucht werden kann, ohne die abgelegte XML-Nachricht am Ende des Datensatzes auslesen und verarbeiten zu müssen. Für die Erstellung eines Datensatzes wird ein `ByteArrayOutputStream` erzeugt, der es erlaubt, den im Byte-Strom abgelegten Datensatz in ein Byte-Array zu überführen. Da der `ByteArrayOutputStream` lediglich Byte-Informationen verwalten kann, wird auf diesem zusätzlich ein `DataOutputStream` erstellt, der das Schreiben von einfachen Datentypen in den Datenstrom unterstützt (vgl. Quellcode 4-7). Werden Daten auf den `DataOutputStream` geschrieben, erfolgt automatisch die Übernahme und Konvertierung in den `ByteArrayOutputStream`. Nachdem die Verwaltungsinformationen wie Exponatnummer, Inhaltstyp, Sprache etc. in den `DataOutputStream` geschrieben wurden, erfolgt die Übergabe der XML-Antwortnachricht des Servers. Dabei wird nicht die Methode `writeUTF()` des `DataOutputStreams` verwendet, da diese die Nachricht in eine UTF-Zeichenkette umwandelt und dadurch etwaig vorhandene Umlaute nicht korrekt wiedergegeben werden. Stattdessen wird die Methode `write()` genutzt, welche die XML-Nachricht als Bytefolge in den Datenstrom übernimmt. Damit bei einem späteren Auslesen ermittelt werden kann, welche Länge die XML-Nachricht aufweist, muss deren Länge vor der eigentlichen Nachricht abgelegt werden, was durch die Methode `writeLong()` erfolgt.

```
   ByteArrayOutputStream loc_BAOStream = new ByteArrayOutputStream();
   DataOutputStream loc_DataOutStream = new DataOutputStream(loc_BAOStream);

   //Verwaltungsinformationen speichern
   loc_DataOutStream.writeUTF(loc_Content.get_ExhibitID());
   loc_DataOutStream.writeInt(StorageRS.ContentType_Text);
   loc_DataOutStream.writeUTF(loc_Content.get_Language());
   [...]

   //Länge der XML-Nachricht
   loc_DataOutStream.writeLong(loc_ReceivedMessage.length());

   //XML-Nachricht in Record Store schreiben
   loc_DataOutStream.write(loc_ReceivedMessage.getBytes(), 0,
       loc_ReceivedMessage.length());

   //Übergabe der Daten zur Speicherung in den Record Store
   loc_RecordStore.WriteContentData(StorageRS.RS_TEXTCONTENT, loc_BAOStream);
}
```

Quellcode 4-7: Zusammenstellung eines Record Store-Datensatzes.

Sind alle erforderlichen Daten für die Speicherung des Datensatzes abgelegt, wird der erzeugte `ByteArrayOutputStream` an die Methode `WriteContentData()` der Klasse `StorageRS` übergeben. Um der Methode mitzuteilen, in welchem Record Store der Datensatz abzulegen ist, werden Konstanten der Klasse `StorageRS` verwendet. Für die Speicherung von Exponatinformationen existieren zwei Record Stores. Der erste Record Store nimmt alle vom Server übermittelten XML-Nachrichten auf und im zweiten werden alle multimedialen Inhalte zu einem Exponat gespeichert. Da es sich bei dem eben erzeugten Datensatz um Textinformationen handelt, wird der Datensatz folglich im ersten Record Store abgelegt.

Bevor der Zugriff auf den Record Store erfolgen kann, muss dieser geöffnet werden. Dazu wird die Konstante der Klasse `StorageRS`, die den Record Store identifiziert, an die Methode `OpenRecordStore()` übergeben (vgl. Quellcode 4-8). Innerhalb dieser Methode indiziert die Konstante den für den Record Store verwendeten Namen in einem String-Array und der benötigte Record Store wird geöffnet. Ergibt die anschließende Überprüfung der Speicherverfügbarkeit, dass der Datensatz im Record Store abgelegt werden kann, wird der neue Datensatz über die Methode `addRecord()` der Klasse `RecordStore` hinzugefügt. Da diese Methode ein Byte-Array als Übergabeparameter erwartet, wird die Methode `toByteArray()` des `ByteArrayOutputStreams` aufgerufen, die den erzeugten Datenstrom in ein solches konvertiert. Die anderen Parameter der Methode bewirken, dass der gesamte Inhalt des Byte-Arrays in den Datensatz übernommen wird. Diese Parameter stellen Indizes auf das Byte-Array dar und ermöglichen das Schreiben von Teilbereichen des Arrays.

```
public void WriteContentData(int IN_RecordStore, ByteArrayOutputStream IN_Content){

   //öffne den ausgewählten RecordStore
   OpenRecordStore(IN_RecordStore);

   //Überprüfung der Speicherplatzverfügbarkeit
   if (get_SizeAvailable(IN_RecordStore) > IN_Content.size()) {
       glb_Recordstore.addRecord(IN_Content.toByteArray(), 0, IN_Content.size());
   }
}
```

Quellcode 4-8: Anlegen eines neuen Datensatzes im Record Store.

Im nächsten Schritt werden die zu den Exponaten gewünschten multimedialen Inhalte auf das Mobiltelefon übertragen. Die verfügbaren Inhalte wurden durch die XML-Antwortnachricht des Servers mitgeteilt. Die Referenzen auf die multimedialen Inhalte stehen im `Content`-Objekt in dem gezeigten `Vector` bereit. In einer Zählschleife wird für jedes in diesem `Vector` enthaltene Element eine XML-Anfragenachricht erzeugt und an den Server übertragen. Der einzige Unterschied zu dem in Kapitel 4.3.4 beschriebenen Verfahren ist die Verwendung der eindeutigen Identifikationsnummer des multimedialen Inhaltes anstelle der Exponatnummer. Alle weiteren Abläufe

gleichen den zuvor erläuterten. Bei der Speicherung der multimedialen Inhalte in den zweiten Record Store werden umfangreichere Verwaltungsinformationen erfasst als bei der Zusammenstellung des Text-Datensatzes, das Grundprinzip bleibt jedoch bestehen.

Sind alle zu einem Exponat gewünschten Inhalte auf das Mobiltelefon übertragen und im lokalen Speicher abgelegt, können die Informationen dem Besucher angezeigt werden.

4.3.8 Darstellung und Wiedergabe der Informationen

Für die Darstellung der Exponatinformationen ist die Klasse `ExhibitionGuide` zuständig. Als Basis für die Erstellung der grafischen Benutzeroberfläche zur Anzeige der Informationen wurde die Klasse `Form` der High-Level LCDUI verwendet. Ein `Form`-Objekt bildet einen virtuellen Bildschirm, der beliebige Text-, Bild- und Videoelemente aufnehmen und darstellen kann.

Zur Anzeige der im `Content`-Objekt vorgehaltenen Textinformationen existieren zwei Alternativen. Entweder wird der Text direkt an das `Form`-Objekt übergeben oder man verwendet die Komponente `StringItem`, die anschließend der `Form` hinzugefügt wird. Die Texte sind in einzelne Absätze untergliedert (vgl. Kapitel 4.3.7 - Klasse `Stanza`), d.h. jeder Absatz besteht aus einer Absatzüberschrift und dem eigentlichen Textinhalt. Bei der direkten Übergabe der Textinformationen an das `Form`-Objekt kann die vorliegende Struktur nur schlecht beibehalten werden, da sich die Absatzüberschriften nicht eindeutig vom Text abgrenzen lassen, z.B. durch eine fettgedruckte Überschrift. Mit Hilfe der `StringItems` lässt sich die Absatzstruktur hingegen realisieren, da jedes `StringItem`-Objekt eine Überschrift und einen Textkörper besitzt. Nach der Erstellung eines `StringItem`-Objektes wird die Überschrift und der darzustellende Text durch Aufruf der entsprechenden Methoden `setTitle()` und `setText()` zugewiesen. Um das `StringItem` der `Form` hinzuzufügen, wird die Methode `append()` des `Form`-Objektes aufgerufen, die als Übergabeparameter das `StringItem`-Objekt erwartet.

Über die Komponente `ImageItem` erfolgt die Darstellung der Bildinhalte auf dem Bildschirm. Die Referenzen auf die zum Exponat gehörenden Bilder sind im `Content`-Objekt abgelegt. Die eigentlichen Bilddaten befinden sich in einem Record Store (vgl. Kapitel 4.3.7). Um ein neues `ImageItem`-Objekt zu erzeugen, muss das Bild zunächst im Record Store gesucht und ausgelesen werden. Ist der Datensatz gefunden, liegen die Bilddaten als Byte-Array vor.

```
ImageItem loc_Image = Image.createImage(loc_ByteArray, 0, loc_ByteArray.size());
```

Quellcode 4-9: Erzeugung eines Bildes zur Darstellung auf dem Bildschirm.

Um aus den Byte-Daten ein Bild zu erstellen, wird die Methode `createImage()` der Klasse `Image` aufgerufen, die aus dem Byte-Array ein `ImageItem` erzeugt (vgl. Quellcode 4-9). Um das Bild auf dem Bildschirm anzuzeigen, wird wiederum die Methode `append()` des `Form`-Objektes aufgerufen und das `ImageItem` als Parameter übergeben.

Zur Wiedergabe von Audio- und Videoinhalten wird der in Kapitel 3.6 erläuterte `Manager` der MMAPI verwendet. Der `Manager` erzeugt das `Player`-Objekt, das den vom `Manager` bereitgestellten Datenstrom verarbeitet und abspielt. Da auch die Audio- und Videoinhalte im Record Store abgelegt sind, muss genau wie bei den Bildinhalten der entsprechende Datensatz im Record Store gesucht und ausgelesen werden. Da die Audio- und Videodaten als Byteinformationen vorliegen, der `Manager` diese jedoch nicht verarbeiten kann, muss das Byte-Array in einen `InputStream` umgewandelt werden. Der `InputStream` wird dann zusammen mit dem im `MultimediaObject` enthaltenen Multimediatyp und dem Multimediaformat an die Methode `createPlayer()` des `Managers` übergeben, der daraus ein `Player`-Objekt erzeugt.

```
InputStream loc_InputStream = new ByteArrayInputStream(loc_ByteArray);

Player loc_Player = Manager.createPlayer( loc_InputStream,
                                          loc_MultimediaObject.get_Type()   +   "/"   +
                                          loc_MultimediaObject.get_Format());

loc_Player.realize();
loc_Player.prefetch();
loc_Player.start();
```

Quellcode 4-10: Erzeugung eines Player-Objektes zur Wiedergabe von Audio- und Videoinhalten.

Ist das `Player`-Objekt erfolgreich erstellt, muss dieses zunächst die Zustände `Realized` und `Prefetched` durchlaufen, um die Wiedergabe der Inhalte anschließend durch Aufruf der Methode `start()` beginnen zu können.

5 Anwendungsbeispiel

Nachdem die Entwicklung und Implementierung der Mobiltelefonanwendung des Museumsführers in Kapitel 4 erläutert wurde, widmet sich dieses Kapitel einer Demonstration der Anwendung anhand eines Beispielrundganges in den Meisterhäusern Dessau.

Im Rahmen der Umsiedlung des Staatlichen Bauhaus von Weimar nach Dessau im Jahre 1925 entstanden neben dem Hochschulgebäude für Kunst, Design und Architektur[80] auch Wohnstätten für die Bauhausmeister (Meisterhäuser). Die Wohnanlage umfasste ein Haus für den Bauhaus-Direktor Walter Gropius und drei Doppelhäuser für die anderen Bauhausmeister. Die Architektur und die Innenausstattung der Meisterhäuser sowie die durch die jeweiligen Bewohner vorgenommene Farbgestaltung der Räume veranschaulichen die Einheit von Form und Funktionalität sowie die Verkörperung eines zur damaligen Zeit neuen Wohngefühls. Mit Schließung des Bauhaus in Dessau 1932 wurden die Meisterhäuser verkauft und als Mehrfamilienhäuser genutzt, wobei der Käufer die architektonische Umgestaltung der Meisterhäuser zur Auflage erhielt. Während des Zweiten Weltkrieges wurden das Direktorenhaus sowie die angrenzende Doppelhaushälfte durch einen Bombentreffer zerstört. Seit Beginn der 1990er Jahre erfolgte die schrittweise Sanierung und Restauration der verbliebenen Meisterhäuser und die Rücksetzung in den ursprünglichen Erbauungszustand von 1925/26. Ferner wurde die durch die ersten Bewohner vorgenommene Farbgestaltung der einzelnen Räume weitestgehend wiederhergestellt. Die Meisterhäuser sowie das Bauhaus-Hochschulgebäude sind 1996 von der UNESCO zum Weltkulturerbe erklärt worden. Im Jahr 2002 wurde die Restaurierung des letzten Meisterhauses abgeschlossen. Seither stehen alle verbliebenen Meisterhäuser den Besuchern zur Besichtigung offen.[81] Das Hauptaugenmerk bei einer Besichtigung der Meisterhäuser liegt auf der Architektur der Gebäude sowie der experimentellen Farbgebung der Räumlichkeiten. Die Häuser sowie die einzelnen Räume sind somit als Exponate zu verstehen, über die sich Besucher informieren können. Des Weiteren werden Gemälde und Zeichnungen der Bauhausmeister präsentiert sowie Gegenstände wie Geschirr, Türklinken und Lampen im Bauhaus-Design ausgestellt.

Für die Durchführung des Beispielrundganges ist das Vorhandensein der in Kapitel 4.2.1 beschriebenen Infrastruktur notwendig. Da bis auf das im Rahmen dieser Arbeit erstellte Museumsführer-MIDlet noch keines der in der Infrastruktur beschrieben Elemente existiert, musste für das Beispiel die Infrastruktur vereinfacht umgesetzt

[80] Vgl. Bauhaus Dessau (2009).
[81] Vgl. Meisterhäuser Dessau 1 (2009).

werden, um das MIDlet demonstrieren zu können. Dazu wurde eine einfache Java-Anwendung erstellt, die sowohl die Funktionalität des Servers als auch des Zugangspunktes erfüllt und auf einem Notebook mit integrierter Bluetooth-Schnittstelle ausgeführt wird. Die Serverfunktionalität beschränkt sich auf das Bereitstellen von Exponatinformationen, die in Text- und Multimediadateien auf der Festplatte des Notebooks abgelegt sind. Über die Bluetooth-Schnittstelle des Notebooks erzeugt die Serveranwendung den Bluetooth-Dienst für den Datenaustausch mit dem MIDlet und realisiert somit die Funktionalität des Zugangspunktes. Die Serveranwendung wartet auf eingehende Anfragen des MIDlets, nimmt diese entgegen und übermittelt anschließend die zu einem Exponat hinterlegten Informationen an das Mobiltelefon.

Das Museumsführer-MIDlet wurde auf einem Sony Ericsson W810i Mobiltelefon installiert und ausgeführt. Der Bildschirminhalt des Mobiltelefons wurde für die im weiteren Verlauf des Beispiels dargestellten Abbildungen in Form von Fotografien festgehalten.

Der Beispielrundgang findet in der Doppelhaushälfte des Bauhaus-Meisters Paul Klee statt, wobei der Besucher versucht, Informationen zum *Treppenaufgang* dieses Hauses mit Hilfe des Museumsführer-MIDlets abzurufen.

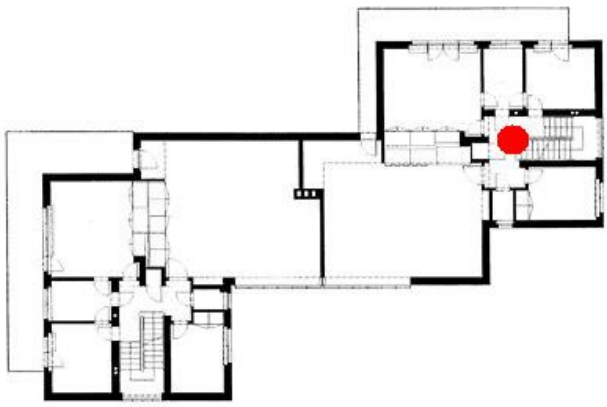

Abbildung 5-1: Grundriss des Obergeschosses im Meisterhaus Kandinsky/Klee.[82]

Der verwendete Datensatz ist im Anhang (vgl. Anhang A) aufgeführt und beinhaltet neben einer Beschreibung des *Treppenaufganges* in Textform jeweils zwei Bilder, Audiokommentare und Videos sowie eine Standortangabe, wie sie in Abbildung 5-1 zu sehen ist. Es wird angenommen, dass sich der Besucher bereits in dem auf dem Grundriss rot markierten Bereich befindet und die Anwendung das erste Mal startet.

[82] Die Abbildung wurde entnommen aus Meisterhäuser Dessau 2 (2009).

Beim Start des Museumsführer-MIDlets wird dem Besucher zunächst der in Abbildung 5-2 dargestellte Begrüßungsbildschirm angezeigt. Dieser enthält eine Grafik und einen kurzen Begrüßungstext. Die Bildschirmelemente sind als Ressourcen in der JAR-Datei enthalten, deren Einbindung zur Laufzeit der Anwendung erfolgt. Damit ist eine Individualisierung für andere Einrichtungen ohne erneute Kompilierung des MIDlets gewährleistet. Es müssen lediglich die in der JAR-Datei enthaltenen Ressourcen ausgetauscht werden.

Abbildung 5-2: Begrüßungsbildschirm des Museumsführer-MIDlets.

Bei der anschließenden Initialisierung erfolgt das Einlesen der Programm- und Benutzereinstellungen sowie das Auslesen der Geräteeigenschaften (vgl. Abbildung 5-3). Den Fortschritt des jeweils ausgeführten Initialisierungsschrittes kann der Besucher auf dem Bildschirm verfolgen. Ein erfolgreich abgeschlossener Teilschritt wird mit einem `"OK"` quittiert. Sollten während der Ausführung eines Initialisierungsschrittes Fehler auftreten, so wird der Besucher über eine entsprechende Fehlermeldung darauf aufmerksam gemacht und die Anwendung gegebenenfalls beendet. Die Programmeigenschaften werden aus einer in der JAR-Datei enthaltenen XML-Konfigurationsdatei ausgelesen, die unter anderem die UUID des Bluetooth-Dienstes beinhalten. Basierend auf der UUID erfolgt im Anschluss an die erfolgreiche Initialisierung ein Suchlauf nach den in der näheren Umgebung befindlichen Bluetooth-Zugangspunkten.

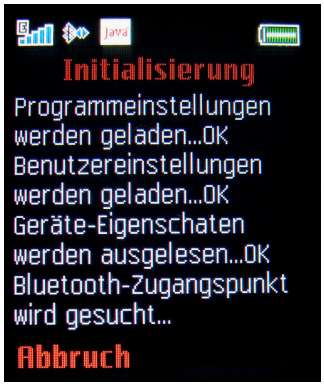

Abbildung 5-3: Initialisierung des Museumsführer-MIDlets.

Endet die Suche nach einem Bluetooth-Zugangspunkt ergebnislos, kann dies mehrere Ursachen haben. Entweder hat der Besucher die Bluetooth-Funktionalität seines Mobiltelefons nicht aktiviert, oder er befindet sich außerhalb der Sendereichweite eines Zugangspunktes. Der Besucher erhält den in Abbildung 5-4 gezeigten Hinweis mit der Aufforderung, die Bluetooth-Schnittstelle des Mobiltelefons zu aktivieren. Weiterhin soll der Besucher überprüfen, ob er sich in der Nähe eines Zugangspunktes befindet. Anschließend kann ein neuer Suchlauf durch Betätigung der `Suchen`-Taste ausgelöst werden.

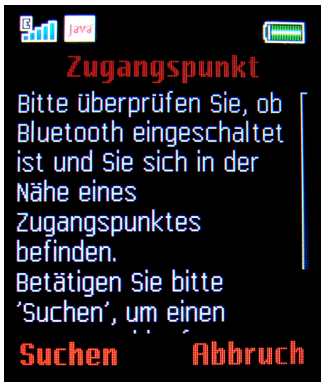

Abbildung 5-4: Erneute Suche nach einem Bluetooth-Zugangspunkt.

Ist ein Zugangspunkt gefunden, erhält der Besucher die Aufforderung, einen an der Kasse erhaltenen Benutzerschlüssel in ein Eingabefeld einzugeben (vgl. Abbildung 5-5). Mittels des Benutzerschlüssels kann eine Autorisierung des Zugriffs auf den angebotenen Dienst erfolgen, womit ein einfaches Bezahlsystem abgebildet wird. Nach Bestätigung der Eingabe wird eine Verbindung über den Zugangspunkt zum Server aufgebaut und der Benutzerschlüssel sowie die bei der Initialisierung ermittelten Geräteeigenschaften an den Server übertragen. Voraussetzung für einen erfolgreichen Verbindungsaufbau ist, dass der Besucher gemäß der in Kapitel 3.3.7 beschriebenen Sicherheitsmechanismen der J2ME-Plattform den Zugriff

auf die Bluetooth-Funktionalität seines Mobiltelefons explizit gestattet (vgl. Abbildung 5-6).

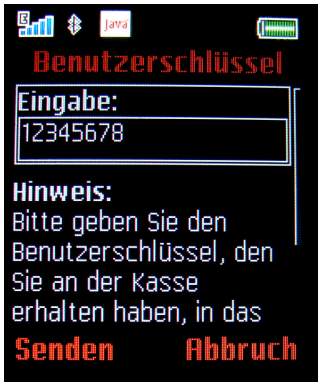

Abbildung 5-5: Eingabemaske für den Benutzerschlüssel.

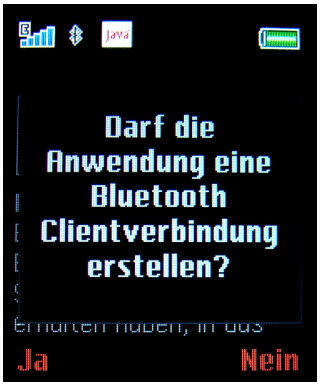

Abbildung 5-6: Sicherheitsabfrage beim Aufbau einer Bluetooth-Verbindung.

Hat der Server den übertragenen Benutzerschlüssel akzeptiert, so sendet dieser eine Bestätigungsnachricht an den Museumsführer und der Besucher gelangt in das Hauptmenü (vgl. Abbildung 5-7). Andernfalls erhält der Besucher eine Meldung, dass der Zugriff aufgrund eines fehlerhaften oder abgelaufenen Schlüssels nicht möglich ist.

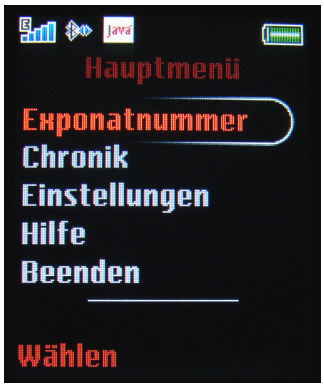

Abbildung 5-7: Hauptmenü des Museumsführer-MIDlets.

Im Hauptmenü stehen dem Besucher die Menüpunkte Exponatnummer, Chronik, Einstellungen, Hilfe sowie Beenden zur Auswahl, welche die folgenden Funktionen bereitstellen:

- Exponatnummer: Abruf von Informationen zu einem Exponat nach Eingabe einer Exponatnummer,
- Chronik: Listenansicht der bereits betrachteten Exponate,
- Einstellungen: Anpassung der Programmeinstellungen,
- Hilfe: Anzeige der Programmhilfe mit Bedienhinweisen sowie
- Beenden: Verlassen der Anwendung.

Bevor darauf eingegangen wird, wie ein Besucher Informationen zum Exponat abrufen kann und die Chronik zu verwenden ist, erfolgt zunächst eine Betrachtung der Programmeinstellungen. Die Einstellungen, die der Besucher anpassen kann, sind in Abbildung 5-8 zu sehen und umfassen die Kategorien Sprache, Wissensstand, Inhaltsauswahl sowie Zielgruppe. Je nach Kategorie kann der Besucher entweder eine Einfach- oder Mehrfachauswahl treffen.

Abbildung 5-8: Programmeinstellungen des Museumsführer-MIDlets.

Über die Einstellungen kann der Besucher Einfluss nehmen auf:

- die Sprache bezogen auf die Menüführung und die Informationen zu Exponaten,
- den Detailgrad der Informationen zu einem Exponat,
- die Aufbereitung der Informationen für eine spezifische Zielgruppe sowie
- die Filterung der gewünschten Informationen in Bezug auf textbasierende und multimediale Inhalte.

Die Einstellungen werden durch Betätigung der `Speichern`-Taste in einen Record Store gesichert und bei der Kommunikation mit dem Server in der XML-Nachricht als Attribute übergeben. Ein Beispiel für eine derartige Anfragenachricht wurde in Kapitel 4.3.4 in Abbildung 4-7 gezeigt.

Nachdem der Besucher die Programmeinstellungen nach seinen Bedürfnissen angepasst hat, wird nun gezeigt, wie dieser Informationen zum Exponat *Treppenaufgang* abrufen kann.

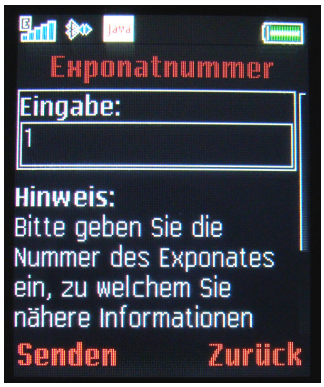

Abbildung 5-9: Eingabemaske für eine Exponatnummer.

Hierfür wählt der Besucher den Menüpunkt `Exponatnummer` aus dem Hauptmenü aus und gibt in der daraufhin angezeigten Eingabemaske die Nummer des Exponates ein (vgl. Abbildung 5-9). Das Exponat *Treppenaufgang* besitzt in diesem Beispiel die Nummer `"1"`. Mit Betätigung der `Senden`-Taste wird erneut eine Verbindung zum Server aufgebaut und die Informationen zum *Treppenaufgang* werden angefordert. Nach erfolgreicher Übertragung der Informationen kann der Besucher die gewünschten Inhalte aus einem Menü auswählen (vgl. Abbildung 5-10).

Abbildung 5-10: Informationsauswahl zu einem ausgewählten Exponat.

Durch die Auswahl des Menüpunktes `Beschreibung` werden dem Besucher Informationen zum Exponat in Textform angezeigt. Zur Strukturierung der Beschreibung enthält diese neben einer Hauptüberschrift eine beliebige Anzahl von Absätzen, die wiederum mit einer Absatzüberschrift versehen sein können (vgl. Kapitel 4.3.7). In Abbildung 5-11 ist der beschreibende Text für das hier betrachtete Exponat zu sehen. Da der gesamte Text aufgrund der geringen Bildschirmgröße des Mobiltelefons nicht auf einmal dargestellt werden kann, ist es

notwendig, dass der Besucher die am rechten Rand zu sehende Bildlaufleiste zur Navigation im Text verwendet.

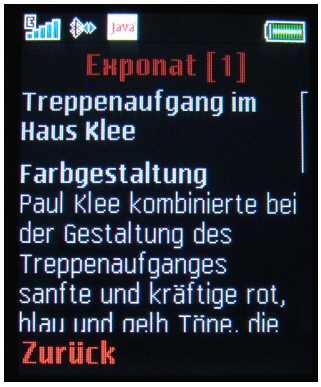

Abbildung 5-11: Beschreibung eines Exponates in Textform.

Alternativ kann der Besucher den in der Beschreibung vorhandenen Text auch in Form eines Audiokommentars anhören. Dazu ist der Menüpunkt `Audio` auszuwählen, wodurch eine Liste mit verfügbaren Audiokommentaren angezeigt wird (vgl. Abbildung 5-12). Durch die Wahl des ersten Listeneintrages wird der entsprechende Audiokommentar selektiert. In dem darauf folgenden Formular kann der Audiokommentar entweder über das Abspiel-Symbol oder über die `Start`-Taste wiedergegeben werden (vgl. Abbildung 5-13). Dieses Vorgehen ist für die Anzeige eines Videos identisch und wird deshalb nicht weiter thematisiert. Sowohl bei Audio- als auch bei Videoinhalten kann dem Besucher ein kurzer erläuternder Textabschnitt zur Verfügung gestellt werden, der den multimedialen Inhalt beschreibt.

Abbildung 5-12: Liste der verfügbaren Audiokommentare zu einem Exponat.

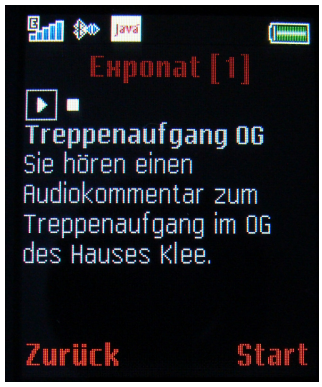

Abbildung 5-13: Abspielsteuerung für Audiokommentare.

Im Gegensatz zu Audiokommentaren und Videos kann der Besucher alle Bilder zu einem gewählten Exponat in einem Formular betrachten. Die Bilder werden dabei automatisch so skaliert, dass diese die maximale Breite des Bildschirmes einnehmen. Zusätzlich kann jedem Bild eine kurze Beschreibung in Textform hinzugefügt werden,

die dem Besucher nähere Informationen zu dem gezeigten Bildausschnitt gibt (vgl. Abbildung 5-14).

Abbildung 5-14: Betrachtung von Bildinhalten.

Zur besseren Orientierung innerhalb der Ausstellung hat der Besucher die Möglichkeit, sich den Standort des ausgewählten Exponates auf einer Karte anzeigen zu lassen. Der Standort ist als roter Kreis auf der Karte dargestellt. In diesem Beispiel ist ein Grundriss des Obergeschosses im Meisterhaus Kandisky/Klee abgebildet, auf dem das entsprechende Exponat *Treppenaufgang* im Haus Klee markiert ist (vgl. Abbildung 5-15).

Abbildung 5-15: Anzeige des Standortes eines Exponates.

Damit wurden die grundlegenden Funktionen des entwickelten Museumsführers am Beispiel des *Treppenaufganges* im Haus Klee vorgestellt. Da der Besucher im Verlauf eines Rundganges durch die Ausstellung mehrere Exponate betrachtet, ist eine Chronik-Funktion in den Museumsführer integriert, die alle bereits betrachteten Exponate in einer Listenansicht aufführt. Die Chronik bietet dem Besucher eine einfache Möglichkeit, Informationen ohne die erneute Eingabe der Exponatnummer nochmalig anzuzeigen. Die Listeneinträge bestehen aus der Exponatnummer, dem Exponattitel sowie der Sprache, in der die Informationen vorliegen. Wählt der Besucher aus der Liste ein Exponat aus, so wird zunächst nach den entsprechenden Daten im

lokalen Speicher des Mobiltelefons gesucht. Nicht mehr im Speicher vorgehaltene Daten werden erneut vom Server nachgeladen.

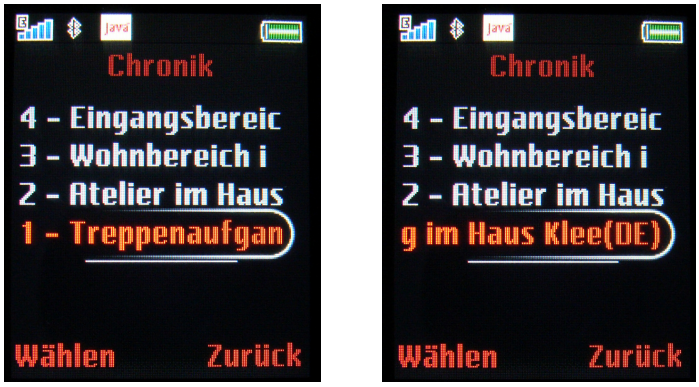

Abbildung 5-16: Chronik der bereits betrachteten Exponate.

In Abbildung 5-16 ist die Listenansicht der Chronik zu sehen. Durch die Auswahl des hervorgehobenen Listeneintrags werden die Daten zum Exponat *Treppenaufgang* geladen und der Besucher gelangt wiederum in das in Abbildung 5-10 dargestellte Menü.

6 Zusammenfassung und Ausblick

Im Rahmen dieser Arbeit wurde die Entwicklung einer Anwendung für Mobiltelefone thematisiert, die innerhalb eines Museums oder einer Ausstellung von den Besuchern als Museumsführer verwendet werden kann. Ziel dabei war es, den Einrichtungen ein System anzubieten, das einfach und kostengünstig umgesetzt werden sowie den Besuchern neben Informationen in Textform auch multimediale Inhalte präsentieren kann.

Im Verlauf der Arbeit wurde erörtert, dass Museen und Ausstellungen bemüht sind, das Lernerlebnis der Besucher durch den Einsatz neuer Medien und die Verwendung elektronischer Hilfsmittel, wie AudioGuide-Systemen, zu unterstützen. Die Einführung von AudioGuide-Systemen scheitert meist an den Investitions- und Folgekosten. Daher wurde ein Ansatz untersucht, der die Anschaffungskosten der Geräte vermeidet, indem die Mobiltelefone der Besucher genutzt werden. Es wurde gezeigt, dass sich Mobiltelefone aufgrund der zahlreichen integrierten Funktionalitäten als Basis einer Museumsführer-Anwendung eignen und den Besuchern bei der Besichtigung der Ausstellungsinhalte zusätzliche Informationen bereitstellen können. Für die Gewährleistung einer möglichst hohen Portabilität der Museumsführer-Anwendung wurde Java als Programmiersprache verwendet. Ferner wurde beschrieben, welche Möglichkeiten und Einschränkungen bei der Entwicklung von Anwendungen für Mobiltelefone unter Java und der J2ME-Plattform bestehen. Ausgehend von den an die Anwendung gestellten Anforderungen und den unter der J2ME-Plattform zur Verfügung stehenden Freiheitsgraden wurde eine Auswahl für die Implementierung des Museumsführer-MIDlets vorgenommen. Die Implementierung als solche wurde anhand der für den Abruf von Informationen zu einen Exponat notwendigen Schritte erläutert. Anhand eines Beispielrundganges in den Meisterhäusern Dessau wurde abschließend die Funktionsweise des Museumsführer-MIDlets demonstriert und aufgezeigt, wie die entwickelte Anwendung in einem realen Umfeld eingesetzt werden kann.

Mit dem beschriebenen Infrastrukturansatz wurde ein System vorgestellt, dass flexibel und universell in Museen und Ausstellungen eingesetzt werden kann. Der Besucher wird bei der Besichtigung der Ausstellungsinhalte durch die Bereitstellung und Anzeige von sowohl Textinformationen als auch multimedialen Inhalten unterstützt. Weiterhin ist das System kostengünstig umzusetzen, da die Anschaffungskosten für die Führungsgeräte eingespart werden und die Infrastruktur mit preisgünstigen PC- und Netzwerk-Standardkomponenten realisiert werden kann. Zusätzlich können die Einrichtungen selbst festlegen, welche Ausstellungsinhalte und in welcher Form diese den Besuchern angeboten werden. Die Besucher hingegen können die von ihnen

gewünschten Inhalte sowie den Detailgrad der Informationen über die Programmeinstellungen selektieren.

Die an das Museumsführer-MIDlet gestellten Anforderungen konnten erfolgreich umgesetzt werden, wobei die nur im Zusammenspiel mit dem Datenserver erfüllbaren Anforderungen vorbereitend implementiert wurden.

Zukünftige, auf der vorliegenden Arbeit aufbauende Untersuchungsansätze könnten unter anderem die Einrichtung des Datenbankservers sowie die Implementierung der für die Zugangspunkte benötigten Anwendung beinhalten. Dem könnte sich ein ausgiebiger Anwendungstest des Museumsführer-MIDlets anschließen, der neben dem Zusammenwirken der Infrastrukturelemente, auch die Ausführbarkeit des MIDlets auf verschiedenen Mobiltelefonen untersuchen sollte, da die Mobiltelefonhersteller bei der Integration der J2ME-Plattform mitunter die vorgestellten Standards nicht wie gefordert umsetzen. Zur Durchführung eines Anwendungstests existieren zwei Möglichkeiten. Entweder das MIDlet wird innerhalb eines Emulators auf einem PC getestet oder der Test findet mit realen Geräten statt. Mobiltelefonhersteller bieten für die Mehrzahl ihrer Geräte Emulatoren an, die in die verwendete Entwicklungsumgebung eingebunden werden können und das Verhalten sowie die Eigenschaften der einzelnen Mobiltelefonmodelle nachstellen. Auf diese Weise ist ein Test zwar möglich, jedoch besitzt dieser keinerlei Aussagekraft. Ein im Emulator erfolgreich bestandener Testlauf der Anwendung garantiert nicht die ordnungsgemäße Ausführung auf einem realen Mobiltelefon. Der Anwendungstest sollte daher auf realen Mobiltelefonen durchgeführt werden. Weiterhin könnte ein Algorithmus für das Museumsführer-MIDlet entwickelt werden, der ein dynamisches Umschalten der Datenübertragung zwischen verschiedenen Zugangspunkten und dem Mobiltelefon erlaubt. Verlässt der Besucher den Sendebereich des für die Kommunikation ausgewählten Zugangspunktes, während eine Übertragung aktiv ist, würde unter den derzeitigen Voraussetzungen die Datenübertragung abbrechen und müsste neu begonnen werden. Mit Hilfe des Algorithmus könnte erreicht werden, dass sich der Besucher frei zwischen den Zugangspunkten bewegen kann und sich die Datenübertragung flexibler gestaltet.

Kurz nach Beendigung der Implementierung des Museumsführer-MIDlets verabschiedete die Bluetooth Special Interest Group die Spezifikation Bluetooth 3.0+HS. Der Standard sieht für die zukünftigen Bluetooth-Geräte eine Nutzung der WLAN-Technologie über Bluetooth vor.[83] Dadurch wird sich das Laden von Informationen aufgrund der erzielten Datenübertragungsraten deutlich beschleunigen und die Sendereichweite der Geräte erhöhen. Ferner hat diese Entwicklung auch

[83] Zivadinovic (2009), S. 52.

Auswirkungen auf die im Lösungsansatz vorgestellte Infrastruktur. Es wird zukünftig nicht mehr notwendig sein, die Zugangspunkte durch PCs mit Bluetooth-Schnittstelle zu realisieren. Stattdessen können handelsübliche WLAN-Router verwendet werden, um eine Verbindung zum Datenserver herzustellen, wodurch sich die beschriebene Infrastruktur vereinfacht und die Kosten für den Aufbau selbiger reduzieren lassen. Bis zur flächendeckenden Verfügbarkeit des neuen Standards in Mobiltelefonen werden jedoch noch einige Jahre vergehen.

Anhang

A Datensatz des Anwendungsbeispiels

Die folgende XML-Antwortnachricht wurde auf die Anfrage des entwickelten Museumsführers vom Server generiert und enthält die Informationen zum Exponat des Treppenaufgangs in der Haushälfte Klee.

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="yes"?>

<content id="1" language="de">
	<text id="132">
		<title>Treppenaufgang im Haus Klee</title>
		<stanza>
			<caption>Farbgestaltung</caption>
			<explanation>Paul Klee kombinierte bei der Gestaltung des
			Treppenaufganges sanfte und kräftige rot, blau und gelb Töne,
			die einen starken Kontrast zueinander bilden. Durch die
			großzügige Glasfassade vermittelt das Treppenhaus je nach
			Tageszeit und Lichteinfall unterschiedliche Stimmungen. Die
			Farbgestaltung vermittelt eindrucksvoll die
			Experimentierfreudigkeit der ersten Bewohner.
			</explanation>
		</stanza>
	</text>

	<multimedia>
		<resource id="1234" type="2" format="jpg" heigth="122" width="233"
		name="Treppenaufgang OG Klee.jpg" hotspot="false">
			<title>Haus Klee</title>
			<stanza>
				<caption>Treppenaufgang OG</caption>
				<explanation>Sie sehen den Treppenaufgang im
				Obergeschoss des Hauses Klee mit den kräftigen rot
				und blau Tönen.
				</explanation>
			</stanza>
			<copyright>
				<author>Hans Mustermann</author>
				<street>Musterweg 12</street>
				<postcode>12345</postcode>
				<city>Musterstadt</city>
				<country>Musterland</country>
				<phone>01234567891200</phone>
				<email>Mustermann@gmx.de</email>
				<homepage>www.Mustermann.de</homepage>
				<info>© 1998 - Dieses Bild ist urheberrechtlich
				geschützt!</info>
			</copyright>
		</resource>
		<resource id="2345" type="2" format="jpg" heigth="222" width="133"
		name="Treppenaufgang OG Klee 2.jpg" hotspot="false">
			<title>Haus Klee</title>
			<stanza>
				<caption>Treppenaufgang OG</caption>
				<explanation>Sie sehen den Treppenaufgang im
				Obergeschoss des Hauses Klee mit den kräftigen
				rot und blau Tönen.</explanation>
			</stanza>
			<copyright></copyright>
		</resource>
		<resource id="3456" type="8" format="3gpp" heigth="222" width="133"
		name="video1.3gp" length="954">
			<title>Video 1</title>
			<stanza>
				<caption>Video 1</caption>
				<explanation>Video 1 zeigt eine Testaufname.
				</explanation>
			</stanza>
			<copyright></copyright>
		</resource>
		<resource id="4567" type="8" format="3gpp" heigth="123" width="133"
```

```
                name="video2.3gp" length="92">
                        <title>Video 2</title>
                        <stanza>
                                <caption>Video 2</caption>
                                <explanation>Video 2 zeigt eine
                                Testaufnahme.</explanation>
                        </stanza>
                        <copyright></copyright>
                </resource>
                <resource id="5678" type="4" format="amr" length="254"
                name="exhibit-01.amr">
                        <title>Audio 1</title>
                        <stanza>
                                <caption>Treppenaufgang OG</caption>
                                <explanation>Sie hören einen Audiokommentar zum
                                Treppenaufgang im OG des Hauses Klee.</explanation>
                        </stanza>
                        <copyright</copyright>
                </resource>
                <resource id="6789" type="4" format="amr" length="200"
                name="exhibit-02.amr">
                        <title>Audio 2</title>
                        <stanza>
                                <caption>Glasfassade</caption>
                                <explanation>Sie hören einen Audiokommentar zur
                                Glasfassade im Treppenaufgang.</explanation>
                        </stanza>
                        <copyright></copyright>
                </resource>
                <resource id="7890" type="2" format="jpg" heigth="12" width="24"
                name="Grundriss Kandisky Klee OG.jpg" hotspot="true">
                        <title>Haus Klee</title>
                        <stanza>
                                <caption>Obergeschoss</caption>
                                <explanation>Sie befinden sich im Treppenaufgang des
                                Hauses Klee im Obergeschoss.</explanation>
                        </stanza>
                        <copyright></copyright>
                </resource>
        </multimedia>
</content>
```

Anhang A: XML-Antwortnachricht auf eine Anfrage des Museumsführers.

Literaturverzeichnis

Abts, D. (2000): Grundkurs Java – Eine Einführung in das objektorientierte Programmieren mit Beispielen und Übungsaufgaben. 2. Aufl., Braunschweig et al.

Apple Inc. (2007): Apple und T-Mobile geben Partnerschaft für den exklusiven Vertrieb des iPhones in Deutschland bekannt.
http://www.apple.com/de/pr/pr-infos2007/september/iphone.html.
03. März 2009.

AudioGuide Ltd. (2009): Geschichte.
http://www.audioguide.ee/index.php?id=48.
11. Februar 2009.

Bauhaus Dessau (2009): Bauhaus 1919-1933.
http://www.bauhaus-dessau.de/index.php?ueberblick.
15. Mai 2009.

Bitkom (2007): Presseinformation - Die SMS wird diese Woche 15 Jahre alt.
http://www.bitkom.org/de/presse/30739_49417.aspx.
20. Februar 2009.

Bluetooth 1 (2009): Architektur - Bluetooth-Basisband.
http://german.bluetooth.com/Bluetooth/Technology/Works/Architecture__Baseband.htm.
26. März 2009.

Bluetooth 2 (2009): Wie Bluetooth funktioniert.
http://german.bluetooth.com/Bluetooth/Technology/Works.
27. März 2009.

Breymann, U.; Mosemann, H. (2006): Java ME – Anwendungsentwicklung für Handys, PDA und Co. München et al.

Bristot, C. (2007): Marketing für Museen als systematischer Managementprozess. In: Mitteilungen und Berichte aus dem Institut für Museumsforschung. Nr. 40 aus dem Jahr 2007, S. 7-124.

Gartner Inc. 1 (2009): Gartner Says Worldwide Mobile Phone Sales Grew 6 Per Cent in 2008, But Sales Decline 5 Per Cent in the Fourth Quarter.
http://www.gartner.com/it/page.jsp?id=904729.
04. März 2009.

Gartner Inc. 2 (2009): Gartner Says Worldwide Smartphone Sales Reached Its Lowest Growth Rate With 3.7 Per Cent Increase in Fourth Quarter of 2008.
http://www.gartner.com/it/page.jsp?id=910112.
11. März 2009.

Iglhaut, S. (2006): Zwischen anklickbarem Exponat und Medieninstallation. In: Mitteilungen und Berichte aus dem Institut für Museumsforschung. Nr. 38 aus dem Jahr 2006, S. 67-81.

Institut für Museumsforschung (2008): Statistische Gesamterhebung an den Museen der Bundesrepublik Deutschland für das Jahr 2007. In: Materialien aus dem Institut für Museumsforschung. Nr. 62 aus dem Jahr 2008.

International Telecommunication Union (2001): ITU-T Recommendation E.161. Arrangement of digits, letters and symbols on telephones and other devices that can be used for gaining access to a telephone network.
http://www.itu.int/rec/T-REC-E.161-200102-I/en.
05. Mai 2009.

Java Community Process (2009): FAQ - General JCP Questions.
http://www.jcp.org/en/introduction/faq.
14. Februar 2009.

JSR-30 (2000): Connected, Limited Device Configuration. Specification Version 1.0a.
http://www.jcp.org/en/jsr/detail?id=30.
28. April 2009.

JSR-37 (2000): Mobile Information Device Profile. Java 2 Platform Micro Edition 1.0a.
http://www.jcp.org/en/jsr/detail?id=37.
28. April 2009.

JSR-75 (2004): PDA Optional Packages for the J2ME™ Platform. FileConnection Optional Package 1.0 Specification.
http://jcp.org/en/jsr/detail?id=75.
23. April 2009.

JSR-118 (2006): Mobile Information Device Profile for Java 2 Micro Edition.
http://www.jcp.org/en/jsr/detail?id=118.
28. April 2009.

JSR-135 (2006): Mobile Media API 1.2.
http://www.jcp.org/en/jsr/detail?id=135.
28. April 2009.

JSR-139 (2007): Connected Limited Device Configuration. Specification Version 1.1.1. http://www.jcp.org/en/jsr/detail?id=139. 28. April 2009.

JSR-280 (2007): XML API for Java™ ME. http://jcp.org/en/jsr/detail?id=280. 13. Mai 2009.

Knudsen, J. (2001): Wireless Java™: Developing with Java™ 2, Micro Edition. Berkeley (Californian).

Kroll, M.; Haustein, S. (2003): J2ME Developer´s Guide – Java-Anwendungen für mobile Geräte. München.

kXML (2009): About kXML. http://kxml.sourceforge.net/about.shtml. 12. Mai 2009.

Meisterhäuser Dessau 1 (2009): Geschichte der Meisterhäuser. http://www.meisterhaeuser.de/de/geschichte.html. 15. Mai 2009.

Meisterhäuser Dessau 2 (2009): Bildarchiv - Historische und Zeitgenössische Aufnahmen der Meisterhäuser. http://www.meisterhaeuser.de/de/bildarchiv.html. 15. Mai 2009.

Motorola (2009): Ein Erbe voller Innovationen: Zeitlicher Überblick der Firmengeschichte von Motorola 1928-2007. http://www.motorola.com/content.jsp?globalObjectId=8714. 15. Februar 2009.

Opitz, R.; Lüders, D. (2006): Smartphones. In: c't Special - Mobil. Nr. 03/2006 im März 2006, S. 60-71.

Richter, V. (2004): Grundlagen der Betriebssysteme. München u. a.

Schiller, J. (2000): Mobilkommunikation - Techniken für das allgegenwärtige Internet. München u. a.

Sun Developer Network (2002): Parsing XML in J2ME. http://developers.sun.com/mobility/midp/articles/parsingxml/. 12. Mai 2009.

Sun Microsystems Inc. (2009): Java ME Technology. http://java.sun.com/javame/technology.

30. März 2009.

Tanenbaum, A. S. (2002): Moderne Betriebssysteme. 2. Aufl., München.

T-Mobile 1 (2009): Funk- und Datennetze.
http://www.t-mobile.de/gprs/0,12732,18100-_,00.html.
15. März 2009.

T-Mobile 2 (2009): Funkversorgung im Inland.
http://www.t-mobile.de/funkversorgung/inland.
15. März 2009.

Topley, K. (2002): J2ME In A Nutshell – A Desktop Quick Reference. Beijing et. al.

Wittenbrink, H.; Köhler, W. (Hrsg.) (2003): XML. Berlin.

Wöhr, H. (2004): Webtechnologien – Konzepte-Programmiermodelle-Archtitekturen. Heidelberg.

Zivadinovic, D. (2009): Nachwachsende Blauzähne - Bluetooth 3.0+HS setzt auf WLAN auf. In: c't. Nr. 11/2009 vom 11. Mai 2009, S. 52.

Abschließende Erklärung

Ich versichere hiermit, daß ich die vorliegende Diplomarbeit selbständig, ohne unzulässige Hilfe Dritter und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe. Die aus fremden Quellen direkt oder indirekt übernommenen Gedanken sind als solche kenntlich gemacht.

Magdeburg, den 22. Juli 2009

Sebastian König