Bachelorarbeit

Dariusch Zohurian

Entwicklung einer J2ME Anwendung für Mobiltelefone zur multimedialen „Live“ Annotation von „Points of Interest“.

Fakultät Technik und Informatik
Studiendepartment Informatik

Faculty of Engineering and Computer Science
Department of Computer Science

Dariusch Zohurian

Entwicklung einer J2ME Anwendung für Mobiltelefone zur multimedialen „Live“ Annotation von „Points of Interest“.

Bachelorarbeit eingereicht im Rahmen der Bachelorprüfung
im Studiengang Technische Informatik
am Studiendepartment Informatik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuende Prüferin: Dipl. Inform. Birgit Wendholt
Zweitgutachterin: Prof. Dr. rer. nat. Bettina Buth

Abgegeben am 16.10.2007

Dariusch Zohurian

Thema der Bachelorarbeit

Entwicklung einer J2ME Anwendung für Mobiltelefone zur multimedialen „Live“ Annotation von „Points of Interest“.

Stichworte

Location-Based Service, GPS, J2ME, Smart-Client Anwendung, Multimedia, Mashup, Points of Interest, Social Network

Kurzbeschreibung

Der Drang, Informationen und Erlebnisse auszutauschen nimmt in unserer heutigen Gesellschaft immer mehr zu. Durch gegenwärtige Technologien kann dieser Austausch auch auf der Basis von mobilen Endgeräten übertragen werden. In dieser Arbeit wird eine mobile J2ME Anwendung realisiert, die vor Ort Multimediadaten, die einen interessanten Standort beschreiben, erzeugen und diese einer Community zur Verfügung stellt kann. Die Informationen erhalten durch das Global Positioning System Verfahren einen direkten Bezug zur realen Welt. Sie können mobil mittels Location-Based Service ortsabhängig abgerufen und kartengestützt angezeigt werden. Um herauszufinden, welche Technologien für die zu entwickelnde Anwendung geeignet sind, werden aktuelle mobile Laufzeitumgebungen, Datenübertragungstechniken und die benötigten Technologien eines ortsbasierten Dienstes untersucht. Anhand der Ergebnisse und Möglichkeiten der genutzten Technologien werden im Anschluss der Realisierung Erweiterungsvorschläge des Systems aufgezeigt.

Title of the paper

Development of a J2ME application for mobile phones that supports a multimedia live annotation of points of interest.

Keywords

Location-Based Service, GPS, J2ME, Smart-Client Software, Mobile Media, Mashup, Points of Interest, Social Network

Abstract

The need to exchange information and experiences increases in our current society. This exchange can be transferred through current technologies on the basis of mobile phones. This thesis realizes a mobile J2ME application that creates locally multimedia data. These data describe interesting locations, and make them available for a community. By the global positioning system the data get a reference to the real world. A location-based service makes it possible to receive the data on the mobile phone. Furthermore, it allows retrieving them on the basis of maps. An investigation of runtime environments, techniques of data transfer, as well as the necessary technologies of location-based services is supposed to show the suitable technologies for the developing application. After the realisation of this application propositions for add-ons will be made via the results and the possibilities of the used technologies.

Danksagung

Ich möchte mich an dieser Stelle bei allen bedanken, die mich über das Studium hinweg begleitet und mir Kraft gegeben haben.

Ein besonderer Dank für die ausführliche Betreuung in der letzten Phase meines Studiums gilt Frau Dipl. Inform. Birgit Wendholt und Frau Prof. Dr. rer. nat. Bettina Buth. Außerdem möchte ich mich bei Dipl. Inform. Birgit Wendholt für die zum Teil langen Freitagabende bedanken. Sie hatten immer ein offenes Ohr und haben mir durch ihre Anregungen, Kritik und fachliche Unterstützung bei der Entstehung dieser Bachelorarbeit sehr geholfen.

Ich möchte mich vom ganzen Herzen bei meiner Freundin Hoa Chi Nguyen bedanken. Du bist immer für mich da und hast mich in jeder Lebenslage unterstützt. Du hast mir die nötige Kraft gegeben, diesen Abschnitt erfolgreich zu beenden.

Ein weiterer Dank geht an meine Freunde, die es verstanden haben, dass das Studium zeitaufwendig ist und nicht immer leicht sein wird. Besonders danke ich Christoph Hennings und Till Kressin, ohne eure Ablenkungen hätte ich so manche Tage nicht überstanden. Till, dir möchte ich außerdem Danken, dass Du mir die tollen Bilder kreiert hast, die den Prototype zu einem echten Produkt aussehen lassen.

Abschließend möchte ich mich bei meinem Kommilitonen Clemens Buchholz für die tolle Studienzeit bedanken. Wir haben zusammen programmiert, gelacht, geschwitzt aber auch sehr gute und interessante Gespräche geführt. Diese werden mir immer in Erinnerung bleiben.

Inhaltsverzeichnis

Abbildungsverzeichnis

1 Einleitung

1.1 Motivation

Wir leben in einer Gesellschaft, in der das Mobiltelefon zu einem stetigen Begleiter des Menschen geworden ist. Mit Hilfe der Verfügbarkeit von Mobilfunknetzen und des steigenden Angebots an Flatrates besteht die Möglichkeit, mit dem Mobiltelefon, dauerhaft online zu sein. Das Bedürfnis, Informationen und Erlebnisse auszutauschen, zieht durch die Web 2.0-Ära[1] immer größere Kreise. Die gegenwärtigen Techniken und Technologien sowie die vorangeschrittene Ausstattung von Mobiltelefonen, mit der Eindrücke vor Ort digital festgehalten werden können, ermöglicht die Realisierung eines ortsabhängigen Austausches von Informationen und Erlebnissen auf Basis von mobilen Endgeräten.

In dieser Arbeit soll der immer größer werdende Drang, Informationen auszutauschen und sich diese zu Nutzen zu machen, mit heutigen Technologien kombiniert werden. Diese Technologien umfassen u. a. den Location-Based Service[2] (LBS), das Positionsbestimmungsverfahren Global Positioning System (GPS) und die Übertragungstechnik Universal Mobile Telecommunications System (UMTS). Es wird ein LBS in Verbindung mit einem mobilen Client entwickelt, der auf den Grundsätzen der *Social Software* (der gemeinschaftlichen Beteiligung an der inhaltlichen Gestaltung eines Angebots) basiert und von einer „passiven“ bzw. „aktiven“ Teilnahme der Community lebt. Um das Vorhaben dieser Arbeit zu verdeutlichen, folgt ein Beispielszenario.

Angenommen man ist in einer fremden Stadt unterwegs, z. B. Coimbra (eine kleine Studentenstadt in Portugal), und ist auf der Suche nach interessanten Orten und Aktivitäten. Leider gibt es in der Nähe kein Internetcafe oder sonstige Möglichkeiten, um auf Informationen zuzugreifen. Wie schön wäre es jetzt, wenn man mit einem Griff auf interessante Orte und Sehenswürdigkeiten zugreifen könnte, die sich in unmittelbarer Nähe befinden?
Alles was man dafür tun müsste ist, eine Software auf dem Mobiltelefon zu installieren und diese bei gegebenem Anlass zu starten. Die Software ermittelt über das Mobiltelefon den Standort des Nutzers und zeigt eine geografische Karte mit den interessanten Orten und Sehenswürdigkeiten als Point of Interest (POI) ortsabhängig auf dem Display an. Nun ist man in der Lage auf Fotos, Videosequenzen oder Beschreibungen der angezeigten Orte zuzugreifen, um auf diese Weise mehr über den Ort zu erfahren. Das Zugreifen auf frei

[1] Eine Definition des Begriffs Web 2.0 ist unter [O’REILLY 2005] nachzulesen.

[2] Ein LBS verbindet Informationen und Dienste mit realen Orten/Positionsdaten.

verfügbare Informationen wird als passive Teilnahme einer Community bezeichnet und stellt den ersten Teil dieser Anwendung dar. Dabei werden die Informationen von Menschen erzeugt, die sich an diesem Ort schon einmal aufgehalten haben. Sie möchten ihre Erlebnisse und Eindrücke mit anderen Menschen teilen und stellen daher per Mobiltelefon ihre Informationen frei zur Verfügung. Das Erstellen dieser Informationen ist die aktive Teilnahme einer Community und stellt den zweiten Teil der Anwendung dar[3].

Befindet sich ein Nutzer an einem interessanten Ort oder hat dieser einen Geheimtipp entdeckt, sollen Fotos, Videosequenzen und Texte, die den Ort beschreiben, erzeugt und für alle Teilnehmer der Community zugänglich gemacht werden. Die erzeugten Daten werden am Ort mit Positionsdaten verknüpft und somit ortsgebunden zu einem zentralen Server gesendet, der als LBS fungiert.

Insbesondere die Aktivität, anderen Menschen an Erlebnissen, Eindrücken sowie Atmosphären durch Verwendung multimedialer Daten teilhaben zu lassen, beschreibt den Wandel in der heutigen Kommunikationswelt und ist eine Bereicherung für jeden, der an der Community teilnimmt.

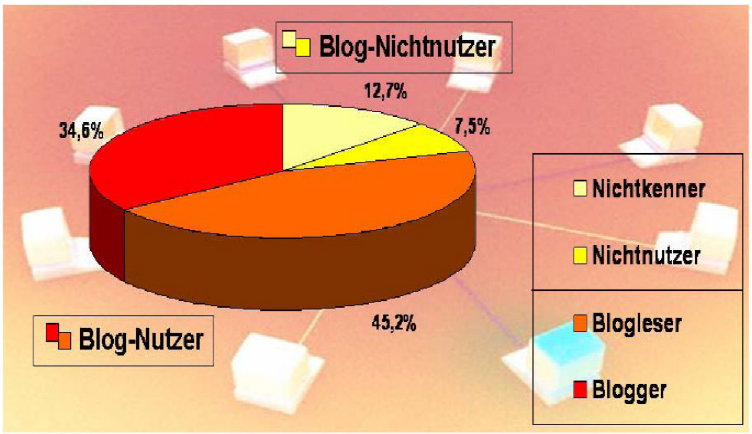


Abb. 1.1: Bekanntheitsgrad und Nutzung von Weblogs [Blog Studie 2007]

Wie stark das Bedürfnis heutzutage in Deutschland verbreitet ist, Informationen auszutauschen und diese zu nutzen, zeigt eine Studie (siehe Abb. 1.1) über Weblog Nutzer. Diese Studie wurde vom Institut für Kommunikations- und Medienwissenschaft der Universität Leipzig (Prof. Dr. Ansgar Zerfaß, Janine Bogosyan) [BLOG STUDIE 2007] mit Unterstützung der Suchmaschine Ask.com als selbstselektive Online-Umfrage durchgeführt. Es ist zu sehen, dass bereits 34,8% der befragten Nutzer Beiträge schreiben

[3] Die Bedeutung der aktiven und passiven Teilnahme einer Community wird im Kapitel Systemanalyse unter der Bezeichnung Produzent und Konsument weiter bestehen bleiben.

und weitere 45,2% auf die Daten zugreifen. Insgesamt wurden dabei 605 Teilnehmer befragt, die mehrmals am Tag im Internet sind.

Social Software bezeichnet vor allem Anwendungen, die Kommunikation, Interaktion und Zusammenarbeit im Internet unterstützen. Vorrangiges Ziel der *Social Software* ist es, die Endnutzer gemeinschaftlich an der inhaltlichen Gestaltung eines Angebots zu beteiligen. Das individuelle Wissen wird damit zu geteiltem Wissen, zu *shared information*. Der Erfolg dieser Anwendungen ist abhängig von der aktiven Teilnahme der Community.
Bekannte Vertreter von *Social Software* Anwendungen sind studiVZ[4], YouTube[5], Wikipedia[6] oder Weblogs. Diese sind klassische Vertreter und beschränken sich auf den reinen Austausch von Informationen.
Die Verbindung von Informationen und Diensten mit realen Orten hingegen werden durch LBS Anwendungen ermöglicht. Sie stellen Informationen immer im Bezug zum Ort dar und erlauben es, in Abhängigkeit vom Aufenthaltsort des Nutzers relevante Informationen zu filtern und zur Verfügung zu stellen.
Für die visuelle Darstellung der realen Orte werden digitale Karten benötigt. Ein Zugriff auf bereits vorhandenes Kartenmaterial im Internet wird durch Schnittstellen, den so genannten Application Programming Interfaces (API) ermöglicht. Mit dem Einzug der Web 2.0-Ära stehen stetig mehr API zur Verfügung, so dass vorhandene Daten mit neuen, eigenen Informationen verknüpft werden können. Anwendungen, die über API auf bereits bestehende Daten im Internet zugreifen und diese mit eigenen Inhalten verknüpfen, werden Mashups genannt. Ein bekannter Vertreter, der u. a. Kartenmaterial über eine API frei zur Verfügung stellt, ist Google-Maps[7].

Durch die Integration eines LBS mit bereits vorhandenen Daten aus dem Internet mittels API, können neue innovative Dienste angeboten werden. Service Provider erkennen diesen Markt. Somit ist es nicht verwunderlich, dass es immer mehr mobile Anwendungen gibt, die genau auf diesen Technologien aufsetzen. Schon heute lassen sich ortsabhängige Dienste (LBS), wie z.B. das Suchen der nächstgelegenen Tankstelle oder Reparaturwerkstätte in Kombination mit einer Kartendarstellung, realisieren.

---

[4] Online im Internet unter http://www.studivz.net/.

[5] Online im Internet unter http://www.youtube.com.

[6] Online im Internet unter http://www.wikipedia.org/.

[7] Online im Internet unter http://www.google.com/apis/maps/documentation/.

Antreiber für neue mobile Anwendungen sind vor allem die Kunden. Bereits heute besitzt laut Bundesverband Informationswirtschaft, Telekommunikation und neue Medien e.V. (BITKOM) jeder Einwohner in Deutschland statistisch gesehen einen Mobilfunkvertrag. Dies wird aus der Abbildung 1.2 deutlich. [BITKOM 2007]

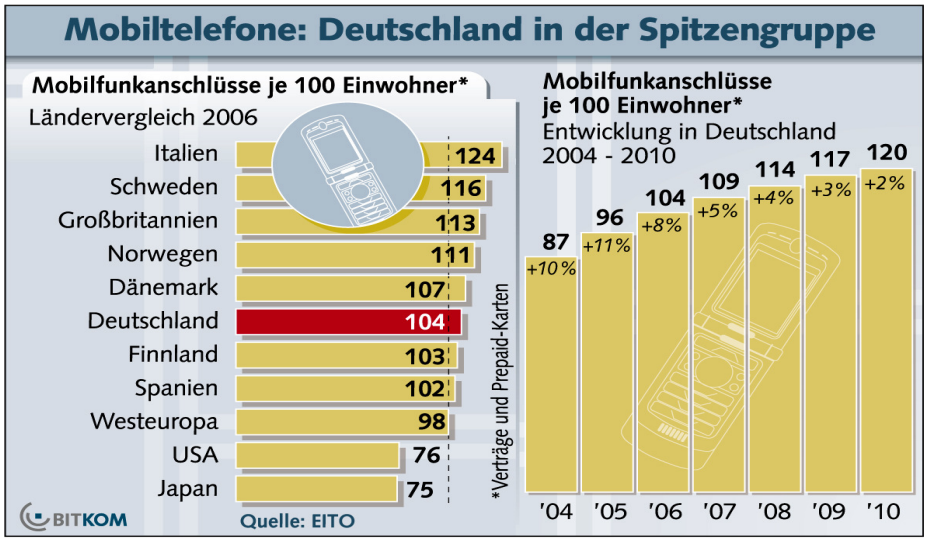


Abb. 1.2: Mobilfunkanschlüsse je 100 Einwohner [BITKOM 2007]

Heutige Mobiltelefone ermöglichen nicht mehr nur das reine Telefonieren, sondern auch das Ausführen von Multimediaanwendungen und selbst entwickelten Programmen. Zudem erlauben sie einen mobilen Zugang zum Internet. Die Internetnutzung wird aufgrund der starken Konkurrenz zwischen den Mobilfunkprovidern vor allem durch die sinkenden Kosten des Datenaustausches sowie des vermehrten Angebots an Flatrates vorangetrieben. Der Nutzer kann somit auch unterwegs *allways on* sein und auf Dienste zugreifen, die einen ständigen Datenaustausch voraussetzen, ohne auf das Volumen oder den Datentransfer achten zu müssen.

Parallel zu den verbesserten Funktionalitäten der Mobiltelefone wurden Datenübertragungstechniken entwickelt, die den heutigen Anforderungen an internetbasierten Anwendungen standhalten. Zu den bekannten Datenübertragungstechniken zählt UMTS. Die Anzahl an Mobiltelefone, die UMTS unterstützen steigt stetig. Ebenso erhöht sich die Zahl der Nutzer von UMTS. Bereits Ende 2007 soll die Anzahl der verkauften Verträge für UMTS Geräte in Deutschland auf 10,5 Millionen wachsen [BITKOM 2007]. Ende 2008 sollen laut der amerikanischen Marktforschungsgesellschaft ABI Research ein Viertel der UMTS fähigen Geräte sogar in der Lage sein, den Standort mit Hilfe von GPS zu bestimmen [ABI-RESEARCH 2006].

Die Nutzung eines LBS ist mit den heutigen Technologien, wie z.B. UMTS sowie GPS, über das Mobiltelefon möglich. Die verbesserten Funktionalitäten der Mobiltelefone erlauben das Ausführen eigener Programme auf dem Telefon sowie das Ausführen von Multimediaanwendungen. API lassen u. a. den Zugriff auf Kartenmaterial zu, so dass reale Orte visualisiert dargestellt werden können. Die steigende Zahl der abgeschlossenen Mobilfunkverträge, der UMTS Nutzer sowie der GPS fähigen Mobiltelefone weisen auf einen größer werdenden Markt für ortsbezogene Dienste hin.

Die Bedingungen für die Entwicklung eines Systems, das den Nutzern erlaubt über das Mobiltelefon eigen erstellte POI einer Community zur Verfügung zu stellen und diese Daten ortsabhängig und kartengestützt abzurufen, sind demnach günstig. Die Realisierung eines solchen Systems würde dem Bedürfnis des Informationsaustausches der Menschen nachkommen.

1.2 Zielsetzung

In dieser Arbeit soll eine mobile Anwendung entwickelt werden, die es Nutzern ermöglicht, spontan Multimediadaten mit dem Mobiltelefon aufzunehmen, zu Verbreiten und diese über einen LBS ortsabhängig abzurufen. Die dabei entstehenden Informationen dienen als Informationsquelle und sollen interessante Orte beschreiben.

Im Gegensatz zu bisherigen mobilen Anwendungen dieser Art, werden die Daten dabei „Live“ vor Ort aufgenommen und, mittels Positionsbestimmungsverfahren direkt der realen Welt zugeordnet. Die Verknüpfung der vor Ort erstellten Daten und der ermittelten Position bilden in diesem System einen vollständigen POI.

Die erstellten POI sollen vom Mobiltelefon aus an einen zentralen Server gesendet werden, damit diese zu jeder Zeit und an jedem Ort verfügbar sind. Der Server ist dann für das Entgegennehmen der Daten verantwortlich und beinhaltet eine Persistenzschicht, damit die Daten zu einem späteren Zeitpunkt abgerufen werden können. Darüber hinaus sollen nur die POI abgerufen werden können, die sich in der Nähe des Nutzers befinden und der gewünschten Kategorie angehören. Um diese Anforderung zu gewährleisten, muss serverseitig ein LBS realisiert werden, der vor dem Abrufen die POI nach der Kategorie filtert und eine Umkreisberechnung durchführt.

Interessierte Nutzer aus der Community können auf die bereitgestellten Daten über das Mobiltelefon ortsgebunden zugreifen. Damit der Nutzer eine Orientierung erhält, soll auf einer digitalen Karte die eigene Position und die der abgerufenen POI mittels Marker angezeigt werden. Für das Bereitstellen der Karten wird ein Zugriff auf einen Map-Server nötig sein, der wie z.B. Google-Map, digitale Karten (Straßenkarten oder Satellitenbilder) in Abhängigkeit zur Position über eine API zur freien Verfügung stellt. Aufgrund des

Zugriffs auf bereits vorhandenes Kartenmaterial über das Internet sowie durch die Verknüpfung dieser Karten mit eignen Inhalten wird ein Mashup resultieren. Mashups verknüpfen Daten miteinander. Die Funktionalität zur Manipulation bzw. Interpretation der Daten stehen in der Regel nicht zur Verfügung. Eine Umkreisberechnung stellt eine Manipulation der Daten dar. Aufgrund dessen muss diese wie oben erwähnt auf dem Server realisiert werden.

Der Nutzer kann über die Karte auf die Daten des jeweiligen POI zugreifen und erhält somit Informationen aus erster Hand. Um bei der mobilen Anwendung die Lazy-Loading[8] Strategie mit einzubeziehen, sollen bei der Erstellung eines POI auch Textinformationen erzeugt werden können. Diese werden beim Selektieren eines POI auf der Karte als erstes angezeigt. Reichen die Informationen nicht aus, ist es möglich auf die Multimediadaten zu zugreifen. Die Lazy-Loading Strategie wird hier eingesetzt, da knappe Ressourcen vorhanden sind (z.B. geringer Speicher) und Performanceprobleme entstehen könnten.

Für das Versenden und Abrufen der Daten muss eine Internetverbindung aufgebaut werden. Eventuelle online/offline Probleme sollten daher berücksichtigt werden. Insbesondere bei Problemen während der Übertragung eines POI an den Server, sollte für ein späteres Versenden das lokale Speichern auf dem Mobiltelefon unterstützt werden.

Ziel dieser Arbeit ist es auch, eine einfache Bedienung und ergonomisch komfortable mobile Software zu realisieren, da es sich hierbei um eine Community-basierte Anwendung handelt. Des Weiteren muss sie plattform- und betriebssystemunabhängig sein, damit eine breite Masse an mobilen Endgeräten unterschiedlicher Hersteller angesprochen wird.

[8] Die Lazy-Loading Strategie besagt, dass abhängige Datensätze erst geladen werden, wenn man sie benötigt.

1.3 Gliederung

Das Kapitel 2 beschäftigt sich mit den Grundlagen. Hier werden Grundbegriffe, Technologien und Besonderheiten von mobilen Endgeräten erläutert. Zum Abschluss folgt ein Fazit aus dem hervorgeht, dass J2ME aufgrund der Plattform- und Betriebssystemunabhängigkeit die hier verwendete Laufzeitumgebung sein wird. Des Weiteren wird sich UMTS, als die vor Ort am besten geeignete Datenübertragungstechnik herausstellen, da diese in Ballungszentren verfügbar ist und eine schnelle Übertragung der Daten gewährleistet.

Das nachfolgende Kapitel 3 befasst sich mit bereits bestehenden Lösungen. Hier wird sich zeigen, dass alle Lösungen Teileigenschaften dieser hier entwickelten Anwendung besitzen. Aufgrund dessen sind sie jedoch in ihrer Funktion eingeschränkt, nicht flexibel genug oder nur auf eine bestimmte Zielgruppe ausgelegt.

In der Systemanalyse, dem Kapitel 4, werden die Anwendungsfälle die vom System umgesetzt werden sollten anhand von Use-Case Diagrammen veranschaulicht. Dabei wird zwischen dem Produzenten (POI Ersteller) und dem Konsumenten (POI Abrufer) unterschieden. Spezifikationen beschreiben die einzelnen Anwendungsfälle detailliert und zeigen die wesentlichen Herausforderungen, die vom System umgesetzt werden sollten, auf.
Resultierend aus den im Kapitel 4 definierten Anforderungen werden in Kapitel 5 die Designentscheidungen getroffen und die Realisierung des Systems erläutert. Innerhalb der Realisierung findet eine Aufteilung der client- und serverseitigen Implementierung statt. Dort werden die umgesetzten Lösungen detailliert beschrieben.

Zum Abschluss findet in Kapitel 6 eine Zusammenfassung des realisierten Systems statt. Außerdem wird auf Probleme, die sich bei der Realisierung ergaben, eingegangen. Anschließend werden mögliche Erweiterungen für eine erfolgreiche Weiterentwicklung, sowohl auf Seiten des Clients als auch des Servers, dargelegt. Ein abschließendes Fazit beendet diese Arbeit.

2 Grundlagen

Da es sich um eine mobile Anwendung handelt, wird im ersten Abschnitt dieses Kapitels auf die Einschränkungen mobiler Endgräte eingegangen, die beim Entwurf der Anwendung zu beachten sind. Es wird sich zeigen, dass den Anwendungen mobiler Endgeräte aufgrund der Mobilität, der Funkverbindung, der Usability und den zur Verfügung stehenden Ressourcen eingeschränkte Mittel und Möglichkeiten zur Verfügung stehen.

In Abschnitt 2.2 werden die beiden populärsten mobilen Plattformen, das *.NET Compact Framework* (.NET CF) und die *Java 2 Micro Edition* (J2ME) erläutert. Diese sind für das Ausführen von Anwendungen auf dem Mobiltelefon verantwortlich. Die Entscheidung für J2ME ist durch die Plattform- und Betriebssystemunabhängigkeit und der daraus folgenden großen Akzeptanz unterschiedlicher Hersteller motiviert.

Eine Untersuchung der aktuell vorhandenen Übertragungstechniken auf dem Mobilfunksektor folgt im Abschnitt 2.3. Aufgrund der Übertragungsgeschwindigkeit und der in den Ballungszentren guten Verfügbarkeit ist die UMTS Technologie die in dieser Arbeit vorzuziehende Übertragungstechnik.

Der abschließende Teil dieses Kapitels, Location-Based Services, erläutert die hier verwendete Basiseigenschaft eines mobilen ortsbasierten Dienstes, die benötigten Technologien, wie z.B. das Global Positioning System sowie die Architektur der für die Nutzung geografischer Karten notwendigen Map-Server.

2.1 Einschränkungen mobiler Endgeräte

Anders als bei herkömmlichen Personal Computern (PC) muss bei mobilen Endgeräten beachtet werden, dass sie ihren Standort wechseln, die Verbindung zum Netz über Funk erfolgt und eingeschränkte Ressourcen zur Verfügung stehen. Auf Grund dessen kommen Probleme und Einschränkungen zum Tragen, die in diesem Abschnitt näher erläutert werden.

Mobilität

Mobile Endgeräte halten sich nie an einem bestimmten Ort auf. Dadurch ist es möglich, dass eine Verbindung zum Funknetz oder zum Internet, die über UMTS hergestellt, durch z.B. das Betreten eines Tunnels oder Gebäudes unterbrochen wird. Aus diesem Grund muss bei der Implementierung von Anwendungen, die auf jegliche Art von Funkverbindungen angewiesen sind, deren Unzuverlässigkeit (online/offline Probleme) mit einbezogen werden. Darüber hinaus ist die Sicherheit ein wichtiger Punkt im Bezug auf die Mobilität.

Benutzerrelevante Daten können durch einen Unfall oder Diebstahl verloren bzw. für illegale Zwecke missbraucht werden.

Funkverbindung

Bei Funkverbindungen handelt es sich nicht um eine klassische Netzwerkverbindung, die über fest verdrahtete Leitungen hergestellt wird. Bedingt durch das Übertragungsmedium Luft kann durch z.B. Reflexionen, keine Zusicherung der Bandbreite erfolgen. Das Datenvolumen im Verhältnis zur verfügbaren Bandbreite muss daher so gewählt werden, dass es bei der Übertragung zu keinen längeren Verzögerungen kommt. Dieser Aspekt spiegelt sich auch in der Usability[9] einer Anwendung wieder, da die Benutzerfreundlichkeit durch aufkommende Störungen an Qualität verliert.

Usability

Die Einschränkungen mobiler Endgeräte werden vor allem durch immer kleiner werdende Displaygrößen, den unterschiedlichen Tastaturfeldern (z.B. Touchscreen) und der beschränkten Möglichkeit an selbst definierten UI-Komponenten hervorgerufen. Komplexe Menüstrukturen sind durch die geringe Bildschirmgröße nicht übersichtlich zu realisieren und sollten sich daher auf das Wesentliche konzentrieren. Die Applikation sollte daher benutzerfreundlich und in ihrer Anwendung intuitiv sein.

Beschränkte Ressourcen

Trotz steigender Funktionen mobiler Endgeräte werden diese immer kleiner. Dadurch werden die Geräte weniger leistungsfähig, weisen eine geringe Prozessorleistung auf und sind auf Akkus angewiesen. Insbesondere Netzwerkverbindungen stellen eine hohe Anforderung an die Akkuleistung und können gegebenenfalls bei der Nutzung von dauerhaften Internetverbindungen von Nachteil werden. Zudem können Daten nicht unbegrenzt gespeichert werden. Der Speicher auf einem Mobiltelefon ist in der Regel durch eine erweiterbare Speicherkarte begrenzt verfügbar.
Außerdem sollten herstellerunabhängige Anwendungen keinen Zugriff auf den Speicher, besonders aber auf benutzerbezogene oder sensible Daten erhalten.

Zusammenfassung

Die Aspekte der Mobilität, Funkverbindung, Usability und beschränkten Ressourcen haben deutlich gemacht, dass bereits im Vorwege der Programmierung einer mobilen Anwendung

[9] Usabilty bezeichnet die Benutzerfreundlichkeit der Anwendung.

auf ihre Verwendbarkeit eingegangen werden muss. Die Darstellung, Bedienbarkeit und Anwendbarkeit müssen an die gegebenen Umstände der Mobilität, Usability, Funkverbindungen und der eingeschränkten Ressourcen von mobilen Endgeräten angepasst werden. Auf diese Weise kann eine stabile und effiziente Anwendung entwickelt werden. Bedingt durch diese Anforderungen ist bereits eine Entscheidung auf Architekturebene des Mobiltelefons zu treffen, auf die im folgenden Kapitel eingegangen wird.

2.2 Mobile Laufzeitumgebungen

Bei der Entwicklung mobiler Anwendungen bieten sich grundsätzlich webbasierte oder speziell für mobile Umgebungen zugeschnittene Laufzeitumgebungen an. Web-Anwendungen, auch *Thin-Clients* genannt, laufen in einem Webbrowser. Sie benötigen eine dauerhafte Verbindung zum Inter-/Intranet und bieten auf mobilen Plattformen durch die Darstellungsmöglichkeiten des Webbrowsers nur eingeschränkte Möglichkeiten zur Gestaltung der Benutzeroberfläche. Des Weiteren ist es zum jetzigen Zeitpunkt nicht möglich, auf mobiltelefonspezifische Funktionen oder Dateisysteme uneingeschränkt zuzugreifen.

Smart-Clients nutzen unmittelbar die Möglichkeiten der Betriebsysteme von Mobiltelefonen und liefern daher den Zugriff auf mobiltelefonspezifische Funktionen. Diese Funktionen können z.B. das Versenden von SMS, das Ansteuern der Kamera oder das Abfragen von GPS Koordinaten sein. Spezielle Laufzeitumgebungen wie z.B. das *.NET Compact Framework* (.NET CF) und die *Java 2 Micro Edition* (J2ME) erlauben die Entwicklung von *Smart-Clients*. Laufzeitumgebungen ermöglichen darüber hinaus das Entwickeln von offline Anwendungen, da sie nicht zwingend eine ständige Verbindung zu einem Webserver voraussetzen. In dieser Arbeit muss auf genau solche mobiltelefonspezifische Funktionen zugegriffen werden. Deshalb wird eine Laufzeitumgebung zur Entwicklung der Smart-Client Anwendungen verwendet. In den folgenden Abschnitten werden zu diesem Zweck die zwei populärsten Laufzeitumgebungen, das .NET CF und die J2ME näher erläutert.

2.2.1 .NET Compact Framework

Das *.NET Compact Framework* (.NET CF) ist eine hardwareunabhängige Umgebung für die Ausführung von Programmen auf Computergeräten mit beschränkten Ressourcen. Unter anderem gehören dazu die *Personal Digital Assistants*[10] (PDAs) wie dem *PocketPC*,

[10] PDA steht für Persönlicher digitaler Assistent. Mit diesem Gerät kann nicht Telefoniert werden.

SmartPhones[11], *Set-Top-Boxen*, Geräte für Autos sowie individuell konzipierte eingebettete Geräte. [MS-NET 2007] Abbildung 2.1 zeigt die Architektur des .NET CF, die im Wesentlichen aus den Schichten *Windows CE*, *Common Language Runtime* und *Framework* besteht. Diese Schichten werden im Folgenden vorgestellt.

Windows CE Schicht

Windows CE gehört zur Familie der Betriebsysteme von *Microsoft* und wurde speziell für die Verwendung in Kleinstcomputern, insbesondere für Industrie-, Automobil- und mobile Geräte entwickelt. Das .NET CF ist eine Betriebssystemkomponente und verwendet *Windows CE* für Kernfunktionen und mehrere gerätespezifische Features. Durch die Interoperabilität mit .NET CF kann auf systemeigene Funktionen zugegriffen und bevorzugte systemeigene Komponenten in eine Anwendung integriert werden. [MS-NET A 2007]

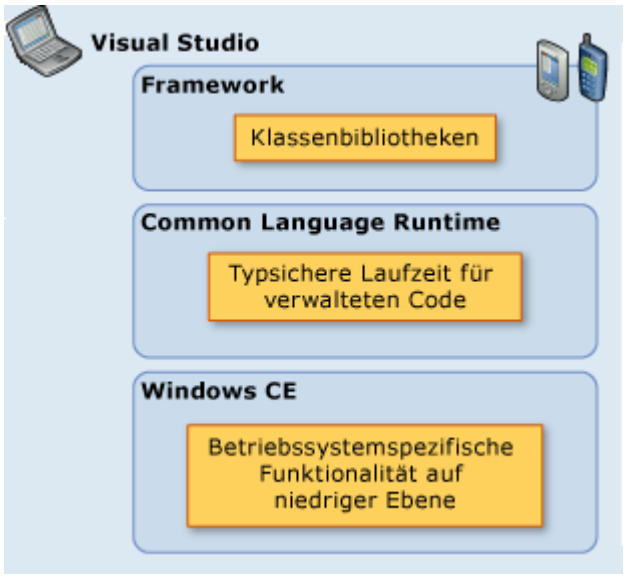


Abb. 2.1: .Net Compact Framework Architektur [MS-NET A 2007]

Common Language Runtime Schicht

Die *Common Language Runtime* (CLR) wurde speziell für die effiziente Ausführung von verwaltetem[12] Code auf ressourcenbeschränkten Geräten konzipiert. Ihre Aufgaben bestehen im Wesentlichen in der Verwaltung der Garbage Collection, Speicherverwaltung und der Just-In-Time (JIT) Kompilierung während der Programmausführung.

---

[11] SmartPhone ist eine Kombination aus Mobiltelefon und PDA.

[12] Spricht man von verwaltetem Code, ist damit eine nichtherstellerspezifische Anwendung gemeint.

Framework Schicht

Das .NET CF bietet Klassenbibliotheken für das Erstellen von mobilen Anwendungen an. Es ist eine Teilmenge des Desktop-.NET Frameworks und wurde speziell für mobile Geräte entwickelt, die über eine reduzierte Bildschirmgröße, niedrigere CPU-Leistung und einen kleinen Speicher verfügen. Im Gegensatz zum großen Framework wurden ab der Version 2.0, die Klassenbibliotheken[13] aufgrund der knappen Ressourcen durch die Hardware vom Umfang eingeschränkt. Das installierte CF 2.0 nimmt auf dem mobilen Gerät 4,5 MByte ein (8 % der Größe des .NET Frameworks), bietet aber 28 % der Eigenschaften, Methoden und Events des Desktop-Frameworks. Des Weiteren können Drittanbieter optionale Komponenten, z. B. Datenbank-, Messaging- und bestimmte Benutzeroberflächenkomponenten erstellen, die die Funktionalität von .NET CF erweitern. [MS-NET M 2007]

Damit Anwendungen auf der Basis von *Microsoft* für *PocketPC* oder *SmartPhones* programmiert werden können, wird die Entwicklungsumgebung *Microsoft Visual Studio* benötigt und Compiler für die Programmiersprachen Visual Basic, C# und C++ unterstützt. Um eine fertig entwickelte Anwendung verteilen zu können, wird eine so genannte CAB-Datei[14] erzeugt. Diese stellt ein Format zur komprimierten Archivierung von Dateien dar. Die Anwendung kann nach der Erzeugung der CAB-Datei auf folgende Weise auf das Endgerät übertragen werden:

- Übertragung mittels eines PC.
- Senden der CAB-Datei als Anlage in einer E-Mail-Nachricht.
- Übermittlung durch das Internet. Hierfür wird die CAB-Datei auf einer HTML Seite hinterlegt. Diese Methode wird auch *Over-the-Air* (OTA) genannt.
- Einrichten einer Speicherkarte, die beim Einschieben das Installationspaket ausführt. [MS-NET S 2007]

Microsoft .NET Anwendungen lassen sich durch das Kompilieren des Codes in die *Microsoft Intermediate Language* (MSIL) auch auf andere Plattformen[15] übertragen, was das so genannte „write once, run everywhere“ beschreibt. Des Weiteren stehen durch die enge Kopplung mit dem Betriebsystem API zur Verfügung, die es einem Entwickler ermöglichen auf hersteller- oder benutzerspezifische Funktionen und Daten zuzugreifen.

---

[13] Eine Liste der unterstützen Klassen ist unter [MS-NET K 2007] nachzulesen.

[14] CAB steht für Cabinet und bedeutet Aktenschrank.

[15] Eine Liste der zur Verfügung stehenden Plattformen ist unter [MS-NET D 2007] nachzulesen.

[MS-NET M 2007] Nach der Übertragung einer Anwendung auf das Endgerät wird beim Starten über die Anwendungsdomäne, die einem Betriebssystemprozess ähnlich ist, eine Instanz der CLR erzeugt. Jede Anwendung wird in einen eigenen Prozess der CLR geladen, wodurch die Anwendung von anderen auf dem gleichen Mobiltelefon ausgeführten Anwendungen isoliert wird. Mit der Isolierung wird sichergestellt, dass der in einer Anwendung ausgeführte Code andere, unabhängig davon ausgeführte Anwendungen nicht beeinträchtigt. Es ist somit möglich, verwaltete und systemeigene Anwendungen gleichzeitig auszuführen. [MS-NET AD 2007]

Zusammenfassung

Das *Microsoft .NET Compact Framework* ist eine hardwareunabhängige Umgebung für die Ausführung von Programmen auf Computergeräten mit beschränkten Ressourcen. Es bietet durch die Interoperabilität mit *Windows* Betriebssystemen die Möglichkeit, auf systemeigene Funktionen zuzugreifen und bevorzugte systemeigene Komponenten in eine Anwendung zu integrieren. Einerseits schafft dies einen gewissen Freiraum für die Erstellung von Anwendungen. Dieser Freiraum erhöht andererseits die Gefahr von Fehlern, sodass dieser gleichzeitig eine Schwachstelle der Anwendung darstellt.

Des Weiteren spricht *Microsoft* von einer Plattformunabhängigkeit bei Anwendungen, die mittels .NET CF erstellt wurden. Dies bezieht sich allerdings nur auf *Microsoft* eigene Betriebssysteme und deckt somit zum jetzigen Zeitpunkt eine eher geringe Anzahl von mobilen Endgräten ab [MS-NET E 2007]. Im kommerziellen Bereich bietet *Microsoft* ein großes Spektrum an Support und stellt mit dem Programm *Microsoft Visual Studio* eine leistungsfähige und übersichtliche Entwicklungsumgebung bereit. Diese ist für Unternehmen jedoch kostenpflichtig, was besonders für Start-up Firmen ein Entscheidungskriterium für die Nutzung von .NET CF darstellt.

2.2.2 Java 2 Micro Edition

Die *Java 2 Micro Edition* (J2ME) von *Sun* ist eine Ansammlung von Technologien und Spezifikationen, um eine Plattform zu erzeugen. Diese Plattform hält den Anforderungen von mobilen Geräten wie z.B. eingebetteten Systemen und hoch entwickelten mobilen Endgeräte stand [SDN-J2ME 2007]. J2ME basiert auf der vor einigen Jahren entwickelten *Java 2 Standard Edition* (J2SE) und wurde speziell für Geräte mit eingeschränkten Ressourcen (siehe Abb. 2.2) entwickelt.

Um eine breite Masse an mobilen Endgeräten zu unterstützen, wurde bei der Konzeption von J2ME darauf geachtet, dass mobile Geräte Eigenschaften haben, wovon nicht alle auf allen Geräten verfügbar sind. Die Java Laufzeitumgebung muss zu den Eigenschaften

passen, so dass Klassen von Geräten definiert wurden, die bestimmten Java Konfigurationen entsprechen.

Eine Anwendung, die für eine Klasse entwickelt wurde, sollte auf jedem Gerät dieser Klasse laufen können. [SDN-J2ME 2007] Um dies zu gewährleisten, basiert die J2ME Architektur zum heutigen Zeitpunkt auf zwei unterschiedliche Konfigurationen, die jeweils eine bestimmte Geräteklasse abdecken.

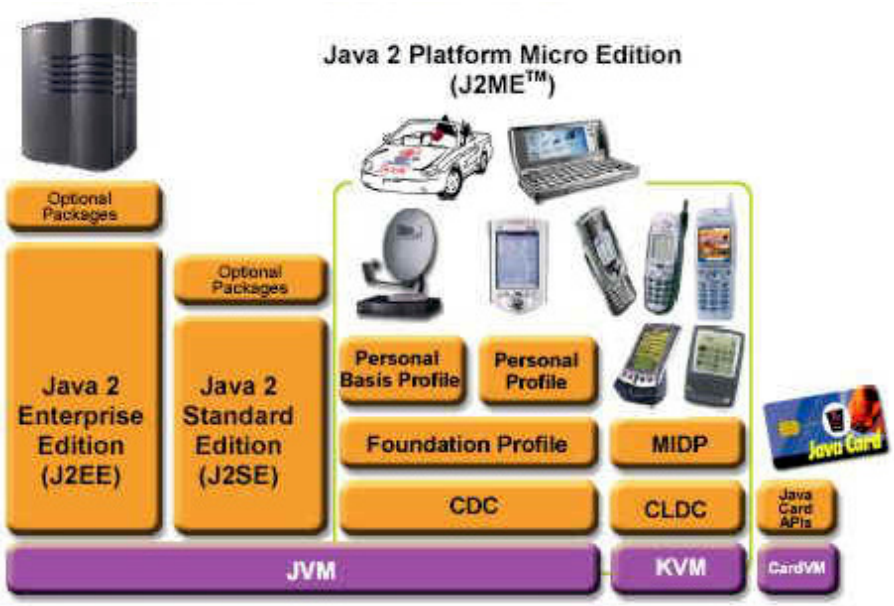


Abb. 2.2: Java Plattform Architektur [J2ME Plattform]

Eine Konfiguration in diesem Sinn ist die Kombination einer *Java Virtual Machine* (JVM) mit einem API, so dass die Kombination eine Entwicklungsplattform für eine breite Klasse an Geräten darstellt. Die Kriterien für diese Differenzierung werden durch die Hardwarevoraussetzungen der einzelnen Geräte geprägt und bestehen im Wesentlichen aus:

- Systemarchitektur
- Leistungsfähigkeit
- Anwendungsbereich

Hinsichtlich der unterschiedlichen Anforderungen im Bereich mobiler Geräte wurde der strukturelle Aufbau der J2ME entsprechend flexibel gehalten.

Die J2ME Plattform besteht im Wesentlichen aus der Virtual Machine (VM), aus Konfigurationen und der Profile Schicht, die je nach Anforderung durch optionale Pakete entsprechend angepasst werden können (siehe Abb. 2.3).

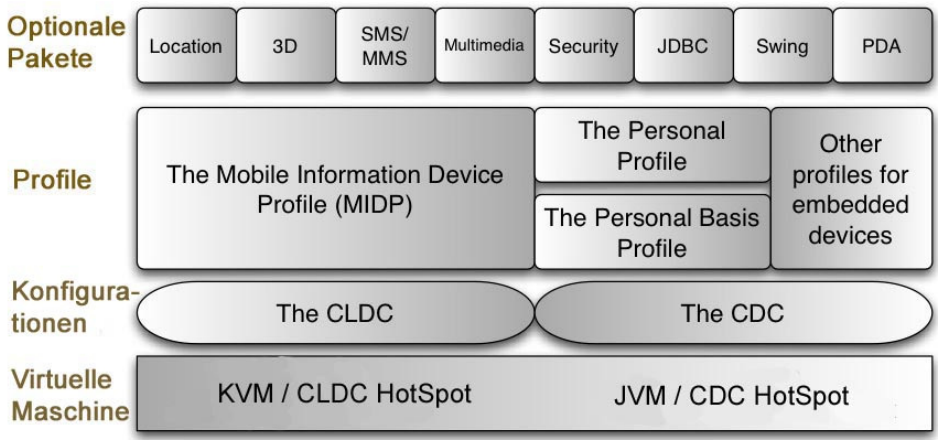


Abb. 2.3: Struktur der Java ME Plattform [J2ME Struktur]

Im Folgenden wird auf die verschiedenen Schichten der Java ME Plattform eingegangen und erläutert, wie die Komponenten zusammenspielen. Da es in dieser Arbeit um die Entwicklung einer Java ME Anwendung für Geräte aus der *Connected Limited Device Configuration* (CLDC) geht, wird auf die Komponenten der *Connected Device Configuration*[16] (CDC), also der CDC HotSpot JVM, CDC und der darüber liegenden Profile, nicht weiter eingegangen.

Java ME Virtual Machine Schicht

Die JVM ist die zentrale Komponente für das Ausführen einer Java Anwendung. Sie ist dafür zuständig, dass alle Java-Funktionen in korrekter Weise auf dem System zur Ausführung gelangen. Sie ermöglicht die Plattformunabhängigkeit von Java. Bei der J2ME ist die VM an das Betriebsystem angepasst und unterstützt spezielle Konfigurationen der darüber liegenden Schicht. Die CLDC, die die Gruppe der PDA und Mobiltelefone abdeckt, baut auf die Kilobyte Virtual Machine (KVM) auf (siehe Abb. 2.3).

Das Hauptziel bei der Entwicklung der KVM war die Erstellung einer kleinstmöglichen vollständigen VM für Geräte mit eingeschränkten Ressourcen, die alle Kernfunktionalitäten von Java beinhaltet, jedoch maximal 40-80 Kilobyte Speicher beansprucht. Um allerdings den heutigen Ansprüchen Genüge zu tragen, wurde die *CLDC HotSpot VM* entwickelt. Sie basiert auf der gleichen Implementierung wie die KVM, erzielt aber, u.a. durch einen optimierten Interpreter, eine schnellere Synchronisation, einen verbesserten JIT Compiler sowie eine höhere Leistungsfähigkeit bei gleichem Akku- und Speicherbedarf. Die *CLDC*

[16] Eine ausführliche Erläuterung der Komponenten und deren Eigenschaften sind unter [SDN-J2ME CDC 2007] nachzulesen.

HotSpot VM unterstützt Geräte, die einen 16/32-bit Prozessor und nur wenige hundert Kilobyte Speicher besitzen. [SDN-J2ME CLDC 2007]

Java ME Konfiguration Schicht

Die Zahl der zur Verfügung stehenden Geräte, wie beispielsweise PDA und Mobiltelefone, steigt mit jedem Tag. Um den gesamten Markt abdecken zu können, ist es sinnvoll gleichartige Geräte zusammenzufassen und jeder Gruppe eine speziell angepasste J2ME Version zur Verfügung zu stellen. Konfigurationen definieren eine horizontale Gruppierung der Produkte, die auf der gleichen Systemarchitektur, Leistungsfähigkeit und dem gleichen Anwendungsbereich basieren [Web Services 2007]. Anhand dieser Basis an Hardwareunterstützung werden in den Konfigurationen die Eigenschaften der JVM und der Java Programmiersprache bestimmt. Die in dieser Arbeit unterstützten Geräte befinden sich in der CLDC Konfiguration.

CLDC Eigenschaften

Die CLDC ist für Geräte gedacht, die mit langsamen Prozessoren, wenig Speicherplatz und einer unzuverlässigen Netzwerkverbindung ausgestattet sind [SDN-J2ME 2007]. Sie hat die Aufgaben, aufgrund der eingeschränkten Ressourcen, eine minimale Java-Plattform zur Verfügung zu stellen und gleichzeitig einen mächtigen Befehlssatz für die möglichst einfache Entwicklung mobiler Anwendungen bereitzustellen. Das Augenmerk der CLDC liegt dabei auf den Gebieten

- der Java Programmiersprache und VM Eigenschaften
- der Kernbibliotheken (*java.lang.**, *java.util.**)
- der Inputs / outputs (*java.io.**)
- der Sicherheit
- der Netzwerkverbindungen
- der Internationalisierung

Die hier aufgelisteten Bereiche bilden die Grundlage für jedes in der CLDC enthaltene Endgerät und beinhalten eine Menge an Spezifikationen. Zum Beispiel geht die CLDC von einer Hardware-Mindestanforderung aus, die wie folgt definiert ist:

- Ein 16 / 32 Bit Prozessor muss vorhanden sein.
- Es müssen für die KVM und die CLDC-Bibliotheken mindestens 192 Kilobyte nicht-flüchtiger Speicher zur Verfügung stehen.
- Mindestens 32 Kilobyte flüchtiger Speicher müssen der JVM zur Verfügung stehen.
- Unterstützt Endgeräte mit mobiler Stromversorgung.

Anhand dieser Vorgaben kann eine minimale VM zur Verfügung gestellt werden, die bei der CLDC die bereits erwähnte *CLDC HotSpot VM* ist. Ein weiterer Punkt der CLDC ist das Bereitstellen von Klassen-Bibliotheken, die, bedingt durch die Speichergröße der J2ME Endgeräte, nicht den Java 2 Standard Edition Klassen-Bibliotheken entsprechen.

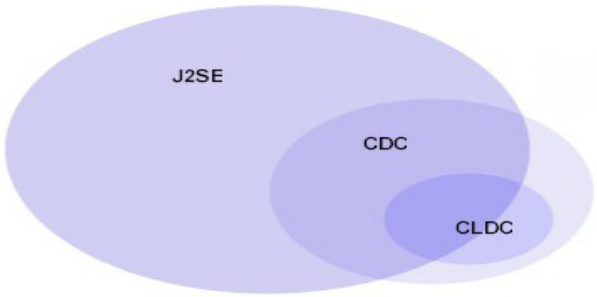


Abb. 2.4: Klassen-Bibliotheken J2SE vs. CLDC [J2ME CLDC]

Wie aus der Abbildung 2.4 ersichtlich ist, bilden die CLDC Klassen-Bibliotheken eine Untermenge der J2SE Bibliotheken inklusive einer Menge aus CLDC spezifischen Klassen. Es soll dadurch gewährleistet werden, dass Java ME Programme aufwärtskompatibel sind, d.h. auf Plattformen wie der Java SE oder Java EE laufen können. In der Kategorie der J2SE Untermenge befinden sich die Packages *java.lang*, *java.util* und *java.io*, während CLDC spezifische Klassen im Package *java.microedition* untergebracht sind.

CLDC Einschränkungen

Dadurch das CLDC der kleinste gemeinsame Nenner ist, der für eine Klasse von Geräten gelten soll, ergeben sich Einschränkungen, die sich auf die API und die Bibliotheken auswirken. Das bedeutet, dass CLDC-Klassen nicht alle Attribute und Methoden implementieren, die von denselben Klassen der Java SE bekannt sind. Einige Einschränkungen gegenüber der Java SE sind z.B.:

- Der Klasse *Object* fehlen die Methoden *clone()* und *finalize()*.
- Der vorhandene *Classloader* kann nicht durch einen eigenen ersetzt werden.
- Es steht nur eine eingeschränkte Fehlerbehandlung zur Verfügung.
- Asynchrone *Exceptions* werden nicht unterstützt.
- Die Abfrage von Typinformationen zur Laufzeit wird nicht unterstützt (*Reflection*).

CLDC gibt es aktuell in zwei Versionen (Version 1.0 und 1.1), wobei die Version CLDC 1.1 die hier verwendete Spezifikation ist. Angesichts der technischen Entwicklung wurde der Funktionsumfang beim Versionswechsel erhöht. Dies erfolgte z.B. durch die

Unterstützung von Floating Point Arithmetik, wodurch hauptsächlich die Erweiterung des zur Verfügung stehenden Speichers von 160 Kilobyte auf 192 Kilobyte resultierte.

CLDC Sicherheit

Der Schwerpunkt der CLDC liegt in der Anwendungsprogrammierung, wobei Systemfunktionen über CLDC-Bibliotheken angesprochen werden können. Aus Sicherheitsgründen, durch das so genannte Sandbox[17]-Model, dürfen Applikationen jedoch nur auf die Bibliotheken der CLDC, der Profile und der optionalen Pakete zugreifen. Dadurch wird verhindert, dass nichtherstellerspezifische Anwendungen z.B. Telefonfunktionen nutzen können. Wäre dies möglich, könnte im Hintergrund des Programms eine 0190 Nummer gewählt werden, so dass der Nutzer ungeahnte Kosten bezahlen müsste. Allgemein kann man sagen, dass das Sandbox-Model Benutzer und Anwendungen vor ungewollten Zugriffen schützt.

Einer der wesentlichen Ziele von CLDC ist das dynamische Laden von Programmen zu ermöglichen. Dadurch, dass Software nicht mehr fest codiert wird, werden Geräte erweiterbar. Diese können dann Anwendungen sowie interaktive Inhalte auch von Drittanbietern laden. [SDN-J2ME CLDC 2007]

Profile Schicht

Die bisher genannten Schichten waren u.a. hinsichtlich der Fähigkeiten von Benutzerschnittstellen entscheidend für das Ausführen von Programmcodes und für die Unterteilung von Endgeräten in bestimmte Gruppen. Die Profile Schicht setzt auf der Konfigurationsschicht (CLDC Schicht) auf und ist im Gegensatz zu den Konfigurationen für den vertikalen Markt bestimmt. D.h., dass Endgeräte durch Profile in bestimmte Arten, wie z.B. Mobiltelefone, Kühlschränke oder Spielzeuge, unterteilt werden und sie gewissen Mindestanforderungen der J2ME unterliegen. Sie stellen eine Art Baukastensystem dar, um Anwendungen für einen speziellen Gerätetyp durch eine Menge von API und den dazugehörigen Bibliotheken zu entwickeln. Je nachdem für welchen Gerätetyp die Anwendung entwickelt werden soll, müssen die Merkmale des Profils umgesetzt werden. Speziell in dieser Arbeit wird das *Mobile Information Device Profile* (MIDP) Version 2.0 genutzt, das für die Fähigkeiten der Mobiltelefone ausgelegt ist. Die Funktionen des MIDP decken folgende Gebiete ab:

- Lieferung von Anwendungen (OTA-Methode)
- Steuerung des Lebenszyklus einer Anwendung

[17] Das Sandbox Model steht für die application level security von J2ME. Weitere Informationen unter [SDN-J2ME S 2002].

- Signierung und Sicherheitsmodell
- Unterstützung von HTTPS (HTTP verschlüsselt)
- Registrierung von Server-Push-Diensten[18]
- Netzwerkverbindungen und Protokolle
- Persistenter Speicher (RecordStore)
- Erzeugung von Tönen
- Timer
- Benutzungsinterfaces (Bildschirm, Tastatur)

Jeder dieser Punkte deckt einen bestimmten Bereich ab. So definiert z.B. das Benutzungsinterface, dass die Bildschirmgröße mit mindestens 96x54 Pixel ausgelegt und eine Einhand/Zweihand Tastatur oder ein Touchscreens vorhanden sein muss.
In der Abbildung 2.5 ist zu sehen, dass auf dem *Mobile Information Device* (MID) verschiedene Arten von Applikationen ablaufen können.

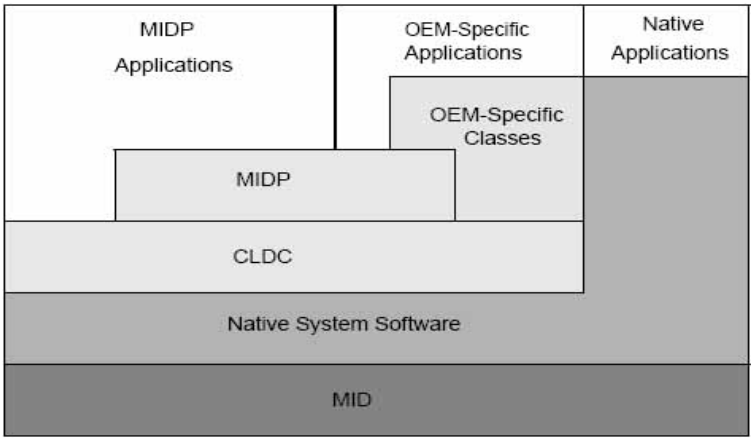


Abb. 2.5: Abhängigkeit von CLDC und MIDP [SDN-J2ME MIDP 2007]

Native[19] Applikationen werden nicht in Java programmiert. Sie stehen also nicht im Zusammenhang mit der CLDC, sondern unterliegen dem Betriebsystem. Außerdem gibt es herstellerspezifische Anwendungen (OEM-Specific Applications). Diese sind nicht abhängig von Klassen der MIDP Spezifikation und können auch nicht auf anderen Mobile Information Devices übertragen werden. MIDP Anwendungen (MIDP Applications) heißen MIDlets und können z.B. mit der frei verfügbaren Entwicklungsumgebung *eclipse*[20] und zwei Erweiterungen [Eclipse-Erweiterungen] erstellt und getestet werden. Wie in der

[18] Der Push-Dienst wird im Kapitel Location-Based Services näher erklärt.

[19] Unter native Applikationen sind systemeigene Anwendungen zu verstehen.

[20] Online im Internet unter http://www.eclipse.org/.

Abbildung 2.5 zu sehen ist, können MIDlets nur auf Klassen-Bibliotheken der MIDP und CLDC zugreifen. Jedes mobile Endgerät, auf dem MIDlets ablaufen sollen, stellt eine *Application Management Software* (AMS) zur Verfügung. Dadurch können MIDlets ausgewählt, installiert, gelöscht und gestartet werden. [JavaME 2006]

MIDP Sicherheit

Wie bereits erwähnt, unterliegt ein MIDlet dem Sandbox-Model der VM, das dafür zuständig ist, dass laufende Applikationen keinen Schaden verursachen können. Es gibt allerdings Szenarien, in denen es sinnvoll ist, dass Anwendungen mehr Freiraum erhalten. Hierbei kommt die Signierung eines MIDlets ins Spiel. Dabei wird in vertrauenswürdigen (trusted) und nicht vertrauenswürdigen (untrusted) Codes bzw. Zugriffsberechtigungen unterschieden. Ein nicht vertrauenswürdiger Code kann z.B. nicht beliebig Verbindungen nach außen aufbauen. Kommt es dennoch zu einem Zugriff, benötigt die Anwendung eine Erlaubnis des Anwenders. Bei der Erstellung kann der Entwickler den Code digital und kostenpflichtig[21] signieren lassen, um ihn als vertrauenswürdig einzustufen. Das Endgerät verifiziert diese Unterschrift dann. Mit MIDP 2.0 werden Zugriffsrechte für unterschiedliche Arten von Netzzugriffen eingeführt. Entsprechend können von optionalen Paketen, die mit sensiblen API arbeiten, auch neue, unterschiedliche Zugriffsrechte definiert werden[22].

Optionale Pakete

Optionale Pakete setzten auf den Profilen auf und erweitern diese um spezielle APIs und Bibliotheken. Um auf Funktionen der optionalen Pakete zuzugreifen, müssen diese auf dem jeweiligen Endgerät installiert sein. Eine nachträgliche Installation ist möglich, erhöht allerdings die Größe der zu installierenden MIDlets.

Die in dieser Arbeit zum Einsatz kommenden optionalen Pakete für die CLDC sind:

- Location API
- Mobile Media API
- File Connection API

Prinzipiell verfügen Geräte je nach Ausrichtung und Funktionsumfang über unterschiedliche optionale Pakete. Durch das Fortschreiten neuer Technologien, wie

---

[21] Aktuell (03.09.2007) kostet ein Authenticated Content Signing for Symbian mit einer Laufzeit von einem Jahr bei der Firma VeriSign ca. 350 US$, Online unter http://www.verisign.com/index.html.

[22] Eine detaillierte Beschreibung des MIDP Sicherheitskonzeptes ist unter [SDN-J2ME MIDP 2007] einzusehen.

beispielsweise dem GPS, werden sie sich allerdings immer mehr angleichen. Im Weiteren wird kurz auf die hier verwendeten optionalen Pakete eingegangen[23].

Das *Location API* ist im [JSR 179] spezifiziert. Mit ihr kann die Position des Mobiltelefons ermittelt werden. Die *Location API* beinhaltet 11 Klassen und 2 Interfaces, die die Entwicklung von LBS unter Java unterstützen. Die Lokalisierung kann anhand der GPS, GSM, UMTS, WLAN oder Bluetooth Technologien ausgeführt werden. Bei der Verwendung von GPS, wird automatisch bei der Lokalisierung ein interner oder externer GPS-Empfänger (über Bluetooth) gesucht. Darüber hinaus lassen sich über die Lokalisierungsschnittstelle auch Kriterien für die Genauigkeit und Häufigkeit der Positionsangaben bzw. Positionsbestimmung spezifizieren. Dadurch kann das Ergebnis bei Auswahl mehrerer Lokalisierungsquellen verbessert werden.

Das *Mobile Media API* (MMAPI) ist im [JSR 135] spezifiziert und definiert Medien- und Benutzungsschnittstellen zur Entwicklung von Spielen und Multimediadaten. Es unterstützt das Aufnehmen und Abspielen von Videos, Bildern und Tönen mittels einer im Mobiltelefon eingebauten Kamera. Die genannte API wird in dieser Anwendung verwendet, um den erzeugten POI multimediale Inhalte hinzuzufügen bzw. diese wieder abzuspielen.

Das *File Connection API* ist im [JSR 75] spezifiziert. Mit Hilfe von bereitgestellten Klassen ermöglicht es auf das im Mobiltelefon interne Dateisystem zuzugreifen. Zur Benutzung werden die üblichen Dateioperationen wie Lesen oder Schreiben einer Datei, Anlegen eines Verzeichnisses etc. angeboten. Außerdem ist es möglich, das Einstecken oder Entfernen einer Speicherkarte festzustellen, welches besonders beim Mobiltelefon von großem Vorteil ist. Ist allerdings das MIDlet nicht signiert, muss der Anwender mit einer Bestätigung die Erlaubnis der Anwendung erteilen.

Das zu Testzwecken zur Verfügung stehenden Mobiltelefon *Nokia N93*[24] besitzt bereits in der Grundausstattung alle benötigten optionalen Pakete.

2.2.3 Bewertung der Laufzeitumgebungen

Der Aufbau von .NET CF besitzt ähnliche Eigenschaften wie der der J2ME. Beide stellen eine für mobile Endgeräte optimierte Laufzeitumgebung zur Verfügung, die eine effiziente Ausführung vom verwalteten Code gewährleistet. Zudem berücksichtigen sie Sicherheitsaspekte und koordinieren den Speicherzugriff von Anwendungen. Darüber

---

[23] Eine vollständige Liste der optionalen Pakete ist unter http://java.sun.com/javame/reference/apis.jsp einzusehen.

[24] Online im Internet unter http://www.forum.nokia.com/devices/N93.

hinaus bieten sie Möglichkeit auf einfache Weise mobile Applikationen zu erstellen, zu übertragen und, bei Bedarf, zusätzliche Schnittstellen zur Verfügung zu stellen. Für eine Bewertung von Laufzeitumgebungen müssen also deren Unterschiede betrachtet werden.

Möchte man effizient und auf der Basis von *Microsoft* .NET CF Anwendungen entwickeln, muss mit einer kostenpflichtigen Entwicklungsumgebung, der Visual Studio IDE, gerechnet werden. Des Weiteren sind .NET Anwendungen nicht plattformunabhängig, sondern können nur, mit Ausnahme des Open Source Projektes *Mono*[25] [Mono Novell 2007], auf dem herstellerspezifischem Betriebsystem Windows CE ausgeführt werden. Durch die enge Kopplung mit dem Betriebsystem bekommen .NET CF Anwendungen zwar einen größeren Spielraum um auf Systemfunktionen zuzugreifen. Sie sind dadurch jedoch fehleranfälliger.

Im Gegensatz dazu können J2ME Anwendungen mit der frei verfügbaren *eclipse* Entwicklungsumgebung inklusive einiger Plugins [Eclipse-Erweiterungen] uneingeschränkt programmiert und mittels *Sun Wireless Tool Kit* [SDN-J2ME WTK 2007] getestet[26] werden. Außerdem ist J2ME plattformunabhängig und setzt keine großen Hardwareanforderungen an die Endgeräte. Um Fehlfunktionen oder Missbrauch vorzubeugen, wird durch das Sandbox-Model jeglicher Zugriff auf Bibliotheken oder nativen Funktionen außerhalb der Applikationsumgebung unterbunden. Der wesentliche Vorteil gegenüber dem .NET CF besteht darin, dass durch die Architektur von J2ME und der Entwicklung heutiger Mobilfunkgeräte eine breitere Masse an Mobiltelefonen unterstützt wird. Aus diesen Gründen wird für die mobile Anwendung dieser Arbeit die J2ME Laufzeitumgebung verwendet.

2.3 Datenübertragungstechnik

In dieser Arbeit geht es unter anderem darum, Multimediadaten vor Ort an einen Server mittels Mobiltelefon zu senden und diese wieder abzurufen. Um die Akzeptanz der Nutzer und eine effiziente Übertragung zu gewährleisten, müssen heutige Datenübertragungstechniken den Anforderungen an eine große Bandbreite, eine flächendeckende Verfügbarkeit und der geringen Kosten standhalten. In den folgenden Abschnitten werden die derzeitig verfügbaren Übertragungstechniken GPRS, UMTS und WLAN vorgestellt und anhand der genannten Kriterien bewertet.

---

[25] Mono bietet bedingte Möglichkeiten der Plattformunabhängigkeit von .NET Anwendungen an.

[26] Das Sun Wireless Tool Kit bietet einen Emulator zum Testen von J2ME Anwendungen an.

2.3.1 GPRS, die Erweiterung im GSM-Netz

Global System for Mobile Communications (GSM) ist der erste Standard, der so genannten zweiten Generation („2G“) in der Übertragungstechnik und Nachfolger der analogen Geräte. Hierbei wird nach dem Verbindungsaufbau ein exklusiver Kanal mit einer konstanten Bandbreite zu Verfügung gestellt, der hauptsächlich für die Sprachübermittlung konzipiert wurde [ELKO GSM 2007].

Mit dem Aufkommen von Datendiensten im Mobilfunk ergaben sich allerdings neue Anforderungen an die GSM-Netze. Die aus Sicht der Anwender gravierendste Unzulänglichkeit des GSM-Netzes ist die geringe Datenrate von maximal 14,4 KBit/s, die für die meisten Dienste unzureichend ist. Wünschenswert ist für einen Anwender ebenfalls, dass er sich für die Datendienste den lästigen Einwahlvorgang sparen oder sogar „always on“ sein kann. Auch an den Netzbetreiber stellen Datendienste neue Anforderungen. Datenverkehr tritt zeitlich betrachtet meist stoßartig und nicht kontinuierlich auf. Eine leitungsvermittelte Verbindung, wie sie standardmäßig im GSM-Netz vorkommt, wird bei einem stoßartigen Datenverkehr[27] schlecht ausgenutzt, da zwischenzeitlich während der Verbindungsdauer keine Daten übertragen werden und die Leitung brach liegt.

GPRS steht für *General Packet Radio Service* und ist eine Erweiterung des GSM-Mobilfunk-Standards um paketorientierte Datenübertragung. Es erfüllt die drei oben angesprochenen Anforderungen für Datendienste: eine höhere Datenrate, eine bessere Ausnutzung der Netzressourcen und kein ständiges Wiedereinwählen. Im GPRS werden Funkkanäle zur Erhöhung der Bandbreite gebündelt und nur für die Dauer einer Datenübertragung, nicht aber für die gesamte Dauer der Verbindung reserviert. In der Übertragungspause des einen GPRS Endgerätes können diese Funkkanäle anderen GPRS Endgeräten zugewiesen werden. Reicht die Anzahl der Funkkanäle nicht aus, um allen Endgeräten in einer Funkzelle die maximal gewünschte Datenrate zur Verfügung zu stellen, wird für alle die Datenrate reduziert [UMTS 2004]. Im Unterschied zur leitungsvermittelten Verbindung kann bei der paketorientierten Datenübertragung per Datenvolumen abgerechnet werden. Dies ist für den Einsatz heutiger Datendienste wie z.B. das Downloaden von Musikvideos, von entscheidender Rolle. Die maximale reale Datenübertragungsrate im Downloadbereich beträgt hier 50 KBit/s und entspricht einem analogen Modem, das sich gut für das Empfangen und Versenden von *Multimedia Messaging Service* (MMS) Nachrichten [3gpp MMS 2007] und E-Mails eignet. GPRS ist der meist genutzte Service auf der Welt und nahezu in jedem GSM-Netz verfügbar [GSM 2007]. Möchte man allerdings Video-Streaming, wie z.B. das TV Angebot von Vodafone

[27] Ein Web Browsing ist z.B. eine stoßartige/burtstartige Anwendung.

Live! [Vodafone 2007], oder das Internet nutzen, stößt man auch hier auf Grenzen. Da die Übertragungsgeschwindigkeit stark von der Qualität der Funkverbindung und der Anzahl der aktiven Nutzer abhängt, bleibt, bei einer hohen Anzahl von Nutzern weniger Bandbreite und damit weniger Übertragungsgeschwindigkeit für den einzelnen Nutzer übrig. Dies erschwert den Zugriff auf Multimediadienste.

2.3.2 UMTS

Universal Mobile Telecommunications System (UMTS) gilt als Mobilfunkstandard der dritten Generation („3G“) und basiert auf einer paketorientierten Vermittlung und dem Internet-Protokoll. Im Gegensatz zum GSM wurde es entwickelt, um auf Dienste, wie z.B. Video-Streaming, das Internet und das Versenden von Bildern und Texten per E-Mail, vom Mobiltelefon aus zugreifen zu können. Ermöglicht wird dies durch die funkzugriffstechnologie CDMA. CDMA steht für *Code Division Multiple Access* und erzielt seine Geschwindigkeit von bis zu 384 KBit/s im Downloadbereich, also die sechsfache ISDN-Geschwindigkeit[28], durch folgende Eigenschaften:
Anstatt die Ressourcen per Frequenz- oder Zeitmultiplex in Kanäle aufzuteilen, wie es bei den klassischen Systemen GSM und GPRS der Fall ist, wird das Signal per Code-Spreizung über das gesamte Frequenzspektrum verteilt.
Jedes übertragene Bit wird mit einer höherartigen Bitfolge multipliziert. Innerhalb des Frequenzspektrums sind dann gleichzeitig verschiedene Verbindungen überlagert.
Durch CDMA ist die Kapazität in einer UMTS Funkzelle flexibel nutzbar. Dies ermöglicht, dass mehrere Teilnehmer gleichzeitig über nur einen Frequenzkanal kommunizieren können, ohne sich gegenseitig zu stören bzw. zu beeinflussen [WiMAX 2007].
Der Vorteil gegenüber GPRS liegt im Wesentlichen in der Übertragungsgeschwindigkeit, die wiederum erst den Zugriff der oben erwähnten Datendienste ermöglicht.

Seit dem Start des ersten UMTS Netzes im Jahre 2002 in Japan ist UMTS das schnellst wachsende Mobilfunknetz in der Geschichte. Laut einer Studie des BITKOM e.V. zählte es im Dezember 2005 eine Teilnehmeranzahl von mehr als 47,3 Millionen Nutzern weltweit (siehe Abb. 2.6) [BITKOM UMTS 2007]. Durch die zunehmende Anzahl von Teilnehmern und Datendiensten ist die Anforderung an eine Verbesserung der Bandbreite, der Netzwerkleistung und der Interaktivität für Datenapplikationen offensichtlich und wurde durch die Implementierung von *High Speed Packet Access* (HSPA) weiter vorangetrieben. HSPA beinhaltet Erweiterungen bei der Download und Upload Übertragung des UMTS

[28] ISDN steht für Integrated Services Digital Network.

Standards. Es unterliegt keinerlei Änderungen an das UMTS Core Netzwerk. *High Speed Download Packet Access* (HSDPA), welches für die Kommunikation von der Basisstation zum Mobiltelefon verantwortlich ist, ermöglicht eine Downloadgeschwindigkeit von momentan maximal 14,4 Mbit/s je Nutzer.

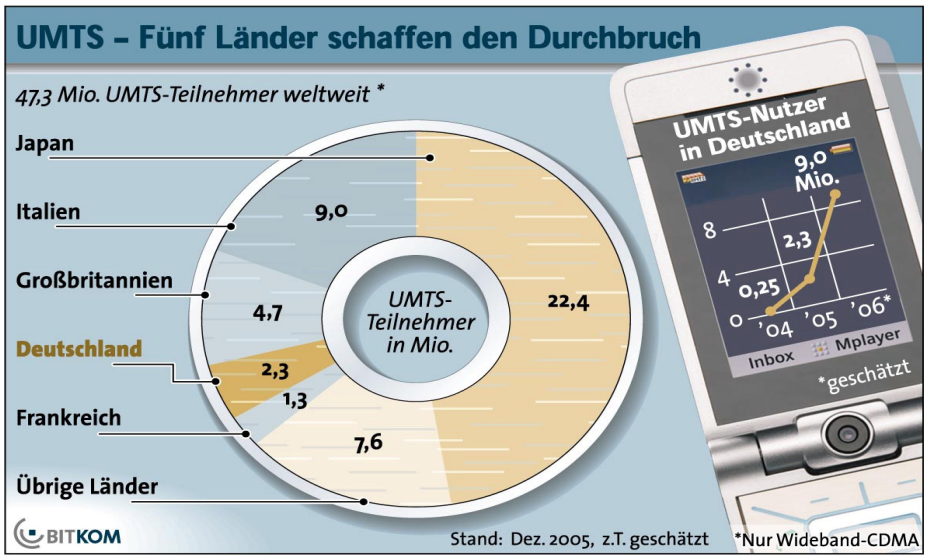


Abb. 2.6: UMTS- Teilnehmer in Millionen [BITKOM UMTS 2007]

Der Gegenpart, nämlich der Upload, wird um das *High Speed Upload Packet Access* (HSUPA) erweitert. Es beinhaltet im Wesentlichen, dass die Reichweite und der Durchsatz sowie die Verringerung der Upload Laufzeit verbessert wurden. Die theoretische maximale Uploadgeschwindigkeit beträgt bei HSUPA 5,5 MBit/s, was auf der Applikationsschicht zu einer Geschwindigkeit von 4 MBit/s führt.

UMTS wird durch den enormen Zuwachs an Übertragungsgeschwindigkeit mittels HSPA den mobilen Markt durch ein hohes Aufkommen von neuen Applikationen fördern. Dienste wie z.B. VoIP, Multiplayer Gaming, Streaming Live TV und Video- Konferenztelefonie werden schon heute angeboten und geben den Betreibern von mobilen Anwendungen jeglichen Spielraum. [WiMAX 2007]

Wie aus der Abbildung 2.6 zu entnehmen ist, ist der Zuwachs an UMTS Nutzern in Deutschland, bezogen auf den privaten Gebrauch, auf Grund der hohen Preise und der Fülle an Tarifoptionen zum jetzigen Zeitpunkt eher verhalten. Hinzukommt, dass UMTS momentan nur in Ballungszentren Deutschlands flächendeckend verfügbar ist, so dass darin eine weitere Hemmschwelle besteht, sich für einen UMTS Vertrag zu entscheiden.

2.3.3 WLAN

Mit der Veröffentlichung des *Wireless Local Area Network* (WLAN) durch das *Institute of Electrical and Electronics Engineers* (IEEE) wurde ein Standard geschaffen, der es ermöglicht, drahtlos Daten auszutauschen und sich an das drahtgebundene IP-Netzwerk anzukoppeln. Im Gegensatz zu den vorherigen Übertragungstechniken hebt sich die Übertragungsgeschwindigkeit enorm ab. Sie ist zum jetzigen Zeitpunkt im IEEE-802.11g Standard mit 54MBit/s verfügbar. In erster Instanz wurde das WLAN eingesetzt, um die Produktivität am Arbeitsplatz zu steigern und die Kosten zu senken. Es mussten keine Leitungen verlegt werden. Mitarbeiter konnten sich mit Ihrem Laptop im Gebäude bzw. Firmengelände frei bewegen. Die Übertragungstechnologie beruht auf dem OSI-Schichtenmodell[29], wobei auf der Bitübertragungsschicht (Schicht 1) im Protokollstack eine Funkübertragung realisiert wurde [BA Buchholz 2007]. Im Unterschied zu GPRS und UMTS ist WLAN keine flächendeckende Lösung, sondern nur auf ein Infrastruktur-Netzwerk beschränkt. Es gibt allerdings WLAN Projekte, die innerhalb einer Stadt flächendeckend installiert sind. Abhängig vom Internetprovider, stellen sie darüber den Internetzugang und die hohe Geschwindigkeit der Öffentlichkeit zur Verfügung. Ein weiterer Aspekt ist, dass die eigentliche Funkverbindung durch Frequenzbänder, die weltweit für einen gebühren- und genehmigungsfreien Betrieb freigegeben sind, realisiert wurde. Es fallen folglich nur die Kosten durch den Internetprovider an, nicht aber durch die Nutzung des Funknetzwerkes. Daher stellt ein Zugang über das WLAN meist eine kostengünstigere Verbindung als das GPRS oder UMTS dar. Durch die beschränkte Verfügbarkeit ist das WLAN jedoch keine wirkliche Konkurrenz.

2.3.4 Bewertung der vorgestellten Übertragungstechniken

Das GSM bzw. GPRS Netz ist flächendeckend verfügbar. Es ist das derzeit meistgenutzte Netz auf der Welt und steht mit über 2 Billionen Nutzern in keiner Konkurrenz zu anderen Netzen [GSM 2007]. In dieser Arbeit geht es darum, Multimediadaten, deren Volumen zum Teil bei heutigen Mobiltelefonen sehr groß werden können, vom Handy zu einer zentralen Einheit zu übermitteln und diese auch wieder abzurufen. Dies ist aufgrund der geringen Datenübertragungsgeschwindigkeit im GPRS Netz nur begrenzt möglich und daher nicht zu empfehlen. Die bessere Wahl für multimediale Anwendungen ist das UMTS Netz. Es wurde für diese Anforderungen entwickelt und stellt die optimale Datenübertragungsgeschwindigkeit im Down- bzw. Upload zur Verfügung. Der derzeitige Nachteil besteht in der fehlenden flächendeckenden Verfügbarkeit des Netzes und der

[29] Das OSI-Schichtenmodell steht für Open System Interconnection Reference Model.

momentan hohen anfallenden Kosten im Down- bzw. Upload. Auf Grund ihrer hohen Datenübertragungsgeschwindigkeit ist das WLAN eine sehr gute Alternative. Die Übertragungsrate gewährleistet eine reibungslose Übermittlung der Daten. Zudem ist sie die kostengünstigere Variante. Doch auch hier mangelt es an einer flächendeckenden Verfügbarkeit, wodurch die Nutzbarkeit des WLAN begrenzt ist.
In dieser Arbeit sollte deshalb eine Kombination aus dem UMTS- und WLAN- Netz zum Einsatz kommen. Das steigende Angebot an Flatrates auf dem Mobilfunkmarkt vergünstigt die UMTS Nutzung im Bereich der Telefonie und der damit zusammenhängenden Nutzung von LBS. Kann vor Ort nicht auf das UMTS Netz zugegriffen werden, sollte beim Versenden eines neu erstellten POI dieser gespeichert und z.B. auf dem Zuhause verfügbaren WLAN Router ausgewichen werden. Möchte ein Nutzer vor Ort auf bereits vorhanden POI über das Mobiltelefon zugreifen, kann bei der Verwendung der reinen Textinformationen auf das GPRS Netz ausgewichen werden. Der Zugriff auf die Multimediadaten über das GPRS Netz ist wegen der geringen Bandbreite zu vermeiden.

2.4 Location-Based Services

In dieser Arbeit wird ein Location-Based Service (LBS) entwickelt, der standortbezogene Daten in Abhängigkeit von der aktuellen Position des Nutzers zur Verfügung stellt. Des Weiteren soll eine geografische Karte, u. a. als Orientierung für den Anwender, auf dem Mobiltelefon angezeigt werden. Um diese Anforderungen zu gewährleisten, müssen Technologien wie die Lokalisierung und das Bereitstellen von geografischen Karten mit einbezogen werden. Im Folgenden wird darauf eingegangen, was ein LBS ist und welche Möglichkeiten es bei der Ortung von Mobiltelefonen gibt. Zudem wird in einem extra Abschnitt die Funktionsweise eines Map-Servers dargestellt, der geografische Karten zur Verfügung stellt.

In [LBS Schiller 2004] wird der LBS definiert als „services that integrate a mobile device's location or position with other information so as to provide added value to a user". Die Rede ist von Diensten, die den aktuellen Standpunkt des Nutzers für die Übermittlung von Informationen berücksichtigen. Klassische Beispiele für LBS Dienste sind der Informationsdienst („Wo finde ich das nächste Fitnessstudio?") und der Navigationsdienst („Wie komme ich von meinem Hotel zur Innenstadt?"). Der Unterschied bei standortabhängigen Diensten liegt im Verfahren der Positionsbestimmung und der Übermittlung von Informationen an den jeweiligen Nutzer. Im Bereich der Positionsbestimmung wird zwischen Indoor- und Outdoor- Lokalisierungssystemen

unterschieden. Da es in dieser Arbeit um eine Outdoor Anwendung geht, wird auf das Indoor System nicht weiter eingegangen.

Das meist verbreitete Outdoor Lokalisierungsverfahren ist das Global Positioning System (GPS). Es wurde ursprünglich in den USA im Jahre 1970 für militärische Zwecke entwickelt, um Personen und Objekte über einen GPS-Empfänger zu Orten. Das Verfahren basiert auf 24 Satelliten, die in der Erdumlaufbahn installiert sind. Damit der GPS-Empfänger die aktuelle Position berechnen kann, wurden die Satelliten so angeordnet, dass mindestens drei Satelliten Signale zur Erde senden (siehe Abb. 2.7).

Abb. 2.7: GPS Satellitenpositionierung [GPS Satelliten]

Dazu vergleicht der GPS-Empfänger die Sendezeit mit der Empfangszeit des Signals. Aus dieser Zeitdifferenz kann die Entfernung des Satelliten berechnet werden. Werden nun zwei weiteren Satelliten Messungen hinzugefügt, kann die aktuelle Position durch Trilateration (Entfernungsmessung von drei Punkten aus) bestimmt werden. Bei gutem Empfang wird eine Position bestimmt, die bis auf 15m genau ist. GPS ist weltweit seit 1980 frei und kostenlos verfügbar, was das Angebot an kostengünstige ortsgebundene Dienste erst ermöglicht. [LBS Schiller 2004]

Ähnlich wie bei GPS gibt es ein europäisches Projekt Namens Galileo, das Ende dieses Jahrzehnt seine Betriebsbereitschaft weltweit aktivieren wird. Es baut auf den Eigenschaften und Positionsberechnungen von GPS auf. Durch die Kombination von Signalen der GPS und Galileo Satelliten soll es sogar eine bessere Genauigkeit erreichen. Im Gegensatz zu GPS wird Galileo speziell für die Bedürfnisse der zivilen Bevölkerung entwickelt [ESNI 2007].

Ein anderes Lokalisierungsverfahren ist das zellenbasierte Verfahren. Dies wird z. B. für die Positionsbestimmung bei einem Notruf über das Mobiltelefon eingesetzt. Im Gegensatz zu GPS werden hierfür keine Satelliten, sondern Zellen der Netzbetreiber verwendet. Die Zellen entstehen durch so genannte stationäre Antennen der Netzbetreiber, die jeweils eine bestimmte Umgebung abdecken. Das Mobiltelefon ist bei Netzempfang automatisch einer Zelle zugeordnet und würde beim Wechsel in eine neue Umgebung auch die Zelle wechseln. Der Zellenwechsel bleibt für den Nutzer unbemerkt. Die Genauigkeit ist abhängig von der Dichte der Zellen und ist je nach Größe der Stadt unterschiedlich. Dies ist der große Nachteil des zellenbasierten Lokalisierungsverfahrens. Die Position ist auf die Zelle beschränkt und kann im schlimmsten Fall einige Kilometer von der eigentlichen Position abweichen. Der Vorteil liegt darin, dass kein Empfänger angeschafft werden muss und es an jedem Ort einsetzbar ist. Dies ist auch dann der Fall, wenn man sich innerhalb eines Gebäudes aufhält, was beim satellitenabhängigen System wegen evtl. Empfangsstörungen nicht möglich ist.

Man unterscheidet bei den LBS zwischen zwei Diensten, die durch die unterschiedliche Art des Zugriffes bestimmt werden. Kommt der Informationsaustausch aufgrund dessen zustande, dass der Benutzer aktiv eingreift und den Zeitpunkt der Dienstnutzung selber bestimmt, nennt man diesen Dienst den Pull-Dienst. Möchte ein Benutzer sich z.B. über eintreffende E-Mails oder über die Börse informieren, erfolgt dies über den so genannten Push-Dienst. Dabei werden Informationen ohne Einwirken des Benutzers zu einem beliebigen Zeitpunkt an das Mobiltelefon übertragen [Mobile Computing 2005]. Da die hier entwickelte Anwendung durch das aktive Einschreiten des Benutzers auf standortbezogene Daten und geografische Karten zugreift, wird in dieser Arbeit der Pull-Dienst verwendet. Damit auf dem Mobiltelefon zur Orientierung geografische Karten (z.B. aus Deutschland) angezeigt werden können, benötigt diese Anwendung einen Zugriff auf ein Geoinformationssystem.

Geoinformationssystem

Ein Geoinformationssystem (GIS) ist ein „rechnergestütztes Informationssystem, das aus Hardware, Software, Daten und Anwendungen besteht. Mit ihm können raumbezogene Daten digital erfasst und redigiert[30], gespeichert und reorganisiert, modelliert und analysiert sowie alphanumerisch und grafisch präsentiert werden.“ [GIS Bill 1994]. Es ist ein System, das sich je nach Anforderung in unterschiedliche GIS-Kategorien aufteilen läst und bestimmte Dienste erfüllt (siehe Abb. 2.8).

Kategorie	Dienstegruppen			
	Erfassung und Verwaltung	Präsentation	Abfrage	Analyse
Geodaten-Server	x	-	-	-
Map-Server	x	x	-	-
Auskunftssystem	x	x	x	-
Online-GIS	x	x	x	x
Funktions-Server	-	(x)	x	x

Abb. 2.8: GIS-Kategorien [Fitzke 1999]

Da in dieser Arbeit der Einsatz von Map-Servern für das Anzeigen von geografischen Karten auf dem Mobiltelefon zum Tragen kommt, wird auf die anderen Kategorien nicht weiter eingegangen.

Ein Map-Server ist ein Dienst, der Karten zur Online-Visualisierung übermittelt [Fitzke 1999]. Es wird zwischen zwei Arten unterschieden. Zum einen gibt es den statischen Map-Server. Dieser liefert vorgefertigte Karten, die typischerweise aus einem GIS exportiert wurden, und stellt diese den Anwendern zur Verfügung. Die andere Variante ist der interaktive Map-Server. Hier können Kartendarstellungen, beispielsweise die Farbgebung, durch den Benutzer beeinflusst werden. Die Karten werden erst nach der Bearbeitung auf dem Server erstellt und anschließend zur Visualisierung freigegeben. Bei Map-Servern kommuniziert der Client nicht direkt mit dem Server, sondern indirekt über einen Web Server[31]. Der Client stellt eine HTTP Anfrage an den Proxy-Server, der diese an den Map-Server weiterleitet (siehe Abb. 2.9). Dieser generiert die entsprechende Antwort, meist in

[30] Redigiert bedeutet in diesem Zusammenhang, dass die raumbezogenen Daten korrigiert, verbessert oder aktualisiert werden können.

[31] Der Web Server wird in diesem Zusammenhang auch als Proxy-Server bezeichnet. Im weiteren Verlauf der Arbeit wird daher der Namen Proxy-Server verwendet.

Form einer Karte, die aus den vorhandenen Geodaten erzeugt wird. Anschließend schickt der Map-Server die Karte über den Proxy-Server zurück an den Client. Die Aufgabe des Proxy-Servers ist u.a., die Übertragung von HTML Dateien erzeugten Graphiken und eingebauten Programmiercodes im Internetbrowser des Clients zu gewährleisten.

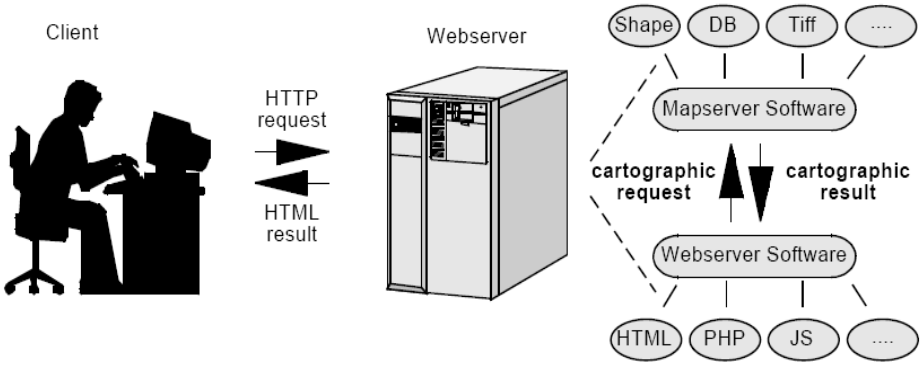


Abb. 2.9: Funktionsweise eines Map-Servers [Dipl. Fürpass 2001]

Der Map-Server hingegen generiert seine Karten lokal aus Geodaten (z.B. ESRI-ShapeFiles[32]), Rasterbildern (z.B. TIFF[33]) oder aus einer Datenbank und fügt diese in eine HTML Seite ein. Zudem bieten heutige Map-Server eine Reihe an Zusatzfunktionen an, wie z.B. räumliches Navigieren, Zoomlevel bestimmen oder Panfunktionen zum Verschieben des Ausschnittes. Dadurch wird das Arbeiten mit den generierten Karten auf einer Internetseite vereinfacht. [Dipl. Fürpass 2001] Beispiel eines leistungsstarken Map-Servers ist Google-Maps. Dieser bietet über eine API an, auf digitale Karten zuzugreifen, um diese auf der eigenen Internetseite Nutzen zu können[34]. In dieser Anwendung wird auf eine externe Lösung zurückgegriffen, die über eine API einen GIS-Client auf dem Mobiltelefon zur Verfügung stellt. Dieser GIS-Client kann auf Map-Server unterschiedlicher Hersteller zugreifen (u.a. auch Google-Maps), um digitale Karten auf dem Display des Mobiltelefons anzuzeigen. Die Möglichkeit, auf Geodaten frei zugreifen zu können, ist ein wesentlicher Bestandteil dieser Arbeit.

---

[32] Nähere Informationen unter [ESRI] nachzulesen.

[33] Nähere Informationen unter http://www.awaresystems.be/imaging/tiff/faq.html nachzulesen.

[34] Das Verwenden von Internetdiensten über bereitgestellte API und somit das Erschaffen neuer Anwendungen wird auch als Mashup bezeichnet.

2.5 Fazit

In diesem Kapitel wurde anfangs gezeigt, dass eine stabile und effiziente mobile Anwendung nur entstehen kann, wenn die Probleme und Eigenschaften der Mobilität, Funkverbindung, Usability und der beschränkten Ressourcen beachtet und in der Designphase mit einbezogen werden. Um ein Verständnis zu bekommen, auf welcher Grundlage heutige Anwendungen auf dem Mobiltelefon entwickelt und ausgeführt werden können, wurden die beiden populärsten Plattformen auf dem mobilen Sektor vorgestellt. J2ME wird aufgrund der Sicherheit, der Vielfältigkeit durch optionale Pakete, des Kostenfaktors und der Plattform- und Betriebssystemunabhängigkeit die in dieser Arbeit verwendete Laufzeitumgebung sein. Im Bereich der Datenübertragung wurde erläutert, dass UMTS trotz geringerer Verfügbarkeit und höheren Kosten gegenüber GPRS aus Nutzersicht den Anforderungen an eine optimale Übermittlung von Multimediadaten am besten gewachsen ist. Zum Ende des Kapitels wurden das GPS- und das zellenbasierte Verfahren zur Lokalisierung mobiler Endgeräte erläutert. Daraus ging hervor, dass GPS eine bessere Genauigkeit erzielt und die derzeitige Standardlösung auf dem Sektor ist. GPS wird in dieser Arbeit in Verbindung mit einem Pull-Dienst, der durch das aktive Eingreifen des Benutzers definiert wird, eingesetzt. Zum Abschluss wurde kurz auf die Funktionsweise eines Map-Servers eingegangen, um zu zeigen, wie Kartenmaterial zusammen- und zur Verfügung gestellt wird.

3 Vergleichbare Arbeiten

In diesem Kapitel werden zwei mobile und eine webbasierte Lösung vorgestellt und anhand von Beispielen gezeigt, welche Vor- und Nachteile sie besitzen. Die vorgestellten Lösungen werden bzgl. der Bedeutung eines POI, der vorhandenen Community und des Konzepts untersucht, um Unterschiede zu der hier entwickelten Smart-Client-Anwendung aufzuzeigen. Die Ergebnisse sollen dazu beitragen, den Nutzen dieser Arbeit zu erkennen, der die Realisierung eines Community gestützten LBS befürwortet.

3.1 Qiro

Qiro[35] ist ein von der Deutschen Telekom entwickelter mobiler Informationsdienst und wurde 2006 auf der IFA[36] zum ersten Mal online geschaltet. Die Idee ist, den eigenen Aufenthaltsort und den der Freunde zu ermitteln und über das Mobiltelefon auf einer digitalen Karte anzuzeigen. Wurden Freunde gefunden, können diese aus der Anwendung heraus kontaktiert und zu spontanen Treffen eingeladen werden (siehe Abb. 3.1 Mitte). Durch die zusätzliche Möglichkeit auf Information von interessanten Orten wie beispielsweise Restaurants, oder Kinos, die sich in der Umgebung befinden, zuzugreifen, kann gleichzeitig die Unternehmung geplant werden (siehe Abb. 3.1 links).

Abb. 3.1: Kategorien, Friend-Finder, Kartendarstellung [Qiro 2007]

Qiro nutzt für die Positionsbestimmung das zellenbasierte Verfahren und kann vom Nutzer per SMS aktiviert oder deaktiviert werden. Um digitale Karten anzeigen zu können, nutzt die *Qiro* Plattform als Map-Server u.a. das Google-Map API. Die mögliche aktuelle Position wird durch einen roten Kreis auf der Karte dargestellt (siehe Abb. 3.1 rechts).
Jeder Nutzer, der sich bei *Qiro* angemeldet hat, ist ein Teilnehmer der Community. Da es sich allerdings bei der Ermittlung des Aufenthaltsortes um die Privatsphäre handelt, stellt

[35] Online im Internet unter https://www.myqiro.de.

[36] Online im Internet untern http://www.ifa-berlin.de.

die Community bei der Suche von Freunden nicht alle angemeldeten *Qiro* Nutzer dar, sondern wird über die so genannten *Buddy*-Liste abgebildet. Ein Nutzer kann somit nur mit jemanden über *Qiro* in Kontakt treten, der die Erlaubnis dazu erteilt hat. Um einen neuen *Buddy* hinzuzufügen, muss die Mobilfunknummer bekannt sein. Mit dieser Nummer kann über das Onlineportal oder aus der Client-Anwendung heraus eine SMS versendet werden, die vom Empfänger bestätigt werden muss.

Befinden sich „Freunde“ in der *Buddy*-Liste, kann deren Position ermittelt und über die Anwendung kontaktiert werden. Es ist allerdings zu jeder Zeit möglich die eigene Sichtbarkeit für alle „Freunde“ aus der *Buddy*-Liste zu deaktivieren oder zu aktivieren. Der Schutz der Privatsphäre ist somit immer möglich.

Qiro bietet außerdem die Möglichkeit, personalisierte POI zu erstellen. Möchte man einen neuen POI hinzufügen, muss die gewünschte Position mit einem Kreuz auf der digitalen Karte markiert werden (siehe Abb. 3.2 links). Außerdem ist es möglich, dem POI mit einem Namen und einer Beschreibung zu versehen (siehe Abb. 3.2 Mitte) und für die ganze Community oder nur der *Buddy*-Liste zur Verfügung zu stellen (siehe Abb. 3.2 rechts). POI werden genutzt, um einen interessanten Ort oder einen neuen Treffpunkt bei einer Verabredung zu markieren. Damit jeder auf diese Informationen zugreifen kann, werde diese auf einem zentralen Server gespeichert.

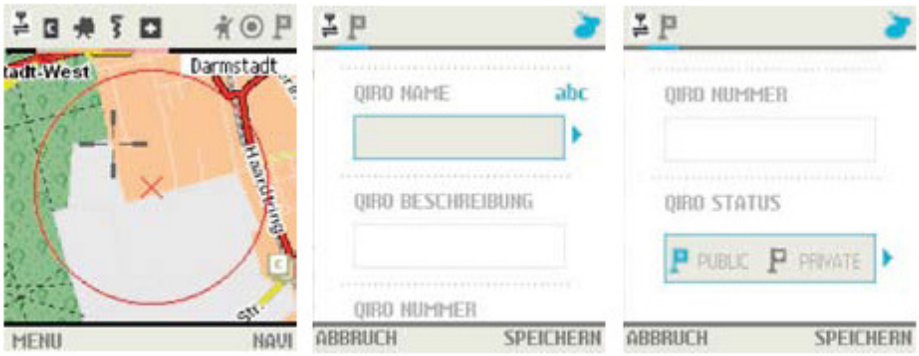


Abb. 3.2: Personalisierten Qiro anlegen [Qiro 2007]

Um bereits bestehende POI auf der Karte darzustellen, können maximal 4 Kategorien selektiert werden (siehe Abb. 3.1 links). Die *Qiro* Software bietet zwei Hauptfunktionen an. Zum einen das Navigieren der Karte und zum anderen das auswählen und anzeigen der POI. Handelt es sich bei einem ausgewählten POI z.B. um ein Restaurant oder einem Freund, kann über die Anwendung ein Anruf für eine Reservierung oder ein Treffen erfolgen.

Die Dienste, die von *Qiro* angeboten werden sind kostenlos, es fallen lediglich die Übertragungskosten durch die Nutzung von GPRS oder UMTS an. Auch für Geschäftskunden ist *Qiro* interessant. So können Restaurants ihre Tageskarten hinterlegen und Geschäftsketten ihre Kunden mit Logos auf der Oberfläche auf ihre Filialen aufmerksam machen. Ebenso können sich Nutzer von *Qiro* von den von ihnen ausgewählten Anbietern spezielle Rabatt-Aktionen melden lassen.

Qiro ist vom Aufbau ähnlich wie die hier entwickelte Smart-Client Software. Informationen über interessante Orte werden ortsabhängig einer Community zur Verfügung gestellt und können mittels Mobiltelefon kartengestützt abgerufen werden.
Es handelt sich bei diesen Informationen allerdings eher um Daten, die bereits auf dem Server vorhanden sind und von Firmen zu Werbezecken in Auftrag gegeben wurden. Werden vom Nutzer eigne personalisierte POI erstellt, stehen diese in keinem Zusammenhang mit dem Standort. Sie werden nicht wie in dieser Anwendung „Live“ vor Ort erstellt und über ein Positionsbestimmungsverfahren der realen Welt zugeordnet. Es ist somit Möglich, einen Punkt auf der Karte zu Markieren und mit Informationen zu füllen, die überhaupt nichts mit dem Ort zu tun haben. Außerdem kann dem POI nur ein Name und eine Beschreibung hinzugefügt werden, der visuelle Aspekt durch die Aufnahme von Multimediadaten wird hier nicht verfolgt.
Darüber hinaus möchte *Qiro* auf die Frage „Wo befindest Du dich gerade?“ eine Antwort geben und verfolgt das Ziel, Freunde aufzuspüren und sich darüber zu verabreden. Die eigentliche Community beschränkt sich somit nur auf die persönliche *Buddy*-Liste und nicht auf die gesamten Nutzer von *Qiro*. Des Weiteren ist die Koordinate der eigenen Position sehr ungenau. Bei der Ermittlung wird nicht wie in der hier konzipierten Anwendung das GPS Verfahren, sondern das zellenbasierte Verfahren verwendet. Je nach Abdeckung kann es zu einer Ungenauigkeit von 200 Metern kommen.

3.2 Google MyMaps

Hinter der seit April dieses Jahres neuen Funktion von Google[37] mit dem Namen *MyMaps* verbirgt sich die Idee, personalisierte, mit multimedialen Elementen versehende Karten zu erstellen und diese, anderen Google-Maps Nutzern zur Verfügung zu stellen.
Bei der hier vorgestellten Lösung handelt es sich nicht um eine mobile Anwendung, sondern um eine Web Applikation, die für Personal Computer ausgelegt und mittels Internetbrowser gestartet werden kann.

---

[37] Online im Internet unter http://www.google.de.

Um eine personalisierte Karte anlegen zu können, muss der Nutzer bei Goolge registriert sein. Ist dies geschehen, kann der Anwender eigene POI, so genannte Ortsmarken, auf der Karte platzieren und mit persönlichen Inhalten versehen. Die Inhalte können in diesem Zusammenhang aus Texten, z.B. einem Titel und einer Beschreibung, und aus Videos bzw. Bilder (siehe Abb. 3.3) bestehen. Multimediale Elemente werden über Links auf die entsprechende Datei mit eingebunden, die durch Dienste von Tochterunternehmen wie Picasa[38], Youtube und Google-Video[39] bereitgestellt werden. Es spielt dabei keine Rolle, ob es sich bei Daten um eigen erstellte oder durch andere Nutzer hinzugefügte handelt. Der Content[40] und die Community sind somit auf mehrere Plattformen verteilt.

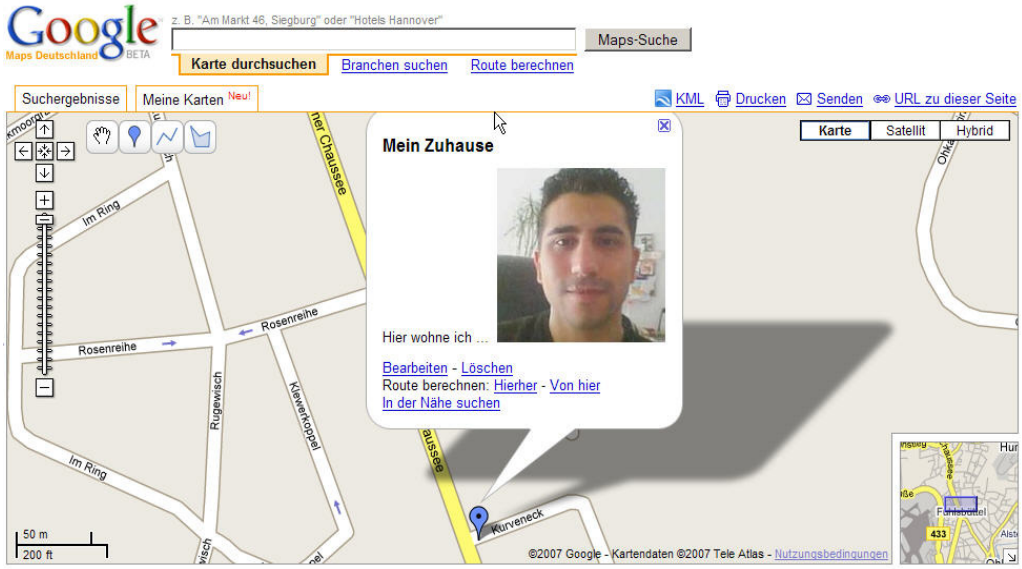


Abb. 3.3: Google MyMaps personalisierte Karte

Um allerdings eigene Videos und Bilder auf der Karte anzeigen zu lassen, bedarf es einer gewissen Vorbereitung. Für das Veröffentlichen von Fotos benötigt man das Web Album Picasa. Es stehen einem dabei 1024 MB, die auf Servern von *Google* bereitgestellt werden, zur freien Verfügung. Die Bilder können entweder über das Webportal oder über eine Software, die auf dem PC installiert werden muss, hochgeladen werden. Nutzt man Google-Video, wird wie bei Picasa keine weitere Registrierung benötigt, da der Account von Google für alle transparent ist. Entscheidet man sich allerdings für das weit verbreitete Youtube, muss, um eigene Videos zu hinterlegen, eine Registrierung bei Youtube erfolgen.

---

[38] Online im Internet unter http://picasa.google.de/.

[39] Online im Internet unter http://video.google.de/.

[40] Der Content bedeutet in diesem Zusammenhang die auf Picasa, Youtube und Google-Video hinterlegten Daten.

Hat man sich für eines der beiden Varianten entschieden, können eigene Videos hochgeladen und wie bei Picasa, per Link auf der personalisierten Karte angezeigt werden. Um seine erstellte Karte anderen Menschen mitzuteilen, gibt es unterschiedliche Varianten. Die einfachste Variante ist das Erzeugen eines Links, der die Karte referenziert und im Browser anzeigt. Dieser Link kann beim Kreieren der personalisierten Karte per Mausklick erzeugt und per Mail an alle Freunde oder Bekannte versendet werden. Eine weitere Möglichkeit besteht darin, den bereits erwähnten Link auf seiner eigenen Homepage zu veröffentlichen. Somit können alle Internetnutzer die auf der Homepage landen, auf die personalisierte Karte zugreifen. Damit allerdings, über die von Google-Maps bereitgestellte Suchfunktion, alle Nutzer einen Zugriff auf den in der personalisierten Karte gehaltenen Content bekommen, muss die Karte mit „öffentlich“ markiert werden.
Wie man sehen kann, existiert die Community aus einer Menge von unterschiedlichen Teilnehmern. Sie beschränkt sich nicht, wie z.B. bei *Qiro*, auf registrierte Nutzer, sondern kann um jeden, der einen Internetzugang besitzt, erweitert werden. Die Möglichkeit, Videos oder Bilder über Youtube, Google-Video und Picasa durch einen Link mit in die personalisierte Karte zu integrieren, bringt unterschiedliche Plattformen und somit unterschiedliche Communities enger zusammen.
Google *MyMaps* ist von der Idee ähnlich wie die hier entwickelte Anwendung. Sie bietet einem die Möglichkeit personalisierte, mit multimedialen Elementen versehende POI einer Community zur Verfügung zu stellen und kartengestützt abzurufen. Der Unterschied liegt allerdings darin, dass die verwendeten Daten keinen direkten Bezug zu der Umgebung haben müssen. Sie können an irgendeinem Ort gemacht worden sein und sind somit, für einen ortsabhängigen Dienst nicht zu verwenden. Darüber hinaus ist diese Anwendung nicht für Mobiltelefone, sondern für den heimischen PC konzipiert. Dies nimmt einem die Möglichkeit auf Informationen vor Ort zuzugreifen, welche von Menschen erstellt wurden, die sich an diesem Ort schon einmal aufgehalten haben.

Google bietet allerdings auch auf dem mobilen Sektor einige Dienste an, was darauf schließen lässt, dass es nur eine Frage der Zeit ist, vom Mobiltelefon ortsabhängig, personalisierte Daten zu versenden und kartengestützt abzurufen.
Dienste wie Youtube und Picasa bieten bereits die Möglichkeit, Videos und Bilder, die über die Plattform veröffentlicht wurden, von jedem Mobiltelefon aus zu betrachten. Auch das Versenden von Videos über das Mobiltelefon via MMS an Youtube wird unterstützt und verschafft somit die Möglichkeit, Erlebnisse vor Ort der Community mitzuteilen. Das Versenden von Fotos über das Mobiltelefon an Picasa soll demnächst folgen. Der Unterschied zu der hier entwickelten Anwendung liegt allerdings darin, dass die

Informationen zwar vor Ort aufgenommen und versendet werden können, aber auch hier keinen Bezug zu der Umgebung haben. Die Daten werden nicht über Positionskoordinaten an die Umgebung gebunden und können somit auch nicht ortsabhängig abgerufen werden. Gerade diese Tatsache trägt dazu bei, dass die Fülle an Daten, die über die Plattformen Youtube und Picasa zustande kommen, nicht als zusätzliche Informationsquelle für ortsabhängige Dienste genutzt werden können.

Google hat mit dem Kauf von Youtube und seinen eigenen Diensten Picasa und Google-Maps einen wichtigen Schritt in Richtung Verbreitung von personalisierten, ortsabhängigen Daten gemacht. Alle Daten befinden sich auf von Google bereitgestellten Servern und sind somit für jeden und zu jeder Zeit abrufbar. Der Nachteil bei der Nutzung von Google-Maps mit Youtube ist allerdings, dass mehr als nur eine Registrierung nötig ist. Ein Single Sign-On wäre aus der Sicht des Nutzer wünschenswert.

Durch die Kombination dieser Dienste und mit einer auf dem Mobiltelefon abgestimmten Smart-Client oder Web Anwendung, die unter anderem die Positionsbestimmung per GPS unterstützt, könnte in naher Zukunft ein ähnlicher, wie in dieser Anwendung entstehender, Dienst angeboten werden.

3.3 Trailblazers

In diesem Beispiel geht es um eine Smart-Client Software, die für Menschen bestimmt ist, die für das Fortbewegen an einen Rollstuhl gebunden sind. Auf der ganzen Welt gibt es Wege zur Umwelt, die für Rollstuhlfahrer nur schwer erreichbar sind. Das *Trailblazers* System bietet einem die Möglichkeit Wege zu finden, welche Hindernisse wie beispielsweise Treppen oder gepflasterte Straßen ausschließt.

Die heutigen Navigationssysteme beschränken sich weitestgehend auf Straßen und sind daher rein für die Automobilindustrie vorgesehen. *Trailblazers* hingegen baut auf detailliertes Wissen auf und beinhaltet barrierefreie Wege wie Fuß- und Fahrradwege.

Die Idee dabei ist, eine große Community zu erreichen, die die nötigen Informationen der barrierefreien Wege liefert. Um dieser Community beizutreten benötigt man ein PDA oder Mobiltelefon mit GPS-Empfänger, die mit dem *Microsoft* Betriebsystem ausgestattet sind. Des Weiteren wird eine Smart-Client Software installiert, die als Bindeglied zwischen einem Server, dem Kernstück, und dem Benutzer fungiert. Da auch in dieser Arbeit der Community Aspekt zum Tragen kommt, hat die Software zwei entscheidende Funktionen. Zum einen können neue Daten, also Informationen über neue barrierefreie Wege, über die Software der Community zur Verfügung gestellt werden (aktive Part in einer Community) und zum anderen, können diese Informationen in Form eines Navigationssystems für Rollstuhlfahrer abgerufen werden (passive Part in einer Community).

Damit allerdings auf einen großen Datenbestand zurückgegriffen werden kann, müssen Informationen über barrierefreie Wege an einen zentralen Server gesendet werden. Um dies zu gewährleisten wird das Scannen von GPS Koordinaten mit einbezogen. Stellen wir uns vor, ein Rollstuhlfahrer der *Trailblazers* Community kommt von dem barrierefreien Weg „A“ (siehe Abb. 3.4) und findet einen neuen barrierefreien Weg, der noch nicht im System erfasst wurde. Tritt nun der Rollstuhlfahrer den neuen Weg an, werden in bestimmten abständen GPS Koordinaten erfasst und dem Server mitgeliefert. Ein neuer barrierefreier Weg wurde geschaffen. Um eine optimale Wegbeschreibung eines neuen barrierefreien Weges zu liefern, nimmt das System jede Koordinate unterschiedlicher Nutzer auf und bildet davon den Durchschnitt. Durch dieses Beispiel wird deutlich, dass die Community selbst die Güte dieser Anwendung bestimmt.

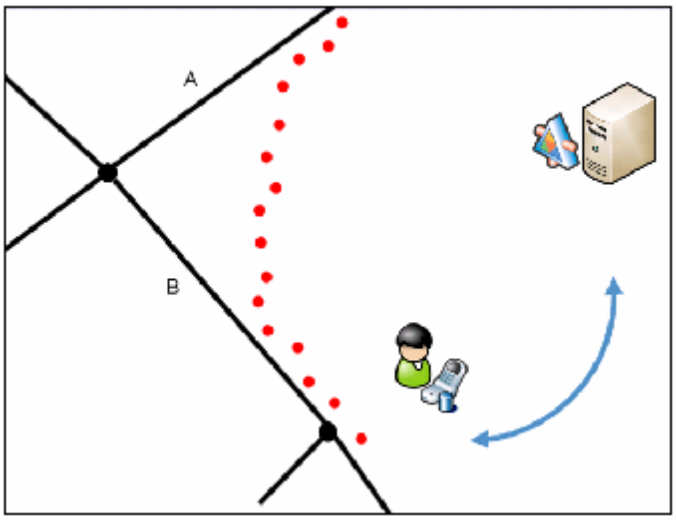


Abb. 3.4: Grafische Repräsentation eines Barriere freien Weges [Trailblazers]

Eine weitere Möglichkeit wird durch das Hinzufügen von POI geboten. Diese gehen dem Problem nach, dass ein barrierefreier Weg für die eine Person, nicht gleich einer anderen sein muss. So könnte es z.B. sein, dass eine Treppe für einen sportlichen Rollstuhlfahrer zu überwinden ist, für andere allerdings nicht. Der POI gibt dem Anwender detaillierte Informationen über ein bevorstehenden Ort oder Punkt, bei dem es zu einem Problem kommen kann. Als Informationsgrundlage können Bilder und Beschreibungen hinzugefügt werden, das in der Abbildung 3.5, mit dem Bild einer Treppe, zu sehen ist.

Der andere Part der Anwendung liegt in der Lieferung eines Navigationssystems. Dieses System baut auf den zuvor besprochenen Informationen über barrierefreie Wege auf und bietet Rollstuhlfahrern eine optimale Wegbeschreibung. Das Prinzip gleicht einem Navigationssystem wie man sie aus der Automobilindustrie her kennt. Ein Rollstuhlfahrer

muss seine aktuelle und die Zieladresse eingeben und bekommt daraufhin den optimalen barrierefreien Weg, der auf einer geografischen Karte grün markiert ist (siehe Abb. 3.5), geliefert. Die Daten der geografischen Karten basieren auf dem von *Microsoft* zur Verfügung stehenden Map-Server *MapPoint*. Diese Daten werden um die erzeugten POI und Wegbeschreibungen der *Trailblazers* Anwender erweitert.

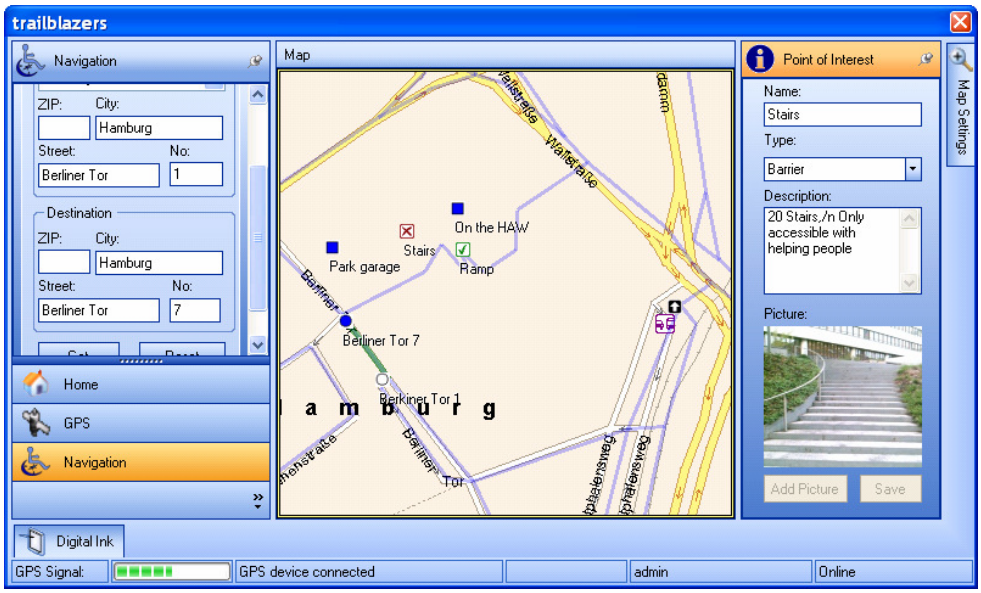


Abb. 3.5: Trailblazers Smart-Client Software [Trailblazers]

Barrierefreie Wege werden auf der Karte blau eingezeichnet. Bei einer neuen Berechnung einer barrierefreien Route werden diese Wege mit einbezogen und mittels grüner Linie auf der Karte kenntlich gemacht (siehe Abb. 3.5). So kann ein Rollstuhlfahrer sehen, welcher Weg der für ihn am günstigsten und bequemsten ist. Wurden auf dieser Route POI eingetragen, die durch blaue Icons auf der Karte dargestellt werden, können diese ausgewählt und deren Informationen angezeigt werden (siehe Abb. 3.5, POI einer Treppe).

Bei der Implementierung der *Trailblazers* Anwendung wurde mit einbezogen, dass es bei der Internetverbindung, in Form von Verbindungsausfällen zu Problemen kommen kann. So bietet die Smart-Client Software einen online und offline Modus an. Sobald ein Nutzer mit dem *Trailblazers* Server verbunden ist, kann jede Route erstellt und auf dem Mobiltelefon gespeichert werden. Sie sind somit auch offline verfügbar. Außerdem ist es möglich, Daten wie beispielsweise das Erstellen eines POI oder einer neuen Route offline durchzuführen, um diese Informationen zu einem späteren Zeitpunkt dem Server mitzuteilen. Der Aufbau der *Trailblazers* Anwendung beruht auf einer modernen Client-Server Architektur (siehe Abb. 3.6), die im Wesentlichen aus einem zentralen Server und

Smart-Clients besteht. Um eine sehr genaue Standortposition zu ermitteln, nutzen die Smart-Clients die so genannte differenzial GPS (D-GPS) Technologie, die den relativen Fehler der GPS Positionsbestimmung weitestgehend eliminiert.

Damit Clients Informationen erhalten können, greifen sie alle auf ein und denselben WebService Provider zu, den *Trailblazers* Server. Dieser Server ist für das Berechnen der Routen, dem Erfassen neuer barrierefreier Wege und für das Bereitstellen von POI der Community verantwortlich. Damit eine gute Auswahl an visueller Darstellung den Anwendern geboten werden kann, wird auf der Serverseite der *Microsoft MapPoint WebService* und auf der Web-Clientseite *Microsoft Virtual Earth* verwendet.

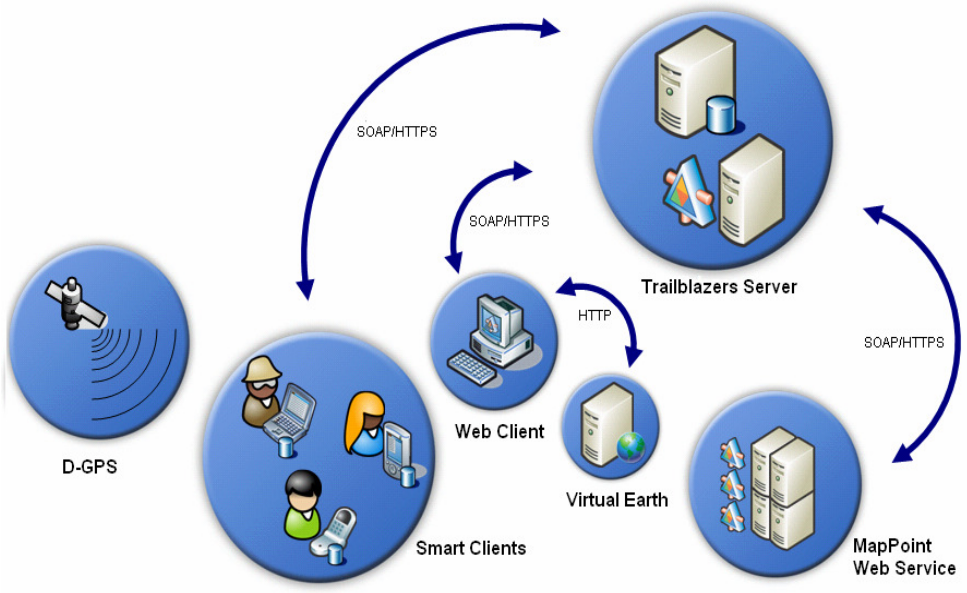


Abb. 3.6: Trailblazers Architektur [Trailblazers]

Die hier entwickelte Smart-Client Software weist einige Parallelen zu der *Trailblazers* Anwendung auf. Der Erfolg und die Qualität der Anwendung werden stark durch eine aktive Community bestimmt. Wenn es zu keinem Datenaustausch an barrierefreien Wegen oder wie hier, an interessanten Orten in Form von multimedialen POI kommt, erzeugt die Anwendung keinen informativen Mehrgewinn und ist somit nicht zu verwenden. Diese Informationen werden durch Teilnehmer der Community erstellt und tragen dadurch zu einem Gemeinschaftsgefühl bei. Darüber hinaus geht es in erster Line darum, Menschen durch zusätzliche Informationen zu helfen und durch visuelle Daten einen besseren Einblick der gegebenen Situation bzw. des Ortes zu liefern. Da diese Informationen ortsgebunden durch Nutzer der Community erstellt werden, geht es auch hier um eine Anwendung, die zeitgemäß auf die Bedürfnisse der heutigen Gesellschaft eingeht.

Im Gegensatz dazu steht die *Trailblazers* Applikation für eine bestimmte Anwendergruppe und verfolgt ein konkretes Problem, nämlich der eingeschränkten Handlungsfreiheit von behinderten Menschen, die auf einen Rollstuhl angewiesen sind. Die Informationen sind somit nur einer eingeschränkten Gruppe von Nutzen, könnten aber bei einer Erweiterung der Zielgruppe und somit der Daten durchaus erweitert werden.
Der große Nachteil ist bei den eingesetzten Softwarekomponenten zu sehen. Die Smart-Client Software ist nur auf Betriebssystemabhängigen, nämlich *Microsoft* properitären Systemen, ausführbar.

3.4 Bewertung der bestehenden Arbeiten

Qiro basiert auf einer Smart-Client Anwendung und bietet die Möglichkeit, eigene POI zu erstellen. Diese können einer Community oder den „Freunden“, den so genannten *Buddies* zur Verfügung gestellt werden. *Qiro* geht ferner der Frage nach „Wo befindest Du dich gerade?“, was soviel bedeutet, dass Freunde auf einer digitalen Karte angezeigt und kontaktiert werden können. Die Community beschränkt sich somit nur auf die *Buddy*-Liste, was sich an der geringen Anzahl der hinzugefügten POI widerspiegelt. Sie dienen eher dem Zweck, einen Treffpunkt mit einem Freund auszumachen, als einen interessanten Ort zu beschreiben. Außerdem müssen die Informationen eines POI nicht zwingend in Verbindung mit dem eigentlichen Standort stehen, da sie nicht „Live“ vor Ort durch ein Positionsbestimmungsverfahren der realen Welt zugeordnet werden. Darüber hinaus kann einem POI nur Textinformationen hinzugefügt werden, so dass die Nutzer einen sehr geringen Informationsgehalt erhalten.
Google *MyMaps* ist eine sehr komplexe Lösung, die nicht für Mobiltelefone, sondern für den heimischen PC ausgelegt ist. Mit ihr können personalisierte Karten erstellt und jedem Internetnutzer, durch die Generierung eines Links auf die Karte, zur Verfügung gestellt werden. Im Gegensatz zu der hier entwickelten Anwendung können die über Youtube, Google-Video oder Picasa bereitgestellten Daten in keinem Zusammenhang mit der Umgebung stehen, da sie nicht vor Ort und in Kombination mit Positionskoordinaten aufgenommen werden. Ein weiterer Punkt ist die Verbreitung und das Hinzufügen eigener Inhalte. Unterschiedliche Plattformen und Communities treffen hier aufeinander. Dies führt dazu, dass es sehr aufwendig ist, Daten bereitzustellen. Im Falle von Youtube erscheint die Plattform durch eine zuusätzliche Anmeldung nicht transparent. Ein Single Sign-On wäre aus der Sicht des Nutzers wünschenswert. Eine Veränderung ist durch das mobile Nutzen von Youtube und Picasa absehbar und somit nur eine Frage der Zeit, bis eine Smart-Client Anwendung von Google eigene ortsabhängige, mit multimedialen Elementen versehende, POI erzeugen und kartengestützt abrufen kann.

Trailblazers baut, wie die hier entwickelte Anwendung, auf eine Community auf und ist abhängig von den bereitgestellten Daten. Es handelt sich dabei auch um eine Smart-Client Anwendung, die die erzeugten Daten über GPS Koordinaten mit der Umwelt verbindet. Im Gegensatz zu der hier entwickelten Anwendung ist das *Trailblazers* System für Menschen ausgelegt, die auf einen Rollstuhl angewiesen sind und einen barrierefreien Weg suchen. Ein Navigationssystem soll diesen Menschen helfen, einen hindernisfreien Weg zu finden. Selbst definierte POI beschreiben Barrieren und dienen nicht als Informationsquelle für interessante Orte. Ein großer Vorteil im Design der Anwendung liegt bei der Implementierung des offline Modus, der ähnlich wie in der hier entwickelten Anwendung z.B. bei der Speicherung von POI, zum Tragen kommt. Ein weiterer Unterschied liegt bei der Wahl der Laufzeitumgebung. Da auf das betriebsystemabhängige .NET CF zurückgegriffen wird, ist die Smart-Client Software nur auf *Microsoft* properitären Systemen ausführbar.

Die hier erstellte Anwendung ist im Gegensatz zu *Trailblazers* für keine spezielle Community bestimmt. Sie baut auf einer Plattform auf, so dass im Unterschied zu *Google MyMaps* ein Single Sign-On gewährleistet wird. Es soll eine Software für mobile Endgeräte entwickelt werden, mit der es möglich ist, „Live“ vor Ort und im Kontext zur Umgebung, Multimediadaten zu erzeugen. Dieses kann durch *Qiro* und *Google MyMaps* nicht gewährleistet werden.
Darüber hinaus soll diese Software dem Anwender die Möglichkeit bieten, auf die durch die Community erstellten Daten, ortsabhängig zu zugreifen. Um dies zu gewährleisten, wird die aktuelle Position nicht wie bei *Qiro* zellenbasiert, sondern via GPS ermittelt. Ist dies geschehen, werden die aktuelle Position und die der abgerufenen POI zur Orientierung auf einer digitalen Karte angezeigt. Ein Filtern der POI nach Interesse soll außerdem implementiert werden. Möchte man nun auf die hinterlegten Informationen eines interessanten Ortes zugreifen, kann man diesen auf der Karte selektieren. Die entsprechenden Multimediadaten können auf dem Mobiltelefon abgespielt werden. Der Informationsgehalt ist somit größer als bei *Qiro* und *Trailblazers*.

4 Systemanalyse

Die Systemanalyse beschreibt die erste Phase des Prozesses der Softwareentwicklung und dient im Wesentlichen:

- zur Anzeige der Ergebnisse, die der Anwender vom System erwartet,
- als Basis für das Design und die Optimierung des Designs,
- als Basis für eine logische Herangehensweise,
- als Basis für Tests während der Entwicklung und
- zur Kommunikation der Grundlagen des Systems in verständlicher, nicht-technischer Form für alle Beteiligten.

Hierfür wird als Erstes die zugrunde liegende Systemidee definiert und die Zielgruppe dargestellt. Anschließend werden die möglichen Anwendungsfälle aus der Sicht des Nutzers beschrieben und anhand von Spezifikationen näher erläutert. Dabei wird zwischen den funktionalen und nicht funktionalen Anforderungen unterschieden. Sie bilden die Grundlage für das Softwaredesign. Mit der Qualität der Anforderungen steht und fällt auch die Qualität des Produkts. Die Anforderungen beschreiben die gewünschten Eigenschaften des hier entstehenden Systems. Sie haben Einfluss auf dem Prozessverlauf, die Produktivität und vor allem auf die Akzeptanz aus der Sicht des Anwenders und die Wartbarkeit aus der Sicht des Entwicklers.

Das Kapitel beschreibt das zu erstellende System aus der Sicht des Nutzers. Zum besseren Verständnis der Abläufe und für die detaillierte Schilderung werden die standardisierte Beschreibungssprache *Unified Modeling Language* (UML) sowie Spezifikation verwendet, wie sie im [REN 2007] definiert sind. Aufgrund der zuvor definierten Anforderungen werden am Ende des Kapitels die wesentlichen Herausforderungen, die bei der hier entwickelten Anwendung umgesetzt werden sollten, genannt.

4.1 Systemidee

Realisiert werden soll eine J2ME Anwendung für Mobiltelefone zur multimedialen „Live“ Annotation von „Points of Interest“.

Dabei steht das „Live“ Annotieren von „Points of Interest“ dafür, das vor Ort, im Augenblick des Geschehens Daten erstellt und diese Daten einer Community zur Verfügung gestellt werden können. Ein POI beschreibt in dieser Anwendung einen interessanten Ort oder ein Erlebnis. Hierfür muss beachtet werden, dass die Daten die den POI beschreiben, einen direkten Bezug zur realen Welt erhalten. Um einen möglichst

großen Informationsgehalt zu bekommen, sollen diese Daten aus Texten und Multimedia-Elementen bestehen, wie z.B. Bilder oder Videos. Ein vollständiger POI besteht somit aus Daten die den Ort beschreiben und den dazugehörigen Standortkoordinaten.
Nutzer können auf diese Daten ortsabhängig und nach Interesse gefiltert zugreifen und erhalten Informationen über interessante Orte aus erster Hand. Darüber hinaus soll dem Anwender zur Orientierung die aktuelle Position und die der abgerufenen POI auf dem Display angezeigt werden. Dies sollte über eine digitale Kartendarstellung auf dem Mobiltelefon geschehen.

4.2 Zielgruppe

Die hier entwickelte Anwendung richtet sich an alle Menschen, die das Interesse haben ihre persönlich erlebten Ereignisse als Informationsquelle für andere interessierte Teilnehmer einer Community zur Verfügung zu stellen. Der große Reiz dabei ist, dass multimediale Inhalte angehängt werden können. Dies führt zu einem größeren Informationsgehalt gegenüber einfachen Texten.
Ferner richtet sich die Anwendung an Menschen, die sich an einem fremden Ort befinden oder einfach nur auf Informationen von Personen zugreifen wollen, die sich an diesen Orten schon einmal aufgehalten haben. Sie können über die Software auf die bereitgestellten Daten zugreifen und erhalten somit ortsabhängig Informationen über interessante Plätze aus erster Hand. Eine Einschränkung im Bezug auf das Geschlecht, das Alter oder der Herkunft, muss bei Einhaltung des seriösen Informationsaustausches, nicht bedacht werden.

4.3 Funktionale Anforderungen

Bei den funktionalen Anforderungen werden die grundsätzlichen Interaktionen des Systems mit der Umgebung aufgelistet. Diese ergeben sich aus den im Kapitel 1.2 definierten Zielen (fachlichen Anforderungen).
Das hier zu implementierende System besteht aus zwei globalen Aufgabenstellungen. Zum einen das Erzeugen eines neuen POI und dessen Annotation von Multimediadaten (Produzentensicht). Dies schließt das Versenden der Daten zu einem zentralen Zugangspunkt mit ein. Zum anderen sollen POI ortsabhängig abgerufen und auf einer digitalen Karte angezeigt werden können (Konsumentensicht). Innerhalb dieser beiden Aufgabenstellungen muss das System bestimmte Anwendungsfälle erfüllen, die in den nun folgenden Abschnitten beschrieben werden.
Dabei werden die funktionalen Anforderungen aus Produzenten- und Konsumentensicht in Form von Use-Case Diagrammen getrennt voneinander betrachtet.

4.3.1 Anwendungsfälle aus Produzentensicht

Abbildung 4.1 zeigt die Anwendungsfälle aus Produzentensicht, die vom System erfüllt werden sollten. Der Produzent in einer Community basierten Anwendung ist derjenige, der einen aktiven Beitrag zur Community leistet. In dieser Anwendung besteht der Beitrag darin, neue informative POI zu erzeugen und diese der Community als Informationsquelle zur Verfügung zu stellen. Die beteiligten Aktoren sind zum einen ein zentraler Zugangspunkt und zum anderen der Nutzer, der über das Mobiltelefon Aktionen anstößt. Der zentrale Zugangspunkt steht jeder Zeit zur Verfügung und ist für das Entgegennehmen der Daten zuständig.

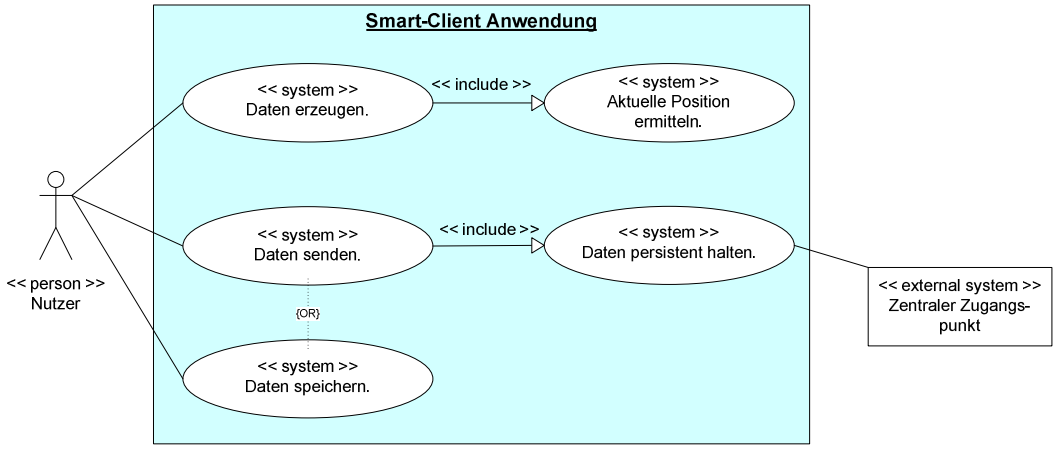


Abb. 4.1: Anwendungsfälle aus Produzentensicht

4.3.1.1 Spezifikationen aus Produzentensicht

Teilnehmern der Community sollen nützliche, ortsabhängige Informationen über interessante Orte oder Erlebnisse geliefert bekommen. Um das zu gewährleisten, sollen Textinformationen und Multimediadaten vor Ort erzeugt werden können.

Zur Vorbeugung von Performanceproblemen bei der Übertragung und von einer Unübersichtlichkeit auf dem Display sollten die erstellten Daten durch die Angabe einer Kategorie eingeteilt werden können. Eine Filtermöglichkeit vor dem Abrufen der Daten ist demnach gegeben. Des Weiteren müssen ein Name und eine Beschreibung des Ortes bzw. des Erlebten mit anzugeben sein. Dadurch kann vor dem Abrufen der Multimediadaten eine kurze Information des POI angezeigt werden. Ist diese Information ausreichend, können die Kosten für das Abrufen der Multimediadaten gespart werden.

Name des Use-Case	Daten erzeugen.
Aufgabe	Informative Daten, die einen Ort oder Erlebtes repräsentieren, sollen erzeugt werden. Dazu gehören Multimediadaten und Textinformationen. Die Textinformationen beinhalten den Namen, eine Beschreibung und die Wahl einer Kategorie, die es ermöglicht, POI nach Interesse zu filtern.
Beteiligter Aktor	Nutzer
Auslöser	Nutzer mittels Smart-Client Anwendung
Vorbedingung	1. Eine Kamera muss auf dem Endgerät vorhanden sein. 2. Die Laufzeitumgebung unterstützt die Ansteuerung der Kamera. 3. Ausreichend Speicher muss für die Aufnahme eines Videos bzw. Bildes vorhanden sein.
Nachbedingung	Daten, die einen interessanten Ort oder ein Erlebnis beschreiben, wurden erstellt und stehen der mobilen Anwendung zur weiteren Verarbeitung zur Verfügung.

Die hier erzeugten Daten sollen den Aufenthaltsort eindeutig zugeordnet werden können, d.h. dass die Daten einen direkten Bezug zur realen Welt erhalten müssen. Ortsbezogene Informationen können nur angezeigt werden, wenn diese Daten mit Koordinaten versehen sind, die den Standort eindeutig identifizieren. Wurde die aktuelle Position ermittelt, steht der Anwendung ein vollständiger POI zur Verfügung. Er beinhaltet die Anwendungsfälle „Daten erzeugen“ und „Aktuelle Position ermitteln“.

Name des Use-Case	Aktuell Position ermitteln.
Aufgabe	Bestimmung des eigenen Standortes, damit ein Bezug der erzeugten Daten zur realen Welt hergestellt werden kann.
Beteiligter Aktor	Nutzer
Auslöser	Nutzer mittels Smart-Client Anwendung
Vorbedingung	1. Der Anwendungsfall „Daten erzeugen“ muss erfolgreich ausgeführt worden sein. 2. Der Anwendung muss es über ein Positionsbestimmungsverfahren möglich sein, die aktuellen Standortkoordinaten zu ermitteln.
Nachbedingung	Die aktuelle Position des Standortes wurde ermittelt. Eine multimediale „Live“ Annotation eines POI wurde somit vollzogen und steht der Anwendung zur Verfügung.

Die erstellten Informationen über interessante Orte und Erlebnisse bekommen nur einen Sinn, wenn sie anderen Menschen mitgeteilt werden können. Es muss somit möglich sein, vor Ort alle Daten an einen zentralen Punkt zu senden, damit die Daten für das ortsabhängige Abrufen persistent gehalten werden können.

Name des Use-Case	Daten senden.
Aufgabe	Einen vollständigen, neu erzeugten POI an einen zentralen Punkt senden. Der Nutzer erhält eine Erfolgsmeldung.
Beteiligter Aktor	Nutzer, ein zentraler Zugangspunkt
Auslöser	Nutzer mittels Smart-Client Anwendung
Vorbedingung	1. Der Anwendungsfall „Aktuelle Position ermitteln“ muss erfolgreich ausgeführt worden sein. 2. Eine Internetverbindung muss vorhanden sein. 3. Der zentrale Zugangspunkt muss erreichbar sein.
Nachbedingung	Die gesamten Daten eines POI wurden an einen zentralen Punkt gesendet. Der Nutzer erhält eine Erfolgsmeldung auf dem Mobiltelefon.

Da beim Senden der Daten ein Austausch zwischen unterschiedlichen Systemen über das Internet stattfindet, können Verbindungsabbrüche den Erfolg verhindern. In diesem Zusammenhang spricht man auch von einem online/offline Problem. Die Anwendung sollte deshalb das lokale Speichern der Daten unterstützen. Zum einen gehen beim Fehlschlagen der Funkverbindung die Daten nicht verloren und zum anderen können dadurch Kosten gespart werden. Ist nämlich, zu einem späteren Zeitpunkt ein kostenloser Zugangspunkt verfügbar, können hier die normalerweise anfallenden Verbindungskosten gespart werden.

Name des Use-Case	Daten speichern.
Aufgabe	Ein zuvor erstellter POI wird lokal für ein späteres Versenden gespeichert. Der Nutzer erhält eine Erfolgsmeldung.
Beteiligter Aktor	Nutzer
Auslöser	Nutzer mittels Smart-Client Anwendung
Vorbedingung	1. Der Anwendungsfall „Aktuelle Position ermitteln“ muss erfolgreich ausgeführt worden sein. 2. Speicher und Zugriffsrechte müssen vorhanden sein.
Nachbedingung	Alle Daten des POI wurden lokal gespeichert und können für ein späteres Versenden genutzt werden. Der Nutzer erhält nach dem Speichern eine Erfolgsmeldung auf dem Display angezeigt.

Damit die erzeugten Daten gesendet und zu einem späteren Zeitpunkt wieder abgerufen werden können, muss ein zentraler Zugangspunkt diese Daten persistent halten.

Name des Use-Case	Daten persisten halten.
Aufgabe	Ein vollständig empfangener POI wird gespeichert.
Beteiligter Aktor	Zentraler Zugangspunkt
Auslöser	Nutzer mittels Smart-Client Anwendung
Vorbedingung	1. Der Anwendungsfall „Daten senden“ muss erfolgreich ausgeführt worden sein. 2. Speicher und Zugriffsrechte müssen vorhanden sein.
Nachbedingung	Der vollständig gesendete POI wurde für das spätere Abrufen gespeichert. Eine Erfolgsnachricht wurde der Smart-Client Anwendung zugeteilt.

4.3.2 Anwendungsfälle aus Konsumentensicht

In der Abbildung 4.2 sind die Anwendungsfälle aus Konsumentensicht dargestellt. Ein Konsument ist in einer Community basierten Anwendungen derjenige, der auf die bereitgestellten Informationen zugreifen möchte, um einen Mehrwert zu erhalten. Die beteiligten Aktoren sind der Nutzer, der zentrale Zugangspunkt und ein Map-Server. Der Nutzer kann über das Mobiltelefon Aktionen auslösen, die im System bestimmte Veränderungen hervorrufen und ein Resultat erzeugen. Der zentrale Zugangspunkt steht zu jeder Zeit zur Verfügung und kann über die Smart-Client Anwendung für bestimmte Anforderungen angesprochen werden. Zu guter Letzt gibt es einen Map-Server, der wie im Kapitel Location-Based Services beschrieben, Kartenmaterial zur freien Verfügung stellt. Im Weiteren werden die Anwendungsfälle spezifiziert. Der Anwendungsfall „POI abrufen“ beinhaltet die mit *include* gekennzeichneten Anwendugsfälle.

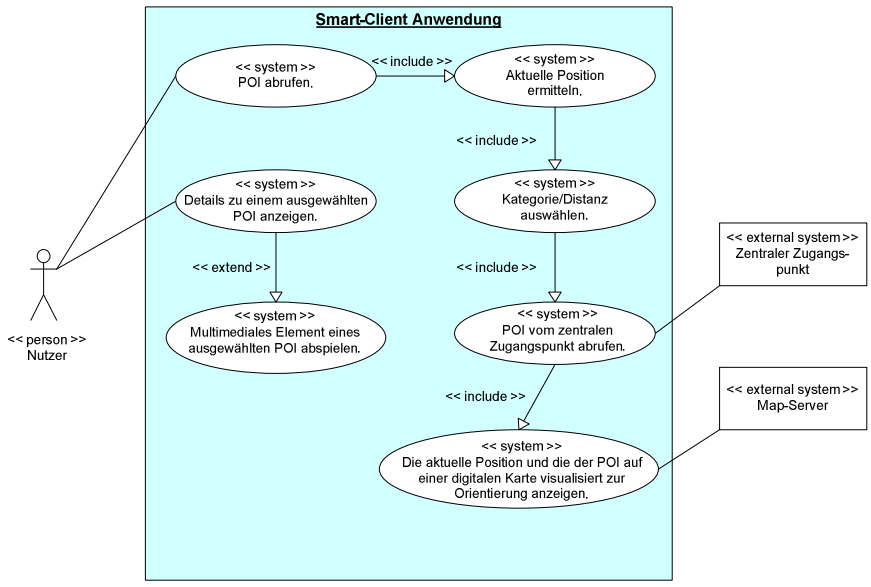


Abb. 4.2: Anwendungsfälle aus Konsumentensicht

4.3.2.1 Spezifikationen aus Konsumentensicht

Damit die beim zentralen Zugangspunkt hinterlegten POI ortsabhängig abgerufen werden können, muss der aktuelle Standpunkt des Nutzers ermittelt werden. Da es sich hierbei um die gleiche Anforderung wie beim Erzeugen eines POI handelt, kann an dieser Stelle auf die Spezifikation „Aktuelle Position ermitteln“ verwiesen werden.

Dem Nutzer sollen nicht alle verfügbaren POI auf einmal angezeigt werden. Deshalb muss vor dem Abrufen der POI eine Kategorie und eine Distanzangabe ausgewählt werden. Die aktuelle Position, die Kategorie- und die Distanzangabe nutzt der zentrale Zugangspunkt (LBS), um die POI nach Interesse und der Entfernung zu filtern. Es werden somit nur die POI angezeigt, die gewünscht sind und sich in der unmittelbaren Umgebung befinden. Dies kann bei großen Datenmengen auch Performanceprobleme und, aufgrund des Internetzugangs, hohe Verbindungskosten verringern.

Name des Use-Case	Kategorie/Distanz auswählen.
Aufgabe	Anzeigen der möglichen Kategorien und Distanzen, für das serverseitige Filtern der POI.
Beteiligter Aktor	Nutzer
Auslöser	Nutzer mittels Smart-Client Anwendung
Vorbedingung	1. Der Anwendungsfall „Aktuelle Position ermitteln.“ muss erfolgreich ausgeführt worden sein.
Nachbedingung	Die ausgewählte Kategorie und Distanz stehen der Anwendung zur Verfügung.

Die erstellten POI sollen an einem zentralen Punkt für das Abrufen persistent gehalten werden. Damit diese Daten als Informationsquelle z.B. einem reisenden Nutzer auf dem Mobiltelefon angezeigt werden können, muss das Abrufen derer möglich sein.

Name des Use-Case	POI vom zentralen Zugangspunkt abrufen.
Aufgabe	Anhand der ausgewählten Kategorie, Distanz und der eigenen Position werden die vom zentralen Zugangspunkt bereitgestellten POI der Smart-Client Anwendung zur Verfügung gestellt.
Beteiligter Aktor	Nutzer, zentraler Zugangspunkt (LBS)
Auslöser	Nutzer mittels Smart-Client Anwendung
Vorbedingung	1. Der Anwendungsfall „Kategorie/Distanz auswählen.“ muss erfolgreich ausgeführt worden sein. 2. Eine Internetverbindung muss vorhanden sein. 3. Der zentrale Zugangspunkt muss erreichbar sein. Dieser filtert die Daten nach der übergebenen Kategorie und der Distanz. Mit der Distanzangabe und der übergebenen Position wird serverseitig eine Umkreisberechnung durchgeführt.
Nachbedingung	Die gefilterten POI stehen der Smart-Client Anwendung für eine weitere Bearbeitung zur Verfügung. Die Übermittelten Daten beinhalten für jeden POI die Position, der Name und die Beschreibung. Die multimedialen Inhalte sollen aufgrund der Lazy-Loading Strategie zu einem späteren Zeitpunkt geladen werden.

Damit der Nutzer sich in der Umgebung orientieren kann, muss eine digitale Karte auf dem Display angezeigt werden. Diese soll anhand der zuvor ausgeführten Positionsbestimmung den Standort des Nutzers und die der abgerufenen POI mittels Marker anzeigen.

Name des Use-Case	Die aktuelle Position und die der POI auf einer digitalen Karte visualisiert zur Orientierung anzeigen.
Aufgabe	Es soll eine digitale Karte mit der aktuellen Position und der abgerufenen POI auf dem Mobiltelefon zur Orientierung angezeigt werden.
Beteiligter Aktor	Nutzer
Auslöser	Nutzer mittels Smart-Client Anwendung, Map-Server
Vorbedingung	1. Der Anwendungsfall „POI vom zentralen Zugangspunkt abrufen.“ muss erfolgreich ausgeführt worden sein. 2. Der Map-Server muss erreichbar sein.
Nachbedingung	Dem Nutzer wurden die Standorte der zuvor abgerufenen POI und die eigene Position auf dem Mobiltelefon visuell mittels Kartendarstellung angezeigt.

Wird die Karte mit den POI angezeigt, sollte es möglich sein auf die Detailinformationen eines jeden einzelnen POI zuzugreifen. Sinnvoll wäre in diesem Zusammenhang auch, dass erst die Textinformationen, also der Name und die Beschreibung angezeigt werden, um im Vorwege eine Beurteilung des Ortes zu erhalten. Scheint der ausgewählte POI interessant zu sein, kann auf die multimedialen Inhalte zugegriffen werden (Lazy-Loading Strategie). Dadurch werden mögliche Performanceprobleme verhindert und unnötige Kosten, die beim Runterladen der Multimediadaten entstehen, eingespart.

Name des Use-Case	Details zu einem ausgewählten POI anzeigen.
Aufgabe	Name und Beschreibung eines ausgewählten POI anzeigen.
Beteiligter Aktor	Nutzer
Auslöser	Nutzer mittels Smart-Client Anwendung
Vorbedingung	1. Der Anwendungsfall „Die aktuelle Position und die der POIs auf einer digitalen Karte visualisiert zur Orientierung anzeigen.“ muss erfolgreich ausgeführt worden sein. 2. Es muss mindestens ein POI vorhanden und aus der Karte heraus ausgewählt worden sein.
Nachbedingung	Dem Nutzer wurden die Detailinformationen des POI angezeigt.

Damit der Nutzer nützliche Informationen über interessante Orte bekommt, sollen auch multimediale Inhalte, die von Teilnehmern der Community zur Verfügung gestellt wurden, abgespielt werden können. Da es sich hier um Performance-, Speicher- und Übertragungsintensiven Anforderungen handelt, sollte das Abspielen nur auf Wunsch des Nutzers und nicht prinzipiell ausgeführt werden.

Anforderung	Multimediales Element eines ausgewählten POI abspielen.
Aufgabe	Das vom Produzenten erzeugte Video bzw. Bild des POI anzeigen.
Beteiligter Aktor	Nutzer, zentraler Zugangspunkt (LBS)
Auslöser	Nutzer mittels Smart-Client-Software
Vorbedingung	1. Der Anwendungsfall „Details zu einem ausgewählten POI anzeigen.“ muss erfolgreich ausgeführt worden sein. 2. Das Abspielen von multimedialen Inhalten muss auf dem Mobiltelefon unterstützt werden. 3. Eine Internetverbindung muss vorhanden sein. 4. Der zentrale Zugangspunkt muss erreichbar sein. Dieser stellt dem Client die multimedialen Inhalte zum Abrufen bereit.
Nachbedingung	Dem Nutzer wurde das vom zentralen Zugangspunkt zur Verfügung gestellte Video bzw. Bild eines ausgewählten POI angezeigt.

4.4 Nichtfunktionale Anforderungen

Im Folgenden werden die nichtfunktionalen Anforderungen des Systems dargelegt. Sie beschreiben die Qualität, die sich aus den Anforderungen aus Nutzersicht ergeben und gehen u.a. der Frage nach, „Wie fehlertolerant das System sein soll?“. Da es sich bei der hier entwickelten Anwendung um eine Smart-Client Anwendung handelt, die einen zentralen Zugangspunkt für das Speichern und Abrufen der erstellten POI benötigt, werden die nun folgenden Anforderungen client- bzw. serverseitig unterschiedlich betrachtet. Der Server beschreibt in diesem Zusammenhang den zentralen Zugangspunkt, der Client, die Smart-Client Anwendung.

Anforderungen der Zuverlässigkeit

Die Zuverlässigkeit umfasst die Anforderungen an die *Robustheit* und die *Dauerhaftigkeit* des hier zu entwickelnden Systems und wird im Folgenden näher erläutert.

Serverseitig

Der hier zu implementierende Server stellt den zentralen Zugangspunkt der Client-Anwendung dar. Dieser darf nur Anfragen entgegennehmen, die vom Client versendet wurden. Fehler bei der Übertragung müssen abgefangen und geloggt werden und dürfen nicht zu einer Störung des Systems führen. Kommt es dennoch zu einem Absturz, muss der Server nach einem Neustart voll funktionsfähig sein.

Clientseitig

Verbindungsprobleme müssen dem Nutzer mitgeteilt werden und dürfen zu keinem Absturz der Anwendung führen. Gegebenenfalls werden die Ausnahmen geloggt, gespeichert und bei nächster Gelegenheit zur weiteren Bearbeitung an den Server gesendet. Zu jeder Zeit muss die Anwendung in einen sicheren Zustand versetzt werden können und nach einem Neustart voll funktionsfähig sein.

Anforderungen der Verfügbarkeit

Im Folgenden werden die Anforderungen an die Verfügbarkeit des hier zu entwickelnden Systems beschrieben. Es sagt aus, wie lange das System operational benutzbar sein soll.

Serverseitig

Der Server sollte zu jeder Zeit verfügbar sein. Dies ist notwendig, da zu jeder Tageszeit Daten abgerufen und gesendet werden können.

Clientseitig

Ist die Client-Anwendung auf dem Mobiltelefon installiert, muss sie zu jeder Zeit ausführbar sein. Kommt es während der Anwendung bei der Übertragung von Daten zu einem Problem, sollte der Nutzer informiert und die Anwendung in einen sicheren Zustand versetzt werden. Ein Speichern der erstellten Daten sollte bei Verbindungsabbruch ebenfalls möglich sein. Die Daten können demnach zu einem späteren Zeitpunkt erneut versendet werden.

Anforderung an die Interoperabilität

Die serverseitige Implementierung sollte so aufgebaut werden, dass sie interoperabel, also von unterschiedlichen Plattformen aus erreichbar ist. Es wäre somit unerheblich, ob die Smart-Client Anwendung auf dem *Microsoft .NET Compact Framework* oder dem *Sun J2ME Framework* aufsetzt.

Anforderung an die Internationalisierung

Die verwendetet Programmiersprache sollte standardisiert und weit verbreitet sein, da diese, im Bezug auf unterschiedliche Benutzergruppen, Möglichkeiten der Internationalisierung bietet.

Anforderungen der Portabilität/Betriebssystemunabhängigkeit

Die Anforderungen an die Portabilität/Betriebssystemunabhängigkeit gehen dem Wunsch nach, dass das System auf möglichst einfache Art und Weise an unterschiedliche Betriebsysteme angepasst werden kann. Um dies zu gewährleisten sollten folgende Anforderungen vom System unterstützt werden:

Clientseitig

Die verwendete Laufzeitumgebung der Client Anwendung muss plattform- und betriebssystemunabhängig sein. Darüber hinaus sollte die Programmiersprache zu einem gewissen Grad portabel sein, damit das Programm mit wenig Aufwand, aber mit nur gering eingeschränkter Funktionalität, auf anderen Plattformen läuft.

Anforderungen an die Modifizierbarkeit und Wartbarkeit

Im Folgenden werden die Anforderungen an die Modifizierbarkeit und Wartbarkeit beschrieben, wovon Aufwand und Erfolg zukünftiger Änderungen an der Architektur oder Systemkomponenten abhängen.

Clientseitig

Die Architektur der Smart-Client Anwendung muss so implementiert werden, dass mehrfach benötigte Eigenschaften durch Komponenten abzurufen sind. Dies schafft eine klare Struktur, gute Wiederverwendbarkeit und einen einfachen Austausch beim Verändern der Eigenschaften. Außerdem kann auf zukünftige Anforderungen gezielt und schnell reagiert werden.

Anforderungen an die Benutzbarkeit und Ergonomie

Im Folgenden werden die Vorgaben der Systemverständlichkeit, Erlernbarkeit und Bedienbarkeit formuliert, die einen wesentlichen Beitrag zum Erfolg dieses Systems leisten.

Clientseitig

Damit diese Anwendung von allen Menschen bedient werden kann, muss sie verständlich, intuitiv in der Handhabung und einfach im Aufbau sein. Die Smart-Client Anwendung hat zwei Hauptfunktionen. Zum einen soll es möglich sein, eigene POI zu erzeugen und sie der Community zur Verfügung zu stellen, und zum anderen, diese Daten wieder abzurufen. Es sollte demzufolge je nach Funktion eine Aufteilung innerhalb des Programms implementiert werden, damit die Struktur klar erkennbar und das Bedienen der jeweiligen

Funktion ohne weitere Kenntnisse ausführbar ist. Bei einem Fehlverhalten der Anwendung ist es sinnvoll, den User durch eine verständliche Erklärung zu informieren. Durch diese Art von Rückkopplung weiß der Benutzer zu jeder Zeit in welchem Zustand sich die Anwendung befindet. Außerdem wird dadurch das Gefühl eines sicheren und gut bedienbaren Systems gesteigert. Ferner darf die Benutzeroberfläche, da es sich bei einem Mobiltelefon um eine sehr kleine Fläche handelt, nicht abschreckend wirken. Sie sollte informativ und ergonomisch[41], durch einfache Menüstrukturen und Farben, aufgebaut sein.
Eine wichtige Herausforderung stellt außerdem die Bedienbarkeit dar. Das Aufnehmen bzw. Abspielen multimedialer Inhalte und das Anzeigen der digitalen Karten muss effizient, ruckelfrei und zu jeder Zeit bei der Ausführung der Smart-Client Anwendung möglich sein.

Anforderungen der Sicherheit

Im Folgenden werden die Anforderungen an die Sicherheit[42], was die Gewährleistung der Vertraulichkeit im Umgang mit sensiblen Informationen des Nutzers umfasst, dargelegt.

Serverseitig

Es dürfen nur Daten entgegengenommen und abgerufen werden, die von der Smart-Client Anwendung aus verschickt bzw. angefragt worden sind. Um einer sicheren Abwehr gegen mutmaßliche Angriffe Serverseitig vorzubeugen, sollte eine Firewall eingesetzt werden.

Clientseitig

Bei der Client-Anwendung sollten Frameworks zum Einsatz kommen, die darauf achten, dass es bei jeder Art von Zugriffen auf Systemkomponenten zu keinem Verlust von sensiblen Daten kommt. Es muss unterbunden werden, dass ohne Einwilligung des Nutzers, kostenpflichtige Verbindungen geschaltet oder Spionageangriffe im Hintergrund der Anwendung ausgeführt werden. Die Client-Anwendung sollte als vertrauenswürdig eingestuft (signiert) werden.

---

[41] Näheres zum Thema Softwar-Engineering [RAASCH SE 2006].

[42] Einen umfassenden Einblick über die Vorgehensweisen der IT-Sicherheit gibt das Bundesamt für Sicherheit in der Informationstechnik [BIS 2005].

Rechtliche Anforderungen

Im Folgenden werden die rechtlichen Anforderungen, die sich durch das Veröffentlichen von multimedialen Daten ergeben, beschrieben.

Bei Community basierten Anwendungen haben prinzipiell alle Nutzer das Recht, einen Beitrag zum Gesamten zu leisten. In dieser Anwendung ist es das Veröffentlichen von selbst erlebten Ereignissen in Form von POI mit multimedialem Inhalt. Es ist somit vorstellbar, dass Nutzer dieser Plattform rechtsradikale, hetz, pornografische oder andere diskriminierende Videos oder Bilder anderen Teilnehmern, vielleicht sogar Kindern, zugänglich machen wollen. Auch das Verbreiten von kopier geschützten Daten sollte verhindert werden, da sonst mit einer Klage[43] von den Urhebern gerechnet werden kann.
Da allerdings eine Berücksichtigung dieser rechtlichen Anforderungen den Rahmen dieser Arbeit sprengen würde, wird an dieser Stelle auf eine ähnliche Lösung mittels Filtern verwiesen[44], die über eine spezielle Erkennungstechnik u.a. pornografische Bilder erkennen kann.

4.5 Zusammenfassung

In diesem Kapitel ging es um die Anforderungen, die bei der Realisierung des hier entstehenden Systems erfüllt werden sollten. Sie bilden die Grundlage für die Entwicklung und bestimmen die Qualität der Anwendung. Die wesentlichen Herausforderungen, die sich bei der Systemanalyse ergaben, sind:

1. Multimediale Inhalte sollen über das Mobiltelefon vor Ort erstellt und abgerufen werden können. Dies stellt eine große Herausforderung in Bezug auf die Performance, Bedienbarkeit, den Transport und den zur Verfügung stehenden Ressourcen wie beispielsweise dem Speicher auf dem Mobiltelefon dar.
2. Die erzeugten Daten sollen einen interessanten Ort beschreiben und ortsabhängig abgerufen werden können. Ein genaues Positionsbestimmungsverfahren muss zum Einsatz kommen, damit die Daten einen direkten Bezug zur realen Welt erhalten.
3. Es muss ein Location-Based Service entwickelt werden, der anhand einer ausgewählten Kategorie, einer Distanzangabe und der aktuellen Position des Nutzers die gespeicherten POI filtert und diese der Client Anwendung zur Verfügung stellt.

---

[43] Das online Portal YouTube wurde Anfang Mai diesen Jahres auf Copyrights verklagt [YouTube 2007].

[44] Online im Internet unter http://www.heise.de/newsticker/meldung/9829.

4. Es muss vom Client aus möglich werden, neu erstellte POI zu versenden und diese vom zentralen Zugangspunkt wieder abrufen zu können.
5. Bei der Übertragung der Daten über eine Funkverbindung kann es durch Verbindungsabbrüche und, wie im Abschnitt 2.3 beschrieben, einer nicht garantierten Bandbreite zu Problemen kommen. Um dabei entstehende online/offline Probleme entgegenzuwirken, sollte das lokale Speichern der POI Daten sowohl beim Senden als auch beim Empfangen möglich sein.
6. Zur Orientierung soll eine digitale Karte auf dem Display angezeigt werden. Der Kartenausschnitt muss dabei den aktuellen Standort des Nutzers widerspiegeln und die eigene Position sowie die der abgerufenen POI mittels Marker visuell anzeigen. Außerdem muss es möglich sein, aus der Karte heraus auf die Detailinformationen zugreifen zu können. Damit Kosten, Performance und unnötige Zeit des Abrufens der Daten verringert werden, sollten zuerst die Textinformationen geladen und angezeigt werden (Lazy-Loading Strategie). Erst wenn diese Informationen nicht ausreichend oder weitere Auskünfte von Nöten sind, kann auf die hinterlegten Multimediadaten zurückgegriffen werden.
7. Aufgrund der Community Abhängigkeit des Systems, muss die Anwendung auf unterschiedlichen mobilen Endgeräten ausführbar sein. Deshalb sollte die Smart-Client Anwendung plattform- und betriebssystemunabhängig sein.

Da der Fokus dieser Arbeit in der Realisierung einer mobilen Anwendung liegt, wird auf die technischen Anforderungen des Servers nicht weiter eingegangen. Die technischen Anforderungen des Clients werden im Abschnitt 5.2 mit einbezogen.

Aufgrund der zeitlichen Begrenzung kann im Rahmen dieser Arbeit keine vollständige Implementierung erfolgen. Um dennoch die Funktionsweise des Systems zu verdeutlichen, wird auf die Umsetzung einiger Anforderungen verzichtet und der Funktionsumfang reduziert. Die umgesetzten Anforderungen des Clients werden im Abschnitt 5.5.1 beschrieben, die des Servers im Abschnitt 5.6.1.

5 Design und Realisierung

In diesem Kapitel wird das Design und die Realisierung des hier entwickelten Systems vorgestellt. Dabei werden zu Anfang, anhand der festgelegten Anforderungen, die Designentscheidungen getroffen. Diese bilden die Grundlage der weiteren Realisierung. Im Abschnitt 5.2 werden die clientseitigen technischen Vorrausetzungen, die für die Nutzung des Systems gewährleistet sein müssen, aufgelistet. In den folgenden beiden Abschnitten wird auf die Architektur und die implementierten Komponenten eingegangen. Ein erster Einblick des umgesetzten Systems wird hergestellt.

Den Hauptteil bilden die Abschnitte 5.5 und 5.6. Hier wird detailliert auf die Realisierung der Smart-Client Anwendung und des Servers eingegangen. Innerhalb des Abschnittes 5.5 findet eine Unterteilung der Anforderungen statt. Zu Anfang werden die allgemeinen Anforderungen beschrieben. Anschließend werden die Anforderungen erläutert, die beim Erstellen und beim Abrufen der POI zum Tragen kommen.

5.1 Designentscheidung

Um die im Kapitel 4 geforderten Anforderungen erfüllen zu können, müssen vor der Implementierung bestimmte Designentscheidungen getroffen werden. Auf die wird im Folgenden eingegangen.

- Eine Anwendung, die das „Live“ Annotieren von POI mit Multimediadaten unterstützen soll, muss über das Mobiltelefon das Ansteuern einer internen Kamera und ein Positionsbestimmungsverfahren unterstützen. Es muss somit ein Zugriff auf systemeigene Komponenten erfolgen. Da browserbasierte Anwendungen dies aus technischen und sicherheitsbedingten Gründen nicht unterstützen, wird eine Smart-Client Anwendung realisiert.
- Aufgrund der Anforderung, eine breite Masse an mobilen Endgeräten ansprechen zu können, wird die hier entwickelte Smart-Client Anwendung auf der Basis von J2ME entwickelt. Somit ist die Anwendung plattform- und betriebsystemunabhängig. Darüber hinaus bietet J2ME, über bereitgestellte API einen Zugriff auf alle erforderlichen Schnittstellen, wie beispielsweise UI- und Systemkomponenten, Transportprotokolle und Positionsbestimmungsverfahren. J2ME ist eine Implementierung auf Basis der *Model-View Controller*[45] (MVC) Architektur. Es werden somit Anforderungen der Modifizierbarkeit und Wartbarkeit erfüllt. Durch MVC wird eine klare Struktur geschaffen, die den Erfolg, auf neue Anforderungen schnell zu reagieren, garantiert.

[45] MVC bezeichnet ein Architekturmuster, bei dem ein Softwaresystem in die Einheiten Präsentation, Steuerung und Datenmodel aufgeteilt wird.

- Dem Nutzer soll zur Orientierung eine digitale Karte angezeigt werden. Da es, wie im Kapitel Location-Based Services beschrieben, bereits existierende GIS Server gibt, wird auf eine bestehende Lösung zurückgegriffen, die clientseitig einen GIS-Client abbildet. Dieser greift über einen Proxy-Server auf das Kartenmaterial von Map-Servern unterschiedlicher Provider zu und stellt sie der Anwendung zur Verfügung. Die zur freien Verfügung gestellte API wurde bereits in der Bachelorarbeit von Herrn Buchholz [BA Buchholz 2007] eingesetzt und erfolgreich von Herrn Landspurg[46] weiterentwickelt.
- Die vom Client erstellten Daten sollen einer Community bereitgestellt werden. Da eine lokale Speicherung aller jemals erzeugten Daten auf dem Mobiltelefon und aufgrund dessen das Bereitstellen dieser Daten mittels Peer-to-Peer nicht möglich ist, wird eine Client-Server Anwendung implementiert.
- Der verwendete GIS-Client macht es möglich, auf bereits bestehendes Kartenmaterial im Internet zuzugreifen und diese mit eigenen Inhalten zu verknüpfen. Der resultierende Mashup stellt die Funktionalität zur Manipulation bzw. Interpretation der Daten nicht zur Verfügung. Da diese jedoch für die Filterung der Daten relevant ist, muss sie serverseitig umgesetzt werden.
- Die erstellten Daten müssen, um für jeden Teilnehmer verfügbar zu sein, an einen zentralen Zugangspunkt versendet werden. Damit gängige Mobiltelefone diese Anforderung erfüllen können, wird das HTTP Protokoll verwendet. HTTP ist ein weit verbreitetes Übertragungsprotokoll auf Anwendungsebene für Hypermedia[47]-Informationen. Es ist ein zustandsloses Protokoll, so dass die zweite HTTP-Anfrage nichts von der ersten weiß. Es sollte somit nicht eingesetzt werden, wo mehrere logisch zusammengehörige Informationen über HTTP hin und her transportiert werden. Dies ist hier nicht der Fall, da die erstellten Daten innerhalb eines Paketes an den Server gesendet werden.
- Um online/offline Probleme zu berücksichtigen, werden die abgerufenen Multimediadaten erst auf das Mobiltelefon geladen und dann abgespielt. Ein Speichern für ein erneutes oder späteres Abspielen wird dadurch möglich. Das Real-Time Transport Protokoll wird aufgrund dessen nicht implementiert. Hier werden die Daten paketweise übertragen und direkt nach Erhalt abgespielt[48]. Es muss somit für die

---

[46] Online im Internet unter http://j2memap.8motions.com/.

[47] Mit "Hypermedia" wird die strukturelle Verknüpfung multimedialer Materialien in digitalen Formaten (Text, Grafik, Bild, Ton, Video) über Querverweise ("link") bezeichnet.

[48] In diesem Zusammenhang spricht man auch vom Streaming.

gesamte Abspieldauer eine Internetverbindung vorhanden sein und ein späteres Abspielen der Daten wäre nur über eine erneute Internetverbindung möglich.

- Um auf die im Abschnitt 2.1 angesprochene Usability einzugehen, werden Benutzeroberflächen der High- und Low-Level API (innerhalb der MIDP) verwenden. Oberflächen der Low-Level API bieten die Möglichkeit, individuell den Bildschirm des Mobiltelefons zu gestalten und werden in dieser Arbeit beim Abspielen und Anzeigen des Videos zum Einsatz kommen.

5.2 Technische Vorraussetzungen

Bei der Realisierung des Systems standen zu Testzwecken das Mobiltelefon N93 von Nokia und der externe GPS-Empfänger von Holux zur Verfügung (siehe Abb. 5.1). Um das System nutzen zu können, mussten diese Geräte einige technische Vorrausetzungen erfüllen. Auf Seiten des Mobiltelefons waren es folgende Eigenschaften:

- Die Datenübertragungstechnologien UMTS und WLAN werden unterstützt.
- Um die aktuelle Position zu ermitteln, ist über die vorhandene Bluetooth Schnittselle der Zugriff auf einen externen GPS-Empfänger möglich.
- Multimediadaten können über die interne Kamera erzeugt werden.
- Das Ausführen von nichtherstellerspezifischen Java Programmen wird unterstützt.
- Verfügt bereits in der Grundausstattung über die benötigten optionalen Pakete[49].

Abb. 5.1: Nokia N93 [NOKIA 2007] und Holux GPSlim 236 [HOLUX 2007]

Da in dieser Arbeit ein LBS implementiert wird, der ortsbezogene Daten zur Verfügung stellt, muss eine Möglichkeit zur Positionsbestimmung vorhanden sein. Dies wird über den externen GPS-Empfänger von Holux, der über eine Bluetooth Verbindung vom Mobiltelefon angesteuert wird, realisiert. Es können allerdings auch Mobiltelefone verwendet werden, die einen internen GPS-Empfänger besitzen.

---

[49] Die benötigten optionalen Pakete MMAPI (JSR 135), Location API (JSR 179) und Bluetooth API (JSR 82) mussten aufgrund dessen nicht nachinstalliert werden, was die Größe des MIDlts beeinflusst hätte.

Zusätzlich wird ein zentraler Server benötigt. Dieser Server wird als LBS realisiert und bietet einen Dienst an, der ortsabhängig und anhand einer Kategorie POI filtert und, aufgrund des implementierten GIS-Clients, eine URL zurückliefert. Außerdem wird der Server eine Persistenzschicht für das Speichern der Daten implementieren.

5.3 Architektur

Die Abbildung 5.2 zeigt die Architektur des hier entwickelten Systems. Die aktuelle Position des Nutzers wird aufgrund der verfügbaren Testgeräte über einen externen GPS-Empfänger ermittelt.

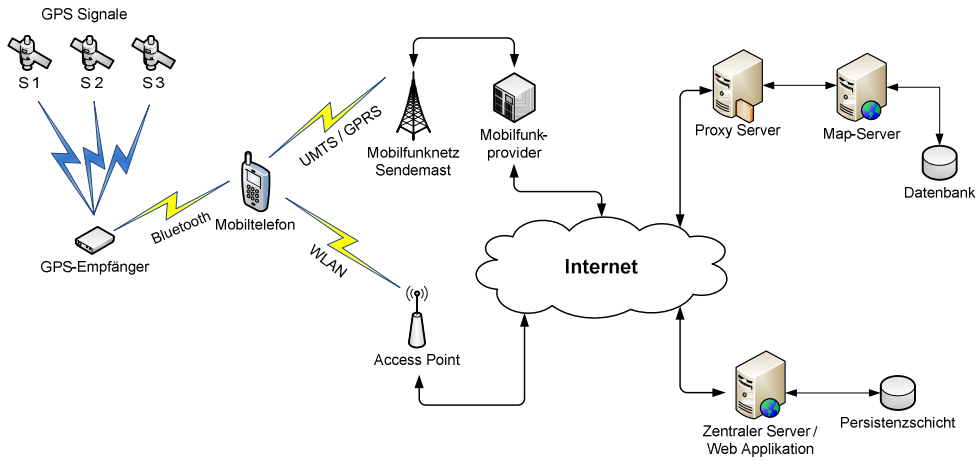


Abb. 5.2: Systemarchitektur

Über das Mobiltelefon können neue POI erstellt werden. Damit sie der Community zur Verfügung stehen, werden diese Daten an einen zentralen Server gesendet. Hierfür stehen die Datenübertragungstechniken UMTS, GPRS und WLAN bereit. Der Zugang kann somit entweder über einen Mobilfunkprovider oder, wenn das Mobiltelefon eine WLAN-Schnittstelle besitzt, über das WLAN hergestellt werden. Wegen der schnelleren Datenübertragung gegenüber GPRS (Kapitel 2.3), sollte der Datenverkehr des Mobiltelefons vor Ort mittels UMTS Technologie durchgeführt werden. Der zentrale Server besitzt eine direkte Verbindung ins Internet und fungiert als LBS. Neu erstellte POI werden auf dem Server persistent gehalten und können über einen Dienst ortsbezogen abgerufen werden. Zur Orientierung wird auf dem Display des Mobiltelefons eine Karte angezeigt. Dies wird durch einen clientseitigen Zugriff auf einen Proxy-Server realisiert. Der Proxy-Server fungiert, wie im Kapitel LBS beschrieben, als Kommunikationsschnittstelle und greift für das Laden von digitalen Karten auf externe Map-Server unterschiedlicher Provider zu.

5.4 Verwendete Komponenten

Die hier entwickelte Anwendung soll auf Veränderungen innerhalb des Systems schnell reagieren und die Möglichkeit der Wiederverwendbarkeit unterstützen. Damit dies möglich ist, wurden die einzelnen Anforderungen innerhalb der Systeme durch einzelne Komponenten realisiert. Die Abbildung 5.3 zeigt die realisierten Komponenten, die im weiteren Verlauf einzeln kurz beschrieben werden. Eine detaillierte Beschreibung der verwendeten Klassen und, die aus den Anforderungen resultierende Implementierung werden in den folgenden Abschnitten näher erläutert.

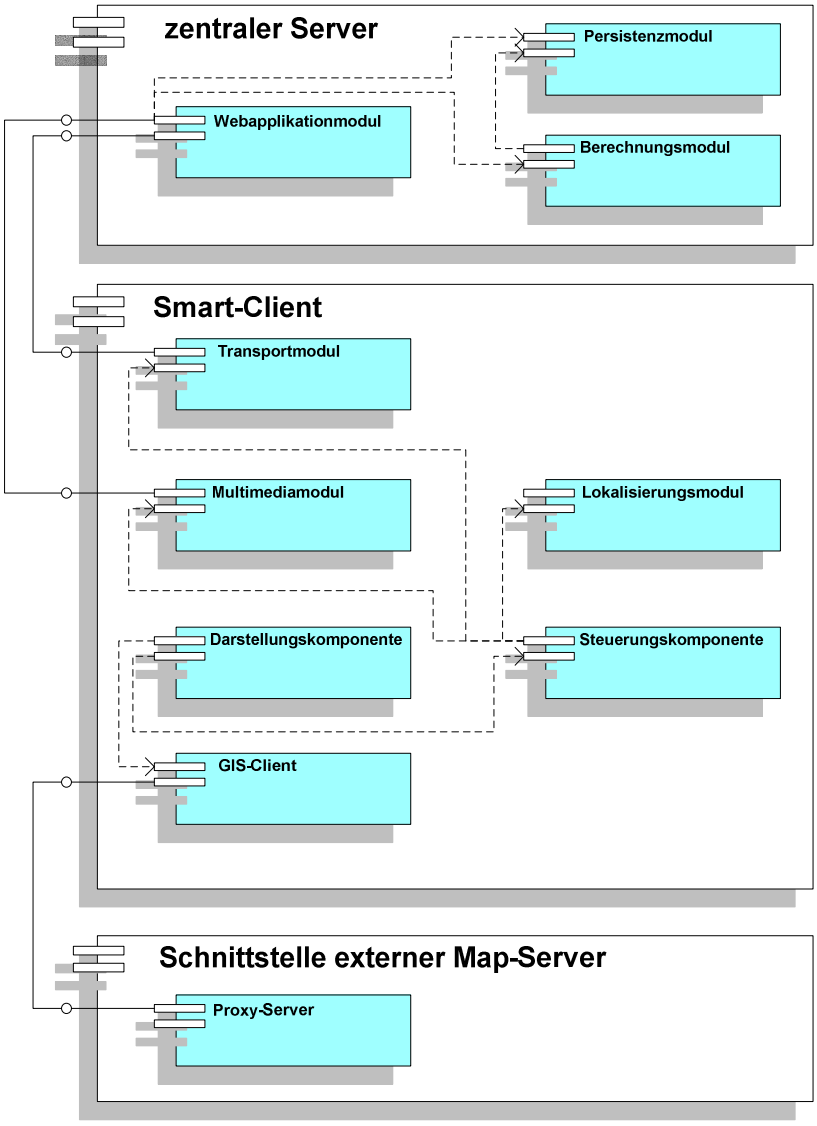


Abb. 5.3: Komponentendarstellung des Systems

Zentraler Server: Webapplikationmodul

Das Webapplikationmodul ist der zentrale Zugangspunkt, um über das Mobiltelefon Daten eines POI zu versenden oder diese abzurufen. Die dabei entstehenden clientseitigen Anfragen werden über das HTTP Protokoll entgegen genommen. Erhält der Server einen

neuen POI, bekommt der Client bei erfolgreicher Übermittlung eine Erfolgsmeldung. Möchte hingegen der Client POI nach der Kategorie und dem Umkreis gefiltert abrufen, wird eine eindeutige URL zurückgeliefert, die auf eine entsprechende xml-Datei verweist. Diese URL benötigt der GIS-Client, um automatisch POI abzurufen und diese auf einer digitalen Karte auf dem Mobiltelefon darzustellen.

Zentraler Server: Persistenzmodul

Das Persistenzmodul ist zum einen für die Speicherung eines neuen POI verantwortlich.
Da der GIS-Client automatisch anhand einer xml-Datei POI abrufen kann, wird die vorgegebene Struktur beim Speichern der Daten eingehalten. Der genaue Aufbau ist in der Abbildung 5.14 zu sehen. Ein POI wird in diesem Format als *item* dargestellt.
Trifft ein neuer POI ein, werden die Textinformationen abhängig von der Kategorie in die entsprechende xml-Datei abgespeichert. Die Multimediadaten werden dabei gesondert behandelt. Eine Referenz, die innerhalb der xml-Datei über einen *Tag* hergestellt wird, verbindet jeden POI mit dem zugehörigen Video. Es gibt somit für jede Kategorie eine xml-Datei, die alle jemals entgegengenommenen POI beinhaltet.
Zum anderen dient das Persistenzmodul zum Speichern von speziellen xml-Dateien. Möchte ein Client POI abrufen, wird eine xml-Datei, die aufgrund einer clientseitig übergebenen ID eindeutig ist und durch das Berechnungsmodul die entsprechenden POI beinhaltet, erzeugt. Das Webapplikationmodul liefert in diesem Fall einen URL zurück, die auf die xml-Datei zeigt und für das Abrufen der POI vom GIS-Client verwendet wird.

Zentraler Server: Berechnungsmodul

Das Berechnungsmodul ermittelt passende POI. Hierfür wird die Distanz zwischen den gespeicherten POI und der aktuellen Position des Mobiltelefons berechnet und mit der vom Nutzer ausgewählten Distanz verglichen. Befinden sich POI innerhalb dieser Distanz, werden diese wie im Persistenzmodul beschrieben, in eine für den Client eindeutigen separaten xml-Datei gespeichert.

Smart-Client: Transportmodul

Das Transportmodul ist für das Versenden der erstellten POI Daten und für das Anfordern der eindeutigen URL, die auf die im Persistenzmodul erstellte xml-Datei zeigt, verantwortlich. Um Anfragen übertragen zu können, wird eine HTTP Verbindung zum zentralen Server hergestellt. Der Auslöser für das Versenden der Anfragen ist der Nutzer, der mittels Darstellungskomponente die Aktion anstößt. Damit die Komponente während der Übertragung nicht blockiert, wird für das Versenden ein eigener Thread verwendet.

Dem Nutzer wird das Ausführen der Übertragung beim Senden der POI Daten mittels Fortschrittsanzeige auf dem Display angezeigt (siehe Screenshot 12).

Smart-Client: Lokalisierungsmodul

Das Lokalisierungsmodul ermittelt den aktuellen Standort des Anwenders und stellt diesen der Anwendung zur weiteren Verarbeitung zur Verfügung. Das Modul nutzt dabei einen externen GPS-Empfänger, der den aktuellen Standort mit Hilfe von Satellitensignalen berechnen kann. Um die standardisierten Schnittstellen für das Ermitteln der Standortkoordinaten ansprechen zu können, nutzt das Modul externe Klassen, die alle durch das optionale Location API bereitgestellt werden.

Damit bei der Standortermittlung die Darstellungskomponente nicht blockiert wird, wird dieser Vorgang in einem eigenen Thread ausgeführt. Dieses Modul legt den wesentlichen Grundstein für die hier entwickelte Smart-Client Anwendung und macht einen LBS erst möglich.

Smart-Client: Multimediamodul

Das Multimediamodul ist eines der Hauptmodule dieser Anwendung. Über dieses Modul können Videos aufgenommen und abgespielt werden. Hierfür wird die interne Kamera des Mobiltelefons verwendet, die über das optionale Mobile Media API angesteuert werden kann. Die API stellt dafür die Klassen *Player*, *VideoControl* und *RecordControl* bereit. Sowohl das Aufnehmen als auch das Abspielen wird in einem eigenen Thread ausgeführt. Die verwendete Darstellungskomponente wurde so implementiert, dass der gesamte Bildschirm für die Kameraansicht genutzt wird (siehe Screenshots 8, 21).

Smart-Client: Steuerungskomponente

Die Steuerungskomponente beinhaltet alle Klassen, die auf die Benutzereingaben der Darstellungskomponenten reagieren und anhand der Eingaben bestimmte Aktionen ausführen[50]. Sie sind innerhalb der MVC Architektur auf der Seite der Steuerung (Controller) einzuordnen. Die Aufgaben sind unter anderem

- Positionsbestimmung starten.
- Datenübertragung via HTTP ausführen.
- Darstellungskomponente wechseln.
- Steuerung des kompletten Workflows der Smart-Client Anwendung.

[50] Das Auslösen durch den Nutzer und das gleichzeitige Reagieren wird durch das *Event-Pattern* beschrieben.

Smart-Client: Darstellungskomponente

Die Darstellungskomponente beinhaltet alle Klassen, die für die grafische Benutzeroberfläche der Anwendung verantwortlich sind. Für das Anzeigen werden unterschiedliche Komponenten der High- und Low-Level API verwendet, die wiederum je nach Anforderung einzelne Ein- und Ausgabekomponenten beinhalten. Eine detaillierte Beschreibung über die verwendeten grafischen Benutzeroberflächen folgt im Abschnitt 5.5.2. Die Darstellungskomponenten sind innerhalb der MVC Architektur auf der Seite der Präsentation (View) einzuordnen.

Smart-Client: GIS-Client

Der GIS-Client ist eine externe Komponente. Es handelt sich hierbei um eine API, die die Funktionalität besitzt, vom Mobiltelefon aus auf unterschiedliche Map-Server zu zugreifen. Satellitenbilder und Straßenkarten werden geladen und können auf dem Display des Mobiltelefons angezeigt werden. Außerdem ist es über den GIS-Client möglich, eigene POI hinzuzufügen und diese auf der Karte anzeigen zu lassen. Hierfür ist ein Zugriff auf eine xml-Datei, die über das Transportmodul vom Server abgerufen wird, nötig. Der Aufbau der xml-Datei ist in der Abbildung 5.14 zu sehen. Die Möglichkeit, digitale Karten und selbst erstellte POI auf dem Display des Mobiltelefons anzeigen zu lassen, wurde in dem hier entwickelten System genutzt.

Schnittstelle externer Map-Server: Proxy-Server

Der Proxy-Server dient als Kommunikationsschnittstelle zwischen dem GIS-Client und den Map-Servern unterschiedlicher Provider. Dieser Proxy-Server ist für private Zwecke frei verfügbar und unterliegt dem Herrn Landspurg, der den GIS-Client entwickelt hat. Der GIS-Client greift somit nicht direkt auf die Map-Server, sondern, wie im Abschnitt 2.4 „Geoinformationssystem“ beschrieben, über einen Proxy-Server zu. Dieser koordiniert das Austauschen von Nachrichten und liefert die clientseitig angeforderten Straßenkarten oder Satellitenbilder.

5.5 Realisierung der Smart-Client Anwendung

Im Folgenden wird die Implementierung der Clientanwendung detailliert beschrieben. Unter 5.5.1 findet eine genaue Definition des funktionalen Umfangs statt, um einen Überblick der umgesetzten Funktionalität herzustellen. Der Abschnitt 5.5.2 beschreibt die Umsetzung der generellen Anforderungen, die unabhängig von den beiden Aufgabenstellungen (POI erstellen/abrufen) umgesetzt werden mussten. Es wird dabei auf die grafischen Benutzeroberflächen und die Positionsbestimmung eingegangen. In den Abschnitten 5.5.3 und 5.5.4 werden die speziellen Implementierungen, die beim Erstellen und Abrufen von POI realisiert wurden, erläutert.

5.5.1 Funktionaler Umfang

Da aus zeitlichen Gründen keine vollständige Umsetzung der im Analysekapitel geforderten Anforderungen erfolgen kann, wurde der Client auf folgende Funktionalitäten reduziert:

Der Nutzer kann vor Ort eigene POI erstellen und diesen an einen zentralen Server senden. Um beispielhaft zu zeigen, dass das Erzeugen von Multimediadaten möglich ist, wird die Aufnahme genau eines Videos pro POI realisiert. Die dabei entstehenden Daten erhalten über ein Positionsbestimmungsverfahren einen direkten Bezug zur realen Welt.

Nutzern wird es möglich sein, die auf dem Server gespeicherten POI ortsabhängig abzurufen. Damit beim Abrufen nicht alle in der Nähe befindenden POI angezeigt werden, wird ein Filter implementiert. Dieser Filter beschränkt sich auf eine fest definiert Anzahl folgender Kategorien: Restaurant, Landmark, Bar, Disco, Others. Außerdem findet serverseitig eine Umkreisberechnung statt. Zur Orientierung wird dem Nutzer beim Abrufen eine digitale Karte auf dem Display des Mobiltelefons angezeigt. Sie beinhaltet die eigene Position und die der abgerufenen POI. Aus der Kartendarstellung heraus wird der Zugriff auf die erzeugten Textinformationen möglich sein. Sind diese Informationen nicht ausreichend, ist das Abspielen des Videos eines jeden POI ausführbar.

5.5.2 Umsetzung genereller Anforderungen

Bedingt durch die Aufgabenstellungen, muss das MIDlet Ein- und Ausgabedialoge anbieten und die aktuelle Position des Nutzers ermitteln. Diese Anforderungen sind sowohl beim Erstellen als auch beim Abrufen von POI zwingend erforderlich und werden in den folgenden Abschnitten näher beschrieben.

Grafische Benutzeroberfläche

Die Smart-Client Anwendung interagiert über grafische Benutzungsschnittstellen mit dem Benutzer. Um über die Anwendung auf den Bildschirm des Mobiltelefons zugreifen zu können, bietet J2ME die Klasse *Display* an. Jedem MIDlet ist genau ein Objekt der Klasse *Display* zugeordnet, das einen virtuellen[51] Bildschirm repräsentiert. Es verwaltet die vom MIDlet anzuzeigenden Objekte, die je nach Anforderung aus dem High-Level oder der Low-Level API stammen können (siehe Abb. 5.4).

Auf Grund der geringen Größe des *Liquid Crystal Display* (LCD) von mobilen Endgeräten, kann jeweils immer nur ein Objekt auf dem Display angezeigt werden. Über das *Display* Objekt wird gesteuert, welches darstellbare Objekt aktuell auf dem Display angezeigt werden soll.

Abbildung 5.4 zeigt die im *javax.microedition.lcdui*[52] Package verfügbare Klassen, die alle von der Oberklasse *Displayable* abgeleitet und auf dem Display des Mobiltelefons anzeigbar sind.

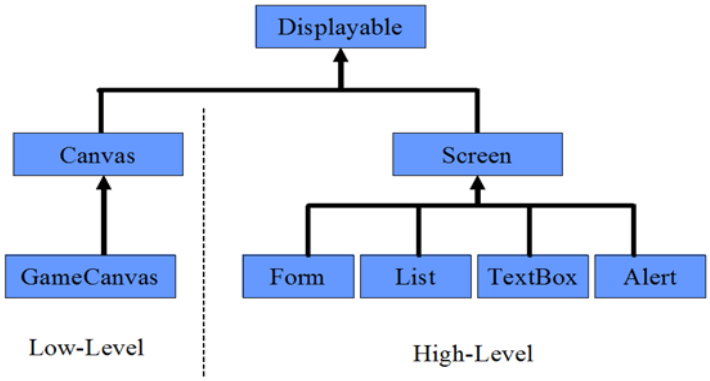


Abb. 5.4: Grafische Benutzeroberflächen [DevX]

Dsplayable ist eine abstrakte Klasse, die ein paar minimale Eigenschaften eines anzeigbaren Bildschirms definieren. Dazu gehören ein Titel, ein Ticker (Schriftlaufband), Kommandos und entsprechende Listener, die auf Eingaben des Benutzers reagieren.

Für die Steuerung dieser Anwendung wurden aus dem High-Level API Objekte der Klasse *Form* verwendet. Allgemein sind diese Klassen einfach in der Handhabung und verwenden Komponenten die, wie in der Abbildung 5.5 zu sehen, vorgefertigte Fähigkeiten besitzen.

---

[51] Mit „virtuell“ ist gemeint, dass dieser Bildschirm nicht unbedingt sichtbar ist, denn es können andere MIDLet oder Programme geben, die um die Anzeige konkurrieren.

[52] lcdui ist eine Abkürzung und steht für „liquid crystal display user interface“.

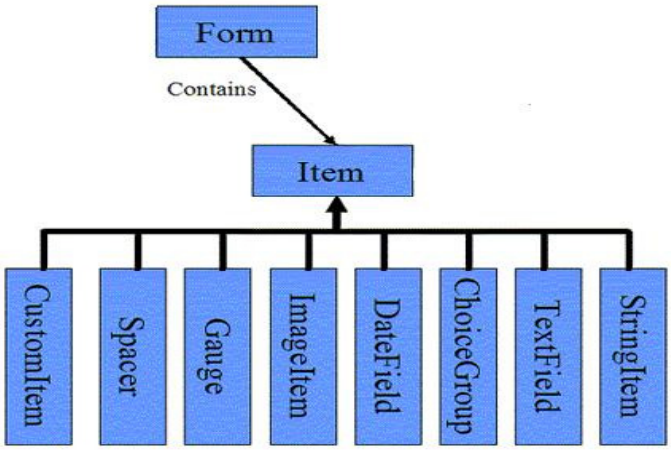


Abb. 5.5: Komponenten der grafischen Benutzeroberfläche Form [DevX]

Als Komponenten stehen den *Form* Objekten u.a. Textfelder, Datumsfelder, Auswahlfelder, Laufticker oder Image Objekte zur Verfügung. In dieser Anwendung werden *Form* Objekte für Eingabe- und Ausgabeformulare verwendet (Screenshot 9).
Ebenfalls in der High-Level API befindet sich die Klasse *Alert*. Ein *Alert* Objekt wird hier verwendet, um den Benutzer auf Fehler oder besondere Umstände hinzuweisen (Screenshots 24, 25, 27).
Der hohe Abstraktionsgrand der zur Verfügung stehenden Komponenten in dem High-Lever API ist beim Anzeigen der Kamerakomponente allerdings hinderlich. Die Größe des Anzeigebildschirmes beim Aufnehmen oder Abspielen eines Videos kann darüber nicht verändert werden und ist Standardmäßig sehr klein. Um pixelgenau zeichnen zu können und den Bildschirm individuell anzupassen, bietet J2ME die Low-Level API an. Innerhalb dieser API gibt es die Klasse *Canvas*. Diese erlaubt die volle Kontrolle über den Bildschirm und ermöglicht einen direkt Zugriff auf Tatstatur oder Zeigergeräte. Die Aufnahme und Wiedergabe von multimedialen Inhalten wurde aufgrund dessen über ein Objekt der Klasse *Canvas* realisiert, die den Anzeigebildschirm optimal ausnutzt (Sreenshots 8, 21).
Jedem Objekt der von *Displayable* abgeleiteten Klassen können mittels der *Command* Klasse Kommandos hinzugefügt werden. Diese sind für die Interaktionen mit dem Benutzer relevant und werden Tastaturfelder[53] auf dem Mobiltelefon zugeordnet. Bei Betätigung des Soft-Keys über den Benutzer wird ein Listener ausgelöst, wodurch die Klasse, die die *CommandListener* Klasse implementiert, individuell reagieren kann. Dadurch können Oberflächen gewechselt oder Menüstrukturen verändert werden.

[53] Tastaturfelder werden auch Soft-Keys genannt.

Das Anzeigen der genannten *Displayable* Objekte und das Reagieren auf Tastatureingaben des Benutzers bilden die Grundlage der hier implementierten Anwendung. Sie steuern den gesamten Ablauf der Anwendung und stehen im ständigen Austausch mit dem Benutzer. Abbildung 5.6 zeigt die für die Steuerung implementierten visuellen Bildschirme, die alle aus der High-Level API stammen und Objekte der Klasse *Form* sind. Die Klasse *MediaPOI* ist das MIDlet.

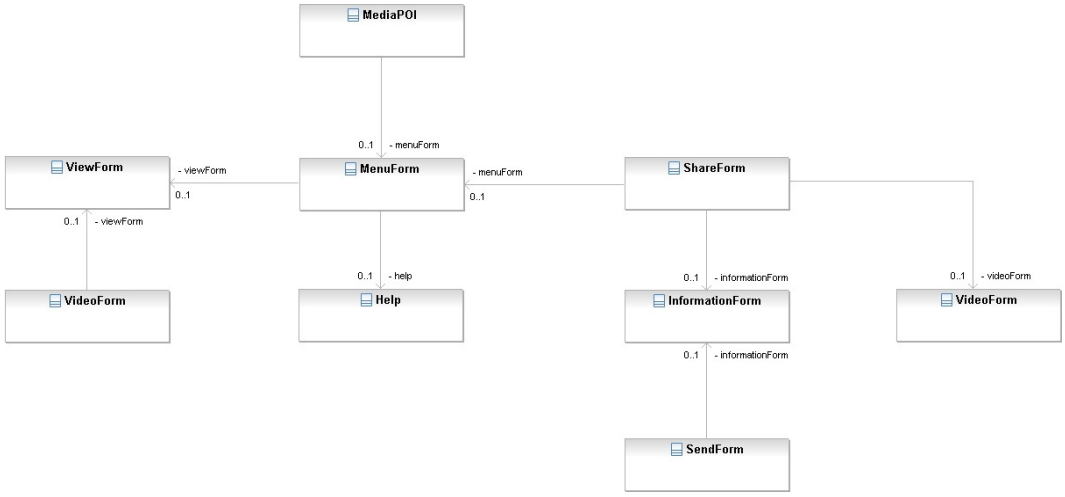


Abb. 5.6: MIDlet und grafische Benutzeroberflächen

Beim Starten des MIDlets *MediaPOI* wird die *startApp()* Methode ausgeführt, die das Hauptmenü *MenuForm* (Screenshot 4) lädt und auf dem Display des Mobiltelefons anzeigt. Von dort aus können eigene POI erstellt (Abb. 5.6 „ShareForm“) oder POI gefiltert angezeigt (Abb. 5.6 „ViewForm“) werden. Es ist, durch das Navigieren der zuvor angezeigten Benutzeroberfläche zu jeder Zeit möglich, wieder ins Hauptmenü zu gelangen. Unterstützend wird im Hauptmenü eine Hilfeseite angeboten, die die Funktionsweise der Anwendung erklärt (Screenshot 23).

Positionsbestimmung

Die Positionsbestimmung wird vor dem eigentlichen Ausführen der jeweiligen Aufgabenstellung vom MIDlet angestoßen. Beim Erstellen der Informationen eines POI wird die aktuelle Position verwendet, damit die Daten einen direkten Bezug zur realen Welt erhalten. Dies ist erforderlich, da Informationen nur ortsabhängig abgerufen werden können, wenn diese eine eindeutige Zuordnung zum existierenden Aufenthaltsort besitzen.
Möchte der Nutzer POI abrufen, wird die aktuelle Position verwendet, damit der Server eine Umkreisberechnung Ausführen kann und die eigene Position zur Orientierung auf der digitalen Karte angezeigt wird.

Abb. 5.7: Klassendiagramm des Lokalisierungsmodul

In der Abbildung 5.7 ist das Klassendiagramm des Lokalisierungsmoduls zu sehen. Die Klasse *GPSDetection* ist die Hauptklasse. Diese stößt zum einen das Ermitteln der GPS Koordinaten (*GPSDetectionThread*) und zum anderen das Anzeigen einer Fortschrittanzeige (*ProgressDisplay*) auf dem Display an. Beide Aktionen werden jeweils in einem eigenen Thread ausgeführt, damit die Darstellungskomponente, die bei der Ermittlung angezeigt wird, nicht blockiert.

Innerhalb der *GPSDetectionThread* Klasse wird für die Ermittlung der GPS Koordinaten die folgenden Schnittstellen des Location API verwendet:

- *Criteria*: Bietet die Möglichkeit Kriterien an den Provider[54] zu stellen. Um die im Abschnitt 2.1 beschriebenen Einschränkungen der mobilen Endgeräte einzubeziehen, wird die *Criteria* Klasse verwendet, um einen durchschnittlichen Stromverbrauch bei der Positionsbestimmung festzulegen.
- *LocationProvider:* Sucht mit Hilfe der Kriterien den besten zur Verfügung stehenden Provider aus. Dieser ist hier der externe GPS-Empfänger, der die Koordinaten anhand von Satellitensignalen berechnet. Damit der Nutzer, bei nicht vorhandener Verbindung vom Mobiltelefon zum GPS-Empfänger, nicht unnötig lange warten muss, wird ein Timeout von 20 sec. festgelegt. Konnte nach 20 sec. keine Koordinaten ermittelt werden, wird der Nutzer durch ein Pop-up darüber informiert (Screenshot 27).
- *QualifiedCoordinates:* Enthält die ermittelten Standortkoordinaten.

Eine Klasse, die das Interface *IGPSDetection* implementiert, kann durch das Interface Pattern individuell auf ein erfolgreiches oder nicht erfolgreiches Ermitteln der GPS Koordinaten mittels der *gpsSuccessful()* und *gpsFailure()* Methoden reagieren.

---

[54] Der Provider ist hier der GPS-Empfänger, es könnte aber auch ein Mobilfunkbetreiber sein. Unterstützt wird dies durch das optionale Location API [J2ME Location API].

Die *ProgressDisplay* Klasse läuft durch die Implementierung eines eigenen Threads parallel zur GPS Ermittlung und aktualisiert ein *Gauge* Objekt, das als Fortschrittanzeige genutzt wird. Diese Fortschrittanzeige zeigt dem Nutzer, dass die Suche der Standortkoordinaten ausgeführt wird (Screenshot 5).
Bei der Ermittlung der GPS Koordinaten über den externen GPS-Empfänger hängt die Genauigkeit des Standortes von der Anzahl der verfügbaren Satellitensignale ab. Sind mehr als drei Satelliten verfügbar, kann der Standpunkt bis auf wenige Meter ermittelt werden. In dieser Arbeit wird vorausgesetzt, dass das verwendete GPS-Gerät genau arbeitet und einen guten Empfang hat. Kann kein Provider gefunden werden, wird eine Fehlermeldung als Pop-up auf dem Display angezeigt. Das Hauptmenü kann nur verlassen werden, wenn eine Positionsbestimmung erfolgreich abgeschlossen wurde (das Anzeigen der Hilfeseite ist hiervon nicht betroffen).

5.5.3 Aufgabenstellung „POI erstellen“

In den kommenden Abschnitten wird die Umsetzung der Aufgabenstellung „POI erstellen“ näher erläutert. Dies betrifft zum einen das Erstellen der Daten und zum anderen das Versenden dieser Daten an einen zentralen Server.

Daten erstellen

Ein Point of Interest besteht in dieser Anwendung aus Positionskoordinaten, die einen Bezug zur realen Welt herstellen, und Daten, die den Standort beschreiben. Damit ein möglichst großer Informationsgehalt über den interessanten Ort vermittelt werden kann, soll die Software „Live“ vor Ort das Annotieren eines Videos ermöglichen und Textinformationen eingetragen werden können.
Die Möglichkeit ein Video zu erzeugen, wird dem Nutzer nach der Positionsbestimmung geboten. Hierfür erfolgt über das Mobile Media API ein Zugriff auf die interne Kamera. Über Systemeigenschaften kann im Vorwege abgefragt werden, ob die interne Kamera das Aufnehmen von Videos unterstützt und wenn ja, welche Formate[55] möglich sind. Die Komponenten, die für das Ansteuern der internen Kamera benötigt werden, sind in der Abbildung 5.8. zu sehen.
Mit der Klasse *Manager*, die nur statische Methode zur Verfügung stellt, wird ein *Player* erzeugt. Der *Player* ist für jegliche Art der Aufnahme und Wiedergabe zuständig. Damit der *Player* weiß, was ausgeführt werden soll, stehen verschiedene Übergabeparameter bei

[55] Je nach Mobiltelefon kann das Video im 3gp oder mpeg Format aufgenommen werden. In dieser Anwendung werden Videos zu Testzwecken im 3gp Format aufgenommen.

der Erzeugung zur Wahl. In dem Fall der Videoaufnahme, wird der String „capture://video“ übergeben.

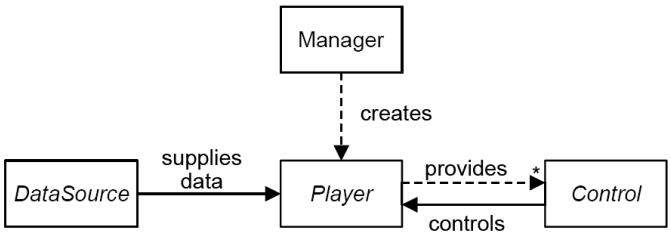


Abb. 5.8: Mobile Media Architektur [NOKIA-MMAPI]

Die Kameraaufnahmen werden wie im Abschnitt 5.5.2 „Grafische Benutzeroberflächen“ beschrieben, über eine Klasse dargestellt, die von der Klasse *Canvas* erbt. Die Eigenschaft wurde genutzt, damit die Kamera den kompletten Bildschirm als Sichtfeld für die Aufnahme verwendet (Screenshots 8, 21). Das Transformieren des über die Kamera aufgenommenen Byte-Stromes in eine visuelle Darstellungsform übernimmt die Klasse *Control*. Sie ist das Bindeglied zwischen dem *Player* und der hier erstellten grafischen Benutzeroberfläche und für das Rendern[56] verantwortlich.
Damit bei der Aufnahme keine Darstellungskomponente blockiert, wird ein separater Thread verwendet. Der während der Aufnahme erzeugte Byte-Strom wird in den Heap-Speicher des Mobiltelefons abgelegt. Um eine *OutOfMemoryException* bei zu großer Aufnahmelänge zu verhindern, muss im Vorwege der zur Verfügung stehende freie Speicher ermittelt werden. Dies bietet die Klasse *Runtime.* Mit ihr kann der freie Speicher ausgelesen werden, die der Java Virtual Machine für die Smart-Client Anwendung zur Verfügung steht. Speziell beim Aufnehmen, muss diese Angabe der Klasse *RecordControl* übermittelt werden. Diese ist für die Aufnahme eines Videos verantwortlich.

Nachdem das Video erstellt wurde, muss dem interessanten Ort ein Name, eine Beschreibung und eine Kategorie als weitere Information hinzugefügt werden. Dies wird über eine Eingabe- Ausgabekomponente, die von der Klasse *Form* erbt, realisiert (Screenshot 9). Die Kategorie dient dem Zweck, dass beim Abrufen der POI eine Filtermöglichkeit vorhanden sein soll. Dadurch werden nicht alle POI, die sich in der näheren Umgebung befinden geladen, sonder nur diejenigen, die von Interesse sind.

---

[56] Rendern in diesem Zusammenhang bedeutet, dass der vom Player erzeugte Byte-Strom in Bilddaten umgewandelt wird.

Zusätzlich wird die benötigte Performance und Downloadzeit beim Laden der Daten reduziert. Zur Auswahl der Kategorie stehen Restaurant, Landmark, Bar, Disco, Others.
Wurden die Daten eingetragen, ist die „Live“ Annotation eines vollständigen POI abgewickelt und steht der Anwendung zum Versenden bereit. Alle Daten können vor dem Senden verändert werden.

Kommunikation mit dem Server „Daten versenden“

Die zuvor erstellten Daten müssen, damit sie von anderen Teilnehmern der Community abrufbar sind, an einen zentralen Server gesendet werden. Der Ablauf, der Clientseitig implementiert wurde, ist in der Abbildung 5.9 als Sequenzdiagramm zu sehen.

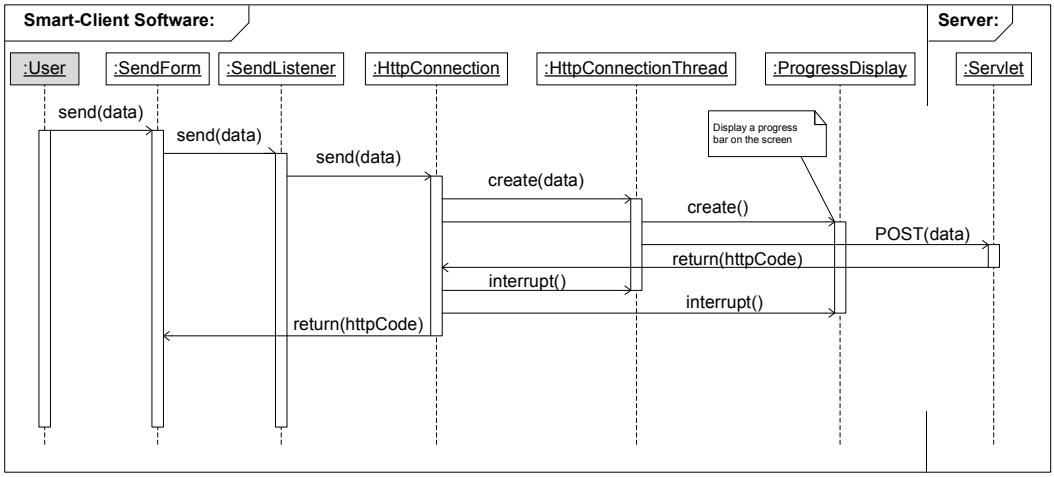


Abb. 5.9 Kommunikation zwischen Client und Server

Der Auslöser ist der Nutzer, der über die Darstellungskomponente *SendForm* (Screenshot 10) die Aktion „send(date)“ anstößt. Durch die Betätigung des Soft-Keys wird ein Listener (hier der *SendListener*) angestoßen, der die Steuerung für das Versenden der Daten übernimmt.
Das Versenden der Daten wurde in dieser Anwendung über das HTTP Protokoll realisiert. HTTP ist ein weit verbreitetes Übertragungsprotokoll auf Anwendungsebene für Hypermedia[57]-Informationen. Um die erstellten Daten an den zentralen Server zu senden wird ein Paket verwendet, das in der Abbildung 5.10 zu sehen ist.
Die Reihenfolge ist, damit sie beim Server korrekt zugeordnet werden kann, fest einzuhalten. Die *Type* Angabe zeigt dem Server, ob es sich bei dem Paket um einen neu

[57] Mit "Hypermedia" wird die strukturelle Verknüpfung multimedialer Materialien in digitalen Formaten (Text, Grafik, Bild, Ton, Video) über Querverweise ("link") bezeichnet.

erstellten POI oder um das Anfordern einer URL, die auf eine vom Persistenzmodul erstellte xml-Datei zeigt, handelt.

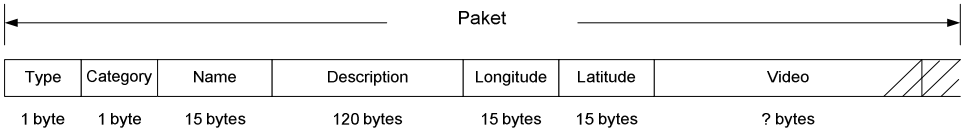


Abb. 5.10: Datenpaket beim Senden eines neuen POI

Da die Aufnahmedauer und somit die Größe des Videos abhängig vom verfügbaren Speicherplatz ist, wurden das Video ans Ende des Datenpaketes gestellt. Nachdem der Listener *SendListener* das Senden der Daten an die Klasse *HttpConnection* weitergeleitet hat, werden zwei parallele Prozesse angestoßen. Zum einen wird über die Klasse *ProgressDisplay* während die Daten zum Server gesendet werden, eine Fortschrittanzeige auf dem Display des Mobiltelefons an (Screenshot 12). Dies zeigt dem Nutzer, dass der Vorgang „Sending data“ gerade läuft. Zum anderen erzeugt *HttpConnection* einen neuen Prozess (Klasse *HttpConnectionThread*), der das Senden der Daten übernimmt. Dieser Ablauf ist in der Abbildung 5.9 zu sehen. Die Daten werden über die HTTP POST Methode, die durch das MIDP unterstützt wird, versendet. Durch die Verwendung eines eigenen Threads wird keine Komponente während der Zeit des Sendens blockiert und kann, wie in diesem Fall, für das Anzeigen einer Fortschrittanzeige verwendet werden. Um dem Server Informationen über das gesendete Paket zu liefern, wird zum einen der Content-Type und zum anderen die Content-Length[58] gesetzt. Der Content-Type sagt aus, wie das nachfolgende Paket zu interpretieren ist. Da die Interpretation der Daten serverseitig durchgeführt und diese in Bytes versendet werden, wurde der Standard MIME-Type[59] verwendet. HTTP ist ein Anfrage-Antwort Protokoll. Nach dem die Daten an den Server gesendet wurden, bekommt der Client eine Antwort. Durch die informativen Codes, die durch das HTTP Protokoll geliefert werden, kann eine Aussage über das erfolgreiche Versenden getroffen werden. Der Client wertet den Code aus und zeigt die entsprechende Meldung auf dem Display des Mobiltelefons an (Screenshots 13, 26). Nachdem die Antwort vom Server im Client empfangen wurde, werden die zuvor gestarteten Prozesse gestoppt (siehe Abb. 5.9, Aufruf der *interrupt()* Methode). Nach der Übertragung kann entweder ins Hauptmenü gewechselt oder die Anwendung geschlossen werden.

---

[58] Gibt dem Server Auskunft über die Länge des Paketes.

[59] Der Standard MIME-Type ist application/x-www-form-urlencoded.

5.5.4 Aufgabenstellung „POI abrufen“

POI abzurufen bedeutet für den Nutzer, hilfreiche Informationen über interessante Orte zu erhalten. Außerdem soll zur Orientierung, die aktuelle Position und der POI auf einer digitalen Karte dargestellt werden. In den kommenden Abschnitten wird die Realisierung der dafür erforderlichen Anforderungen erläutert und auf die wesentlichen Problemlösungen eingegangen.

Kommunikation mit dem LBS

Der GIS-Client bietet anhand einer URL die Möglichkeit, eigen erstellte POI automatisch zu Laden. Aufgrund dessen wurde der LBS auf dem zentralen Server so realisiert, dass anstatt der geforderten POI eine URL zurückgeliefert wird. Diese URL zeigt auf eine xml-Datei, die sich auf dem Server befindet und die kategorisierten und mittels Umkreisberechnung in der nähe befindenden POI enthält. Dieser Vorgang wird clientseitig ausgeführt, bevor der GIS-Client anhand der URL die gewünschten POI abrufen kann.
Um allerdings die geforderte URL und somit die xml-Datei Erzeugen zu können, braucht der Server Informationen über den Client.
Die Abbildung 5.11 zeigt das übermittelte Paket, dass der Client bei jeder Anfrage der URL an den Server sendet.

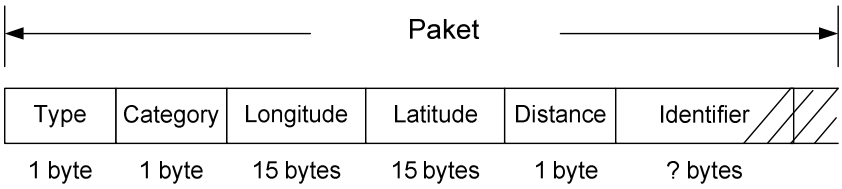


Abb. 5.11: Datenpaket beim der Nutzung des LBS

Die *Type* Angabe zeigt dem Server, dass es sich hierbei um eine Anfrage nach der URL handelt. Außerdem kann der Nutzer vor dem Abrufen die gewünschte Kategorie (*Category*) und die maximale Entfernung (*Distance*) der POI auswählen (Screenshots 16, 17). Der *Identifier* beschreibt einen Key und ist bei jeder Auslieferung der Clientsoftware einmalig. Dieser Key wurde eingeführt, damit die serverseitig erzeugte xml-Datei eindeutig dem Client zugeordnet werden kann. Zusammen mit den aktuellen Positionskoordinaten (*Longitude*, *Latitude*) des Nutzers werden diese Daten an den Server übermittelt, der die benötigte URL zurückliefert. Die Übertragung wird, wie schon beim senden der POI Daten, über das Transportmodul durchgeführt.

Abrufen und Anzeigen der POI

Die Kartendarstellung der Smart-Client Anwendung wurde mit Hilfe einer externen Komponente, der J2MEMap von Herrn Landspurg realisiert. Diese ist für den privaten Gebrauch kostenlos und steht nach der Registrierung auf der Homepage[60] zum Download bereit. Das von J2MEMap zur Verfügung gestellte API bietet die Funktionalität eines GIS-Clients auf dem Mobiltelefon an. Insbesondere können unterschiedliche Map-Server[61] für das Anzeigen von Straßenkarten und Satellitenbilder vom Client aus abgefragt werden. Der Vorgang läuft dabei wie im Kapitel Location-Based Services beschrieben über einen Proxy-Server[62], der als Kommunikationsschnittstelle zwischen dem Client und den Map-Servern fungiert. Das komplette Laden und Anzeigen der Karten auf dem Mobiltelefon wird über eine bereitgestellte Klasse realisiert, die von der Low-Level Klasse *Canvas* erbt. Vor dem Abrufen der Karte, wird der aktuelle Standort des Nutzers als Koordinate mit angegeben. Die Ermittlung des Standortes wurde wie im Abschnitt 5.5.2 „Positionsbestimmung“ beschrieben, über GPS realisiert. Es wird damit gewährleistet, dass der angezeigte Kartenausschnitt der Position des Nutzers entspricht. Außerdem wird die eigene Position mittels rotem Marker auf der Karte dargestellt (Screenshot 18).

J2MEMap bietet jedoch noch weitere Funktionalitäten an, die in der hier entwickelten Smart-Client Anwendung verwendet wurden. Eigen erstellte Informationen können über den GIS-Client geladen werden. Diese Daten werden verwendet, um POI mittels Marker auf der Karte anzuzeigen. Die Daten müssen dafür nicht auf dem Client gespeichert sein, sondern können anhand einer URL geladen werden. Der entfernte Zugriffspunkt ist in diesem System der zentrale Server. Die dafür erforderliche URL wird wie im Abschnitt 5.5.4 „Kommunikation mit dem LBS“ beschrieben, vor dem Abrufen der POI über den Server ermittelt. Damit der GIS-Client anhand der xml-Datei POI erstellen kann, muss diese Datei die in der Abbildung 5.14 dargestellte Notation einhalten. Wurde die xml-Datei vom GIS-Client geladen, wertet dieser die *Tags* wie folgt aus:

- Der *titel* steht für den Namen des POI.
- Die Angabe eines *icon* ist optional. Der Link referenziert ein Bild, das den Standartmarker, der auf der Karte angezeigt wird, ersetzt. (Wird bei Angabe automatisch von der GIS-Client Komponente geladen)
- Die *description* steht für die Beschreibung des POI.

---

[60] Online im Internet unter http://www.8motions.com/.

[61] Die möglichen Map-Server sind u.a. GoogleMap, YahooMap und Ask.com.

[62] Dieser Proxy-Server wird von Herrn Landspurg betreut und steht jedem Anwender zur freien Verfügung.

- Die Werte der *longitude* und *latitude* beschreiben den eindeutigen Standort des POI.
- Das *video Tag* beinhaltet eine URL, dass auf das Video des POI zeigt. Auf diese Eigenschaft wird im Abschnitt „Multimediales Element abspielen“ näher eingegangen.

Anhand dieser *Tags*, erstellt der GIS-Client für jeden POI einen eigenen Marker. Alle Marker werden entsprechend der Koordinaten auf der Karte angezeigt. Um die Detailinformationen eines jeden POI anzuzeigen, muss dieser auf der Karte ausgewählt werden (Screenshot 19). Dies ist der Fall, wenn der Marker über Tastenkombinationen ungefähr in die Mitte des Bildschirms navigiert wird.

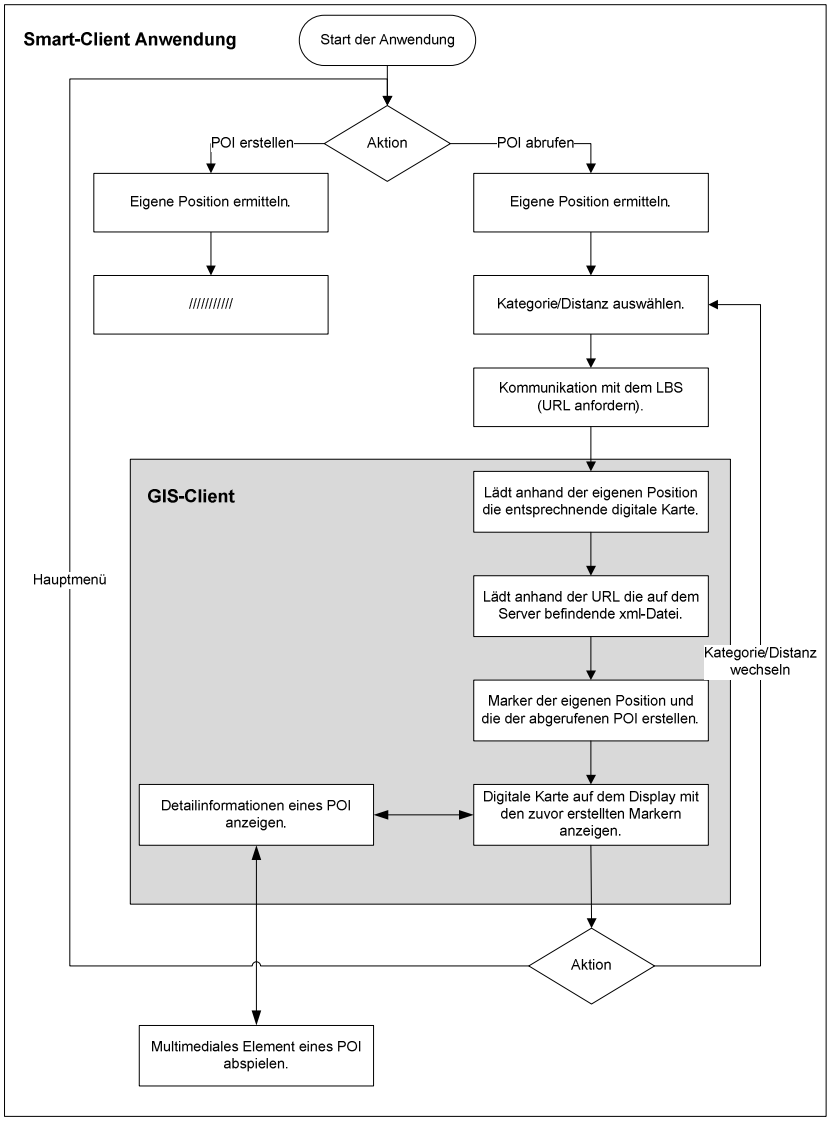


Abb. 5.12: Flussdiagramm „POI abrufen“

Abbildung 5.12 soll den Ablauf, vom Auswählen der Kategorie und der Distanz bis hin zum Abspielen des Videos eines POI veranschaulichen. Insbesondere wird durch den

grauen Rahmen deutlich, das das Abrufen der digitalen Karte und der auf dem Server befindenden xml-Datei vom GIS-Client ausgeführt und kontrolliert wird. Ein erneutes Auswählen der Kategorie oder Distanz wird von der Anwendung unterstützt.
Reichen dem Nutzer die Detailinformationen nicht aus, kann das Video geladen und angezeigt werden (Screenshot 20, 21).

Multimediales Element abspielen

Bei der Erstellung eines POI ist es möglich, ein Video zu erzeugen. Es soll dem Konsumenten nützliche Informationen über den interessanten Ort liefern. Damit beim Abrufen der POI nicht jedes Video mit auf das Mobiltelefon geladen wird, wurde in der bereits erwähnten xml-Datei ein zusätzliches *Tag* eingefügt. Dieses *Tag* beinhaltet eine URL, die das Video eindeutig referenziert.
Die Verwendung der URL wurde durch die Eigenschaften zweier Komponenten bestimmt. Zum einen ist es über den GIS-Client möglich, einen sich auf der Karte selektierten POI anzuklicken (Screenshot 19), wodurch ein Event[63] ausgelöst wird. Dieses Verhalten wird genutzt, um aus der Kartendarstellung heraus eine selbst implementierte grafische Benutzeroberfläche (Screenshot 20) auf dem Display des Mobiltelefons anzuzeigen. Dieser Oberfläche werden die kompletten Daten (die Daten aus der xml-Datei) des selektierten POI für das Abspielen des Videos übergeben. Der Zugriff über die Kartendarstellung auf das Video wurde dadurch erst möglich.
Die andere Eigenschaft lieferte der von dem Mobile Media API bereitgestellte *Player*. Dieser ist, wie bereits im Abschnitt 5.5.3 „Daten erstellen“ erläutert, über eine statische Methode der Klasse *Manager* verfügbar. Der *Player* kann je nach Anforderung unterschiedlich erzeugt werden. In dem Fall des Abspielens von Videodaten, geschieht dies über die URL. Die URL liefert das benötigte Netzwerkprotokoll, hier das HTTP-Protokoll und den Ort, an dem sich die Videodatei befindet (auf dem zentralen Server).
Der *Player* befindet sich ab dem Zeitpunkt des Erzeugens bis hin zum Schließen in unterschiedlichen Zuständen, die in der Abbildung 5.13 zu sehen sind.

[63] Das Auslösen und gleichzeitige Reagieren eines Events, wird durch das *Event-Pattern* beschrieben.

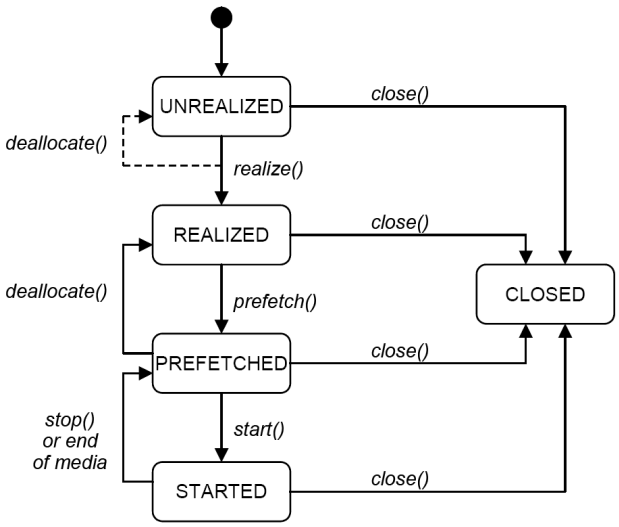


Abb. 5.13: Mögliche Zustände des Media Players [NOKIA-MMAPI]

Nachdem der *Player* mittels URL erstellt wurde, befindet sich dieser in dem Zustand UNREALIZED. Hier werden interne Vorbereitungen des Abspielens geprüft. Anschließend wird über die *prefetch()* Methode, die intern auch die *realized()* Methode ausführen lässt, das komplette Video über HTTP auf das Mobiltelefon geladen. Ein späteres Neuladen des Videos wäre beim Speichern[64] des *Players* nicht erforderlich. Wurde das Video geladen, kann mit *start()* das Video gestartet werden. Für das Anzeigen des Videos wird wie bei der Aufnahme (siehe Abschnitt 5.5.3 „Daten erstellen“) eine von *Canvas* abgeleitete Klasse verwendet, die den vollen Bildschirm für die Wiedergabe des Videos nutzt.

Die Möglichkeit, über Kommandos Zustände des *Players* zu wechseln, wurde beim Abspielen des Videos durch das Bereitstellen einer Pausefunktion mit einbezogen.

Denn, wenn der Zustand PREFETCHED durch den Aufruf von *stop()* erreicht wurde, setzt der nächste *start()* Aufruf die Wiedergabe an der Stelle fort, an der abgebrochen wurde, andernfalls am Anfang (siehe Abb. 5.13). Diese Möglichkeit bietet einen gewissen Komfort und steigert die Zufriedenheit des Nutzers.

Damit auch hier die Darstellungskomponente nicht blockiert, wird das Abspielen des Videos in einem separaten Thread ausgeführt.

Um auf eventuelle Fehler beim Laden oder Wiedergeben des Videos reagieren zu können, werden Ausnahmen abgefangen und Meldungen in Form eines Pop-up auf dem Display angezeigt. Nachdem das Video abgespielt wurde, kann dies erneut geladen oder zur Kartendarstellung gewechselt werden.

---

[64] Das Speichern des Players wurde nicht implementiert.

5.5.5 Screenshots der Smart-Client Anwendung

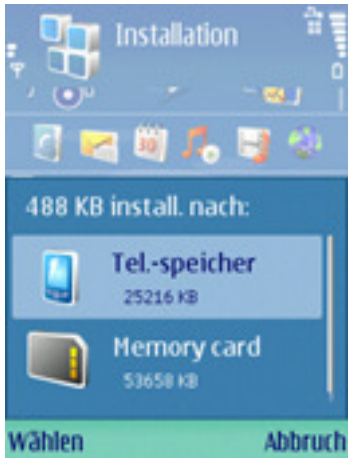
Um einen Eindruck der hier implementierten Smart-Client Anwendung zu erhalten, wird in diesem Abschnitt die Funktionsweise anhand von Screenshots erläutert.

Installation der Smart-Client Anwendung

1

Die Installation einer MIDlet Anwendung muss vom Nutzer bestätigt werden.

2



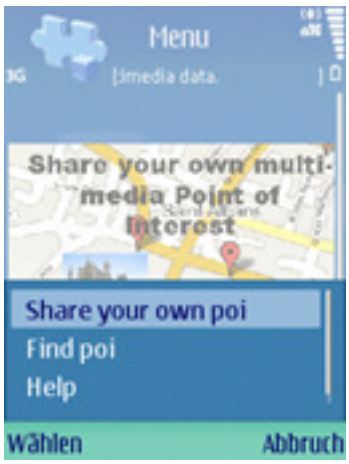
Hat der Nutzer eingewilligt, kann das Installationsverzeichnis gewählt werden.

3

Das MIDlet wird auf dem Mobiltelefon installiert. Dies wurde über ein Verbindungskabel durchgeführt.

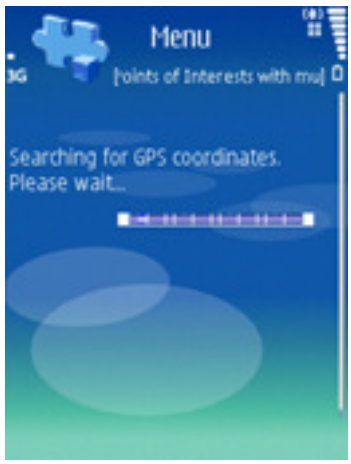
Starten der Anwendung und erstellen eines neuen POI

4



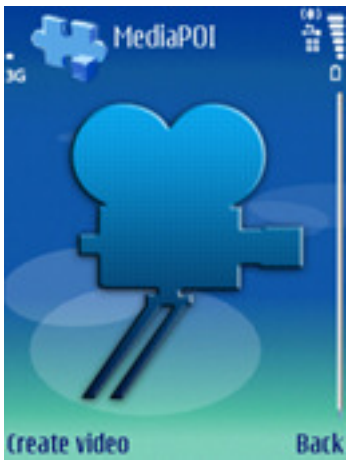
Hauptmenü. Von hier aus können POI erstellt oder abgerufen werden. Eine Beschreibung der Anwendung ist unter *Help* einzusehen.

5



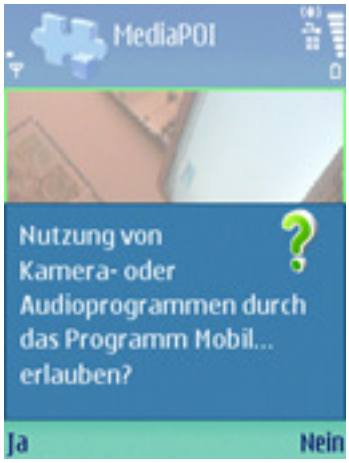
Möchte der Nutzer einen neuen POI erstellen oder abrufen, wird in beiden Fälle die aktuelle Position via. GPS ermittelt.

6



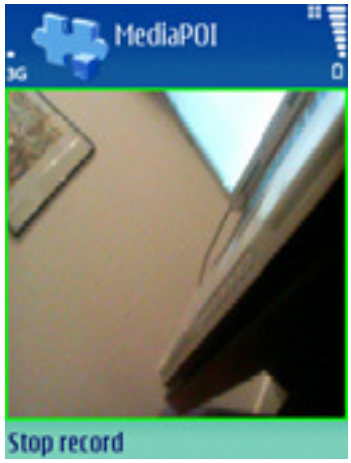
Nachdem die Position ermittelt wurde, muss ein Video erzeugt werden. Ein erneutes Aufnehmen eines Videos ist vor dem Senden möglich.

7



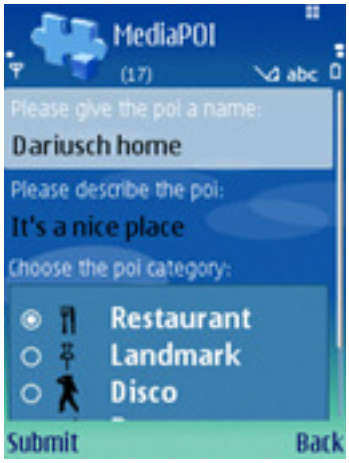
Die Nutzung der internen Kamera über die Smart-Client Anwendung muss, da diese nicht signiert ist, vom Nutzer bestätigt werden.

8



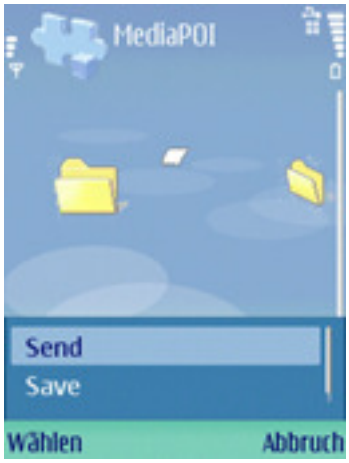
Nachdem der Nutzer den Zugriff auf die Kamera erlaubt hat, kann die Aufnahme gestartet und gestoppt werden.

9



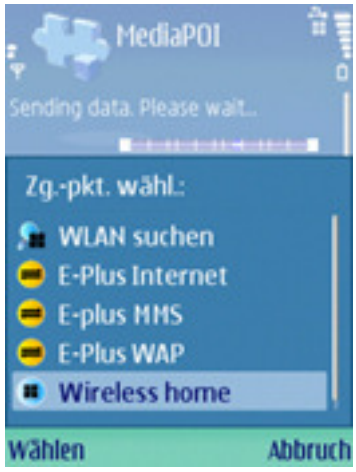
Anschließend muss ein Name, eine Beschreibung und die Kategorie des POI vergeben werden.

10



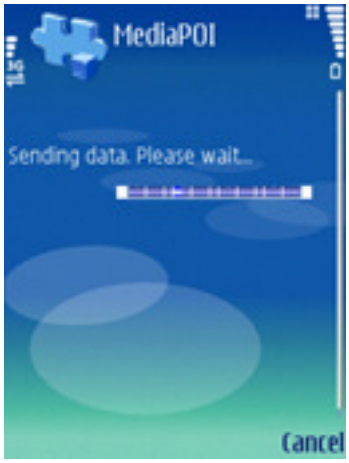
Das Video und die beschreibenden POI Daten können zum jetzigen Zeitpunkt nur an den Server gesendet werden. Ein lokales Speichern der Daten ist nicht möglich.

11



Vor dem Versenden der Daten muss der zur Verfügung stehende Zugangspunkt ausgewählt werden.

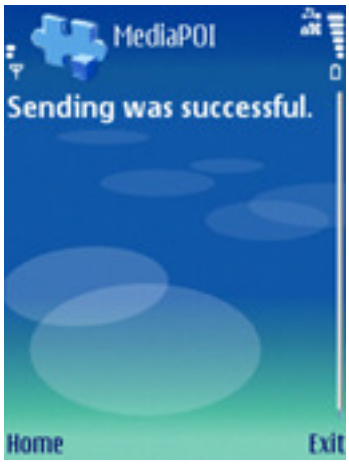
12



Während die Daten übermittelt werden, wird dies dem Nutzer über eine Fortschrittanzeige auf dem Display angezeigt.

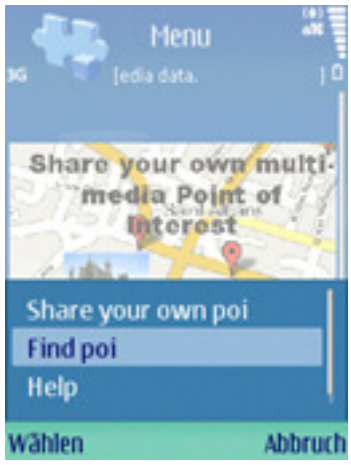
Abrufen interessanter Orte

13



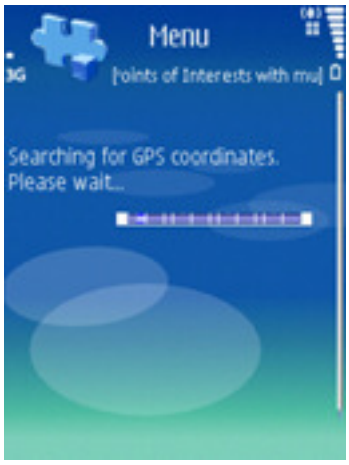
Das erfolgreiche Versenden der Daten wird dem Nutzer über eine Mitteilung auf dem Display angezeigt. Anschließend kann ins Hauptmenü gewechselt oder die Anwendung geschlossen werden.

14



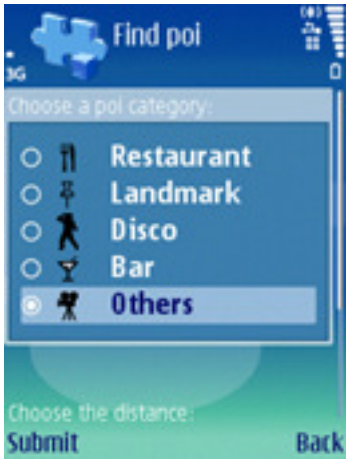
Im Hauptmenü kann unter „Find poi“ auf die in der Community vorhandenen POI zugegriffen werden.

15



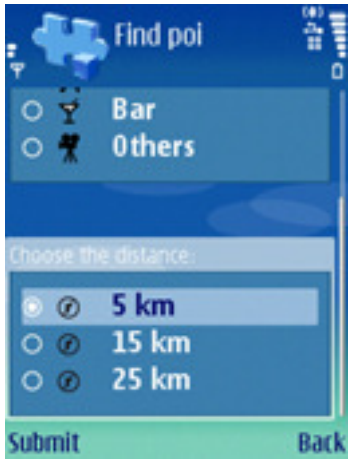
Bevor die aktuelle Position und die der abgerufenen, ortsabhängigen POI auf einer digitalen Karte angezeigt werden können, wird via GPS der Standort des Nutzers ermittelt.

16



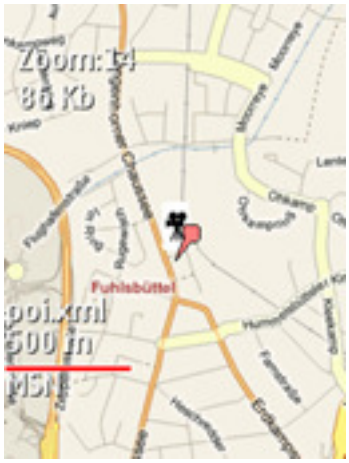
Damit die gewünschten POI auf der Karte angezeigt werden, muss der Nutzer vor dem Abrufen eine Kategorie auswählen. Das entsprechende Icon wird zum Verständnis mit auf der Karte als Marker angezeigt.

17



Außerdem muss der Nutzer die maximale Distanz wählen, damit serverseitig eine Umkreisberechnung durchgeführt werden kann. Anschließend findet der Zugriff auf den LBS statt.

18

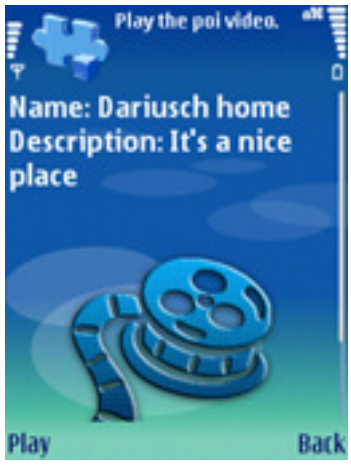


Die Karte zeigt zur Orientierung den aktuellen Standort des Nutzers (das rote Icon) und die der POI (die Kamera) an. Innerhalb der Karte kann der Zoomlevel geändert und navigiert werden.

19

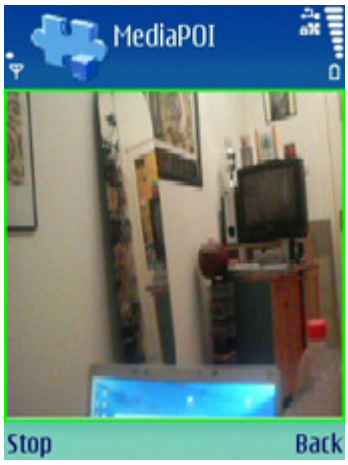
Um Detailinformationen Anzeigen zu lassen, muss der Nutzer mit dem *Soft-Key* das entsprechende Icon zur Mitte der Karte navigieren.

20



Wird ein POI auf der Karte ausgewählt, kann über den Mittelknopf auf eine Anzeige navigiert werden, die die Details anzeigt und das Abspielen des Videos erlaubt.

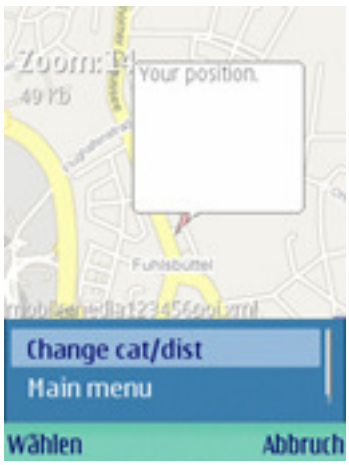
21



Drückt der Nutzer den zuvor sichtbaren *Play-Key*, wird das Video abgespielt. Während des Abspielens, kann das Video in den Zustand Pause versetzt oder zurück zur Detailsicht gewechselt werden.

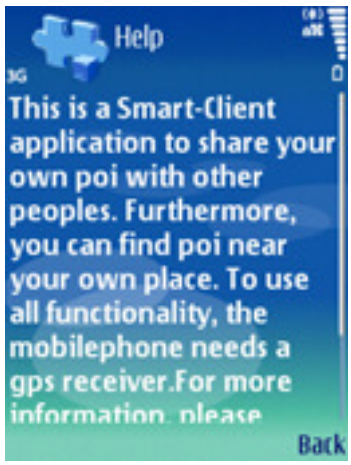
Informations- und Fehlermeldungen

22



Innerhalb der Kartendarstellung kann zurück zum Hauptmenü gewechselt oder ein erneutes Filtern von POI angestoßen werden.

23

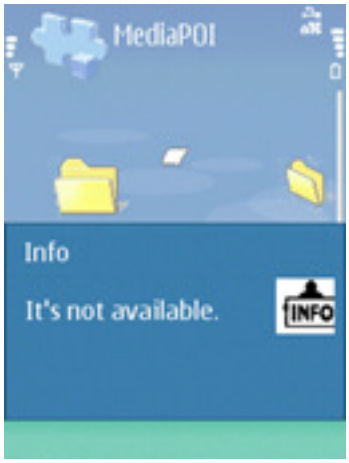


Eine Hilfe kann im Hauptmenü ausgewählt werden. Diese liefert Informationen zur Verwendung der Smart-Client Anwendung.

24

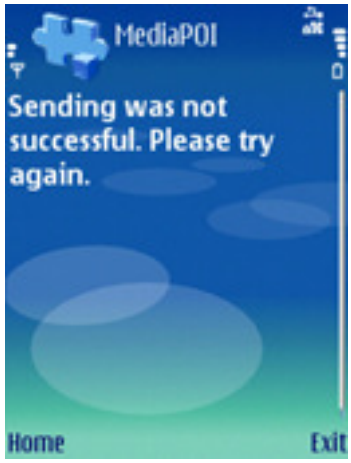
Der Nutzer muss dem POI einen Namen, eine Beschreibung und einer Kategorie zuordnen, falls dem nicht so ist, wird eine Informationsmeldung als Pop-up angezeigt.

25



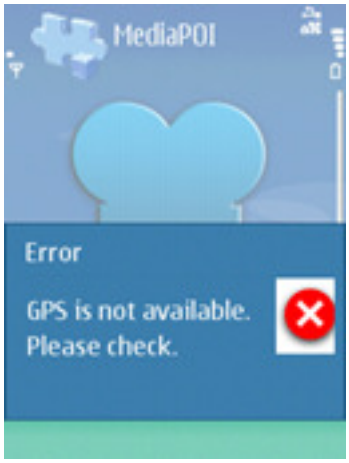
Das lokale Speichern der Daten ist zum jetzigen Zeitpunkt nicht möglich. Der Nutzer wird über ein Pop-up informiert.

26



Kommt es bei der Übertragung der Daten zu einem Fehler, wird eine Meldung auf dem Display angezeigt. Da ein Speichern zum jetzigen Zeitpunkt nicht unterstützt wird, kann zum Hauptmenü gewechselt oder die Anwendung geschlossen werden.

27



Kann kein Provider für die Positionsbestimmung gefunden werden, wird eine Fehlermeldung angezeigt, da diese Information zwingend erforderlich ist.

5.6 Realisierung des Servers

In den folgenden Abschnitten wird auf die serverseitige Implementierung eingegangen. Da der Fokus dieser Arbeit in der Realisierung einer J2ME Anwendung für Mobiltelefone lag, wurden die Anforderungen des Servers reduziert. Abschnitt 5.6.1 gibt einen Überblick der umgesetzten Funktionalitäten wieder. Im Anschluss wird auf das Entgegennehmen der clientseitig gestellten Anfragen eingegangen. Die Umsetzungen, neu erstellte POI abzuspeichern und diese mittels GIS-Client abzurufen, werden in den Abschnitten 5.6.3 und 5.6.4 näher beschrieben.

5.6.1 Funktionaler Umfang

Der Server nimmt clientseitige Anfragen entgegen. Sendet der Client einen neuen POI, wird dieser gespeichert und eine Erfolgsmeldung zurückgeliefert. Außerdem stellt der Server einen Dienst (LBS) zur Verfügung. Mit ihm ist es möglich, eine URL zu erhalten, die auf eine xml-Datei zeigt. Dies wurde realisiert, da der auf dem Client befindende GIS-Client selbst erstellte POI mittels URL automatisch Abrufen kann. Da dieser GIS-Client allerdings keine Funktionalität zur Manipulation bzw. Interpretation der Daten besitzt, wurde serverseitig eine Filterung der POI implementiert. Die gefilterten POI befinden sich in der für den Client speziell erstellten xml-Datei.

5.6.2 Serverseitig Anfragen entgegennehmen

Das Entgegennehmen der clientseitigen Anfragen wird mittels Servlet[65] realisiert. Servlets sind Java Klassen, die innerhalb eines Webcontainers[66] betrieben und verwaltet werden. Damit das Servlet über HTTP angesprochen werden kann, implementiert es die Schnittstelle HttpServlet. Mit ihr können die beiden wichtigsten HTTP-Methoden, die GET- und POST-Methode verarbeitet werden. Da dies eine standardisierte Vorgehensweise und der Zugriff unabhängig von der verwendeten Plattform ist, wird die Anforderung an eine serverseitige Interoperabilität erfüllt.

Die Anfragen des Clients wurden mittels POST realisiert. Folglich wird im Servlet nur die doPost Methode der Superklasse überschrieben. Diese nimmt den gesendeten Byte-Strom entgegen (Request) und leitet ihn entweder für das Speichern eines neuen POI oder für das Abrufen der URL weiter. Das Servlet erkennt den Anfragetyp anhand des ersten

---

[65] Mehr zum Thema Servlets unter [Servlets 2007].

[66] In dieser Arbeit kam der Apache Tomcat Server 6.0 zum Einsatz. Mehr zum Thema Architektur und Anwendung eines Tomcat Servers unter [Tomcat 2006].

empfangenen Bytes (siehe Abb. 5.10 o. Abb. 5.11, erstes Byte). Abschließend wird je nach Anfragetyp eine Antwort (Response) zum Client gesendet.

5.6.3 POI speichern

Das in der Abbildung 5.10 zu sehende Paket wird beim Speichern eines neuen POI im Server entgegengenommen. Um die Daten (ohne das Video) eines POI auf dem Server abzuspeichern, gibt es für jede Kategorie (Restaurant, Landmark, Disco, Bar, Others) genau eine xml-Datei. Diese xml-Datei muss, damit der GIS-Client automatisch POI abrufen kann, eine bestimmte Notation einhalten (siehe Abb. 5.14).

```
<items>
  <item>
    <title>Dariusch home</title>
    <icon>http://xxx.dyndns.org:8080/Datenbank/other.png</icon>
    <description>It's a nice place</description>
    <longitude>10.016393333333</longitude>
    <latitude>53.638253333333</latitude>
    <video>http://xxx.dyndns.org:8080/Datenbank/video0.3gp</video>
  </item>
</items>
```

Abb. 5.14: Notation eines POI

Innerhalb dieser xml-Datei entspricht ein *item* Eintrag genau einem POI. Die empfangenen Daten werden vom Server mittels DOM-Parser[67] in die entsprechenden *Tags* der xml-Datei eingetragen und abgespeichert. Dabei entspricht der empfangene Name dem *title*-Tag, die Beschreibung dem *description*-Tag und die Längen- bzw. Breitengerade dem *longitude*- bzw. *langitude*-Tag.

Das zusätzliche *icon*-Tag wird verwendet, um das Standardbild des Markers zu ändern. Ein Marker kennzeichnet einen POI auf der digitalen Karte (Screenshot 18, schwarze Kamera). Für jede Kategorie wurde ein eigenes Icon entworfen. Das übertragende Video wird gesondert gespeichert[68]. Der Grund dafür ist zum einen, clientseitig ein Lazy-Loading[69] Verhalten zu erzwingen und zum anderen aus der Gegebenheit, dass J2ME das Abspielen eines Videos mittels URL unterstützt. Somit beinhaltet jedes *item*-Tag genau ein *video*-Tag. Es referenziert das Video über eine eindeutige URL. Diese URL wird auf dem Client verwendet, um das Video eines auf der digitalen Karte selektierten POI abzuspielen.

---

[67] Nähere Informationen des DOM-Parsers sind unter [DOM- w3schools] einzusehen.

[68] Sowohl die empfangenden Videos als auch die erzeugten xml-Dateien werden direkt ins Ordnersystem gespeichert.

[69] Lazy-Loading bedeutet in diesem Zusammenhang, dass nur die zum Anzeigen der POI benötigten Daten geladen werden. Das Video eines POI wird erst geladen, wenn es der Nutzer möchte.

Innerhalb der xml-Datei können beliebig viele *item* (POI) Einträge gespeichert werden. Somit beinhaltet eine xml-Datei alle jemals gesendeten POI einer Kategorie. Alle 5 xml-Dateien mit den jeweiligen Videodaten bilden die serverseitige Persistenzschicht ab.

5.6.4 URL abrufen

Der GIS-Client kann anhand einer URL automatisch eigen erstellte POI laden. Aufgrund dessen liefert der serverseitig implementierte LBS nicht die ortsabhängigen POI Daten, sondern eine URL, die die benötigte xml-Datei referenziert.

Möchte der Client POI in Abhängigkeit der vom Nutzer ausgewählten Kategorie und Distanz abrufen, wird das in der Abbildung 5.11 dargestellte Paket an den Server gesendet.

Anhand der übergebenen Kategorie wird über das Persistenzmodul die entsprechende xml-Datei geladen. Es stehen somit, entsprechend der Kategorie, alle jemals erstellten POI zur Verfügung. Dadurch, dass die aktuelle Position des Nutzers übermittelt wurde, kann das Berechnungsmodul[70] die Distanz zwischen den abgespeicherten POI und dem Nutzer berechnen. Befinden sich POI innerhalb der vom Nutzer ausgewählten Distanz, werden diese in eine separate xml-Datei gespeichert. Diese xml-Datei hat die gleiche Notation wie sie in der Abbildung 5.14 dargestellt ist. Die xml-Datei enthält somit alle POI, die zum einen der übergebenen Kategorie entsprechen und zum anderen sich in der näheren Umgebung des Nutzers befinden. Der clientseitig übermittelte *Identifier* wird verwendet, um eine eindeutige URL zur erzeugen. Diese URL referenziert die zuvor erzeugte xml-Datei und wird nach der Anfrage des Clients zurückgeliefert. Somit steht der Clientanwendung eine URL zur Verfügung, die der GIS-Client zum automatischen Abrufen der POI verwendet. Dieser Ablauf wird jedes Mal ausgeführt, wenn entweder die Kategorie oder die Distanz auf dem Client geändert wird. Da die xml-Datei durch den *Identifier* dem Client zugeordnet werden kann, wird diese nicht immer wieder neu erzeugt, sondern überschrieben.

[70] Die zur Berechnung der Distanz verwendete Formel ist aus [BA Buchholz 2007] zu entnehmen.

6 Fazit und Ausblick

Im folgenden Kapitel wird zu Anfang eine Zusammenfassung des in dieser Arbeit realisierten Systems geliefert. Die dabei entstandenen Probleme werden im Abschnitt 6.2 beschrieben. Um das Potential zu zeigen und eine denkbare Weiterentwicklung anzuregen, wird im Abschnitt 6.3 auf möglichen Erweiterungen der Client- und der Serverseite hingewiesen. Abschnitt 6.4 befasst sich mit der Qualitätssicherung, die bei der Softwareentwicklung zwingend erforderlich, aber aufgrund des Zeitmangels, hier leider nicht zum Einsatz kam. Abschließend folgt ein Fazit.

6.1 Zusammenfassung

In dieser Arbeit wurde ein Location-Based Service entwickelt, der standortbezogene Daten über eine Smart-Client Anwendung zur Verfügung stellt. Der Fokus lag dabei auf der Clientseite, der das „Live“ Annotieren von Multimediadaten unterstützt. Die dabei entstandenen Daten beschreiben einen POI.

Nutzer können mit der hier entwickelten Client Anwendung vor Ort POI erstellen und diese kartengestützt abrufen. Ein POI besteht dabei aus eindeutigen Standortkoordinaten, einem erstellten Video, einem Namen und einer Beschreibung. Damit vor dem Abrufen der Daten ein Filtern möglich ist, wird der POI zusätzlich einer Kategorie zugeordnet. Die zur Auswahl stehenden Kategorien sind Restaurant, Landmark, Bar, Disco und Others. Zur Orientierung wird dem Nutzer eine digitale Karte auf dem Display des Mobiltelefons angezeigt. Auf der Karte befinden sich Marker, die die eigene Position und die der abgerufenen POI kennzeichnet. Die Detailinformationen können aus der Kartenansicht angezeigt werden. Benötigt der Nutzer weitere Informationen, wird das jeweilige Video auf das Mobiltelefon geladen und abgespielt.

Ein zentraler Server steht für das Entgegennehmen und Abrufen der clientseitigen Anfragen bereit. Dieser implementiert zum einen eine Persistenzschicht, die eine Datenbank simuliert und zum anderen ein Berechnungsmodul, das die Distanz zwischen zwei Standorten berechnen kann.

6.2 Probleme bei der Realisierung

In der Anfangsphase wurden Komponenten auf einem Emulator getestet, der innerhalb des *J2ME Wireless Toolkit* (WTK) der Firma *sun*[71] zur Verfügung steht. Dieser soll gleichwertig gegenüber heutigen Mobiltelefonen sein und die CLDC, das MIDP und die gängigen optionalen Pakete unterstützen. Es erwies sich allerdings, dass es nur mit relativ

[71] Online im Internet unter http://de.sun.com/.

großem Zeitaufwand möglich war, die erforderlichen optionalen Pakete wie beispielsweise die Location API oder die Mobile Media API zu simulieren. Es wurde deshalb direkt auf dem Testgerät, dem Nokia N93 getestet. Wird die Entwicklung des Systems allerdings weitergeführt, sollte aufgrund der vielfältigen Möglichkeiten und des guten Supports durch die Firma *sun*, auf das WTK zurückgegriffen werden.

Der verwendete GIS-Client macht es möglich, auf bereits bestehendes Kartenmaterial im Internet zuzugreifen und diese mit eigenen Inhalten zu verknüpfen. Der resultierende Mashup stellte die Funktionalität zur Manipulation bzw. Interpretation der Daten nicht zur Verfügung. Diese waren jedoch für die Filterung der Daten relevant. Um dennoch ein Filtern der Daten zu unterstützen, wurde serverseitig das Berechnungsmodul implementiert. Ein weiterer Nachteil ergab sich aus der mangelnden Dokumentation der API. Die J2MEMap bietet eine Fülle an Funktionalitäten, die erst durch mühseliges Testen, Suchen oder einem E-Mail Kontakt mit Herrn Landspurg aufgedeckt wurden. Eine Dokumentation, wie sie beispielsweise bei den verwendeten optionalen Paketen der J2ME angeboten werden, wäre wünschenswert gewesen. Es ist zu überlegen, selbst einen aktiven Teil zur J2MEMap Community beizutragen und die vorhandenen Erkenntnisse in Form eines Dokumentes der Community zur Verfügung zu stellen.

Innerhalb des Grundlagenteils wurde einschlägig auf die aktuellen Datenübertragungstechniken eingegangen. Es stellte sich heraus, dass aufgrund der Übertragungsgeschwindigkeit und der heutzutage relativ guten Verfügbarkeit das UMTS-Netz dem GPRS- Netz vorzuziehen ist Dies ist besonders der Fall, wenn es sich bei den Daten um Multimediadaten bzw. größere Datenmengen wie z.B. MMS handelt. Um das UMTS Netz zu Testzwecken nutzen zu können, wurde eine entsprechende Karte zur Verfügung gestellt. Es stellte sich allerdings heraus, dass obwohl zuvor eine UMTS Verbindung vorhanden war, diese nicht immer von der Mobilfunkkarte bzw. vom Provider verwendet wurde. Dies wurde durch das 3G Symbol, das bei der Verwendung des UMTS Netzes auf dem Mobiltelefon angezeigt wird, deutlich. Es ist somit abzuwarten, wann ein flächendeckendes und vor allem ein problemlos funktionierendes UMTS Netz verfügbar ist. Bis dahin sollte darauf geachtet werden, möglichst im UMTS Netz eingeloggt zu sein oder, wenn verfügbar, auf das WLAN Netz auszuweichen.

Die Positionsbestimmung wurde mittels GPS-Verfahren realisiert. Es wurde hierfür der externe GPS-Empfänger der Firma Holux GPSlim 236 verwendet. LEDs zeigen bei diesem Gerät die Bluetooth Verbindung, den Energiestand und das Vorhandensein der benötigten Satelliten zur Ermittlung des Standortes an. In einigen Situationen war die Positionsbestimmung jedoch nicht möglich, obwohl eine optimale Verbindung zu den Satelliten angezeigt wurde. Es sollten daher vergleichbare GPS-Empfänger getestet werden,

um eine genauere Aussage der Verfügbarkeit machen zu können. In Zukunft wird es im Mobilfunkbereich allerdings nur noch interne GPS-Empfänger geben.

Die Wiedergabe des Videos wurde durch das HTTP Protokoll realisiert. Hierbei wird das Video vorab geladen und dann abgespielt. Diese Variante erwies sich als unvorteilhaft, da bei großen Datenmengen, eine für den Nutzer unangenehme Latenzzeit hervorgerufen wurde. Die Möglichkeit, das Video mittels Streaming abspielen zu lassen, wird bei den Erweiterungsmöglichkeiten genannt. Es ist allerdings darauf zu achten, ob eventuelle online/offline Problemen das genaue Gegenteil bewirken. Denn Streaming ist nicht zu empfehlen, wenn durch ständige Verbindungsabbrüche das Video nicht ruckelfrei abgespielt wird.

6.3 Erweiterungsmöglichkeiten

Die Realisierung des hier entwickelten Systems stellt derzeit einen Prototyp dar. Es wurde gezeigt, dass aus der Kombination von vorhandenen Technologien neue und nützliche Ideen umgesetzt werden können. Um allerdings produktreif zu werden, können und müssen einige Erweiterungen umgesetzt werden, auf die in den beiden folgenden Abschnitten näher eingegangen wird.

6.3.1 Clientseitige Erweiterungen

Da es bei der Übertragung des erstellten POI zu online/offline Problemen kommen kann, sollte das lokale Speichern möglich sein. Die Daten könnten somit zu einem späteren Zeitpunkt übermittelt werden. Hierfür kann das optionale FileConnection API oder das Record Management System (RMS) verwendet werden. Da das RMS innerhalb der MIDP Spezifikation definiert ist, sollte dies aufgrund der größeren Verfügbarkeit die erste Wahl für das Speichern sein.

Die digitalen Karten werden über den GIS-Client je nach Bedarf auf das Mobiltelefon geladen. Hierfür baut der Client eine Verbindung zu einem zentralen Proxy-Server auf, der Karten von unterschiedlichen Map-Servern anfordert und abspeichert. Hat der Client somit keine Internetverbindung, kann die Karte nicht angezeigt werden. Um dies zu umgehen, sollte das lokale Speichern von Karten z.B. einer kompletten Stadt, unterstützt werden. J2MEMap[72] bietet diese Möglichkeit im begrenzten Umfang an. Die Karten werden dafür lokal im jar gespeichert und je nach Bedarf geladen.

---

[72] Ein Beschreibung, wie J2MEMap das lokale Speichern der Karten umsetzt, ist unter [J2MEMap Lokal] nachzulesen.

In der jetzigen Lösung wird das Video per HTTP geladen und abgespielt. Ein erneutes Abspielen benötigt auch ein erneutes Laden. Dem kann abgeholfen werden, indem das *Player* Objekt gespeichert wird. Ein erneutes Abspielen ist somit möglich, ohne das Video wiederholt vom Server abzurufen. Da es allerdings bei großen Daten zu Latenzzeiten kommen kann, sollte außerdem das Streaming angeboten werden. Das Video wird somit ohne Verzögerung abgespielt und trägt im hohen Maße der Benutzerfreundlichkeit bei.
Das clientseitig erforderliche Real-Time Transport Protokoll ist bereits im Mobile Media API definiert.
Zum jetzigen Zeitpunkt kann genau ein Video pro erstellten POI erzeugt werden. Um die Kreativität und den Spaß beim Erstellen zu steigern, sollten auch Fotos oder Tonaufzeichnungen aufgenommen werden können. Die benötigten Schnittstellen liefert das Mobile Media API. Allerdings muss darauf geachtet werden, dass die Übertragung der Daten für den Nutzer nicht zu lange dauert. Beim Erzeugen sollte deshalb die Länge des Videos nicht von dem verfügbaren Speicherplatz auf dem Mobiltelefon abhängen. Es sollten anhand von Performance-, Übertragungs- und Usability[73]- Tests Durchschnittsgrößen der zu erzeugenden Daten ermittelt werden. Werden diese Vorgaben vom Nutzer eingehalten, bietet das Produkt ein stabiles und benutzerfreundliches System.

6.3.2 Serverseitige Erweiterungen

Damit überhaupt eine Community entstehen kann, sollte es serverseitig ein Portal geben. Nutzer melden sich über die Plattform an und erhalten ein Benutzerkonto. Hierüber kann auch die individuelle Smart-Client Anwendung (individuell, wegen dem eingeführten *Identifier*) zum Download bereitgestellt werden. Um einen Bezug zu den Daten herzustellen, werden die Benutzerdaten clientseitig beim Erstellen und Abrufen von POI mit an den Server übermittelt. Es wäre über das Portal möglich, seine eigen erstellten POI zu Verwalten und anderen Menschen im Internet zur Ansicht zur Verfügung zu stellen. Die Community beschränkt sich somit nicht nur auf diejenigen, die die Client Anwendung auf dem Mobiltelefon installiert haben. Jeder könnte über das Internet darauf zugreifen.
Die serverseitig gespeicherten Daten sollten aufgrund der Performance, Verfügbarkeit und Sicherheit in einer Datenbank gespeichert werden. Insbesondere, wenn Benutzerkonten hinzugefügt und eine Zuordnung der Daten erfolgen muss, sollte ein geeignetes Objekt-Model für die Kapselung der Daten eingesetzt werden.

---

[73] Bei einem Usability-Test wird die Interaktion zwischen dem Produkt und dem Anwender anhand konkreter Aufgabenstellungen beobachtet, analysiert und ausgewertet, mit dem Ziel, evtl. Schwachstellen zu beheben und somit zum Wohlbefinden des Anwenders beizutragen.

Beim Erstellen der Daten kann ein Video erzeugt werden. Dieses Video ist je nach Aufnahmedauer und verfügbarem Speicherplatz auf dem Mobiltelefon, unterschiedlich groß. Möchte ein Nutzer dieses Video abrufen, muss es bei der jetzigen Lösung erst komplett geladen werden, damit es abgespielt wird. Dies dauert je nach Größe des Videos unterschiedlich lang und ist, im Bezug auf die Effizienz und Performance ein Manko. Es sollte daher die Möglichkeit des Streamings implementiert werden. Hierfür müsste ein Streaming Server[74] in das System integriert werden. Die Wiedergabe erfolgt dabei sofort nach Eintreffen des ersten Datenpaketes. Der Nutzer muss nicht mehr Warten und kann bei nicht gefallen des Videos, den Stream vorzeitig beenden.

Im Zuge der Plattformerweiterung sollte für eine einfache und leichte Installation der Software das Over-The-Air[75] Verfahren angeboten werden. Es unterstützt das direkte Installieren der Anwendung über eine Funkverbindung.

Momentan wird dem Nutzer vertraut, dass die Daten, die er über die Client Anwendung erzeugt korrekt (z.B. die Kategorie des POI) und nicht rechtsradikale, pornografische oder andere diskriminierende Informationen beinhalten. Es sollte somit überlegt werden, ob anhand der Positionskoordinate und einer bestehenden Datenbank, eine Zuordnung zu bestehenden POI realisiert werden kann. Außerdem sollten Filter, die zum einen die Copyrightrechte und zum anderen strafrechtliche Aufnahmen erfassen können, eingesetzt werden (siehe Abschnitt 4.4 „Rechtliche Anforderungen“).

Die entgegengenommen Daten werden zum jetzigen Zeitpunkt ins Dateisystem gespeichert. Es sollte bei einer Weiterentwicklung eine reale Datenbank eingesetzt werden, damit ein schneller und sicherer Zugriff auf die Daten erfolgen kann. Sinnvoll wäre in diesem Zusammenhang eine Cluster Architektur. Mehrere Server erscheinen nach außen wie eine Einheit. Vorteile eines Clusters sind höhere Ausfallsicherheit und Skalierbarkeit.

6.4 Qualitätssicherung

Der Fokus in dieser Arbeit lag in der Entwicklung einer J2ME Anwendung für Mobiltelefone. Es konnte aus Zeitgründen keine erforderliche Qualitätssicherung des Systems durchgeführt werden. Da der Mensch jedoch im Allgemeinen dazu neigt, Fehler zu machen, ist die Qualitätssicherung bei allen Tätigkeiten der Softwareentwicklung notwendig und sollte bei einer Weiterentwicklung dieses System eingeführt werden. Mit ihr können die Qualität und die Zuverlässigkeit eines Produktes gesichert werden. Außerdem

[74] Eine Möglichkeit wäre der Open Source Streaming Server von Apple [Apple 2007].

[75] Für nähere Informationen [J2ME OTA].

werden bei frühzeitiger Anwendung Fehler verhindert, Seiteneffekte entdeckt und Entwicklungszeiten verkürzt.

Damit die Qualität des Prototyps gesichert wird, sollten unterschiedliche Methoden angewendet werden. Der Prototyp wurde auf der Basis von Komponenten realisiert. Es sollten somit als erstes Modultests eingeführt werden, die auf Klassen- bzw. Methodenebene die Funktionalität des Moduls automatisiert testen. Ein Beispiel hierfür ist J2MEUnit[76]. J2MEUnit ist vergleichbar mit JUnit[77] und ist ein Java-Framework für das Schreiben und Ausführen automatischer Tests. Der Unterschied liegt im Wesentlichen darin, dass die *Java Micro Edition* weniger Bibliotheken unterstützt (z.B. keine Reflection).

Aufbauend auf den Modultests werden über Integrationstests die verschiedenen voneinander abhängigen Komponenten des hier entwickelten Systems im Zusammenspiel miteinander getestet. Es können frühzeitig Probleme aufgedeckt und ein mögliches Scheitern des Systems verhindert werden. Da der Aufwand mit der Fülle an Komponenten steigt, sollten hier die speziellen Szenarien wie beispielsweise das Senden der Daten oder die Auf- bzw. Wiedergabe des Videos getestet werden. Es werden somit Integrationstests für einzelne Subsysteme erstellt.

Abschließend sollten Systemtests integriert werden. Insbesondere bei der Übertragung der POI Daten und beim Abrufen der multimedialen Inhalte müssen Funktions- und Leistungstests durchgeführt werden, um dem Nutzer die Zuverlässigkeit und Benutzbarkeit zu zusichern. Alle vorgestellten Softwaretestmethoden setzen die Kenntnis über die Implementierung des Systems voraus. Dadurch kann es passieren, dass Entwickler um ihre eigenen Fehler herum entwickeln und somit Lücken in der Implementierung entstehen. Um dies zu vermeiden, sollten Black-Box Tests eingeführt werden. Sie überprüfen die Funktion des Systems nach außen hin und greifen dabei auf die zuvor festgelegten Spezifikationen zurück.

Durch den Einsatz der unterschiedlichen Methoden nimmt das System an Qualität und Zuverlässigkeit zu. Dies steigert das Vertrauen des Nutzers in das Produkt. Folglich wäre eine Weiterentwicklung des Systems gesichert.

---

[76] Eine weiterführende Beschreibung ist Online unter http://j2meunit.sourceforge.net/doc.html einzusehen.

[77] Online im Internet unter http://www.junit.org/.

6.5 Fazit

Die hier entwickelte Anwendung geht auf das steigende Bedürfnis Informationen auszutauschen, ein. Es wurde ein Prototyp entwickelt, mit dem es vor Ort möglich ist, Erlebnisse multimedial festzuhalten und einer Community zur Verfügung zu stellen. Damit die Daten einen direkten Bezug zur realen Welt erhalten, wurde das GPS-Verfahren eingesetzt. Es wurde ein LBS entwickelt, der die erzeugten POI der mobilen Anwendung ortsabhängig zur Verfügung stellt. Um digitale Karten auf dem Mobiltelefon anzeigen zu können, wurde ein GIS-Client genutzt. Dieser Client bot außerdem die Möglichkeit, eigene Inhalte mit der Karte zu verknüpfen. Da der Client die Funktionalität zur Manipulation bzw. Interpretation der Daten nicht unterstützt, wurde das Filtern der POI serverseitig realisiert. Zur Orientierung wurden auf der Karte die eigene Position und die der POI mittels Marker dargestellt.

Das lokale Speichern der POI und der abgerufenen Videos wurden aus Zeitgründen nicht realisiert.

Zudem bedarf es einer Qualitätssicherung des Systems. Dafür eignen sich Modultests, Integrationstests, Systemtests und Blackboxtests.

Eine mögliche clientseitige Erweiterung ist das Abspielen des Videos per Streaming. Serverseitig sollte für das Verwalten der POI ein Online-Portal umgesetzt werden. Darüber hinaus sollte aus Performancegründen das Speichern der POI in einer realen Datenbank gewährleistet werden.

Die hier entwickelte Anwendung stellt einen Prototyp dar, auf dem aufgebaut werden kann und sollte, um das Bedürfnis an Informationsaustausch weiterhin befriedigen zu können. Dies bestätigt der immer größer werdende Markt für mobile Anwendungen. Die heutigen Technologien sowie die vorangeschrittene Ausstattung mobiler Endgeräte stehen der Erweiterung in keinem Fall entgegen.

7 Literaturverzeichnis

[ABI-Research 2006] Studie über den Zuwachs von GPS Modulen im GSM Mark, ABI Research , 10.Oktober 2006

URL: http://www.abiresearch.com/abiprdisplay.jsp?pressid=740

[Apple 2007] Open Source Streaming Server, Apple, 2007

URL: http://developer.apple.com/opensource/server/streaming/index.html

[BA Buchholz 2007] Clemens Buchholz, Mobile Mitfahrbörse, 2007

PDF: BA - Mobile Mitfahrboerse.PDF

[BIS 2005] Bundesamt für Sicherheit in der Informationstechnik, 2005

URL: BSI Standard_1.pdf, BSI Standard_2.pdf, BSI Standard_3.pdf

[BITKOM 2007] Bundesverband Informationswirtschaft, Telekommunikation und neue Medien e.V., Umfrage: Zahl der Mobilfunk-Anschlüsse, 13 März 2007

URL: http://www.bitkom.org/de/presse/30739_44673.aspx

[BITKOM UMTS 2007] Bundesverband Informationswirtschaft, Telekommunikation und neue Medien e.V., Umfrage: UMTS Teilnehmer Weltweit, 15 Februar 2006

URL: http://www.bitkom.de/de/presse/30739_37663.aspx

[Blog Studie 2007] Universität Leipzig. Univ.-Prof. Dr. Ansgar Zerfaß. Janine Bogosyan, Blogstudie 2007

URL: http://www.blogstudie2007.de/

[Dipl. Fürpass 2001] Christian Fürpass, Mapserver als Hilfsmittel zur Datenvisualisierung im Internet, 2001

PDF: Diplomarbeit-Fuerpass.pdf

[DevX] Jupitermedia Corporation, 2007

URL: http://www.devx.com/wireless/Article/21262/0/page/2

URL: http://www.devx.com/wireless/Article/21262/0/page/3

[Eclipse-Erweiterungen] Erweiterungen der *eclipse* IDE zur Entwicklung von J2ME Anwendungen

Sun Java Wireless Toolkit for CLDC, URL: http://java.sun.com/products/sjwtoolkit/

EclipseME, URL: http://eclipseme.org/

[ELKO GSM 2007] Elektronik Kompendium, Patrick Schnabel, 2007

URL: http://www.elektronik-kompendium.de/sites/kom/0910201.htm

[ELKO UMTS 2007] Elektronik Kompendium, Patrick Schnabel, 2007

URL: http://www.elektronik-kompendium.de/sites/kom/0910231.htm

[REN 2007] Chris Rupp & die SOPHISTen, Requirements- Engineering und Management, Professionelle, iterative Anforderungsanalyse für die Praxis, HANSER Verlag, 2007
ISBN-10: 3-446-40509-7
ISBN-13: 978-3-446-40509-7

[ESNI 2007] European Sattelite Navigation Industries GmbH, 2007
URL: www.european-satellite-navigation-industries.net

[ESERI] White Paper, ESRI-Shapefile, Technical Description, 1998
PDF: ESRI-Shapefile-Technical-Description.pdf

[Fitzke 1999] Dipl. Geograph Jens Fitzke, Universität Bonn, GIS-Tutorial, 1999
URL: http://www.giub.uni-bonn.de/gistutor/

[FORUM 2007] Offizielles UMTS Forum, 2007
URL: http://www.umts-forum.org
PDF: MultiMedia_Papers_White-Paper-HSPA.pdf

[GIS Bill 1994] Ralf Bill, Grundlagen der Geoinformationssysteme. Heidelberg 1994, ISBN 3-87907-265-5

[GPS Satelliten] Advanced Wireless Planet, Round Solutions Ltd., 2007
URL: http://www.gsm-modem.de/gps/gps-ortung.html

[GPRS 2007] GSM World - the www site of the GSM Association, 2007
URL: http://www.gsmworld.com/technology/gprs/index.shtml

[GSM 2007] GSM World - the www site of the GSM Association, 2007
URL: http://www.gsmworld.com/technology/gsm.shtml

[GoTo Java2 2001] GoTo Java 2, Guido Krüger, Addision-Wesley, 2001
ISBN: 3 8273 1710 X, Seite 742

[HOLUX 2007] Holux, GR-236 Bluetooth GPS
URL: http://www.holux-gps.de

[JavaME 2006] Ulrich Breymann, Heiko Mosemann, JavaME Anwendungsentwicklung Für Handys, PDA und CO. Carl Hanser Verlag München Wien 2006,
ISBN: 3-446-22997-3

[J2ME CLDC] SE-EDUCATION Public Company Limited, 2007
URL: http://internet.se-ed.com/content/IN84/IN84_43.asp

[J2ME Location API] J2ME Location API
PDF: JSR179_MR_101.pdf

[J2ME Plattform] J2ME Micro Edition Platform, 2003
URL: http://j2me.winwinfaisal.info/

[J2ME Struktur] Michael Juntao Yuan, JavaWorld.com, 2003
URL: http://www.javaworld.com/javaworld/jw-11-2003/jw-1107-wireless.html

[J2ME OTA] sun, Over the Air, Over The Air User Initiated Provisioning

URL: http://java.sun.com/products/midp/OTAProvisioning-1.0.pdf

[J2MEMap Lokal] Landspurg, Thomas: J2MeMap, Forum, 2007

URL: http://8motions.com/forum/viewtopic.php?f=3&t=13&st=0&sk=t&sd=a&start=45

[LBS Schiller 2004] Jochen Schiller, Agnès Voisard, Location-Based Services, Verlag Elsevier 2004, ISBN: 1-55860-929-6

[Mobile Computing 2005] Jörg Roth, Mobile Computing, Verlag dpunkt 2005 Heidelberg, ISBN: 3-89864-366-2 (Seite 257)

[Mono Novell 2007] Mono, Open Source Project 2006

URL: http://www.mono-project.com/Main_Page

[MS-NET 2007] msdn, .NET CF, 2007

URL: http://msdn2.microsoft.com/de-de/library/f44bbwa1(VS.80).aspx

[MS-NET A 2007] msdn, .NET CF Architektur, 2007

URL: http://msdn2.microsoft.com/de-de/library/9s7k7ce5(VS.80).aspx

[MS-NET AD 2007] msdn, Anwendungsdomäne in .NET CF

URL: http://msdn2.microsoft.com/de-de/library/18adbxba(VS.80).aspx

[MS-NET D 2007] msdn, .NET - Unterstützte Geräte und Plattformen, 2007

URL: http://msdn2.microsoft.com/de-de/library/ms172550(VS.80).aspx

[MS-NET E 2007] msdn, Plattformabhängigkeit .NET CF Anwendungen, 2004

URL:http://www.microsoft.com/germany/msdn/library/net/compactframework/ErsteSchritteMitDemNETCompactFramework.mspx?mfr=true

[MS-NET K 2007] msdn, .NET Framework- unterstützte Klassen, 2007

URL: http://msdn2.microsoft.com/de-de/library/w8xd02k7(VS.80).aspx

[MS-NET M 2007] msdn, .NET CF Windows Mobile 5.0, 2007

URL:

http://www.microsoft.com/germany/msdn/library/net/compactframework/MobileAnwendungenMitDemNETCompactFramework20.mspx?mfr=true

[MS-NET S 2007] msdn, Entwickeln für Smartphone 2003 mit dem .NET CF, 2007

URL:

http://www.microsoft.com/germany/msdn/library/mobility/windowsmobile/smartphone/EntwickelnFuerSmartphone2003MitDemNETCompactFramework.mspx?mfr=true

[NOKIA 2007] Nokia, Finland, Modellübersicht

URL: http://www.nokia.de/de/mobiltelefone/modelluebersicht/4198.html

[NOKIA-Streaming] Nokia Forum, Video And Streaming In Nokia Devices

PDF: Video_And_Streaming_In_Nokia_Devices_v3_0_en.pdf

[NOKIA-MMAPI] Nokia Forum, Mobile Media Developers Guide

PDF: MIDP_Mobile_Media_API_Developers_Guide_v1_0_en.pdf

[O'Reilly 2005] Tim O'Reilly, What is Web 2.0, 2005

URL: http://www.oreillynet.com/pub/a/oreilly/tim/news/2005/09/30/what-is-web-20.html

[Qiro 2007] Qiro GmbH, Zimmerstraße 55 D-10117 Berlin Germany

PDF: qiro_tutorial.pdf

[RAASCH SE 2006] Vorlesungsskript SE, Raasch, 2006

Skript: Raasch-Ergonomie.pdf

[SDN-J2ME 2007] Sun, Java Micro Edition Technology, 2007

URL: http://java.sun.com/javame/technology/index.jsp

[SDN-J2ME S 2002] Sun, Wireless Java Security, 2002

URL: http://developers.sun.com/mobility/midp/articles/security/index.html

[SDN-J2ME CDC 2007] Sun, CDC: Java Platform Technology for Connected Devices, 2007

URL: http://java.sun.com/products/cdc/wp/cdc-whitepaper.pdf

[SDN-J2ME CLDC 2007] Sun, CLDC HotSpot™ ImplementationVM

URL:http://java.sun.com/j2me/docs/pdf/CLDC-HI_whitepaper-February_2005.pdf

[SDN-J2ME MIDP 2007] Sun, JSR-000118 MIDP 2.0, 2007

URL: http://jcp.org/aboutJava/communityprocess/final/jsr118/index.html

[SDN-J2ME WTK 2007] Sun Java Wireless Toolkit for CLDC

URL: http://java.sun.com/products/sjwtoolkit/

[Servlets 2007] Vorlesung: Webservices/Java Webapplications SoSe 2007, Frau Wendholt

Skript: Servlets.pdf

[Tomcat 2006] Vorlesung: Webservices/Java Webapplications SoSe 06, Frau Wendholt

Skript: Tomcat.pdf

[UMTS 2004] Franz-Josef Banet, Anke Gärtner, Gerhard Teßmar. Hüthig: UMTS Netzwerktechnik, Dienstarchitektur, Evolution Verlag, 2004 - ISBN: 3-8266-5034-4

[Ubiquitous Computing 1991] Mark Weiser: The Computer for the 21st Century, 1991

URL: http://www.ubiq.com/hypertext/weiser/SciAmDraft3.html

[Vodafone 2007] Vodafone Live! Vodafone D2 GmbH, 2007

URL: http://www.vodafonelive.de/cp/lang/en/pid/143

[Web Services 2007] Vorlesung: Webservices/Java Webapplications Wendholt, Birgit, Web Services, HAW Hamburg, SoSe2007

Skript: MobileWebservicesRuntime.pdf

[WiMAX 2007] Johannes Maucher, Jörg Furrer, Der IEEE-802.16-Standard: Technik Anwendung Potenzial, Heise Verlag, 2077

ISBN-10: 3-936931-33-X

ISBN-13: 978-3-936931-33-4

[DOM- w3schools] XML DOM Tutorial

URL: http://www.w3schools.com/dom/default.asp

[YouTube 2007] Musikverlage beteiligen sich an YouTube-Klage, 2007

URL: http://www.computerwoche.de/nachrichten/597888/

[3gpp MMS 2007] Multimedia Messaging Service (MMS), 2007

URL: http://www.3gpp.org/ftp/Specs/html-info/26140.htm

Inhalt der beiliegenden CD

- /Bachelorarbeit.pdf: dieses Dokument
- /Referenzen: Beinhaltet alle aus dem Literaturverzeichnis verwendeten Quellen, die lokal gespeichert werden konnten.
- /Sourcecode:
    - Webapplikation.zip: Eclipse Tomcat Project (serverseitige Realisierung)
    - Smart-Client-Software.zip: Eclipse J2ME MIDlet Suite (clientseitige Realisierung)
    - Readme.txt: Hinweise

Versicherung über Selbständigkeit

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

Hamburg, 16.10.2007 _____

Ort, Datum

_____

Unterschrift