Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Projekt 1

Christian Wagner

Fakultät Technik und Informatik
Studiendepartment Informatik

Inhaltsverzeichnis

1 Einleitung

Im Fach „Projekt“ soll über zwei Semester eine definiertes Ziel umgesetzt werden. Dabei soll im ersten Teil des Projektes (1. Semester) die Infrastruktur zur gesamten Umsetzung geschaffen werden. In dem folgenden Projektbericht wird die Zielsetzung, die geplante Umsetzung und der Infrastrukturaufbau genauer erläutert.

1.1 Ziel des Projektes

Das Projektziel besteht darin, eine Navigations-Applikation zu entwickeln. Im Vordergrund sollen hierbei Wegmarken stehen, die für die Benutzer von besonderer Bedeutung sind - sogenannte „Points of Interest“, kurz POI. Dabei sollen POIs von den Benutzern ausgewählt werden und die Applikation erstellt daraufhin eine Route, die die gewählten POIs auf dem kürzesten Weg verbindet. Diese Routen sollen des Weiteren gespeichert werden und als Stadtrundführung für andere Benutzer verfügbar sein. Zusätzlich soll es möglich sein, Start-Ziel-Routen auch unter Einbeziehung von bestimmten POIs oder Interessen errechnen zu lassen.
Die folgenden Szenarios sollen das Ziel der Anwendung verdeutlichen:

Szenario 1:
Ein Tourist möchte gerne in Hamburg eine Stadtrundführung ohne Reiseführer durchführen. Dafür wählt er in der Applikation die veröffentlichten Routen in Hamburg aus. Hierbei wird er zusätzlich von seinen eigenen Profildaten (z.B. eigenen Interessen) bei der Auswahl einer passenden Route unterstützt. Wählt er eine Route, führt die Applikation ihn durch die Stadt. An eingetragenen POIs werden ihm zusätzlich hinterlegte Informationen wie Text, Audio (Text-To-Speech) oder Video angeboten. Außerdem hat der Benutzer die Möglichkeit, weitere POIs der Route hinzuzufügen oder eine eigene Route aus verschiedenen POIs zu erstellen und diese nach Auswahl bestimmter POIs generieren zu lassen.

Szeanario 2:
Ein Geschäftsmann landet am Münchener Flughafen und hat bis zu seinem geplanten Termin noch 2 Stunden Zeit. Er beschließt, vorher noch ein Restaurant zu besuchen und sich danach noch nach einer Lektüre für den Rückweg umzuschauen. Er gibt folgende Informationen in das System ein:

```
Standort: aktueller Standort
POI1: Interesse: Essen (italienisch)
POI2: Interesse: Buchhandlung
Ziel: Firma, Uhrzeit: 16.00
```

Das System berechnet eine sinnvolle Route anhand von POI-Bewertungs-Bibliotheken und unter Einbeziehung der zeitlichen Informationen.

2 Hauptteil

2.1 Architektur

Die Routen und POIs sollen zentral auf einem Server abgelegt werden, von denen sich Mobile Clients die gewünschten Informationen über Webservices abrufen können. Zusätzlich wird ein Datenbankserver benötigt, der die zentralen Information verwaltet. Dafür wird in diesem Projekt eine postgreSQL-Instanz verwendet. Die abgelegten Informationen werden vom Webserver abgefragt und an den Client weitergeleitet. Die Kommunikation zwischen den Clients und dem Server findet ausschließlich über den Webserver (Webservices) statt.

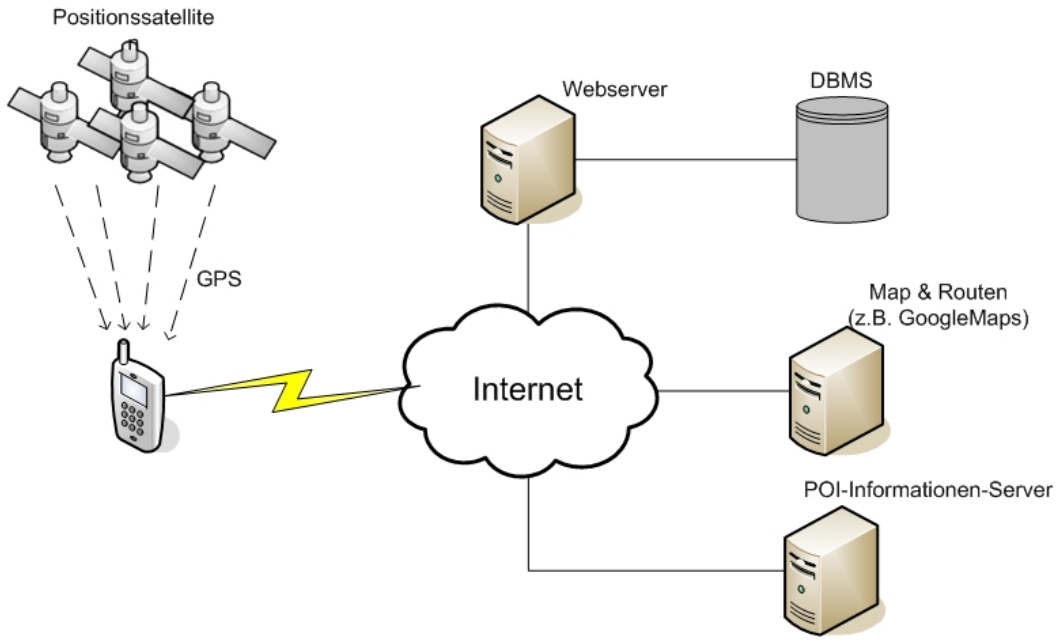

Abbildung 1: Architekturübersicht

Die Maps- und Routenserver, sowie die POI-Verzeichnisserver sind Fremdsysteme, die durch die vordefinierten Schnittstellen der Anbieter angesteuert werden können, um benötigte Informationen abzufragen.
Die bisherige Umsetzung des Client wird auf Basis eines Emulators durchgeführt. Die Ser-

veranwendungen (Datenbank, Webservices) werden lokal auf dem Rechner betrieben. Im zweiten Projektteil werden die Architekturkomponenten auf getrennte Systeme portiert.

2.2 Server

Für die zentralen Aufgaben werden verschiedene Server benötigt. Die serverseitige Applikation teilt sich in drei Hauptaufgaben auf:

- Datenbank
- Webserver / Webservices
- Anwendung

Im Folgenden werden die aufgelisteten Punkte näher betrachtet.

2.2.1 PostGreSQL-Datenbank

Als Datenbank zur Speicherung der zentralen Informationen wird ein PostGreSQL-DBMS verwendet. In dieser Datenbank werden alle Routen, POI und Userinformationen vorgehalten und dem Benutzer bei Bedarf zur Verfügung gestellt. Ein zusätzlich zu entwickelndes Front-End für den Datenbank-Zugriff wird nicht benötigt, da der Zugriff ausschließlich über die Webservices ermöglicht werden soll, um die dynamische Verwendung verschiedener Client-Anwendungen zu gewährleisten.

Das folgende ER-Diagramm (s. Abb. 2) stellt den Aufbau der PostgreSQL-Datenbank da. Einige Tabellen, die zur Datenverarbeitung notwendig sind, wurden hier nicht dargestellt. Das Diagramm soll vor allem die wichtigsten Komponenten zeigen und zum Verständnis dienen. Zudem wurden zur besseren Übersichtlichkeit nur die wichtigsten Tabellenfelder aufgeführt.

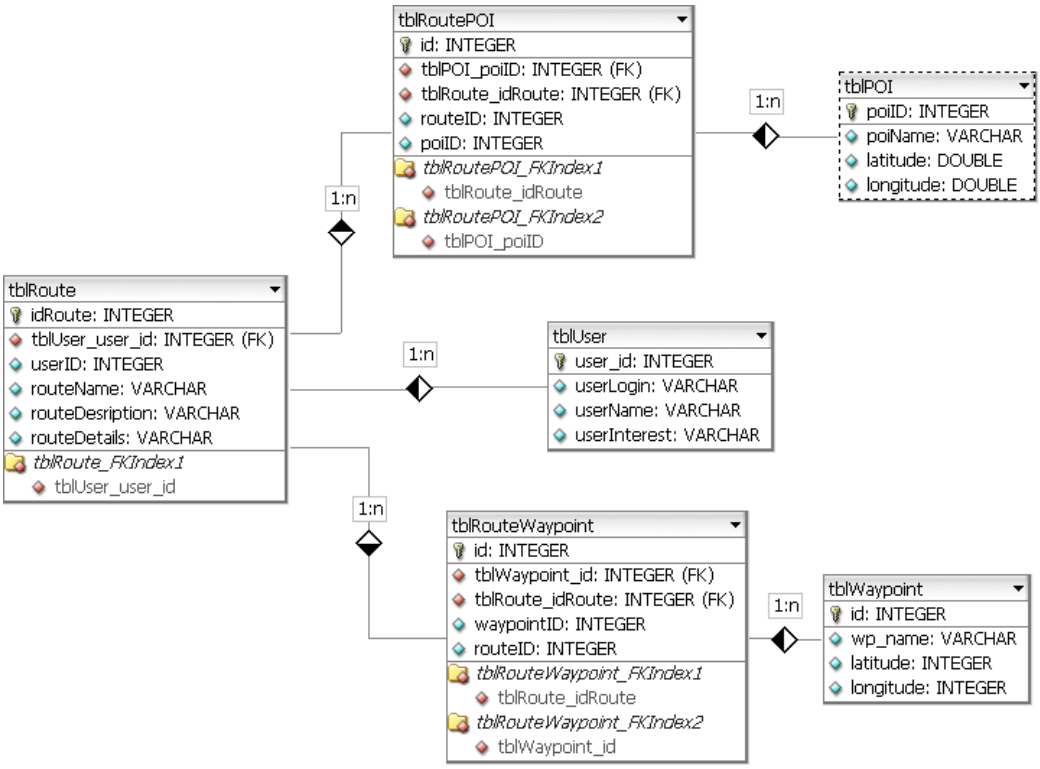

Abbildung 2: ER-Diagramm

2.2.2 Webservice

Apache Tomcat / Apache Axis

Der Tomcat Server stellt auf Basis eines Apache Webserver Java-Code auf Webseiten bereit. So können die Webservices auf Java-Basis implementiert werden. Zusätzlich wird die SOAP-Engine AXIS eingesetzt, die die Erstellung und Veröffentlichung von Webservices auf einem Apache Webserver unterstützt. Auf dieser Plattform werden die benötigten Webservices umgesetzt, die über den Tomcat-Server aufgerufen werden können.

Webservices

Die serverseitigen Informationen werden über Webservices zur Verfügung gestellt. Über diese Webservices können Daten abgefragt sowie hinterlegt werden. Im Allgemeinen werden Zeichenketten, die nach festgelegter Struktur aufgebaut sind, abgefragt bzw. dem Server übertragen. Aus diesen Zeichenketten können die übertragenen Informationen extrahiert werden und somit entweder von der Client-Anwendung verarbeitet oder vom Server in der

Datenbank abgelegt werden. Der Vorteil der XML-basierenden Nachrichten der Webservices ist die Betriebssystemunabhänigkeit. Dadurch können auch weitere Client-Anwendungen mit unterschiedlichen Betriebssystemen auf die Server-Applikation zugreifen. Damit wären auch zukünftige Implementierungen weiterer Clients (z.B. für IPhone) möglich.

Im Folgenden werden die bisher implementierten Webservices aufgelistet:

int login (String user, String passwort)

string getUserRouteList(String authString)

string getShareRouteList(string country, string region, string category)

string getRouteDetails(String authString, int routeID)

string changeRoute(String authString, int routeID, string routeName, String routeDescription, String routeDetails, String wayPoint)

string deleteRoute(String authString, int routeID)

string addRoute(String authString, String routeName, String routeDescription, String routeDetails, String wayPoint)

int addUser(String authString, String userInfo)

bool deleteUser(String authString, int delUserID)

bool changePasswort(String oldAuthString, String newPasswort)

bool changeUserDetails(String authString, int userID, String memberGroup, String userDetails)

2.2.3 Text-To-Speech - OpenMary

Für die Sprachwiedergabe von Texten zur weiteren Informationsübermittlung der POIs wird die Applikation „OpenMary“ getestet. OpenMary beinhaltet die bekannte „mbrola“-Bibliothek, die in vielen Text-To-Speech-Werkzeugen verwendet wird. OpenMary basiert auf einer Serveranwendung, zu der verschiedene OpenSource-Implementierung für Clients verfügbar sind.
OpenMary ermöglicht es, Texte in gesprochene Audiodaten zu übersetzen. Dafür stehen verschiedene Sprachbibliotheken zur Verfügung. Die Audiodaten können als Stream empfangen oder als MP3/WAV-Datei vom Server heruntergeladen werden. Die verschiedenen Empfangsmöglichkeiten der Audiodaten sind für die zu entwickelnde Applikation sehr wichtig, da so auch die Möglichkeit eines Offline-Modus implementiert werden kann.

Um hier an vorlesbare Informationen zu gelangen, wird ein Wikipedia-Parser geschrieben, der die Webseiten von Wikipedia.org nach einfachen vorlesbaren Seitenteilen filtert. Leider

ist der Aufbau der Wikipedia-Webseiten nicht so trivial, da dieser einfache Parser nicht für alle Wikipedia-Seiten funktioniert. Aus Komplexitätsgründen wird an dem Parser nicht weiter gearbeitet. Dafür wird in der Datenbank die Möglichkeit geschaffen, Texte für POIs manuell zu hinterlegen, die ggf. auch von der OpenMary-Applikation vorgelesen werden können.

2.3 Client

Die zu entwickelnde Anwendung soll die Mobilität des Benutzers unterstützen. Daher muss ein Teil der Anwendung für mobile Endgeräte entwickelt werden. Durch die Verwendung von Webservices bieten sich viele verschiedene betriebssystemunabhängige Implementierungsmöglichkeiten an. Hier wird das Google-Betriebssystem „Android“ verwendet.

2.3.1 Android

Android ist ein mobiles Betriebssystem sowie eine Software-Plattform und basiert auf einem Linux-Kernel. Ein großer Teil der Software ist openSource.
Eine weitere Verbreitung von mobilen Devices, die Android als Betriebssystem verwenden, scheint sehr wahrscheinlich. Während Anfang des Jahres 2009 nur Googles eigenes Mobiltelefon (G1) auf Basis von Android verfügbar war, bieten einige Hersteller, wie Samsung, HTC, SonyErricson jetzt ebenfalls Mobiltelefone mit Android als Betriebssysteme an. Zusätzlich haben auch weitere Hersteller Android-Phones angekündigt. Eine starke Verbreitung dieses Betriebssystems ist daher anzunehmen.
Ein weiterer Grund für die Wahl von „Android“ ist, dass es auf Java-Basis aufsetzt und viele Java-Bibliotheken bereits enthalten sind. Das macht es Entwicklern mit Java-Erfahrung einfacher, sich in die Entwicklung von Android-Applikationen einzuarbeiten.

Zur Entwicklung von Android-Applikationen bietet Google ein SDK (Software Development Kit) kostenlos als Download-Paket an. Für die Entwicklung des Clients wird das Android-SDK1.1 verwendet. Das SDK kann als Plugin in eine Eclipse-Umgebung eingebunden werden. Zusätzlich stellt das SDK weitere nützliche Tools wie den Emulator zur Verfügung, der ebenfalls über Eclipse bedient werden kann.

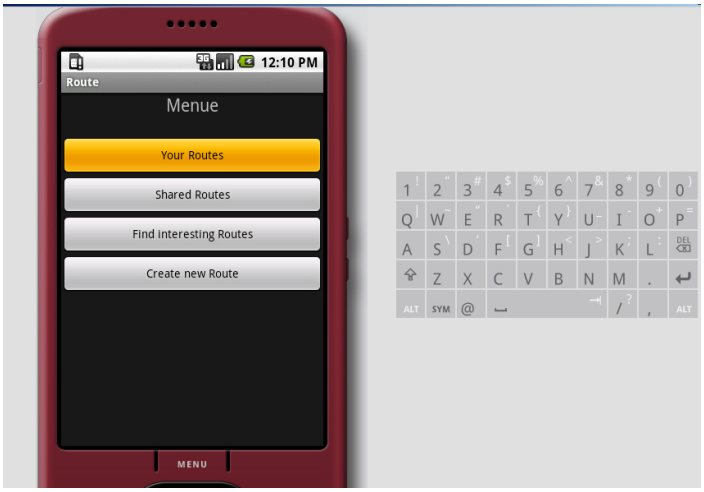

Abbildung 3: Screenshot - Android-Emulator

Die Entwicklung von Android-Anwendungen basiert auf sogenannten „Intents“, die einzelne Aktionen definieren. Für die zu entwickelnde Anwendung sind verschiedene Intents implementiert. Das abstrakte Ablaufdiagramm (s. Abb. 4) zeigt die Unterteilungen, die in der Anwendung als Intents realisiert sind.

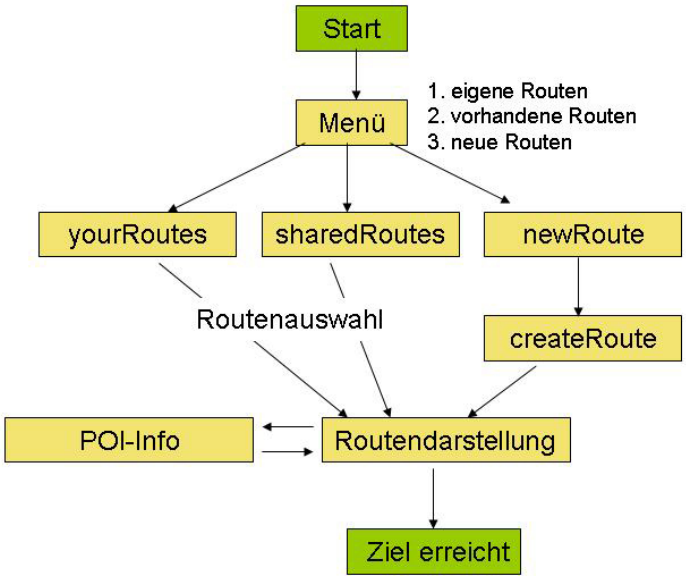

Abbildung 4: Ablaufdiagramm - Clientanwendung

Das Layout wird für jeden Intent als XML-Datei definiert. Somit sind auch wechselnde Layout-Ansichten zum größten Teil als eigenes Intent implementiert.

Update auf Android-SDK1.5
Während der Entwicklung des Clients wurde die neue SDK Version 1.5 von Google zur Verfügung gestellt. Das Update des Android SDK während der Entwicklung verlief nicht problemlos. Anders als bei der Sun-Strategie (Java) ist eine komplette Abwärtskompabilität beim Android-SDK nicht gegeben. Einige Bibliotheken wurden entfernt, verändert oder hinzugefügt. Somit musste der bisherige Code angepasst werden.

2.3.2 Maps, Routen, GPS

Routenberechnung
Zur Routenberechnung wird die Unterstützung von GoogleMaps verwendet. GoogleMaps errechnet für die Applikation die Routen zwischen den POIs. Die Routenoptimierung bei nicht sortierten Routen ist noch nicht implementiert. Bisher werden die POIs nur nach Eingabereihenfolge in die Route eingebaut. (s. dazu Kapitel 3.1)
Für die Routendarstellung kann die in Android integrierte GoogleMaps-API verwendet werden. Die API ermöglicht eine einfache Kartendarstellung und verwendet hierzu das online verfügbare Kartenmaterial von GoogleMaps.
Der Vorteil der API ist, dass man nicht nur die reine Kartendarstellung sondern auch zusätzliche Funktionen von GoogleMaps verwenden kann. Beispielsweise ist eine Adressen-nach-Koordinaten-Umwandlung bereits enthalten, die für die zu entwickelnde Anwendung sehr hilfreich ist. Allerdings ist eine Routendarstellung nur bis zur veralteten Android-Version 0.9 vorhanden. Die Routendarstellung muss daher selbst implementiert werden.
Hierfür werden über GoogleMaps die Routeninformationen ermittelt:

```
http://maps.google.de/maps?f=d&saddr=Berliner Tor, Hamburg
&daddr=Brandenburger Tor,Berlin&output=kml
```

Als Attribut „saddr“ wird die Startadresse, als „daddr“ das Ziel angegeben. Zusätzlich wird das Attribut ouput als „kml“ angegeben. GoogleMaps errechnet die Route und gibt als Ergebnis eine kml-Datei zurück. Diese Datei enthält alle wichtigen Routeninformationen und ist XML-ähnlich aufgebaut. Zur Darstellung einer Routenlinie sind zusätzlich alle notwendigen Geo-Koordinaten enthalten. Diese Geo-Koordinaten werden auf die Display-Koordinaten umgerechnet, als Linie verbunden und somit wird die Route als Linie dargestellt.

GPS-Daten

Um GPS-Daten im Emulator verwenden zu können wird eine GPS-Mouse über Bluetooth mit dem Rechner verbunden. Allerdings ist eine direkte Anbindung zwischen der GPS-Mouse und dem Emulator nicht möglich, da dieser die GPS-Daten nur über eine Telnet-Verbindung empfangen kann. Daher ist der Einsatz eines weiteren Tools notwendig, das die Daten von der GPS-Mouse empfängt und an den Emulator weiterleitet.
Hier existieren einige Tools, die die GPS-Daten über Bluetooth empfangen und eine definierte Weiterleitung der Daten ermöglichen. Allerdings können diese Tools die GPS-Daten nicht an die notwendige Telnet-Verbindung weiterleiten. Für das Projekt wird vorerst die Software GPSGate verwendet. Zusätzlich dazu muss eine Anwendung zur Weiterleitung der GPS-Daten von GPSGate an den Emulator entwickelt werden.
Die Software GPSGate empfängt somit die Daten der GPS-Mouse, die dann per UDP an das selbstentwickelte Tool weitergeleitet werden. Das Tool baut eine Telnet-Verbindung zum Emulator auf und übertragt die GPS-Daten.

Nach der Umsetzung des Tools und einer Testphase stellte sich heraus, dass der Emulator fehlerhaft ist und nur eine Koordinate richtig empfängt. Weitere übertragene Koordinaten werden ignoriert, auch wenn diese manuell per Telnet-Verbindung übermittelt werden. Dieses Problem wird auch in zahlreichen Foren diskutiert, allerdings konnte bisher keine Lösung gefunden werden.

Über eine Telnet-Session können an den Emulator auch weitere Informationen gesendet bzw. Situationen simuliert werden. Beispielweise können über diesen Weg Anrufe oder eingehende Kurzmitteilungen simuliert werden.
Eine verfügbare Hilfsapplikation zur Übertragung verschiedener Telnet-Befehle ist „AnETTe“ (Android Emulator Telnet Tester) die in der Version 0.9 verwendet wird.

3 Fazit

Die Arbeit am ersten Projektteil hat gezeigt, dass die Komplexität der Aufgabe leicht zu unterschätzen ist.
Besonders das Zusammenspiel der vielen verschiedenen Komponenten bietet ständig neue Herausforderungen. Aktivitäten, wie z.B. das Auslesen der GPS-Daten, die zunächst für trivial gehalten wurden, waren viel komplexer als erwartet. Zudem kommen noch einige Probleme durch die Entwicklung mit Android hinzu. Hier merkt man, dass die Android-Umgebung noch in der Anfangsphase (siehe SDK-Update) steckt und nicht für alle Anforderungen, wie z.B. die Routendarstellung, schnelle und einfache Lösungen vorhanden sind.
Im Allgemeinen sind der Infrastrukturaufbau und die damit verbundenen Tests weitestgehend durchgeführt worden.

3.1 Ausblick - Projekt 2

Das Projekt 1 diente dazu, die Infrastruktur für das Projektziel zur Verfügung zu stellen. Im zweiten Teil, Projekt 2, soll auf Basis dieser Infrastruktur das geplante Ziel umgesetzt werden. Folgende Punkte sind im zweiten Teil des Projektes geplant.

Routenberechnung
Die Routenberechnung mit den ausgewählten POIs erfolgt zurzeit nur über die Eingabereihenfolge. Hierfür soll ein Algorithmus erstellt werden, der nicht nur die kürzester-Weg-(Luftlinie)-Reihenfolge zwischen den POIs erstellt. In vielen Fällen mag die Luftlinien-Entfernung auch mit der tatsächlich zu bewältigenden Entfernung übereinstimmen. Allerdings können z.B. Flüsse dafür sorgen, dass die Luftlinien-Entfernung zu einem POI auf der anderen Flussseite sehr gering ist, doch der Weg , der zurückzulegen ist, deutlich höher ist und an anderen POIs vorbeiführt, die dann sinnvollerweise vorher zu besuchen sind. Zusätzlich sollen die verschiedenen Fortbewegungsmittel berücksichtigt werden. Beispielsweise kann eine Route zu verschiedenen POIs mit den öffentlichen Verkehrsmitteln anders sein als mit dem Auto oder zu Fuß.

Anbindung an Bewertungssystem
Um zu vermeiden, das POIs manuell eingepflegt und bewertet werden müssen, sollen POI-Datenbanken mit Bewertungen, wie z.B. Qype, angebunden werden. Dadurch könnte auf eine Vielzahl von POIs mit Bewertung zurückgegriffen werden.

Umsetzung auf Live-System
Die bisherige Entwicklung fand nur im Emulator statt. Deshalb konnten Usability-Aspekte, die für eine solche Applikation wichtig sind, im Feldtest nicht berücksichtigt werden. Im zweiten Teil des Projektes soll eine ausreichende Feldstudie durchgeführt werden, um die Anwendung auch in der Realität bedienbar zu gestalten.

Multimedia-Unterstützung
Im Projekt 1 wurde bereits „OpenMary“ zum Vorlesen von Texten getestet. Im zweiten Teil des Projektes soll die Multimedia-Unterstützung auf den mobilen Devices umgesetzt werden. Ebenfalls soll auch das Anschauen von Videos zur weiteren Information über die erreichten POIs auf einer Route ermöglicht werden.

Web-Site
Um aufwendige Eingaben, wie z.B. die Routeneinrichtung, nicht ausschließlich von mobilen Devices durchführen zu müssen, soll zusätzlich zur Client-Eingabe eine Web-Site erstellt werden, über die Routen-, POI- und Benutzerpflege möglich ist. Dieses erleichtert den Umgang mit der Applikation erheblich, sodass das mobile Device mit seinen typischerweise eingeschränkten Eingabe-/Ausgabemöglichkeiten nur für die Anzeige und kleine Änderungen von Routeninformationen benutzt wird. Die Erstellung bzw. Änderung von Routen kann dann bequem am „heimischen PC“ durchgeführt werden.

Literatur

[1] Android Development Community - anddev.org
. http://www.anddev.org.

[2] GpsGate.com
. http://www.gpsgate.com.

[3] Offizielle Website - PostgreSQL
. http://www.postgresql.de.

[4] Qype. http://www.qype.com.

[5] Anwendung AnETTe
. http://android-telefonie.de/anette-german/.

[6] Apache <Web Services /> Project. Web Services - Axis
. http://ws.apache.org/axis/.

[7] Ed Burnette. *Hello, Android*. 1. Aufl. edition, 2008.

[8] Google Inc. GoogleMaps
. http://maps.google.de.

[9] Google Inc. Offizielle Android-Developer Website
. http://developer.android.com.

[10] Google Inc. Offizielle Android Website
. http://www.android.com/.

[11] Projekt - OpenMary. OpenSource Text-To-Speech-Applikation
. http://mary.dfki.de/.

[12] The Apache Software Foundation. offizielle Apache Tomcat Website
. http://tomcat.apache.org/.