Master Thesis

im Rahmen des
Universitätslehrganges „Geographical Information Science & Systems“

(UNIGIS MSc) am Zentrum für GeoInformatik (Z_GIS)
der Paris Lodron-Universität Salzburg

zum Thema

Individuelle Wegzeitberechnung für „Human Powered Mobility“

Standortbezogene Ankunftszeitprognose für Wanderer anhand eines mobilen Prototyps

vorgelegt von

Dipl. Ingenieur FH Jürg Liechti
U1409, UNIGIS MSc Jahrgang 2008

Zur Erlangung des Grades

„Master of Science (Geographical Information Science & Systems) – MSc(GIS)”

Gutachter:

Ao. Univ. Prof. Dr. Josef Strobl

Köniz, 28.12.2010

Vorwort / Danksagung

Besonders danken möchte ich Peter Gsteiger, der mich während der ganzen Arbeit begleitet und unterstützt hat. Mit ihm hatte ich jederzeit einen kritischen Mitdenker, der mich auf dem ganzen Weg begleitete. Auch Nicolas Lenz möchte ich speziell danken für seine Unterstützung im mathematischen Bereich. Ausserdem möchte ich meinem Arbeitgeber geo7 AG danken, der mich während des ganzen UNIGIS-Lehrganges unterstützt und ein hohes Mass an Arbeitszeitflexibilität geboten hat.

Zudem möchte ich außerordentlich meinen Eltern und Schwiegereltern, sowie auch Barbara Lüthi und Martina Zysset danken, die während der ganzen Zeit viel für die gesamte Familie geleistet und mich dadurch entlastet haben.

Ein grosses Dankeschön gilt auch meinen Mitstudierenden Mary Brown, Tuxa Ayus und Horst Zimmerlein, mit denen ich zusammen viele schöne, lehrreiche (und anstrengende) UNIGIS-Lernstunden verbrachte.

Weiter möchte ich meinen Freunden Bernhard Künzle, Simon Meister, Stefan Liniger und Mathias Schmocker danken, mit denen ich mich beim Joggen oder sonst wo austauschen konnte und die mich motiviert und unterstützt haben weiterzumachen.

Am meisten möchte ich aber meiner Frau Patricia danken, die immer für mich da ist!

Eigenständigkeitserklärung

Ich versichere, diese Master Thesis ohne fremde Hilfe und ohne Verwendung anderer als der angeführten Quellen angefertigt zu haben, und dass die Arbeit in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegen hat. Alle Ausführungen der Arbeit die wörtlich oder sinngemäß übernommen wurden sind entsprechend gekennzeichnet.

Köniz, 28.12.2010

Jürg Liechti

Kurzfassung

Die Distanz, der Höhenunterschied und die individuelle Konstitution sind die wesentlichen Faktoren um Wegzeiten, die mit eigener Muskelkraft zurückgelegt werden (HPM), berechnen zu können. Je nach Fortbewegungsart und Person spielen diese Faktoren eine unterschiedliche Rolle.

Wer sich mit eigener Muskelkraft fortbewegt, interessiert vor allem, wo der geplante Weg durchführt und wann man wo ist. Die möglichst präzise individuell berechnete Ankunftszeit an dem geplanten Zielort wäre sehr hilfreich, damit Pausen und Abfahrtszeiten von öffentlichen Verkehrsmitteln möglichst in die Routenplanung mit einbezogen werden können.

Da die meisten Mobiltelefone heute bereits ein GPS integriert haben, wäre es sehr praktisch, wenn damit die individuelle Wegzeit berechnet werden könnte. Dazu braucht es folgende Grundlagen:

- Idee, wo die Route durchgehen soll (digitale Strecke)
- Digitales Höhenmodell um die Höhendifferenz abzuleiten -> 3D Route
- Mobiles Endgerät mit GPS und geplanter 3D-Route
- Startort und Zeit
- Algorithmus, der mit der GPS-Position und der Dauer anhand der geplanten 3D-Route die Ankunftszeit berechnet.

Ziel dieser Masterarbeit ist es, einen Algorithmus zu formulieren, der die individuelle Wegzeit für muskelgetriebene Fortbewegung berechnet. Der Algorithmus wurde anschliessend in einem Prototyp implementiert und getestet. Bei der Umsetzung des Prototyps wurden die aktuellen technischen Mittel und Vorgehensweisen optimal eingesetzt. Die daraus resultierenden Erfahrungen und Erkenntnisse flossen als Resultat in die vorliegende Arbeit ein.

Stichworte: Wegzeitberechnung, GPS, Android

Abstract

Distance, altitude difference and individual constitution are the main factors to calculate the way time by muscle driven human powered mobility. Depending on the type of human powered mobility and individual person, these factors play a different role.

As an individual person, the interests in muscle driven mobility are focused on, where am I on my scheduled way and when will I reach the target. An individually estimated time of arrival for human powered mobility would be very helpful, so that pauses and schedules of public transportations could be included in the route planning.

Most of the today delivered mobile phones have GPS already integrated, so the basic requirement to calculate the estimated time of arrival for human powered mobility is set. The following fundamentals are also needed:

- Idea where the route should go trough (digital route)
- Digital elevation model where the altitude difference could be derived
- Mobile device with GPS and planned 3D route
- Starting time and starting point
- Algorithm that calculates the time of arrival with the GPS position and the duration based on the planned 3D route.

The aim of this master thesis is to build an algorithm that calculates the estimated time of arrival for human powered mobility. The developed algorithm is implemented and tested in a prototype. With this prototype, today's technical state of the art is optimally implemented. The resulting experiences and findings are covered in this paper.

Keywords: way time calculation, GPS, Android

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

Aufl. = Auflage

Bd. = Band

DHM = Digitales Höhenmodell

ed. = Edition

et al. = und andere

ERD = Entity Relationship Diagram

erw. = erweitert(e)

ETA = Estimated Time of Arrival (vorausberechnete Ankunftszeit)

f. = folgende Seite

ff. = fortfolgende Seiten

GPS = Global Positioning System

HPM = Human Powered Mobility

Hrsg. = Herausgeber

Jg. = Jahrgang

KML = Keyhole Markup Language

Nr. = Nummer

OGC = Open Geospatial Consortium

OMG = Object Management Group

RDBMS = Relationales Datenbank Managementsystem

S. = Seite

s. = siehe

Sp. = Spalte

SRTM = Shuttle Radar Topography Mission

SWW = Schweizer Wanderwege (Dachorganisation)

UML = Unified Modeling Language

uvm. = und viele(s) mehr

WYSIWYG = What you see is what you get (W*as du siehst, ist [das,] was du bekommst)*

XML = Extensible Markup Language

z.B. = zum Beispiel

zit. nach = zitiert nach

1. Einführung

Problemstellung und Motivation

Wandern und Fahrradfahren gehören zu den beliebtesten Freizeitaktivitäten in der Schweiz und gewinnen laufend neue Anhänger. Die Infrastruktur dafür ist überaus gut ausgebaut. Es gibt sehr viele Wegweiser oder Hinweistafeln (s. Abbildung 1), die in der ganzen Schweiz platziert sind und gepflegt werden. Bei den Wanderwegtafeln gibt es meistens sogar einen Hinweis auf die Dauer der zurückzulegenden Strecke. Diese Dauer ist eine Angabe für den „Durchschnittswanderer“ und wird in der Schweiz nach einer komplexen Formel berechnet (s. Kapitel *Wandern*).

Abbildung 1 Wanderwegsignalisationstafel

Vereinfacht gilt folgende allgemeingültige Faustregel:

Alleingeher bzw. Kleingruppen legen in 1 Stunde zurück:

- 400 Höhenmeter im Aufstieg
- 800 Höhenmeter im Abstieg
- 5 km Horizontalentfernung

Größere Gruppen legen in 1 Stunde zurück:

- 300 Höhenmeter im Aufstieg
- 600 Höhenmeter im Abstieg
- 5 km Horizontalentfernung

Nach der offiziellen Norm für die Wanderwegzeitberechnung (DIN 33466, Nr. 4.3.2.) benötigt ein ganz normaler DIN-Wanderer in Deutschland 1 Stunde für 4 Kilometer in der Ebene. Beim Aufstieg braucht er 1 Stunde für 300 Höhenmeter, beim Abstieg schafft er 500 Höhenmeter pro Stunde (vgl. Wegzeitberechnung beim Wandern – Outdoor.de).

Mit all den oben stehenden Näherungswerten erhalten „Durchschnittswanderer" ziemlich gute Wegzeitschätzungen. Jedoch sind die Zeitangaben für Leute mit beispielsweise einer kleinen Gehbehinderung oder für Extremsportler nicht aussagekräftig. Zudem gibt es messbare Eigenschaften wie Körpergrösse, Gewicht, Kondition uvm., die für jede Person individuell sind und die sich auf die Wanderleistung auswirken. Auch nicht messbare Eigenschaften wie z.B. die Tagesform, das Wetter oder Muskelkater spielen für die HPM Fortbewegung eine nicht zu vernachlässigende Rolle.

Das genaue Vorausbestimmen der Wegzeit ist bei allen HPM-Fortbewegungsarten schwierig. Allgemein kann aber gesagt werden, dass im Bereich Wandern und Fahrradfahren Durchschnittszeitangaben oder Erfahrungswerte, für das Zurücklegen bestehender Routen, vorhanden sind. Für alle anderen HPM-Aktivitäten gibt es dagegen kaum Angaben darüber, wie lange es dauert, eine Strecke zurückzulegen.

Der Vergleich zur Wegzeitberechnung im Strassenverkehr mit dem Auto zeigt die unterschiedlichen Ausprägungen:

Eigenschaft	**Fiat Punto S**	**Jürg Liechti (Fahrrad)**
Kraft	40‘000 W	ca. 400 W
Ausdauer/Reichweite	Tankfüllung/500 km	Nahrungs- und Fettspeicher/50 km
Geschwindigkeit	max. 180 km/h	max. 80 km/h
Verbrauch	6 l / 100 km	ca. 3 l Wasser & 350 kcal / 100 km
Gewicht	ca. 1100 kg	ca. 75 kg

Tabelle 1 Vergleich Auto – Mensch

Der Hauptunterschied liegt darin, dass das Durchschnittsauto - im Gegensatz zum Fahrradfahrer oder Wanderer - meistens genügend Kraft hat, alle Steigungen mit konstanter Geschwindigkeit zu überwinden. Somit kann für die Ankunftszeitberechnung im Strassenverkehr mit dem Auto mit einer konstanten Geschwindigkeit, allein abhängig von der Höchstgeschwindigkeit des Strassentyps, gerechnet werden.

Abbildung 2 Geschwindigkeitsbegrenzung (Quelle: Peter Hürzeler)

Die vorliegende Arbeit beschäftigt sich mit den Problemen, die bei der Berechnung der individuellen Wegzeit für muskelgetriebene Fortbewegung entstehen. Um die Grundlagen möglichst realitätsnah verstehen zu können, wurde ein Prototyp erstellt. Somit konnten wichtige Erfahrungen gesammelt werden, welche in die diese Arbeit einflossen.

Hypothesen

Aus den oben erwähnten Problemen für die individuelle HPM - Wegzeitberechnung, wurden folgende Hypothesen abgeleitet:

1. Mit den heutigen technischen und infrastrukturellen Mitteln lässt sich eine dem Individuum angepasstere Wegzeitberechnung als die vorberechneten Zeiten der Signalisationstafeln realisieren.
2. Es kann für (alle) muskelgetriebenen Fortbewegungsarten ein Algorithmus für die individuelle Wegzeitberechnung gefunden werden.
3. Der Benutzer kann zu jedem Zeitpunkt der Fortbewegung die aktuelle, individuell berechnete Ankunftszeit verfügbar haben.

Ziele

Das Ziel dieser Arbeit ist, einen einfachen Prototyp zu realisieren, welcher die individuelle Wegzeit während der muskelgetriebenen Fortbewegung, entlang einer vordefinierten Route, berechnet und aktualisiert. Es sollen Erfahrungen mit GPS, Fortbewegungspausen und anderen Störfaktoren gesammelt werden, um daraus einen möglichst guten Algorithmus für die Ankunftszeit abzuleiten. Dabei sollen die Bestandteile des Prototyps nach den heutigen relevanten IT- und GIS-Standards realisiert werden. Die konkreten Ziele für den Prototyp werden im Kapitel 4. Anforderungen aufgeführt.

Lösungsansatz

Folgende Grundlagen müssen vorhanden sein, um schlussendlich die Ankunftszeit vorausbestimmen zu können:

- GPS-Gerät mit geplanter Strecke (3D)

- Startzeit und Ort (geplante Strecke kann von 2 Seiten her begangen werden)
- Startformel (berechnet erste Ankunftszeit)

Alle muskelgetriebenen Fortbewegungsarten gehen von einer idealen Startformel für die Weg-Zeit-Berechnung aus. Auf der geplanten 3D-Strecke kann jederzeit die aktuelle Position mit der geplanten (mit Startformel berechneten) Position verglichen werden. Anschliessend soll anhand der aktuellen und geplanten Position eine neue Ankunftszeit berechnet werden, welche laufend den aktuellen Gegebenheiten angepasst wird. Die gesammelten Erfahrungswerte (Steigung und Geschwindigkeit pro Fortbewegungsart) können gebraucht werden, um damit ein persönliches Profil zu erstellen, das anschliessend wieder für weitere Vorausberechnungen verwendet werden kann.

Abgrenzung

Der erstellte Prototyp beschränkt sich im Wesentlichen nur auf das Wandern. Es gibt natürlich auch noch andere muskelgetriebene Fortbewegungsmöglichkeiten wie Fahrradfahren, Inlineskaten, Rudern (uvm.) die implementiert werden könnten. Die Grundüberlegungen bleiben aber bei allen HPM-Fortbewegungsarten ausser vielleicht bei den Fortbewegungsarten im Wasser ungefähr gleich.

Auf die GPS-Genauigkeit der Position wird bei der Arbeit nicht eingegangen. Die absolute Position spielt bei der Ermittlung der Weg-Zeit keine so wichtige Rolle. Wichtiger ist die relative Position auf der geplanten Route.

Es ist nicht Ziel dieser Arbeit, eine marktreife Applikation zu erstellen. Vielmehr sollen bei der Realisierung des Prototyps Erfahrungen und Ideen gesammelt und dokumentiert werden.

Es geht auch nicht darum, einen möglichst genauen Startzeit-Algorithmus (Startformel) zu finden, als vielmehr einen optimalen Algorithmus, welcher die individuelle Ankunftszeit bestmöglich berechnet.

Zielpublikum

Das Zielpublikum sind in erster Linie Lösungsverantwortliche, Software-Architekten und Ingenieure die im Bereich HPM arbeiten. Die hier erarbeiteten Ideen und Überlegungen könnten als Grundlage für eine spätere Produktimplementierung dienen.

Struktur der Master Thesis

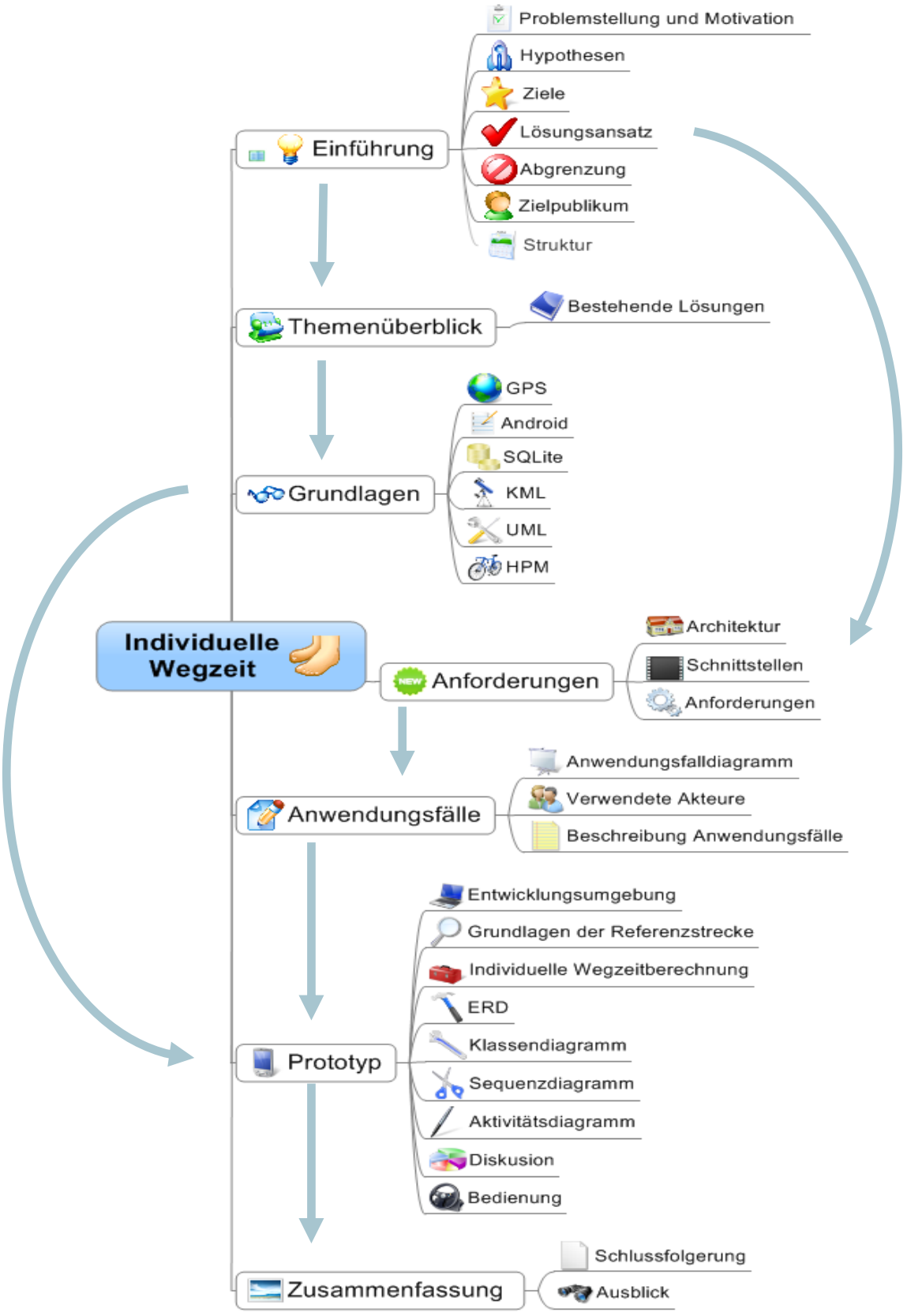


Abbildung 3 Struktur der Master Thesis

2. Themenüberblick

"Die Wissenschaft sucht, allgemeine Regeln aufzustellen, die den gegenseitigen Zusammenhang der Dinge und Ereignisse in Raum und Zeit bestimmen. Für diese Regeln beziehungsweise Naturgesetze wird allgemeine und ausnahmslose Gültigkeit gefordert. (...) Die Tatsache, dass wir aufgrund solcher Gesetze den zeitlichen Verlauf von Erscheinungen auf gewissen Gebieten mit großer Genauigkeit und Sicherheit vorherzusagen vermögen, sitzt tief im Bewusstsein des modernen Menschen, selbst wenn er vom Inhalt jener Gesetze sehr wenig erfasst hat." (Einstein 1979)

Das gewählte Thema steht in starkem Zusammenhang von Raum, Zeit und deren Voraussagungen. Der Anspruch auf eine hundertprozentige Genauigkeit der Vorhersage wird aber klar nicht erhoben. Ziel soll es vielmehr sein, den Inhalt von bestimmten Naturgesetzen soweit kennenzulernen und zu verstehen, um die individuelle Wegzeitberechnung mit einer gewissen Verlässlichkeit voraussagen zu können.

Bestehende Lösungen

Es bestehen in erster Linie Planungs- und Dokumentationswerkzeuge für die muskelgetriebene Fortbewegung (HPM). Im Zusammenhang mit GPS ist es sehr einfach, einen Track der zurückgelegten Route aufzuzeichnen. Die aufgezeichneten Routen können mit den heutigen technischen Mitteln einfach ausgetauscht werden. Es ist aber auch möglich, im Voraus die Wanderung digital zu planen. Hier nun ein Überblick der vorhandenen Möglichkeiten.

Planungswerkzeuge

Es gibt verschiedenste mögliche Planungswerkzeuge. Diese werden vor der Fortbewegung, meistens auf Basis einer Karte, eines Wegnetzes oder eines Ausgangs- und Zielorts, zur Planung verwendet.

In Google Earth kann eine selbst digitalisierte Linie via KML-Export ausgegeben und weiterverarbeitet werden. In Google Maps kann die Route mit Start- und Zielort für Auto, öffentlicher Verkehr und zu Fuss automatisch berechnet werden. Die Berechnung für Fussgänger berücksichtigt die Höhe und gibt beim Auf-, respektive

Abstieg unterschiedliche Zeiten an. Aus Google Maps gibt es keine Exportmöglichkeit der Route.

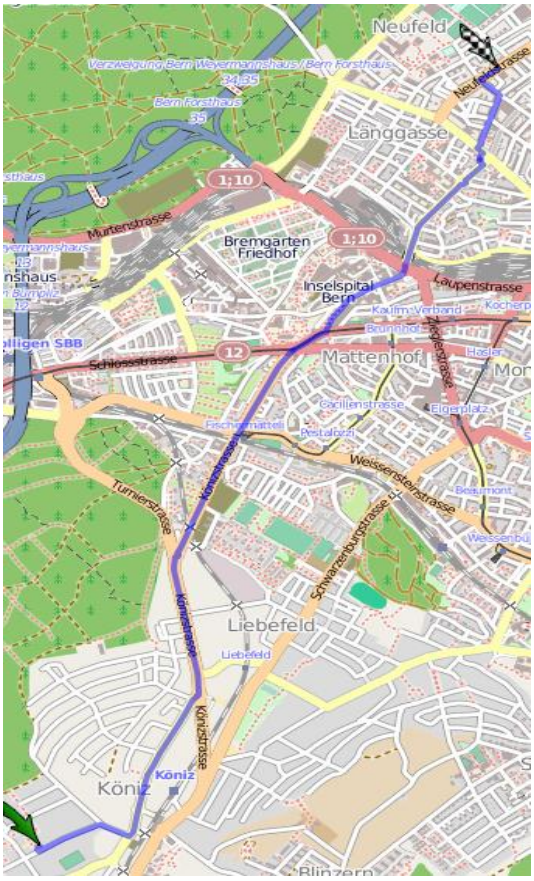
OpenRouteService.org ist ein klassischer Routenplaner, der eine Berechnung von Start- zu Zielort für Auto, Fussgänger (kürzester Weg) und für das Fahrrad (kürzester Weg, Mountainbike, Rennrad, sicherster Weg, bevorzugter Weg) berechnet. Es wird ein Höhenprofil der geplanten Strecke angeboten sowie ein Routen-Link, GPX und XML Export. Die Berechnung der Zeit berücksichtigt keine Höheninformationen. Leider enthalten auch die angebotenen Exporte keine Höheninformationen.

Das folgende Bild zeigt einen Vergleich der zwei erwähnten Routenplaner für die Referenzstrecke (Fahrrad).

Google Maps

Total Distanz: 4.7km Total Zeit: 57 min

OpenRouteService



Total Distanz: 5km Total Zeit: 58 min

Im Online-Werkzeuge www.wanderprofi.ch kann direkt auf dem Wanderwegnetz die individuelle Wanderung zusammengestellt werden. Als Ausgabe kann zwischen GPS-Track, KML oder PDF mit Karte, Höhenprofil und Routenbeschreibung gewählt werden.

Die Zeiten werden mit der Formel (s. Kapitel *Wandern*) der SWW berechnet.

Auch von Swisstopo gibt es unter dem Namen „Swiss Map Mobile“ in Zusammenarbeit mit Schweiz Mobil Lösungen, um die Langsamverkehrs-Routen und Pixelkarten auf mobile Endgeräte verfügbar zu machen (Swiss Map Mobile – Swisstopo). Nach direkten Abklärungen mit Swisstopo gibt es noch kein Bestreben, eine individuelle Wegzeitberechnung zu integrieren.

Mit ape@map und MagicMaps2Go können Kartendaten auf mobilen Endgeräten dargestellt werden. Touren können geladen und zur Navigation verwendet werden. Tracks können aufgezeichnet, verwaltet und geöffnet werden. Statistiken über die Zeit, Geschwindigkeit, Dauer, Weg/Höhenmeter, Track-Höhenmeter und Track-Länge können erstellt werden. Es werden jedoch keine Angaben über die Wegzeit gemacht.

Weiter gibt es noch diverse Desktop-Planungswerkzeuge wie den Rad.RoutenPlaner oder den ADAC Reiseplaner. Diese Planungswerkzeuge bieten komfortable Planungswerkzeuge mit diversen vorkonfigurierten Routenvorschlägen. Zudem gibt es Export-Schnittstellen für mobile Endgeräte damit die geplanten Routen in einem GPS Gerät geladen werden können. Daneben bieten diese Produkte noch zahlreiche Zusatzinformationen wie Restaurants oder Übernachtungsmöglichkeiten an.

Die meisten Mobiltelefonhersteller bieten mittlerweilen auch diverse Dienste für die Navigation an. So stellt z.B. Nokia auf dem OVI Portal (http://maps.ovi.com) weltweit Kartenmaterial von Navteq zur Verfügung. Dies kann auch zur Fussgängernavigation verwendet werden. Alleine auf der Tatsache, dass der Hinweg mit der gleichen Zeit angegeben wir wie der Rückweg, lässt sich schliessen, dass die Höheninformationen nicht einbezogen werden und somit keine individuelle Wegzeitberechnung stattfindet.

Zusammenfassend kann gesagt werden, dass alle Planungswerkzeuge, welche die Wegzeit berechnen, dies anhand bestehender Formeln bewerkstelligen. Es existiert jedoch nirgends eine Lösung, welche die individuelle Wegzeit berücksichtigt.

Dokumentationswerkzeuge

Unter Dokumentationswerkzeugen werden alle Lösungen bezeichnet, die bestehende z.B. mit GPS aufgezeichnete Routen bearbeiten, erweitern, speichern, konvertieren

und verwalten können. Im Gegensatz zu den Planungswerkzeugen, die vor der Fortbewegung die Route planen, wird bei den Dokumentationswerkzeugen im Nachhinein eine Route dokumentiert.

Unter GPS-Tracks.com können, wie der Name schon sagt, GPS-Tracks hoch - und heruntergeladen werden. In den GPS Daten sind zum Teil auch die Wegzeiten mit aufgezeichnet. Jedoch können diese Wegzeiten nur als grobe Abschätzung verwendet werden, da es zu viele Unsicherheitsfaktoren gibt, um eine repräsentative Aussage machen zu können.

Auch unter Alpinetouren.com können GPS-Daten hoch und runtergeladen werden. Es werden diverse Informationen zu der Tour informativ dargestellt. Zur Wegzeit gibt es eine Angabe, die vom Verfasser eingegeben werden kann.

Im Gipfelbuch.ch werden Touren beschrieben und auf einer Karte dargestellt. Es gibt keine GPS Daten oder andere Exportformate. Alle Daten werden vom Verfasser eingegeben, so auch Wegbeschreibung, Höhenmeter, Verhältnisse und Wegzeiten.

Während dieser Arbeit wurde vor allem mit dem Onlineportal www.gpsies.com gearbeitet (vgl. Lau 2008). Dort können geplante Routen direkt in der Karte digitalisiert werden. Weiter können aufgezeichnete Tracks einfach hochgeladen werden. Zudem unterstützt die Plattform diverse Ausgabeformate mit Höheninformationen. Es gibt sogar auch eine Darstellung für mobile Browser unter m.gpsies.com. Außerdem gibt es spezifische GPSies App‘s für das iPhone und Android. Leider ist die Version für Android 1.5 noch nicht so stabil und konnte daher nicht getestet werden.

Zusammenfassend kann gesagt werden, dass bei den Dokumentationswerkzeugen die Wegzeiten grösstenteils aus Erfahrungswerten bestehen. Diese geben zumindest einen Anhaltspunkt, sind jedoch hier noch schwieriger einzuschätzen als bei den Planungswerkzeugen, da sie immer von anderen Personen erfasst worden sind. Bei den Planungswerkzeugen basieren die Zeitangaben auf verschiedenen Berechnungsalgorithmen. Die vorhandenen Werkzeuge bieten aber keinesfalls eine individuelle Wegzeitberechnung für HPM an.

3. Grundlagen

Um die aufgestellten Hypothesen untersuchen zu können, mussten einige Grundlagen genauer betrachtet werden. Die nachfolgenden Kapitel bilden die Basis, auf denen die Anforderungen des Prototyps implementiert werden.

GPS, Global Positioning System

GPS, das in den 1970er Jahren vom US-Verteidigungsministerium entwickelt wurde, ist ein globales Satellitennavigationssystem, das zur Positionsbestimmung und Zeitmessung verwendet werden kann. Seit dem 17. Juli 1995 ist das System voll funktionsfähig und stellt, seit der Abschaltung der künstlichen Signalverschlechterung im Mai 2000, auch für zivile Zwecke eine Ortungsgenauigkeit in der Größenordnung von ca. 10 Meter sicher. Durch die Differenzmethoden (dGPS) lässt sich die Genauigkeit sogar auf Zentimeter steigern.

GPS zeichnet sich vor allem durch folgende Eigenschaften aus (vgl. Thurston et al. 2003):

- 24 Stunden weltweiter Betrieb
- Extrem genaue, dreidimensionale Positionsinformation
- Extrem genaue Geschwindigkeitsinformation
- Präzise Zeitangaben
- Weltweit bekanntes Koordinatengitter (WGS-84), das einfach in ein lokales Koordinatensystem konvertiert werden kann.
- Erreichbar für eine unbeschränkte Anzahl Benutzer

Positionsbestimmung

Die GPS Satelliten strahlen ständig ein kodiertes Radiosignal mit ihrer aktuellen Position und der genauen Uhrzeit aus. Aus den Signallaufzeiten kann der GPS-Empfänger seine eigene Position und Geschwindigkeit berechnen. Theoretisch reichen die Signale von drei Satelliten (s. Abbildung 4) aus, um die genaue Position und Höhe

auf der Erde bestimmen zu können.

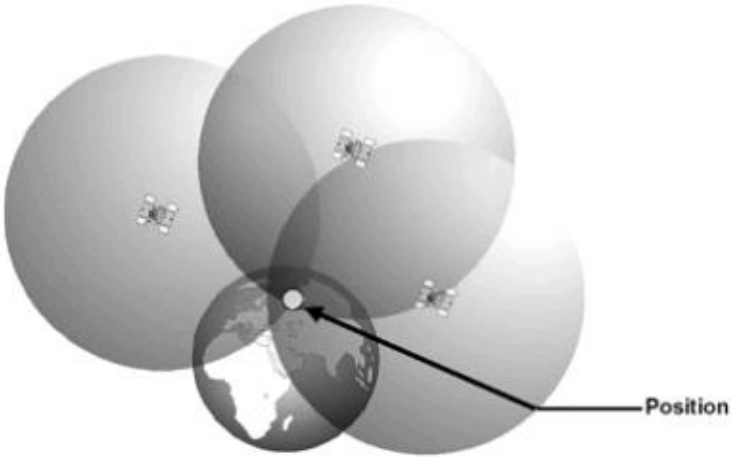


Abbildung 4 Positionsbestimmung mit 3 Satelliten (Quelle: Zogg 2004)

Aus Kostengründen haben die GPS–Empfänger aber keine genügend genaue Uhr, um die Laufzeiten korrekt messen zu können. Deshalb wird das Signal eines vierten Satelliten benötigt, mit dem die genaue Zeit im Empfänger bestimmt werden kann. Die folgende Abbildung zeigt grafisch die Auswirkung des Uhrenfehlers, wobei die Darstellung auf die zweite Dimension reduziert wurde.

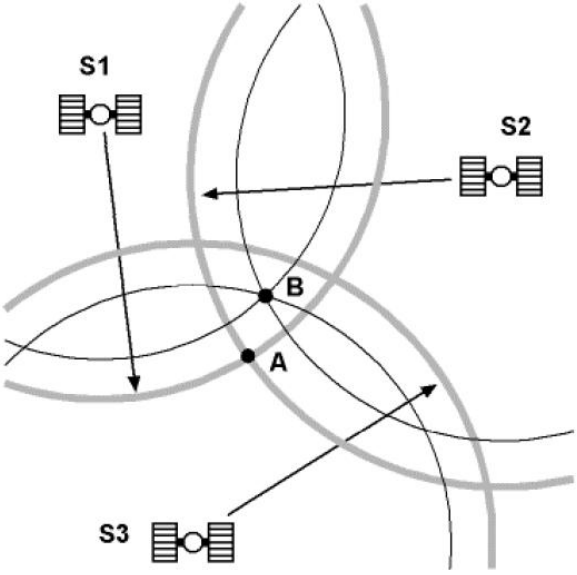


Abbildung 5 Auswirkung des Uhrenfehlers (Quelle: Hurn 1989)

Der Schnittpunkt A ergibt sich durch die Messung der Pseudostrecken vom Benutzersegment zum Satellit 1 und zum Satellit 2. Aufgrund der verfälschten Messungen durch den Uhrenfehler weicht der ermittelte Schnittpunkt A jedoch vom

wahren Schnittpunkt B ab. Kommt ein weiterer Satellit hinzu, kann der Uhrenfehler beseitigt werden, obwohl vorerst kein eindeutiger Schnittpunkt erzeugt werden kann. Die Algorithmen der GPS-Empfänger verändern die Distanzen der Uhrenfehler solange, bis sich die Kreise genau in einem Punkt treffen und die wahre Position (Schnittpunkt B) wiedergeben (vgl. Hurn 1989).

Damit ein GPS-Empfänger immer zu wenigstens vier Satelliten Kontakt hat, werden insgesamt mindestens 24 Satelliten eingesetzt. Sie umkreisen die Erde jeden Tag zweimal in einer mittleren Bahnhöhe von 20‘200 km. Auf den sechs Bahnebenen bewegen sich mindestens vier Satelliten die 55° gegen die Äquatorebene geneigt und gegeneinander um jeweils 60° verdreht sind. Ein Satellit ist damit zweimal in 23 Stunden 55 Minuten und 56,6 Sekunden über demselben Punkt der Erde (U.S. Naval Observatory).

Die Lebenserwartung eines GPS-Satelliten ist ca. 7,5 Jahre. Jedoch funktionieren die Satelliten häufig deutlich länger. Um Ausfälle problemlos zu verkraften, wurden daher bis zu 31 Satelliten in den Orbit gebracht, so dass man auch bei schlechten Bedingungen fünf oder mehr Satelliten verwenden kann. Der aktuelle Status der GPS-Satelliten kann auf der Internetseite vom U.S. Coast Guard Navigation Center entnommen werden.

Formate

Da GPS ein weltweites System ist, wird das Koordinatensystem WGS-84 verwendet. Das World Geodetic System 1984 (WGS 84) ist ein geodätisches Referenzsystem für Positionsangaben auf der Erde und im erdnahen Weltraum. Das System besteht aus einem Referenzellipsoid, das in seiner Einfachheit bestmöglich der Erdoberfläche angepasst ist. Die Kommunikation zu einem GPS-Empfänger basiert meistens auf dem Protokoll NMEA 0183 (National Marine Electronic Association). Für den umzusetzenden Prototyp kann direkt auf eine GPS-Bibliothek, die WGS-84 Koordinaten liefert, zugegriffen werden.

Android

Android ist ein Betriebssystem wie auch eine Software-Plattform für mobile Geräte, die von der Open Handset Alliance entwickelt wird. Als Basis wird der Linux Kernel 2.6 verwendet. Android ist quelloffen und frei verfügbar.

Bedient wird Android über einen Touchscreen und eine Anzahl an definierten Hardwaretasten. Mindestanforderung bilden die Tasten „Home“, „Menü“ und „Zurück“.

Der Linux Kernel von Android ist zuständig für die Speicherverwaltung, Prozessverwaltung und die Netzwerkkommunikation. Zudem beinhaltet er die Gerätetreiber des Systems und bildet die Hardwareabstraktionsschicht für den Rest der Software. Wichtige Bausteine bilden die virtuelle Maschine Dalvik und die dazugehörigen Android-Java-Klassenbibliotheken, die auf der Basis von Java entwickelt wurden (vgl. Becker et al. 2010).

Zum Programmieren von eigenen Android-Anwendungen bietet das Entwicklungssystem ca. 1448 Java Klassen und 394 Schnittstellen. Hier ein Überblick der vorhandenen Bibliotheken.

Klassenbibliothek	Beschreibung	Anzahl Klassen	Anzahl Schnitt-stellen	Ursprung
Android	plattformspezifisch	500	126	Google Inc.
com.google.android	plattformspezifisch	11	2	Google Inc.
Java	Standardklassen, Teil des JDK	612	150	Apache Harmony-Projekt
Javax	Standardklassen, Teil des JDK	145	51	Apache Harmony-Projekt
Junit	Test-Framework	12	3	JUnit-Project

org.apache.commons	allgemein verwendbare Funktionen	128	20	Apache Commons-Projekt
org.bluez	Bluetooth-Unterstützung	15	6	BlueZ-Projekt (ursprünglich entwickelt von Qualcomm)
org.json	Klassen zur Verarbeitung von Daten im JSON-Format	5	0	Douglas Crockford
org.w3c.dom	Klassen zur Manipulation des DOM	1	17	World Wide Web Consortium
org.xml.sax	Klassen zum Einlesen von XML-Daten	19	19	ursprünglich von David Megginson, siehe SAX

Tabelle 2 Android Klassen (Quelle: Android (Klassen und Schnittstellen) – Wikipedia)

Die momentan aktuelle Android Version ist 2.2 mit dem Namen Froyo (Frozen Yogurt). Einen aktuellen Überblick über die Versionen mit den wesentlichen Neuerungen erhält man unter Wikipedia (Android (Versionsverlauf) – Wikipedia).

Um die Hypothesen zu prüfen, ist vorgesehen, ein Prototyp auf Android 1.5 (vorhandene Hardware) mit Java zu implementieren.

SQLite

SQLite ist eine prozessinterne Bibliothek, die mit Android mitgeliefert wird. Sie ist eine selbständige, serverlose, konfigurationsfreie, ausführende SQL - Datenbank - Engine.

Der Code von SQLite ist lizenzfrei und dadurch frei zur Benutzung für jeglichen Zweck, ob kommerziell oder privat.

Anders als bei den meisten anderen SQL - Datenbanken, hat SQLite keinen eigenen Server Prozess. SQLite liest und schreibt direkt auf gewöhnlichen Festplattendateien. Eine komplette SQL - Datenbank mit mehreren Tabellen (tables), Indexen (indices), Trigger und Sichten (Views) ist in einer einzigen Datei enthalten. Das Datenbank Dateiformat ist plattformunabhängig. Es kann frei zwischen 32-Bit und 64-Bit Systemen oder zwischen big-endian (UTF-16)(Englisch) und little-endian (UTF-16l)(Englisch) Architekturen kopiert werden.

Der Einsatz für SQLite ist vor allem praktisch als Ersatz zu herkömmlichem Dateihandling, da die abgelegten Daten sehr einfach mit SQL abgefragt und verändert werden können. Ein echtes RDBMS jedoch kann SQLite nicht ersetzen (vgl. Owens 2006).

Weitere wichtige Eigenschaften von SQLite (vgl. SQLite):

- Transaktionen sind ACID (atomic, consistent, isolated and durable) und das auch nach Systemabstürzen und Energieausfällen.
- Enthält fast alle Standards von SQL-92 (Date et al. 1999)
- Unterstützt terabytegroße Datenbanken, gigabytegroße Zeichenketten und "blobs"
- Kleine Codegrößen: weniger als 300 KB (Englisch) voll konfiguriert oder weniger als 180 KB beim Weglassen von optionalen Fähigkeiten.
- Unabhängiger Kommandozeilen-Schnittstellen (CLI = command-line interface) Client
- Keine äußeren Abhängigkeiten
- Schneller als andere beliebte Client/Server Datenbanken Engines bei fast allen allgemeinen Operationen.

Im geplanten Prototyp sollen Daten, die persistent gespeichert werden müssen, in eine SQLite Datenbank abgelegt werden.

KML

KML ist eine Auszeichnungssprache, ursprünglich von der Firma „Keyhole Corp." entwickelt, zur Beschreibung von Geodaten. Ende Oktober 2004 wurde Keyhole Corp. und somit KML von Google übernommen. KML wurde bis zur Version 2.2 von Google weiterentwickelt. Im April 2008 wurde die Version 2.2 vom Open Geospatial Consortium (OGC) als Standard anerkannt und wird seitdem auch von diesem unterhalten.

In KML-Dokumenten können Geodaten, sowohl in Vektor wie auch in Rasterform abgebildet werden. Als Placemark-Elemente werden Vektorobjekte wie Punkte, Linien, lineare Ringe oder Polygone modelliert, als GroundOverlay-Elemente die Luft- und Satellitenbilder.

Um Routen für den Prototyp zu definieren, werden ausschliesslich KML Dateien mit Vektorobjekten verwendet. Deshalb wird der Hauptfokus nur auf die Vektordefinitionen gelegt.

Neben der Geometrie können die Vektorelemente Placemark (KML - OGC 2008:58-60) auch Name, Beschreibung, vordefinierten Stil, Betrachtungswinkel und -höhe, einen Zeitstempel, aber auch beliebige untypisierte oder typisierte Daten beinhalten. Für das Abbilden einer Route wird der Geometrietyp LineString (KML - OGC 2008:83-85) verwendet. Als geodätisches Referenzsystem wird in KML-Dokumenten ausschließlich das World Geodetic System 1984(WGS-84) (KML - OGC 2008:14-15) verwendet. Somit werden alle Koordinaten, zur Definition einer Route, mit geografischer Länge und Breite sowie Höhe über Meer, angegeben. Die Höhe bezieht sich dabei auf das WGS84 EGM96 Geoid. Jedoch muss gerade bei der Höhe genau betrachtet werden, mit welchem Höhenmodell die Höhe für das KML digitalisiert wird. So wurde für den Prototyp die Route auf der Onlineplattform www.gpsies.com digitalisiert. Dort wird das STRM Version 2.1 Höhenmodell verwendet, das auch auf dem Geoid EGM96 basiert. Folgende Abbildung zeigt die Grundstruktur von KML.

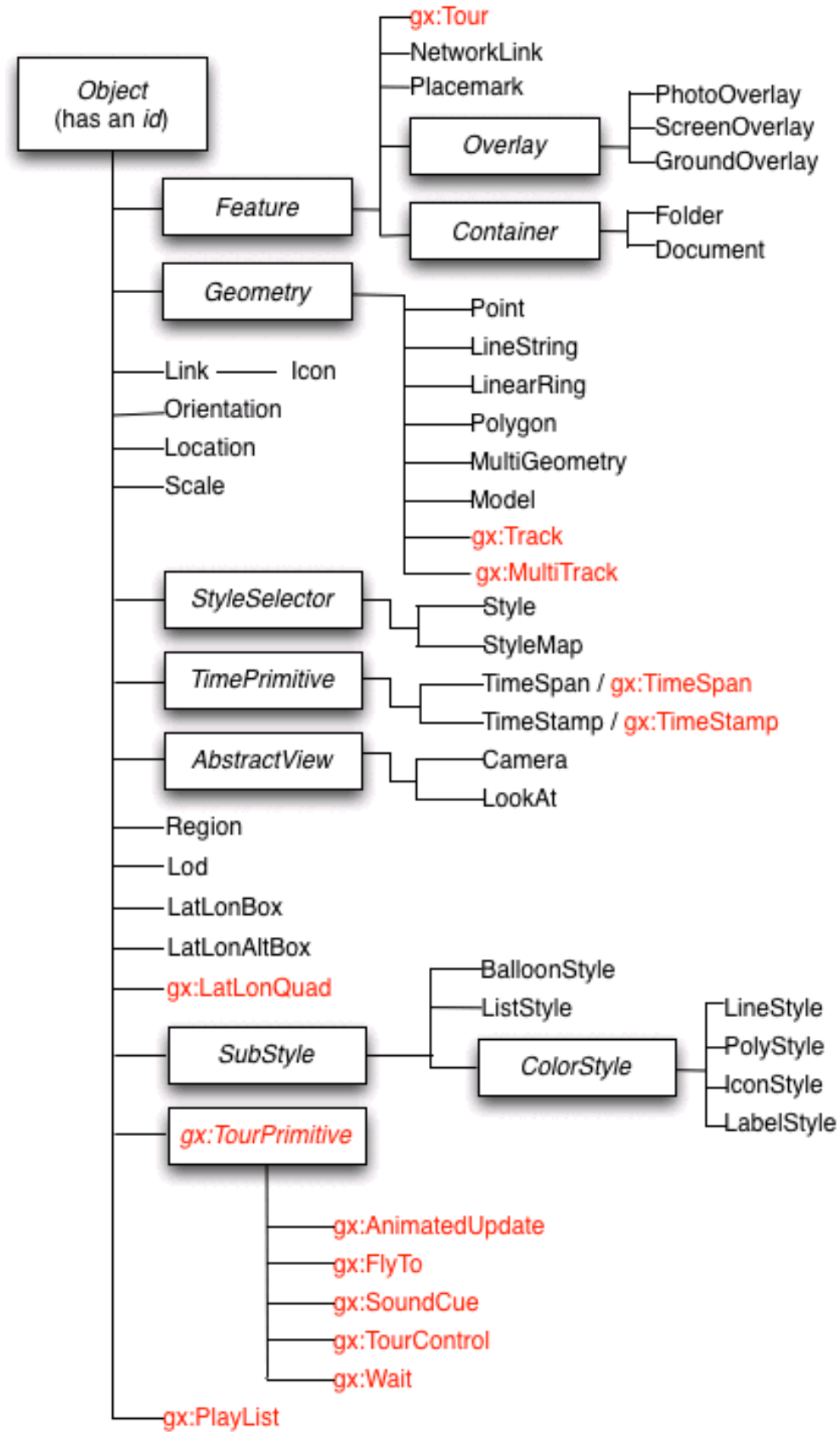


Abbildung 6 Überblick KML (Quelle: KML - Google Code)

UML

Die Unified Modeling Language (UML -OMG) ist eine Sprache und Notation zur Spezifikation, Konstruktion, Visualisierung und Dokumentation von Modellen für Softwaresysteme. Die UML berücksichtigt die gestiegenen Anforderungen bezüglich der Komplexität heutiger Systeme, deckt ein breites Spektrum von Anwendungsgebieten ab und eignet sich für konkurrierende, verteilte, zeitkritische, sozial eingebettete Systeme uvm.

Mit UML ist es möglich, die Modellierung der Lösung unabhängig von der späteren konkreten Umsetzung zu entwerfen und zu dokumentieren. Technische Details wie Betriebssystem, Programmiersprache oder Datenbanksystem werden dabei bewusst ausgeklammert. Es soll eine gemeinsame Diskussionsbasis zwischen Entwickler, Anwender und Auftraggeber geschaffen werden (vgl. Oestereich et al. 2009).

UML wird in dieser Arbeit zur formalen Beschreibung mit verschiedenen, ausgewählten UML-Diagrammtypen angewendet (vgl. Pilone 2006). Folgende Diagramme wurden für die Beschreibung des Prototyps ausgewählt:

Anwendungsfalldiagramm

Das Anwendungsfalldiagramm beschreibt eine Menge von Aktivitäten eines Systems aus Sicht seiner Akteure, die zu einem wahrnehmbaren Ergebnis führen. Anwendungsfälle werden immer durch einen Akteur initiiert. Ein Anwendungsfall ist eine komplette, unteilbare Beschreibung, welche im Diagramm zur besseren Lesbarkeit sehr knapp gehalten wird (vgl. Oestereich et al. 2009). Die Anwendungsfälle werden anschliessend in tabellarischer Form noch detaillierter beschrieben und es wird ein Bezug zu den definierten Anforderungen hergestellt. Wie detailliert die Beschreibung der Anwendungsfälle ausgeführt wird, ist von Projekt zu Projekt unterschiedlich.

Für diese Arbeit wurde die Beschreibung bewusst relativ kurz gehalten, da ein Prototyp implementiert wird, der in erster Linie die Kernfunktionalität unter Beweis stellen soll.

Mit den definierten Anwendungsfällen lässt sich anschliessend einfach überprüfen, ob alle Anforderungen direkt oder indirekt erfüllt werden. Am Ende der Implementierung

können die Anforderungen anhand der Anwendungsfälle in ein Testprotokoll integriert und getestet werden.

ERD / Klassendiagramm

Das Klassendiagramm wird verwendet, um die verschiedenen Klassen, deren Aufbau und ihre Beziehungen, zu beschreiben. ERD und Klassendiagramme werden hier gleich behandelt, obwohl je nach Notation gewisse Unterschiede bestehen. Mit einem UML-Klassendiagramm können, die für die objektorientierte Softwareentwicklung typischen Eigenschaften wie Vererbung, Kapselung und Polymorphismus, dargestellt werden. Es ist auch möglich, ein Klassendiagramm zur Beschreibung von Datenmodellen zu verwenden. Die Rolle der Klassen wird dabei durch die der Entitäten übernommen und Relationship-Attribute werden mit Hilfe von Assoziationsklassen modelliert.

In der vorliegenden Arbeit ist das Entity-Relationship-Diagramm, das zur Datenmodellierung verwendet wird, sehr einfach, da es keine Beziehungen, sondern nur einfache Entitäten aufweist.

Sequenzdiagramm

Das Sequenzdiagramm zeigt eine bestimmte Sicht auf die dynamischen Aspekte des modellierten Systems. Beim Sequenzdiagramm steht der zeitliche Verlauf der Nachricht im Vordergrund. Die Objekte werden mit einer senkrechten Lebenslinie gezeigt. Die Zeit verläuft dabei von oben nach unten (vgl. Oestereich et al. 2009).

Um ein besseres Verständnis zu erlangen, werden komplexe Abläufe in der vorliegenden Arbeit mittels Sequenzdiagrammen abgebildet.

Aktivitätsdiagramm

Auch das Aktivitätsdiagramm wird für die Modellierung dynamischer Aspekte verwendet. Ein Aktivitätsdiagramm stellt die Vernetzung von elementaren Aktionen und deren Verbindungen mit Kontroll- und Datenflüssen grafisch dar (vgl. Oestereich et al. 2009).

Komplexe Ausschnitte des Systems werden zur besseren Verständlichkeit mit einem Aktivitätsdiagramm abgebildet.

HPM (Human Powered Mobility)

Human Powered Mobility ist Fortbewegung durch eigene Muskelkraft. Die aufgestellten Hypothesen basieren genau auf diesen Fortbewegungsarten. Je nach Fortbewegungsart kommt man schneller oder langsamer und effizienter oder ineffizienter vorwärts. In der folgenden Abbildung wird die Effizienz nach Fortbewegungsmittel zur Veranschaulichung dargestellt.

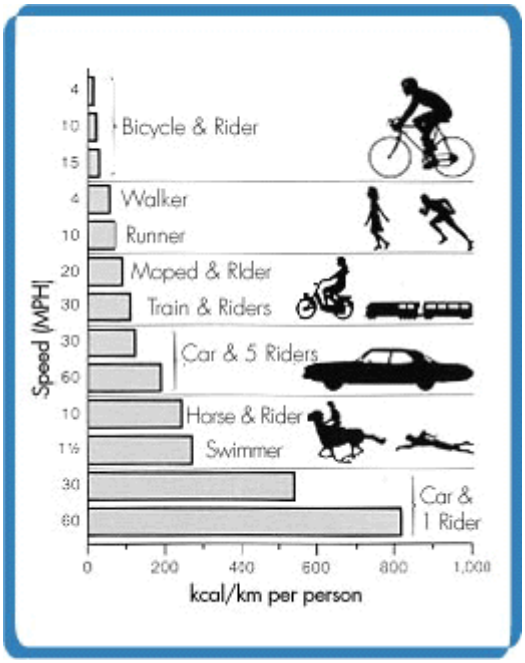


Abbildung 7 Energieverbrauch nach Fortbewegungsmittel (Quelle: Wilson 2004: 166)

Fahrradfahren und Wandern gehören zu den effizientesten Fortbewegungen! In der vorliegenden Arbeit ist jedoch nicht die Effizienz vordergründig, vielmehr ist die Geschwindigkeit je nach Steigung und Fortbewegungsart zentral. Hier wurde vor allem die Fortbewegungsart „Wandern“ genauer betrachtet.

Wandern

Die nationale Dachorganisation der Schweizer Wanderwege entwickelte eine Gehzeit-Formel, die für den Durchschnittswanderer gilt und mit der alle Zeiten der Wanderwegsignalisationstafeln berechnet werden:

$$t = (d * (14.271 + 0.36992 * s + 0.025922 * s^2 - 0.0014384 * s^3 + 0.000032105 * s^4 + 8.1542E-06 * s^5 - 9.0261E-08 * s^6 - 2.0757E-08 * s^7 + 1.0192E-10 * s^8 + 2.8588E-11 * s^9 - 5.7466E-14 * s^{10} - 2.1842E-14 * s^{11} + 1.5176E-17 * s^{12} + 8.6894E-18 * s^{13} - 1.3584E-21 * s^{14} - 1.4026E-21 * s^{15})) / 1000$$

Gehzeit Berechnungsformel (Quelle: Gehzeit Formel – Schweizer Wanderwege 2007)

t = Gehzeit [min]

d = Horizontaldistanz [m]

s = Steigung [%]

Diese Gehzeit Formel wird im Prototyp als Wander-Startformel verwendet und wird schlussendlich in der Tabelle „Formel“ abgebildet. Die Gehzeit Formel wird an dieser Stelle nicht genauer auf Korrektheit und mathematische Feinheit untersucht. Sie wurde jedoch minimal in der Darstellung angepasst.

Allgemein für alle HPM-Fortbewegungen wird der Begriff Wegzeit als Zeitangabe für das Zurücklegen eines definierten Weges verwendet. Für den spezifischen Fall Wandern, wird der Begriff Gehzeit verwendet, der mit Wegzeit gleichzusetzen ist.

Der wichtigste Parameter bei der Berechnung der Gehzeit ist die Steigung. In der Formel vom SWW wird die Steigung bei -40% und 40% begrenzt. Dies ist sicher sinnvoll, denn bei Steigungen über 40% oder unter -40% kann man nicht mehr von Wandern sprechen. Visualisiert man die Formel vom SWW, so ergeben sich folgende Abbildungen:

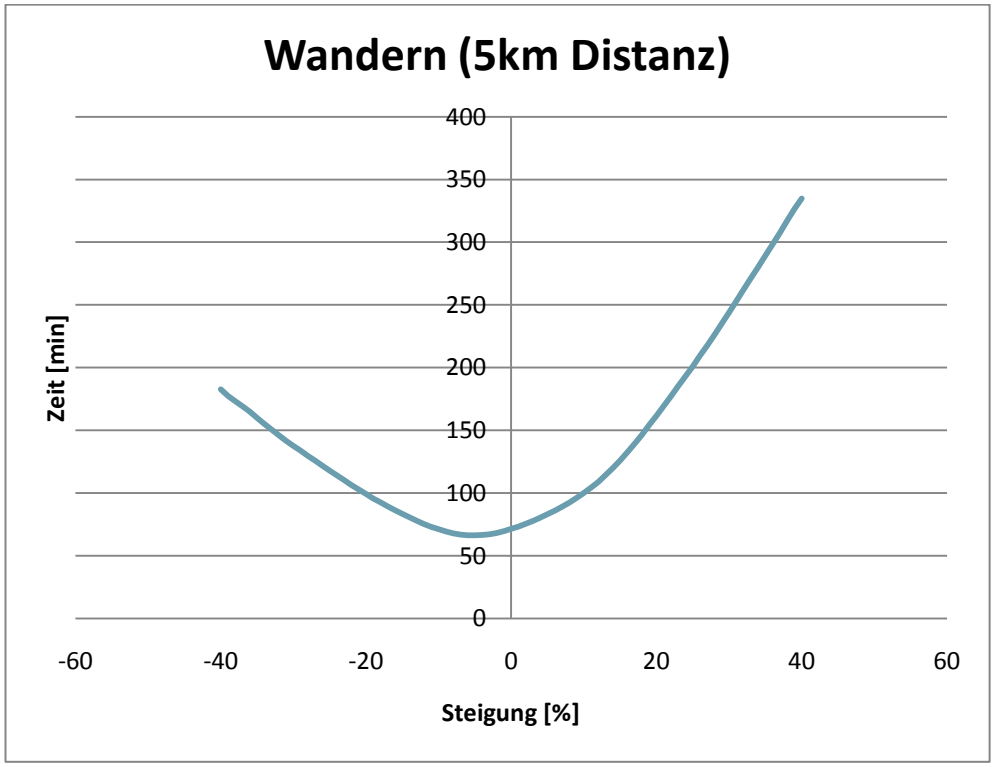


Abbildung 8 Gehzeit [Min] für 5km in Abhängigkeit der Steigung

Die kürzeste Zeit, um 5km zurückzulegen, liegt bei ca. 5% Gefälle (ca. 66 Minuten). Somit liegt die höchste Geschwindigkeit ca. 1,26 [m/s] auch bei 5% Gefälle, wie das nachfolgende Diagramm zeigt.

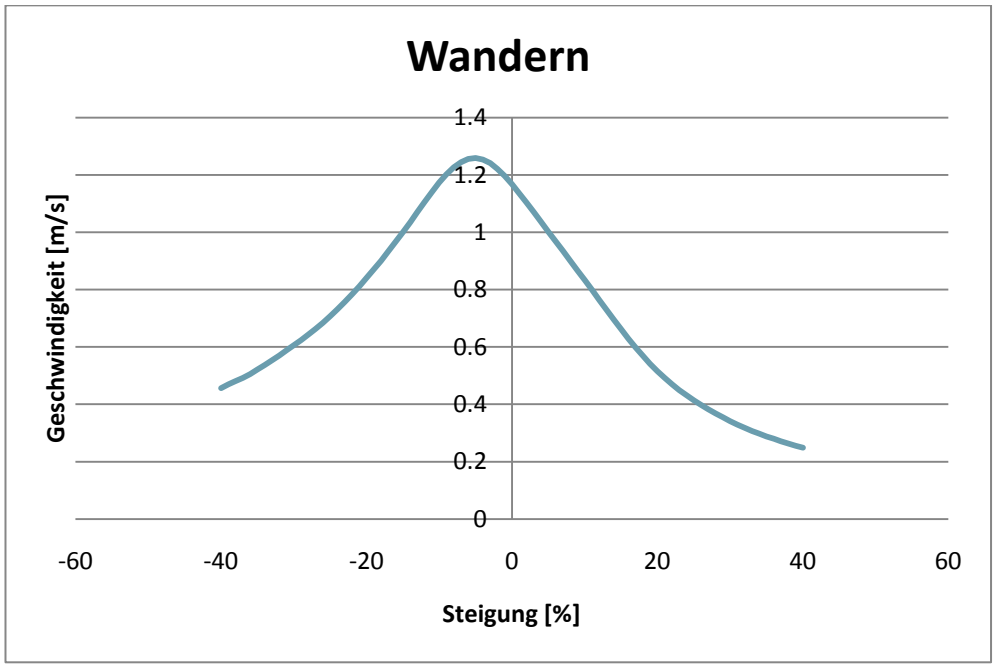


Abbildung 9 Gehgeschwindigkeit in Abhängigkeit der Steigung

4. Anforderungen

Damit die Idee der individuellen Wegzeitberechnung umgesetzt werden kann, sollen bestimmte Anforderungen implementiert werden. Ziel ist es, den Kern der individuellen Wegzeitberechnung in einem Prototyp zu realisieren.

Architektur

Das folgende Bild zeigt grob den Überblick über die wesentlichsten Systemkomponenten. Für die individuelle Wegzeitberechnung werden eine 3D-Route zur Navigation und die aktuelle Position benötigt. Die Applikation baut auf Android auf und wird mit Java implementiert.

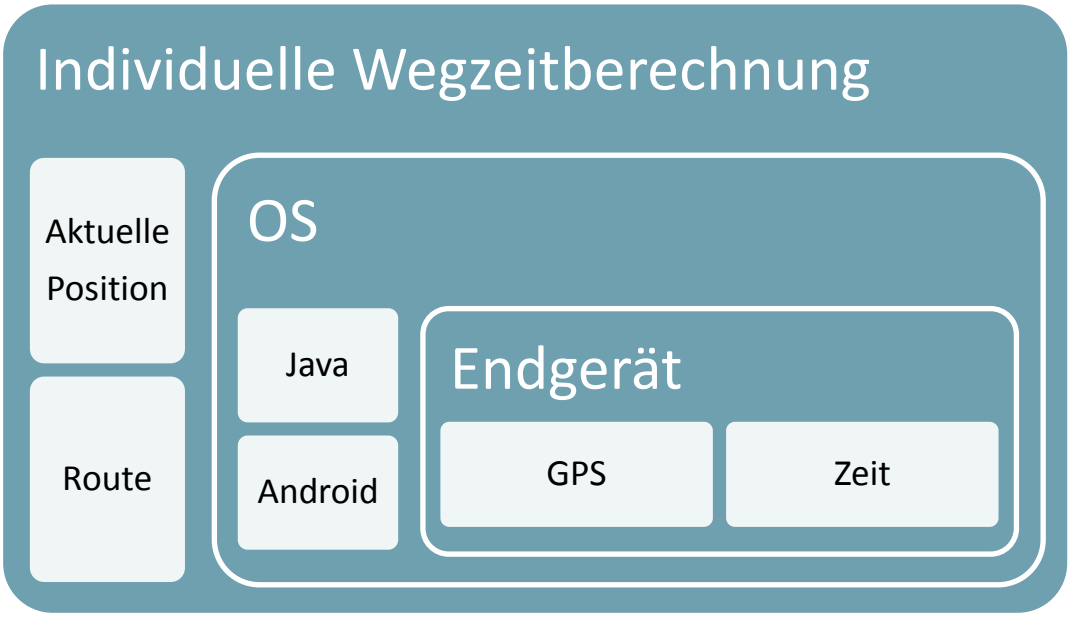


Abbildung 10 Architekturüberblick

Zudem soll der Prototyp mit einem MVC (Model-View-Controller) Muster implementiert werden (vgl. Buschmann et. al 2000: 124), damit bei einer späteren Erweiterung die entsprechenden Bestandteile einfach ausgebaut werden können.

Schnittstellen

Die Applikation soll auf dem Betriebssystem Android (Linux Kernel) aufgebaut werden. Auf dem folgenden Bild wird der Umfang von Android dargestellt. Das Betriebssystem bietet eine vollumfängliche virtuelle Maschine für Java (Dalvik). Zudem gibt es zahlreiche Bibliotheken, die bereits mit installiert sind und welche die Kommunikation

mit der Hardware des mobilen Endgerätes wesentlich vereinfachen.

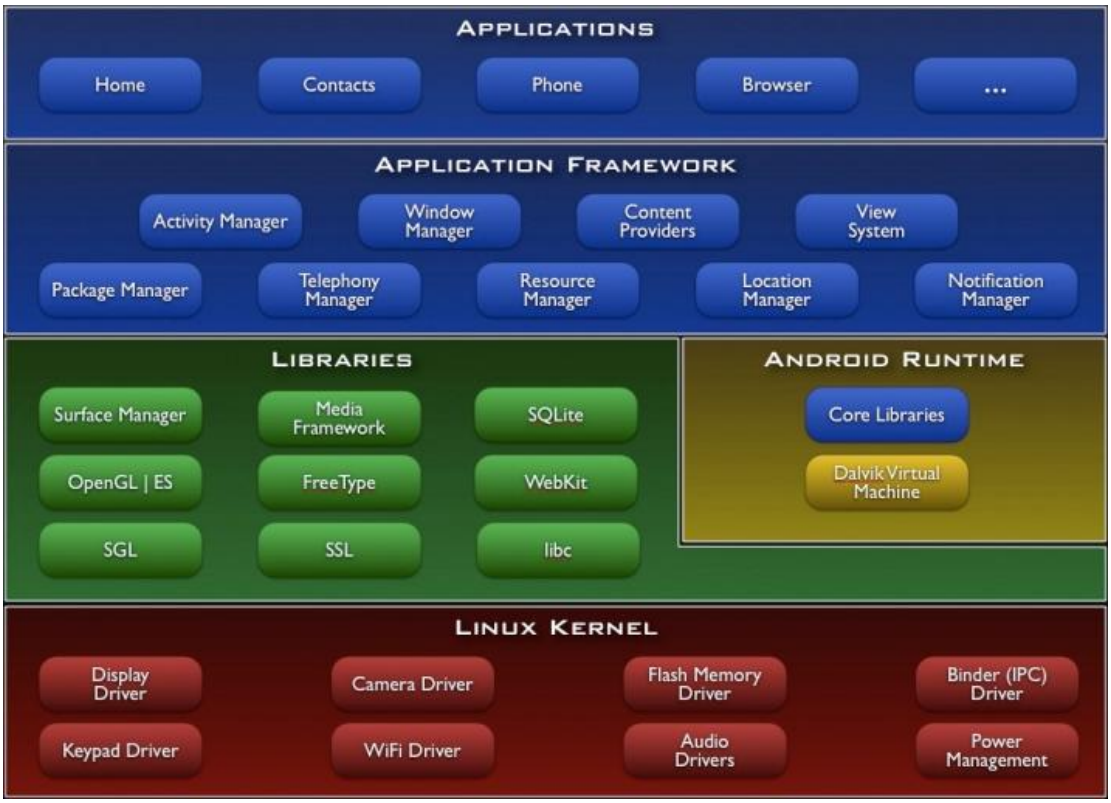


Abbildung 11 Überblick Android (Quelle: Android (Architecture) - Android Developers)

Der geplante Prototyp soll folgende Schnittstellen aufweisen, damit ein realitätsnaher Test durchgeführt werden kann:

Schnittstelle	Beschreibung
GPS -> Applikation	Diese Schnittstelle empfängt die GPS Daten und aktualisiert die momentane Position in der Applikation. Die Positionsdaten werden im Koordinatensystem WGS84 empfangen und so weiterverarbeitet. Android bietet im Application Framework den „Location Manager“ zur GPS Datenverwaltung an.
Internet -> Applikation	Diese Schnittstelle wird verwendet, um Routen aus dem Internet auf das mobile Endgerät zu laden. Die Routen sollten in einem standardisierten Format (KML) vorliegen und so im Endgerät abgespeichert werden.

DB -> Applikation	Diese Schnittstelle wird verwendet, um anfallende Daten dauerhaft zu speichern. Da Android eine interne DB Engine (SQLite) beinhaltet, bietet es sich an, Daten hier abzulegen.

Tabelle 3 Schnittstellen

Anforderungen für den Prototyp

Folgende Anforderungen gelten für den Prototyp. Sie wurden anhand der Hauptziele (s. *Ziele*) mit einem geplanten Zeitbudget definiert. Es gibt 3 Prioritäten für die Realisierung:

- Priorität 1: Muss-Anforderung
- Priorität 2: Soll-Anforderung
- Priorität 3: Kann-Anforderung

ID	Beschreibung	Priorität
A.001	Die Route soll geladen und gespeichert werden können (z.B. via Internet -> gpsies.com).	1
A.002	Startzeitpunkt und Ort (via GPS) soll definiert werden können.	1
A.003	Die Route soll in einem standardisierten Format (z.B. KML) verarbeitet werden.	2
A.004	Zu einer Route sollen ein Name, die Routendatei und Bemerkungen erfasst werden können.	2
A.005	Zu jedem Zeitpunkt soll eine Ankunftszeit berechnet werden.	1
A.006	Die Ankunftszeit wird mit den Informationen der zurückgelegten Strecke laufend aktualisiert.	1

A.007	Sobald die Route geladen wurde, soll kein Internet-Datenverkehr mehr stattfinden.	3
A.008	Es muss gewählt werden können, wie man sich fortbewegt.	3
A.009	Aus der Aktualisierung der Ankunftszeit sollen pro Fortbewegungsart Erfahrungswerte abgeleitet werden.	1
A.010	Die persönlichen Erfahrungswerte werden pro Fortbewegungsart für die Berechnung der Ankunftszeit mit einbezogen.	1
A.011	Das Höhenprofil soll (vereinfacht) dargestellt werden können.	2
A.012	Der aktuelle Standort soll auf dem Höhenprofil dargestellt werden.	2
A.013	Die vorausberechnete Dauer der zurückzulegenden Strecke soll angezeigt werden.	1
A.014	Die geometrisch zurückgelegte Zeit soll angezeigt werden.	3
A.015	Die geometrisch noch zurückzulegende Zeit soll angezeigt werden.	3
A.016	Die Ankunftszeit soll angezeigt werden (ETA).	1
A.017	Die Navigation soll unterbrochen werden können -> Pause.	2
A.018	Das Koordinatensystem soll WGS-84 sein.	1
A.019	Die Erfahrungswerte sollen gelöscht werden können.	2
A.020	Eine gespeicherte Route soll gelöscht werden können.	2

A.021	Die aktuelle Position wird mit GPS bestimmt.	1
A.022	Die Zeit für die Dauer soll vom Endgerät verwendet werden.	1
A.023	Die Applikation soll beendet werden können.	1
A.024	Nach einem Unterbruch (Pause) soll die Navigation wieder aufgenommen werden können.	2

Tabelle 4 Anforderungen

5. Anwendungsfälle

Um die Anforderungen möglichst umfassend abzudecken, wurden folgende Anwendungsfälle für den Prototyp definiert. Die Anwendungsfälle haben, wenn vorhanden, eine Referenz auf die Anforderung und einen Status, der für die Release-Planung verwendet wird. Für den Prototyp im Umfang der Masterarbeit ist ein Release geplant, der alle Anforderungen bis Priorität 2 umsetzt. Im Statusfeld der Anwendungsfälle wird je nach Stand einer der folgenden Status gesetzt:

- In Arbeit
- Bereit zum Review
- Im Review
- Abgelehnt
- Abgenommen
- Auf nächsten Release verschoben

Die aufgeführten Kategorien der Beschreibung der Anwendungsfälle, können je nach Projekt und Anforderung unterschiedlich sein. In dem vorliegenden Fall wurden die Wichtigsten, wie sie in Oestereich et al. 2009 definiert wurden, aufgeführt.

Anwendungsfalldiagramm

Das folgende Anwendungsfalldiagramm zeigt die zu realisierenden Anwendungsfälle, die sich aus den Anforderungen ableiten.

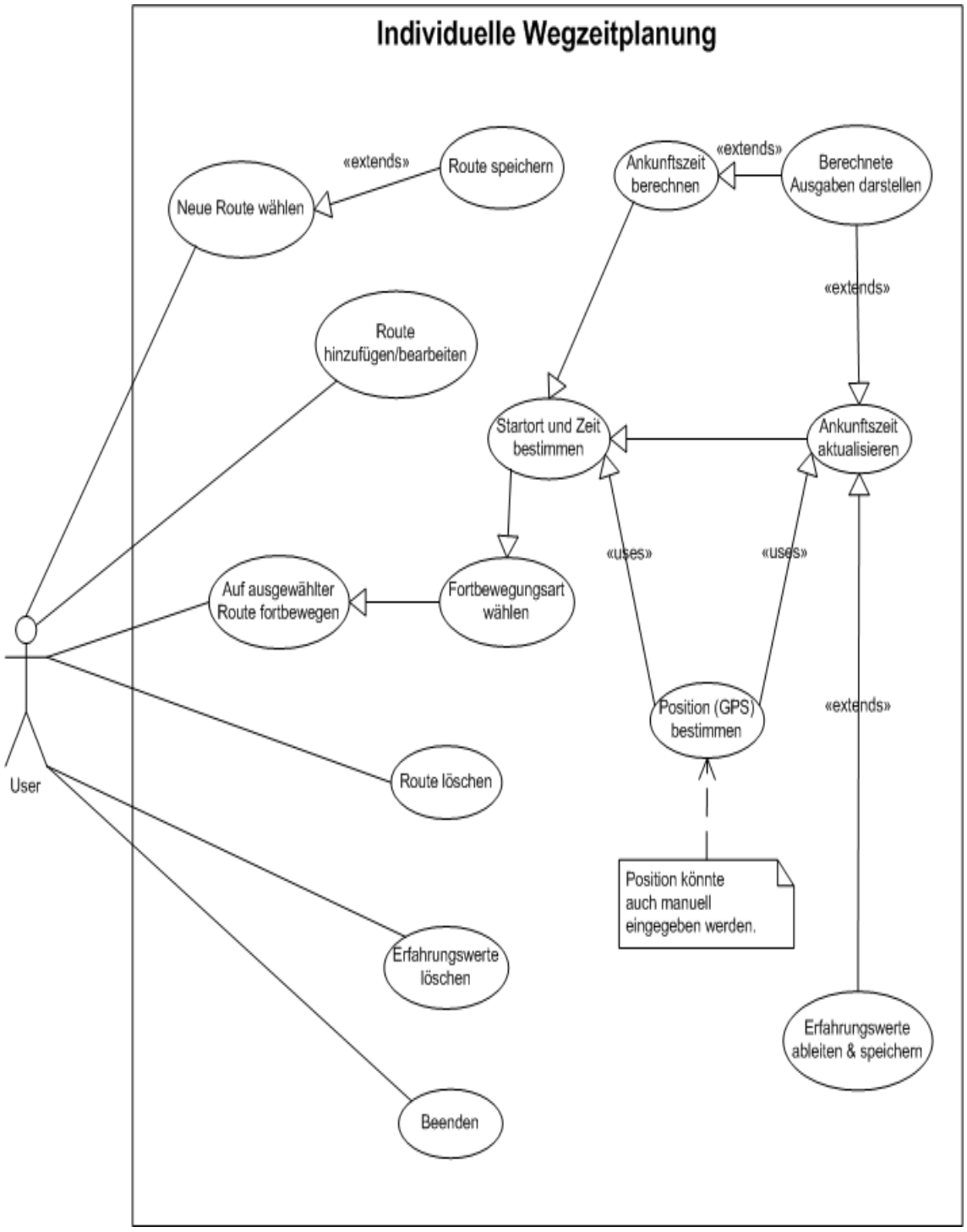


Abbildung 12 Anwendungsfälle

Verwendete Akteure

Für den Prototyp wurde nur ein Akteur definiert.

Akteur	Beschreibung
User	User ist ein Benutzer der individuellen Wegzeitberechnung.

Tabelle 5 Akteure

Beschreibung der Anwendungsfälle

Name	**Neue Route wählen**
ID	U.001
Vorbedingungen	Internet muss vorhanden sein. Geplante Route muss als KML Datei auf der Internetseite existieren.
Nachbedingungen	Geplante Route wurde als KML Datei für das Herunterladen selektiert.
Ablaufbeschreibung	Eine neue Route wird als KML Datei von einer Internetseite selektiert.
Fehlersituation	Ist das Internet nicht verfügbar, wird dies in einer Fehlermeldung mitgeteilt und zur Hauptseite der Applikation zurückgekehrt. Ist die KML Datei nicht verfügbar, wird dies in einer Fehlermeldung mitgeteilt und zur Hauptseite der Applikation zurückgekehrt.
Erfüllt Anforderung	A.001 / A.018

Status	Abgenommen

Tabelle 6 Anwendungsfall "Neue Route herunterladen"

Name	**Auf ausgewählter Route fortbewegen**
ID	U.002
Vorbedingungen	Eine existierende Route wurde ausgewählt. Die dazugehörige KML Datei wurde gelesen und für die Fortbewegung vorbereitet.
Nachbedingungen	Geplante Route ist geladen und bereit für die Fortbewegung.
Ablaufbeschreibung	Eine gespeicherte Route wird geladen und kann für die Fortbewegung verwendet werden.
Fehlersituation	KML Datei kann nicht gelesen werden. Der Benutzer wird mit einer Fehlermeldung informiert. Der Anwendungsfall wird abgebrochen und es wird zur Hauptseite der Applikation zurückgekehrt.
Erfüllt Anforderung	A.003 / A.007 / A.018
Status	Abgenommen

Tabelle 7 Anwendungsfall "Auf ausgewählter Route fortbewegen"

Name	**Route löschen**
ID	U.003
Vorbedingungen	Eine existierende Route wurde aus der Routenliste ausgewählt.
Nachbedingungen	Die selektierte Route wurde aus der Routenliste entfernt. Die

	dazugehörige KML Datei bleibt auf dem Dateisystem bestehen.
Ablaufbeschreibung	Eine selektierte Route wird aus der Routenliste gelöscht. Die KML-Datei bleibt auf dem Dateisystem bestehen.
Fehlersituation	Die selektierte Route kann nicht gelöscht werden. Der Benutzer muss mit einer Fehlermeldung informiert werden. Der Anwendungsfall wird abgebrochen.
Erfüllt Anforderung	A.020
Status	Abgenommen

Tabelle 8 Anwendungsfall "Route löschen"

Name	**Route hinzufügen / bearbeiten**
ID	U.004
Vorbedingungen	Geplante Route muss auf dem Dateisystem als KML Datei vorliegen.
Nachbedingungen	Route wurde in die Routenliste mit den definierten Attributen aufgenommen.
Ablaufbeschreibung	Eine KML-Datei wird als Route zur Routenauswahlliste hinzugefügt. Es können die Attribute Routenname, Routendatei und Bemerkung bearbeitet werden.
Fehlersituation	Die Route kann nicht in die Routenliste aufgenommen werden. Der Benutzer wird mit einer Fehlermeldung informiert. Der Anwendungsfall wird abgebrochen.
Erfüllt Anforderung	A.004

Status	Abgenommen

Tabelle 9 Anwendungsfall "Route hinzufügen / bearbeiten"

Name	**Erfahrungswerte löschen**
ID	U.005
Vorbedingungen	Die Datenbank mit den Erfahrungswerten wurde angelegt.
Nachbedingungen	Die Erfahrungswerte wurden gelöscht.
Ablaufbeschreibung	Alle gesammelten persönlichen Erfahrungswerte werden gelöscht.
Fehlersituation	Die Erfahrungswerte können nicht gelöscht werden. Mit einer Fehlermeldung wird der Benutzer informiert. Der Anwendungsfall wird abgebrochen.
Erfüllt Anforderung	A.019
Status	Abgenommen

Tabelle 10 Anwendungsfall "Erfahrungswerte löschen"

Name	**Route speichern**
ID	U.006
Vorbedingungen	Eine Route wurde als KML Datei im Internet selektiert.
Nachbedingungen	Die KML Datei wurde auf dem Dateisystem des mobilen Endgerätes gespeichert.
Ablaufbeschreibung	Die KML-Datei wird heruntergeladen und auf dem Dateisystem des Endgerätes abgelegt.

Fehlersituation	Die Datei kann nicht gespeichert werden. Der Benutzer wird mit einer Fehlermeldung informiert. Der Anwendungsfall wird abgebrochen und es wird zur Hauptseite der Applikation zurückgekehrt.
Erfüllt Anforderung	A.001
Status	Abgenommen

Tabelle 11 Anwendungsfall "Route speichern"

Name	**Fortbewegungsart wählen**
ID	U.007
Vorbedingungen	Ausgewählte Route muss geladen und bereit für die Fortbewegung sein.
Nachbedingungen	Wenn noch nicht vorhanden, müssen die Geschwindigkeiten pro Steigung für die Startfunktion der gewählten Fortbewegungsart in die Datenbank gespeichert werden. Damit kann die Dauer der geplanten Route berechnet werden.
Ablaufbeschreibung	Die Fortbewegungsart soll ausgewählt werden können. (Im aktuellen Prototyp wird immer die Fortbewegung Wandern verwendet).
Fehlersituation	keine
Erfüllt Anforderung	A.008
Status	Auf nächsten Release verschoben

Tabelle 12 Anwendungsfall "Fortbewegungsart wählen"

Name	**Startort und Zeit bestimmen**
ID	U.008
Vorbedingungen	Der Benutzer befindet sich am Startort der geplanten Route. Die Route wurde auf dem mobilen Endgerät ausgewählt und geladen. Die Fortbewegungsart wurde definiert. Das GPS hat eine gültige Position.
Nachbedingungen	Der Startort und die Startzeit wurden definiert. Die Fortbewegungszeit läuft.
Ablaufbeschreibung	Der aktuelle Standort wird mit dem GPS vom mobilen Endgerät zur Bestimmung des Startortes verwendet. Die Zeit wird aus dem mobilen Endgerät gelesen. Der Navigationsthread wird gestartet.
Fehlersituation	Das GPS kann keine gültige Position bestimmen. Der Anwendungsfall muss abgebrochen werden. Mit einer Fehlermeldung wird der Benutzer informiert. Die vom GPS bestimmte Position ist mehr als 1 km von der geplanten Strecke entfernt. Der Benutzer wird informiert. Die Fortbewegung kann trotzdem fortgesetzt werden.
Erfüllt Anforderung	A.002 / A.017 / A.022 / A.024
Status	Abgenommen (Fehlersituation nicht vollständig implementiert)

Tabelle 13 Anwendungsfall "Startort und Zeit bestimmen"

Name	**Ankunftszeit berechnen**
ID	U.009

Vorbedingungen	Die geplante Route ist geladen und die Fortbewegungsart definiert. Die Startposition ist festgelegt.
Nachbedingungen	Die Ankunftszeit wurde mit der Startfunktion für die geplante Route berechnet.
Ablaufbeschreibung	Anhand der Startfunktion soll die Dauer und die Ankunftszeit der geplanten Route berechnet werden.
Fehlersituation	Die Dauer kann mit der angegebenen Startfunktion nicht berechnet werden. Der Benutzer muss mit einer Fehlermeldung informiert werden und die Fortbewegung wird abgebrochen.
Erfüllt Anforderung	A.005
Status	Abgenommen

Tabelle 14 Anwendungsfall "Ankunftszeit berechnen"

Name	**Ankunftszeit aktualisieren**
ID	U.010
Vorbedingungen	Die Route ist geladen, Startort und –zeit sind vorhanden, die Dauer wurde mit der Startfunktion berechnet und das GPS liefert die aktuelle Position.
Nachbedingungen	Die Dauer und Ankunftszeit wurde aktualisiert.
Ablaufbeschreibung	Die Ankunftszeit wird mit der aktuellen Position und Zeit anhand des Algorithmus für die individuelle Wegzeitberechnung aktualisiert.

Fehlersituation	Die Ankunftszeit kann nicht aktualisiert werden. Der Benutzer wird informiert und die Fehlermeldung wird in die Logdatei protokolliert. Der Benutzer kann den Anwendungsfall abbrechen und in die Hauptansicht zurückkehren.
Erfüllt Anforderung	A.005 / A.006 / A.010
Status	Abgenommen

Tabelle 15 Anwendungsfall "Ankunftszeit aktualisieren"

Name	**Berechnete Ausgaben darstellen**
ID	U.011
Vorbedingungen	Die Route ist geladen und das Höhenprofil wird dargestellt.
Nachbedingungen	Die aktuell berechneten Werte werden auf dem Display dargestellt.
Ablaufbeschreibung	Alle berechneten Angaben wie Höhenprofil mit aktueller Position, Dauer, Ankunftszeit, geometrisch zurückgelegte Zeit und geometrisch noch zurückzulegende Zeit werden angezeigt.
Fehlersituation	Nicht alle Werte sind verfügbar. Der nicht verfügbare Wert wird nicht dargestellt. Darstellung nicht möglich. Der Benutzer wird informiert und die Navigation wird abgebrochen. Es wird zur Hauptseite der Applikation zurückgekehrt.
Erfüllt Anforderung	A.011 / A.012 / A.013 / A.014 / A.015 / A.016
Status	Abgenommen

Tabelle 16 Anwendungsfall "Berechnete Ausgaben darstellen"

Name	**Erfahrungswerte ableiten und speichern**
ID	U.012
Vorbedingungen	Die individuelle Wegzeitberechnung konnte durchgeführt werden.
Nachbedingungen	Die Erfahrungswerte wurden abgeleitet und in die Datenbank gespeichert.
Ablaufbeschreibung	Anhand der aktuellen Zeit und Position im Vergleich zur vorausgesagten Position und Zeit sollen Erfahrungswerte (Geschwindigkeit pro Steigung je nach Fortbewegungsart) abgeleitet werden, die für die aktuellen und zukünftigen Berechnungen verwendet werden können.
Fehlersituation	Erfahrungswerte können nicht abgeleitet werden. Es wird eine Warnung ausgegeben. Die Fehlermeldung wird in die Logdatei protokolliert. Die Applikation kann jedoch weiter laufen auch ohne aktualisierte Erfahrungswerte.
Erfüllt Anforderung	A.009 / A.010
Status	Abgenommen

Tabelle 17 Anwendungsfall "Erfahrungswerte ableiten und speichern"

Name	**Position (GPS) bestimmen**
ID	U.013
Vorbedingungen	Route wurde geladen und Fortbewegungsart bestimmt.
Nachbedingungen	Aktuelle Position wurde bestimmt.

Ablaufbeschreibung	Die mit GPS gemessene Position wird auf die geplante Route projiziert. Somit kann die Position auf der geplanten Route bestimmt werden, die für die Berechnung der individuellen Wegzeit massgebend ist.
Fehlersituation	GPS kann die aktuelle Position nicht bestimmen. Wenn nach längerer Zeit (> 5 min) keine gültige Position bestimmt werden kann, muss der Benutzer informiert werden. Dieser hat anschliessend die Möglichkeit, die Anwendung zu stoppen.
Erfüllt Anforderung	A.021
Status	Abgenommen (Fehlersituation nicht vollständig implementiert)

Tabelle 18 Anwendungsfall "Position (GPS) bestimmen"

Name	**Beenden**
ID	U.014
Vorbedingungen	Der Benutzer befindet sich im Hauptmenü.
Nachbedingungen	Die Applikation wurde beendet. Alle verwendeten Ressourcen wurden freigegeben.
Ablaufbeschreibung	Alle Aktivitäten wie GPS und Verbindung zur Datenbank abbauen. Anschliessend wird der Prozess der Applikation beendet. (In Android müssen Prozesse nicht unbedingt beendet werden. Normal wird einfach der Fokus der Applikation verlassen. Da im Prototyp GPS verwendet wird, was viel Batterie benötigt, wurde entschieden, die Applikation mit der Beenden-Funktionalität auszustatten).
Fehlersituation	Kann eine Ressource nicht abgebaut werden, so wird der

	Prozess der Applikation dennoch beendet. Die Fehlermeldung wird in die Logdatei protokolliert.
Erfüllt Anforderung	A.023
Status	Abgenommen

Tabelle 19 Anwendungsfall "Beenden"

6. Prototyp

Entwicklungsumgebung

Hardwareinfrastruktur (VMWare)

Damit die Entwicklungsumgebung einfach und überall mitgenommen werden kann, wurde eine virtuelle Maschine auf einer externen Festplatte eingerichtet. Folgende Parameter dienten zur Erstellung der virtuellen Maschine:

Hardware Komponenten	Beschreibung
VMWare Player	Version 3
Hauptspeicher	1.5 GB
CPU	1 CPU
Harddisks	IDE Disk 1 10 GB IDE Disk 2 20 GB
CD/DVD	Auto detect
Netzwerk	Bridged

Tabelle 20 Hardwareinfrastruktur

Softwareinfrastruktur

Die Softwareinfrastruktur hängt stark von der Entwicklungsumgebung des Endgerätes ab. Am Anfang der Masterarbeit standen folgende Umgebungen für die Entwicklung des Prototyps zur Auswahl:

- Nokia (N95) -> Symbian / Java ME (bereits vorhandenes Mobiltelefon)
- Trimbel GeoXT -> Windows Mobile / .Net CF (Gerät im Büro)
- iPhone -> iOS / XCode (Mobiles Gerät könnte von Swisstopo ausgeliehen werden, jedoch fehlt die Arbeitsstation zum entwickeln)

- Samsung -> Android / Java (neu beschafftes Gerät)

Zuerst wurde versucht, die Symbian-Entwicklungsumgebung und die benötigten Bibliotheken für das bestehende Nokia N95 Mobiltelefon zu installieren. Leider gibt es für Symbian verschiedenste Ausprägungen für die diversen Hersteller und Modelle. Zudem gibt es verschiedenste kompatible oder inkompatible Bibliotheken, welche die Implementierung auf dieser Basis zusätzlich erschweren. Nach einem kleinen Test mit Android wurde schnell klar, dass diese Umgebung für den Prototyp zweckmäßiger ist. Folgende Gründe waren für den Entscheid zu Gunsten von Android und zu Ungunsten von Symbian ausschlaggebend (vgl. Bierer et al. 2010):

- Entwicklungsumgebung gut in Eclipse integriert
- Guter Emulator mit Zusatzapplikation für Eventsteuerung (GPS, SMS, u.s.w.)
- Volle Java Unterstützung
- Durchgängige Dokumentation
- Einfache Installation und Konfiguration

Somit wurden folgende Softwarekomponenten installiert:

Software Komponenten	Version
Windows	XP, SP3
Eclipse	Pulsar-Galileo
Java	1.6.0_20
Android SDK	0.9.7
GIMP	2.6.10
ArcGIS Desktop	9.3.1 SP2
MS Office	2007

Firefox	3.6.8

Tabelle 21 Softwareinfrastruktur

Die Hauptentwicklung konnte mit Eclipse (Burnette et al. 2010) und dem Android SDK (Android (SDK) – Android Developers) mit integriertem Emulator durchgeführt werden (s. Abbildung 13). Sehr praktisch war der Dalvik Debug Monitor, in dem alle Logeinträge dargestellt und die Eventsteuerung simuliert werden konnte. Einzig mit dem Simulieren der KML Höhe gab es im Emulator Probleme, respektive die Höhe wurde nicht richtig übertragen. Dies wurde schlussendlich auf einem echten Endgerät getestet.

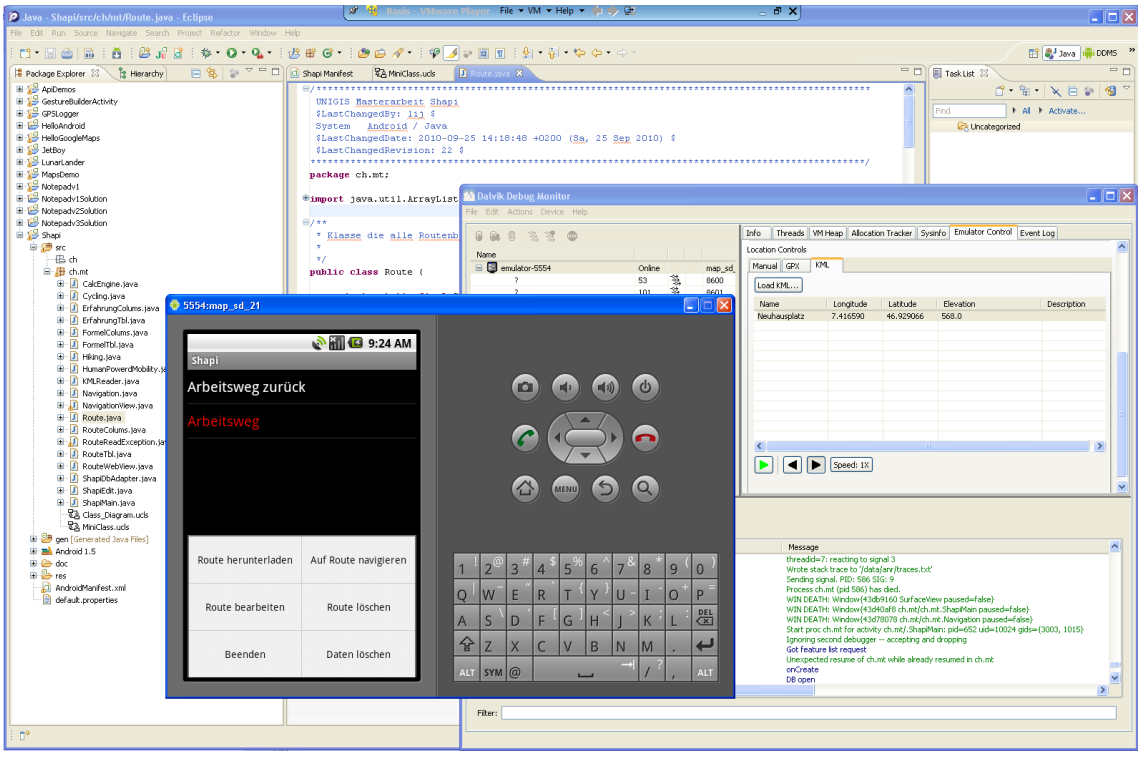


Abbildung 13 Entwicklungsumgebung mit Android-Emulator

Auch auf die interne Datenbank SQLite konnte sehr einfach zugegriffen werden. Auf der Kommandozeile konnte direkt die Datenbank des Emulators angesprochen und mit SQL abgefragt oder manipuliert werden (s. Abbildung 14).

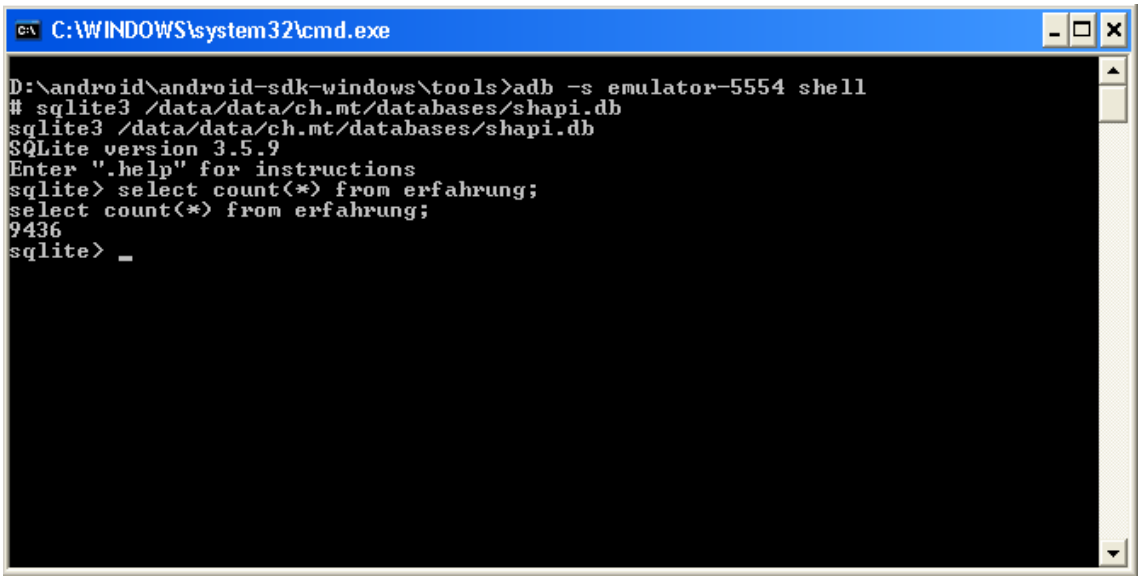


Abbildung 14 Datenbankabfrage Android-Emulator

Die Layouts wurden grundsätzlich in Eclipse entwickelt. Dazu gibt es in Eclipse eine gute Integration, welche die View-XML Dateien (St. Laurent et al. 2006) visualisiert (wysiwyg), wie Abbildung 15 zeigt.

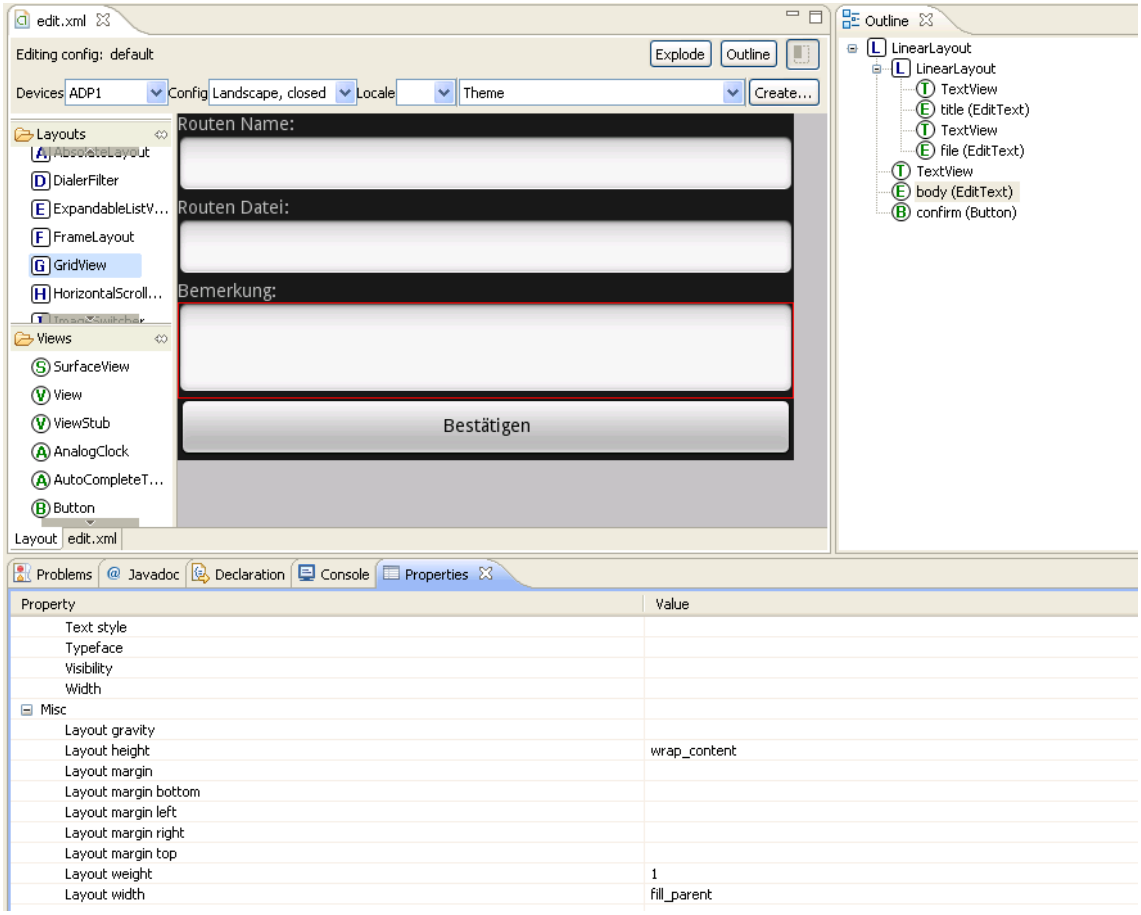


Abbildung 15 View XML Editor

Anschliessend konnten die Views einfach mit dem Hierarchy Viewer (s. Abbildung 16), der im Android SDK mitgeliefert wird, betrachtet und auf Fehler überprüft werden. Anbei das Beispiel der Routenedit View edit.xml (vgl. Anhang C: Layout View edit.xml).

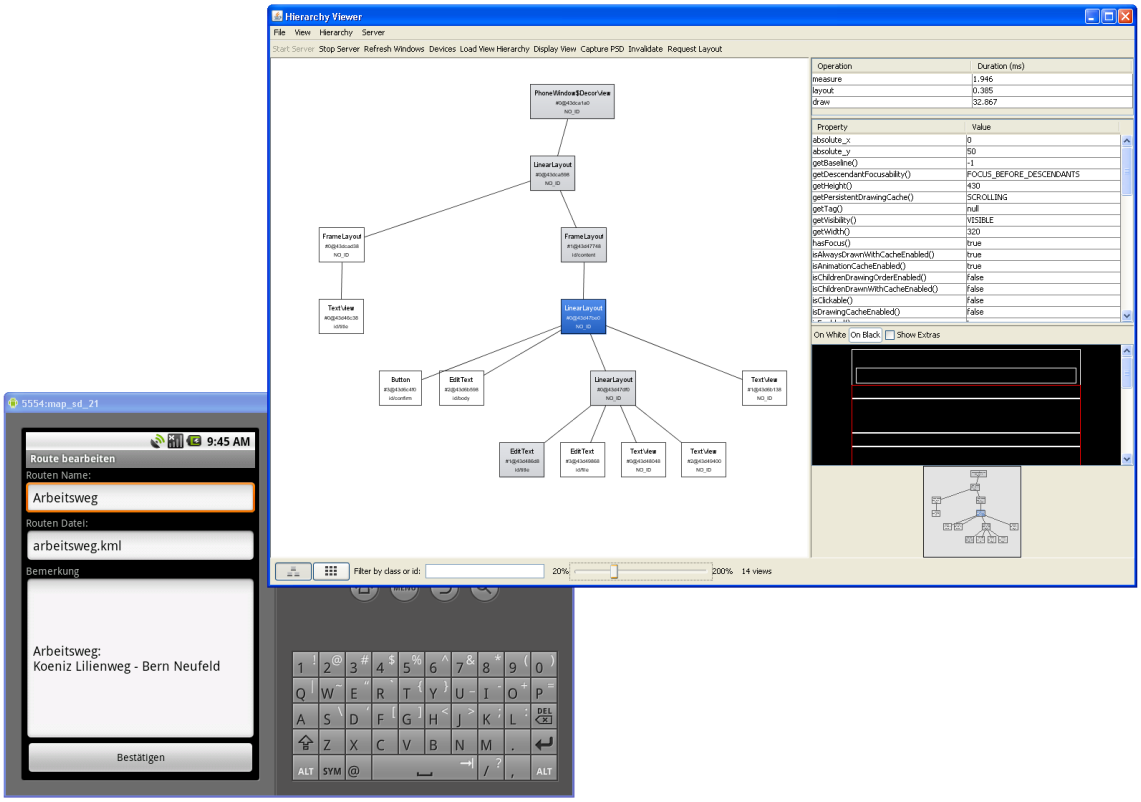


Abbildung 16 Hierarchy Viewer für erstellte Layouts

In Android werden die Rechte für einen Hardwarezugriff einer Applikation in der Datei AndroidManifest.xml definiert. Für den vorliegenden Prototyp mussten folgende Rechte konfiguriert werden:

- ACCESS_FINE_LOCATION -> für den Zugriff auf das GPS
- ACCESS_LOCATION_EXTRA_COMMANDS -> für den Zugriff auf das GPS
- INTERNET -> für den Zugriff auf das Internet
- WRITE_EXTERNAL_STORAGE -> für den Zugriff auf den externen Speicher

(vgl. Anhang B: AndriodManifest.xml).

Allgemein kann gesagt werden, dass die Entwicklung für Android Endgeräte in Eclipse sehr komfortabel ist. Die Entwicklung lediglich mit dem Emulator und ohne ein

richtiges Endgerät wäre aber in diesem Fall fehlgeschlagen, da die Simulation des GPS nicht 100% funktionstüchtig war.

Grundlagen zur Referenzstrecke

Die Referenzstrecke wurde auf der Onlineplattform www.gpsies.com auf einem Desktop-PC digitalisiert (s. Abbildung 17 & Abbildung 18).

Die Digitalisierung wurde auf einem Orthofoto durchgeführt. Es wurde versucht, immer innerhalb der Strasse zu digitalisieren. Somit kann mit einer 2D-Genauigkeit von ca. 10m gerechnet werden. Eine höhere 2D-Genauigkeit ist nicht gefordert, da beim Wandern oder Fahrradfahren zum Teil auch die ganze Strasse ausgenützt wird. Problematischer ist es bei der Höhe. Bei der verwendeten Plattform wird das SRTM Höhenmodell verwendet. Das SRTM Höhenmodell hat eine Auflösung von 90 x 90 Meter und eine absolute horizontale Genauigkeit < 20 Meter.

Es wurde bewusst diese Online-Plattform und Digitalisierungsart gewählt, da dies einem späteren produktiven Arbeitsablauf entsprechen könnte. Die gewählte Plattform bietet auch eine Android Applikation, um Routen herunterzuladen. So könnte in Zukunft die Route an einer Arbeitsstation erfasst werden. Anschliessend kann im Feld die erfasste Strecke abgerufen werden.

Als Referenzstrecke wurde der Arbeitsweg des Autors gewählt. Dadurch konnte die geplante Strecke einfach überprüft und getestet werden.

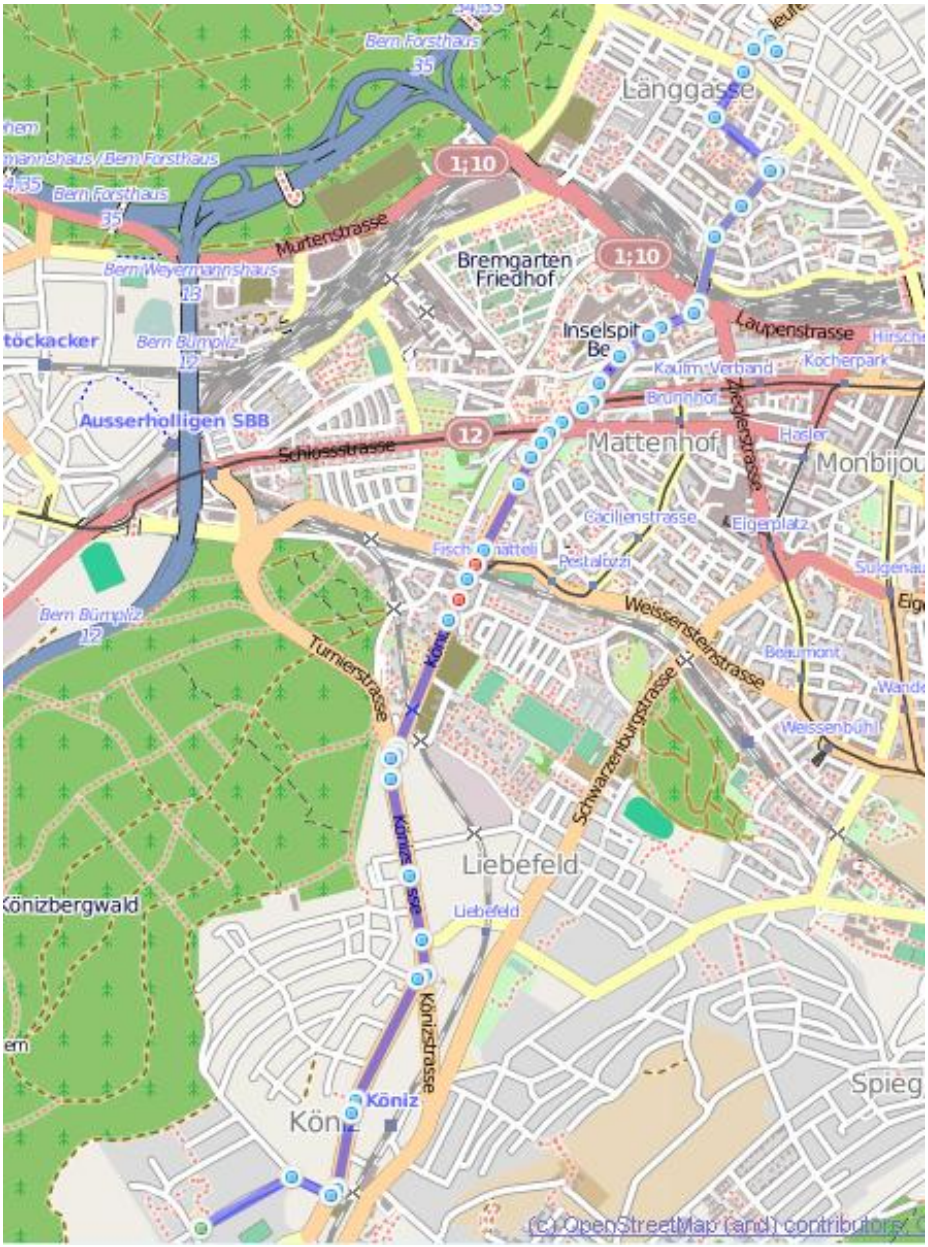


Abbildung 17 Digitalisierte Referenzstrecke auf gpsies.com

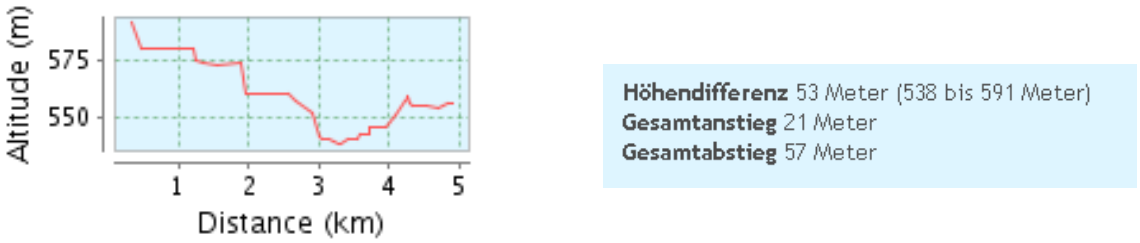


Abbildung 18 Höhenprofil der Referenzstrecke

Anschliessend wurde die Strecke im GIS (Esri) analysiert und dargestellt.

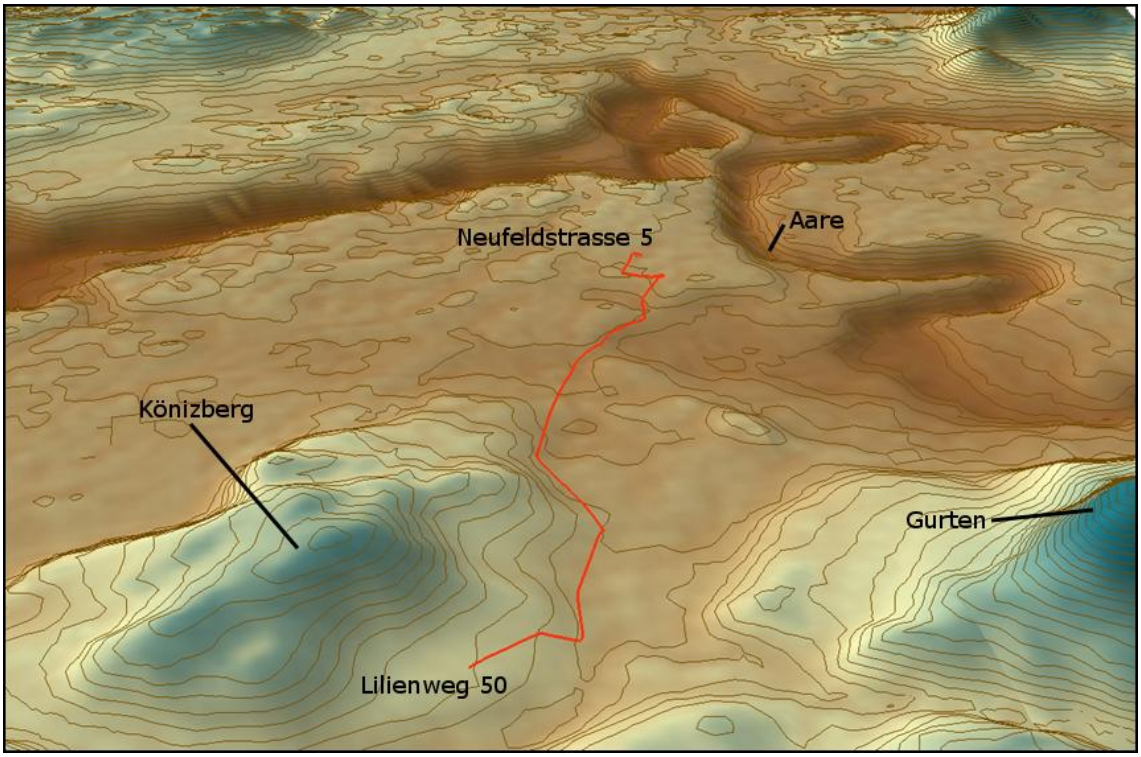


Abbildung 19 Strecke mit 3D Ansicht auf dem verwendeten SRTM DHM

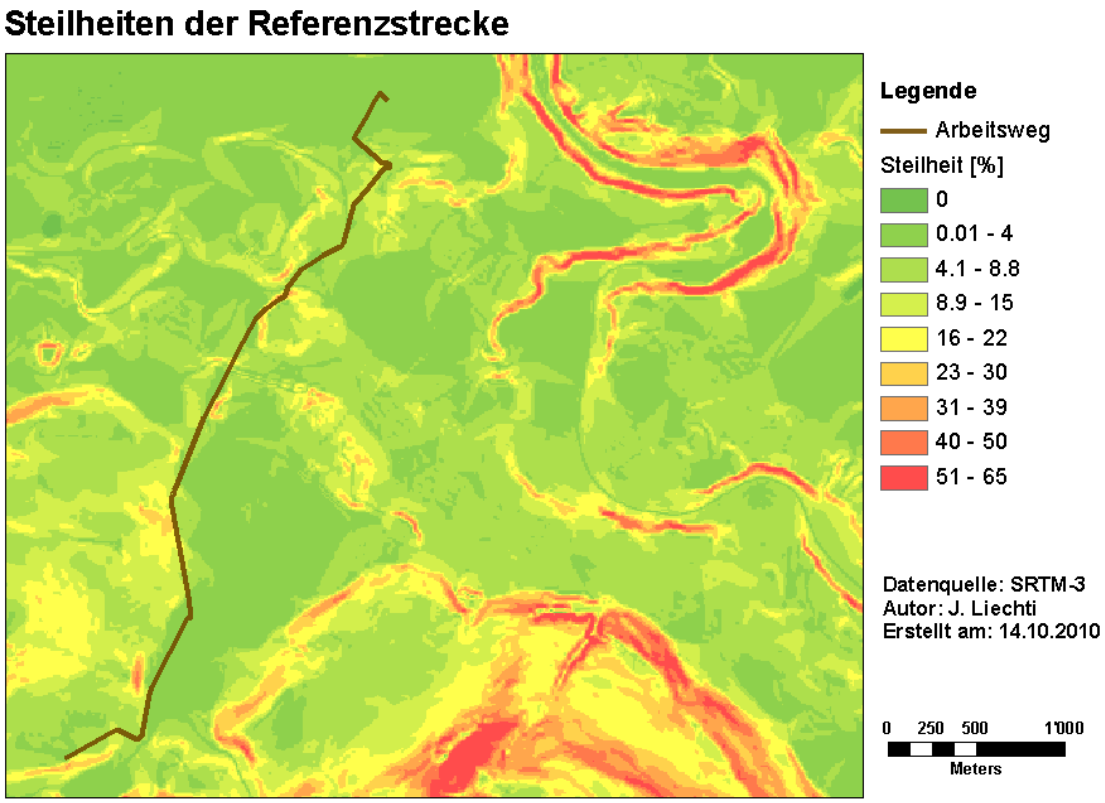


Abbildung 20 Digitalisierte Strecke dargestellt mit dem Gefälle

Die beiden Karten (Abbildung 19 und Abbildung 20) zeigen übersichtlich, wo die grössten Steigungen anfallen.

Anschliessend wurde die Gehzeit mit dem Excel-Formular der Schweizer Wanderwege, das für die Berechnung der Wanderwegtafeln benötigt wird, berechnet. Es wurden nur ein paar markante Standorte, die mit dem GIS identifiziert wurden, für die Berechnung ausgewählt. Dadurch wird die Berechnung der Dauer vereinfacht. Die folgende Abbildung zeigt die Berechnung des Arbeitsweges.

Schweizer Wanderwege
Suisse Rando
Sentieri Svizzeri
Sendas Svizras

Schweizer Wanderwege
Gehzeitberechnung

04.08.2010

Kanton	Bern	Distanz 5100 m	
Routennummer	1	Aufstieg 25 m	
Routenname	Arbeitsweg	Abstieg 68 m	
Bemerkungen	Lilienweg 50 - Neufeldstrasse 5	Profil	

				Wegzeiten:	berechnet		gerundet		Wegweiser	
				Summe in min:	72	76	70	75		
Standort-Nr.	Flurname	Höhe (M. ü. M.)	Distanz (m)	Hinweg (min)	Rückweg (min)	Hinweg (min)	Rückweg (min)		Hinweg (h min)	Rückweg (h min)
	Lilienweg 50	598	1250	17	19	15	20			1 h 15 min
	Neuhausplatz	575	700	10	10	10	10		15 min	55 min
	Wald	568	1300	18	20	20	20		25 min	45 min
	Loryplatz	535	700	10	10	10	10		45 min	25 min
	Insel	543	450	7	6	5	5		55 min	15 min
	Bühlplatz	560	700	10	10	10	10		1 h	10 min
	Neufeldstrasse 5	555							1 h 10 min	

Abbildung 21 Berechnung Arbeitsweg (Gehzeit Formel -Schweizer Wanderwege, 2007)

Der Arbeitsweg vom Lilienweg 50 nach Neufeldstrasse 5 dauert somit 72min (Wegweiser 1 h 10 min) und zurück 76min (Wegweiser 1 h 15 min). Es wird eine Distanz von ca. 5100m, 25m Auf- und 68m Abstieg zurückgelegt (Hinweg).

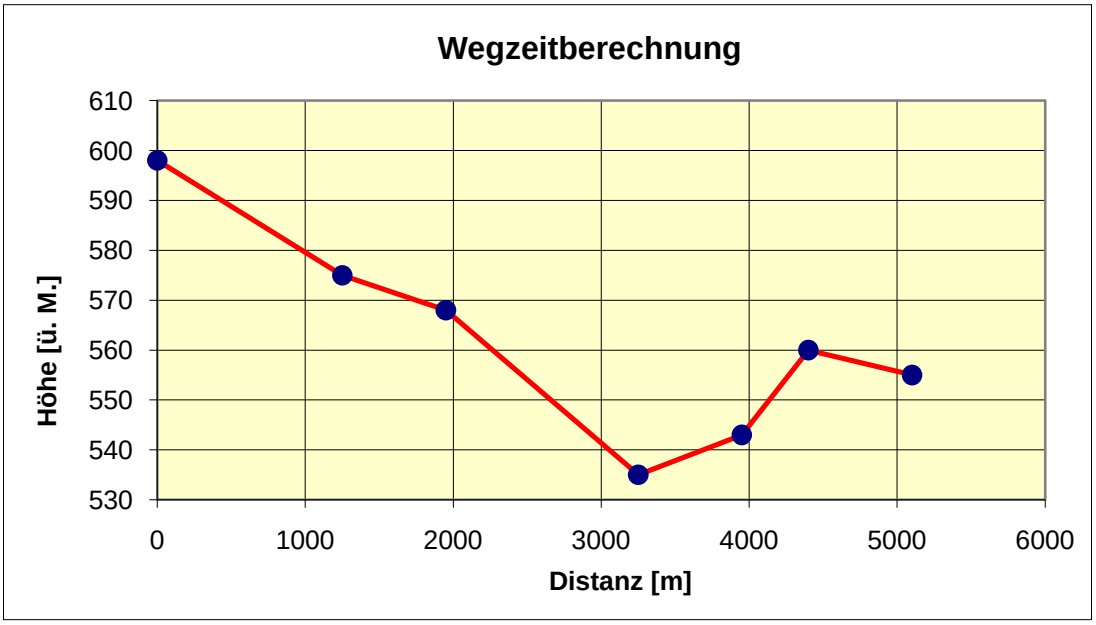


Abbildung 22 Wegzeit (Gehzeit Formel -Schweizer Wanderwege, 2007)

Das generierte Höhenprofil (s. Abbildung 22) zeigt, dass bis ca. Streckenmitte ein leichtes Gefälle zurückgelegt wird. Anschliessend wird eine kleine Steigung überwunden und das Ziel wird auf fast flachem Gebiet erreicht.

Diese Referenzstrecke wurde im Prototyp für Tests verwendet. Beim Laden im Prototyp gibt die digitalisierte KML-Route einen Hinweg von 69,5 min und Rückweg von 72 min an. Die Differenz vom Prototyp zur Berechnung im Excel ergibt sich durch die unterschiedliche Anzahl Stützpunkte und deren Genauigkeit.

Individuelle Wegzeitberechnung

Die individuelle Wegzeitberechnung erfolgt auf der Analyse der bereits zurückgelegten Strecke. Aus den Erfahrungswerten der zurückgelegten Strecke wird für den noch zurückzulegenden Streckenteil eine Vorausberechnung mit den Erfahrungswerten gemacht.

Analyse der zurückgelegten Strecke

Die geladene Route besteht aus den grundlegenden Elementen ($n \in N^*$) Kanten und ($n+1 \in N^*$) Knoten. Zur einfacheren Darstellung wird hier in der Illustration auf die dritte Dimension verzichtet.

Um die zurückgelegte Strecke zu analysieren muss zuerst herausgefunden werden, wo genau man sich befindet. Da das GPS (und vielleicht auch die Person, die sich auf der geplanten Route fortbewegt) eine gewisse Ungenauigkeit hat, ist es höchstwahrscheinlich, dass die aktuelle Position nicht genau auf der vordefinierten Route liegt. Somit muss die aktuelle Position C1 auf die geplante Route projiziert werden wie Abbildung 23 zeigt.

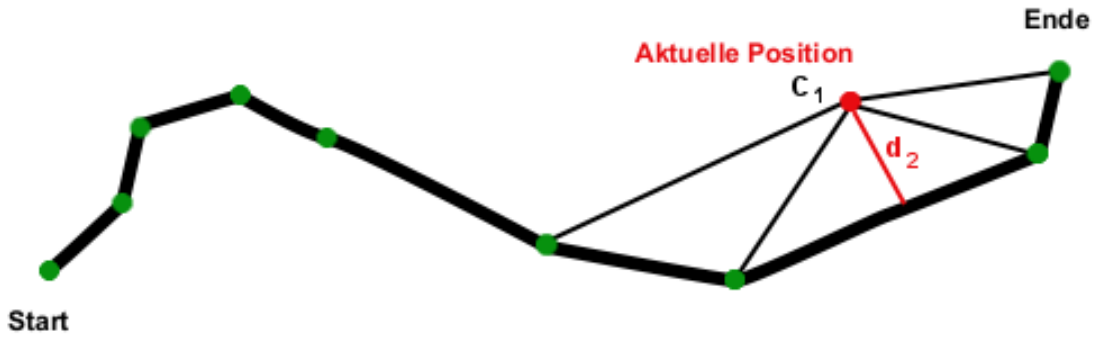


Abbildung 23 Aktuelle Position auf Route projizieren

Dazu werden alle Kanten und Knoten in eine Liste geladen und durch iteriert. Dabei wird die Distanz der aktuellen Position zu allen Kanten und Knoten verglichen.

Kantendistanz

Die Kantendistanz der aktuellen Position C1 zur jeweiligen Kante wird am einfachsten

durch Aufspannen eines Dreiecks ermittelt. Jede Kante bildet mit der aktuellen Position ein Dreieck (s. Abbildung 24). Die Höhe des Dreiecks *d* ist der Abstand der aktuellen Position zur Kante (s. Formel 3).

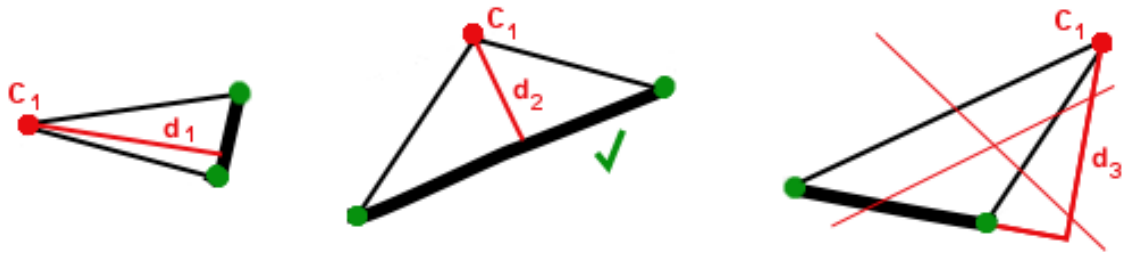


Abbildung 24 Kanten mit aktueller Position bilden Dreiecke

Um eine gültige Höhe zu erhalten, muss die Höhe *d* immer innerhalb des aufgespannten Dreiecks liegen. Um einfach zu bestimmen, wann eine Höhe nicht mehr auf eine Kante projiziert werden kann, können die Winkel α und β zur Hilfe genommen werden. Abbildung 25 zeigt wenn α oder β > 90°, dann liegt die Höhe immer ausserhalb des aufgespannten Dreiecks zur aktuellen Position.

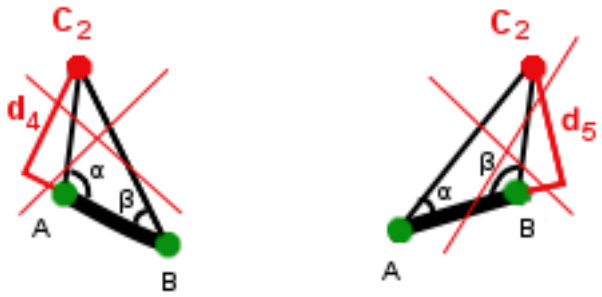


Abbildung 25 Projektion auf Kante nicht möglich

Knotendistanz

Die Knotendistanz wird durch die aktuelle Position C2 zum jeweiligen Knoten bestimmt wie Abbildung 26 zeigt. Es kann in jedem Fall eine gültige Knotendistanz ermittelt werden.

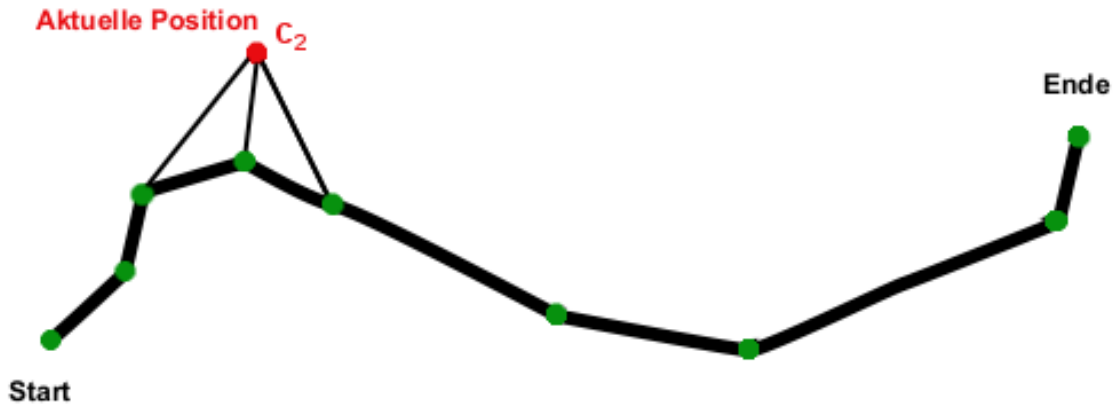


Abbildung 26 Knotendistanz

Position auf Route

Sobald alle Distanzen der aktuellen Position zu allen Kanten und Knoten vorliegen, wird die kürzeste gültige ausgewählt. Wenn die kürzeste Distanz von einem Knoten stammt, gibt derjenige Knoten die aktuelle Position auf der geplanten Route an (s. Abbildung 27).

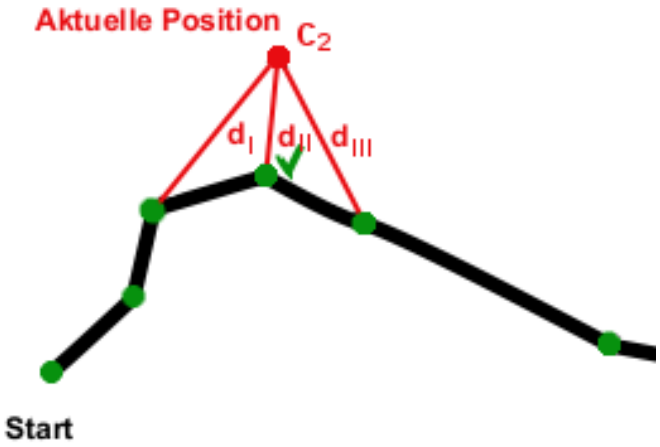


Abbildung 27 Kürzeste Distanz zum Knoten

Liegt die kürzeste Höhe *d* auf einer Kante, so muss diese aufgeteilt werden in die Strecke vor der aktuellen Position (q) und die Strecke nach der aktuellen Position (p) (s. Abbildung 28).

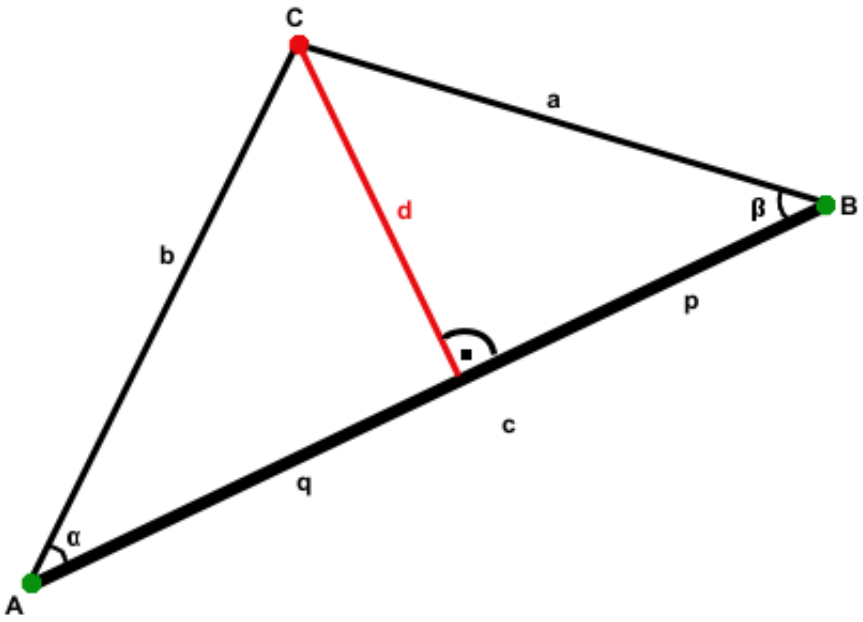


Abbildung 28 Berechnungen auf der Kante

Die Punkte *A*, *B* und *C* liegen alle im Koordinatensystem von WGS84 vor. Damit keine Transformationen gemacht werden müssen, wird versucht, alles mit relativen Distanzen zu berechnen. Mit der Funktion Location.distanzTo(Location) (vgl. Android (Location) – Android Developers) kann die Distanz in Meter zwischen zwei Orten ermittelt werden. Somit ist es möglich, alle Kantenlängen *a,b,c* des Dreiecks zu berechnen. Anschliessend kann mit dem Kosinussatz (s. Formel 1 & 2) die Distanz *q* und *p* in einem nicht rechtwinkligen Dreieck berechnet werden.

$$q = \frac{b^2 + c^2 - a^2}{2c}\ [m] \qquad (1)$$

$$p = c - q\ [m] \qquad (2)$$

Mit dem Satz des Pythagoras (s. Formel 3) kann danach noch die Höhe *d* berechnet werden.

$$d = \sqrt{b^2 - q^2}\ [m] \qquad (3)$$

Somit können alle notwendigen Berechnungen, um die aktuelle Position auf der geplanten Route abzubilden, mit relativen Distanzen durchgeführt werden.

Vergleich geplante Zeit / verstrichene Zeit

Nachdem für alle Fälle die Position auf der geplanten Route bestimmt wurde, kann nun die geplante Zeit zur verstrichenen Zeit verglichen werden. Mit der Totaldistanz ($\boldsymbol{d_{tot}}$) der geplanten Route und der aktuell zurückgelegten Distanz ($\boldsymbol{d_a}$) kann berechnet werden (s. Formel 4), wie viel Prozent der Distanz ($\boldsymbol{p_d}$) bereits zurückgelegt wurde (vergleiche Abbildung 29).

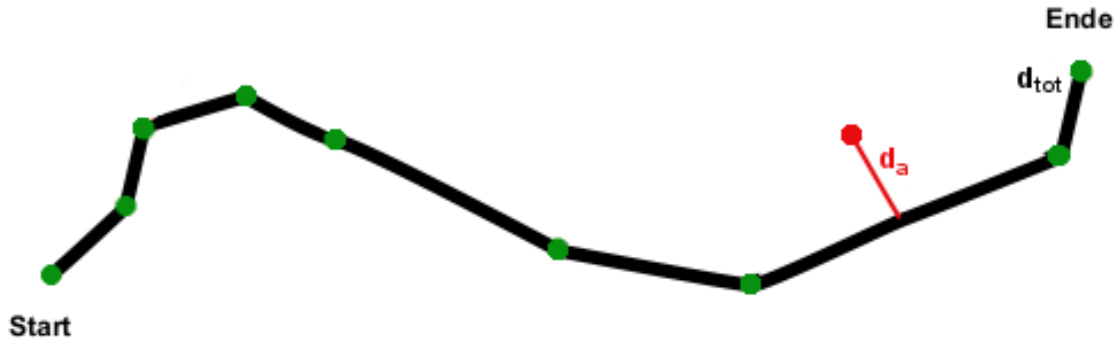


Abbildung 29 Total Distanz und aktuelle Distanz

$$p_d = \frac{100}{d_{tot}} \times d_a \ [\%] \qquad (4)$$

Für die geplante Strecke wurde zuerst mit der Wegzeitformel (s. Wandern) die Dauer ($\boldsymbol{t_b}$) berechnet. Mit der prozentual zurückgelegten Strecke kann nun die prozentual geplante Zeit ($\boldsymbol{t_p}$) berechnet werden (s. Formel 5).

$$t_p = p_d \times \frac{t_b}{100} \ [s] \qquad (5)$$

Nun kann die prozentual geplante Zeit ($\boldsymbol{t_p}$) mit der effektiven Zeit ($\boldsymbol{t_e}$) verglichen werden. Am besten berechnet man nun das Verhältnis (v), damit ersichtlich ist, wie viel man schneller respektive langsamer gegenüber der geplanten Zeit ist (s. Formel 6). Ist v grösser 1, so ist man langsamer gegenüber der geplanten Zeit - ist v kleiner 1, so ist man schneller.

$$v = \frac{t_e}{t_p} \qquad (6)$$

Erfahrungswert berechnen

Nun kann für jede Kante ein neuer Erfahrungswert berechnet werden. Hierzu wird jede geplante Kantenzeit ($\boldsymbol{t_k}$) bis zur aktuellen Position durchlaufen. Für jede Kante wird eine neue Zeit ($\boldsymbol{t_n}$) mit dem vorher berechneten Vergleichswert (v) berechnet (s. Formel 7).

$$t_n = t_k * v\ [s] \quad (7)$$

Mit der neuen Kantenzeit ($\boldsymbol{t_n}$) und der Länge (l) der Kante, kann die neue Geschwindigkeit (g) für die Steigung der jeweiligen Kante berechnet werden (s. Formel 8).

$$g = \frac{l}{t_n}\ [\frac{m}{s}] \quad (8)$$

Diese Geschwindigkeit (g) wird zur entsprechenden Steigung als neuer Erfahrungswert pro Fortbewegungsart in die Datenbank gespeichert.

Vorausberechnung der noch zurückzulegenden Strecke

Für die Vorausberechnung der noch zurückzulegenden Strecke werden alle Kanten (und das angebrochene Kantenstück p) ab der aktuellen Position benötigt. Für alle diese Kanten wird die berechnete Geschwindigkeit anhand der Formel vom SWW aus der Datenbank (Tabelle: Formel) und die Erfahrungsgeschwindigkeit pro Kante aus der Datenbank (Tabelle: Erfahrung) gelesen. In der Datenbank gibt es in der Tabelle Formel zu jedem Prozent Steigung eine entsprechende Geschwindigkeit. Für die Erfahrungsgeschwindigkeit kann es mehrere Geschwindigkeiten (g_i, i ϵ N*) pro Prozent Steigung geben. Hier wird zuerst der Durschnitt aller Erfahrungsgeschwindigkeiten (g_e) für diese Steigung berechnet (s. Formel 9).

$$g_e = \frac{1}{n}\sum_{i=1}^{n} g_i\ ;\ n \neq 0\ [\frac{m}{s}] \quad (9)$$

Um die zukünftige Geschwindigkeit (g_z) zu berechnen, wird der Mittelwert aus der Erfahrungsgeschwindigkeit (g_e) und der Formelgeschwindigkeit (g_f) ermittelt (s. Formel 10).

$$g_z = \frac{g_e + g_f}{2} \left[\frac{m}{s}\right] \quad (10)$$

Die zukünftige Zeit (t_z) einer Kante wird mit der Länge der Kante (*l*) und der zukünftigen Geschwindigkeit (g_z) der Steigung der Kante berechnet (s. Formel 11).

$$t_z = \frac{l}{g_z} [s] \quad (11)$$

Die Summe aller zukünftigen Zeiten (t_s) der noch zurückzulegenden Kanten ergibt schlussendlich die Dauer des noch zu bewältigenden Weges (s. Formel 12).

$$t_s = \sum_{i=1}^{n} t_{z_i} [s] \quad (12)$$

Die neue Dauer (*t*) des noch zurückzulegenden Weges wird mit dem Mittelwert der Summe aller zukünftigen Zeiten (t_s) und der geplanten noch zurückzulegenden Zeit (t_f) berechnet (s. Formel 13).

$$t = \frac{t_s + t_f}{2} [s] \quad (13)$$

Um daraus noch die Ankunftszeit (*eta*) zu berechnen, muss die aktuelle Zeit (t_c) mit der neuen Dauer addiert werden (s. Formel 14).

$$eta = t_c + t [s] \quad (14)$$

Diskussion zum Berechnungsalgorithmus

Die Berechnung der Ankunftszeit kann sicher noch verbessert werden. Hier jedoch noch einige Überlegungen und Begründungen zu den aktuellen Formeln.

Da die Messung der aktuellen Position mit GPS diverse Unsicherheitsfaktoren aufweist, wurde versucht, eine zu starke Abweichung von der Startformel zu verhindern. Deshalb wird der Mittelwert aus der Erfahrungsgeschwindigkeit und der Formelgeschwindigkeit berechnet.

Wird nicht auf der vordefinierten Route fortbewegt, verhält sich der Berechnungsalgorithmus fehlerhaft. Es wird immer versucht, die aktuelle Position auf die geplante Route zu projizieren. Je nach Richtung bleibt diese Position im Extremfall gleich und die Dauer der Fortbewegung läuft weiter. Dies verfälscht die ganze

Berechnung und die abgeleiteten Erfahrungswerte.

Es kann auch vorkommen, dass eine Fortbewegung gestartet wird und die erste GPS Messung, woraus Erfahrungswerte abgeleitet werden könnten, erfolgt erst fast am Ende der Route. In diesem Fall kann nicht genau gesagt werden, auf welcher Steigung welche Verbesserung / Verschlechterung gemacht wurde. Die Verbesserung / Verschlechterung wird über die gesamte Route bis zur aktuellen Position verteilt.

Die besten Erfahrungswerte könnten gesammelt werden, wenn das GPS dauernd (alle 10 Sekunden) die aktuelle Position messen würde (s. Abbildung 30). Hier muss jedoch beachtet werden, dass jede Messung eine gewisse Ungenauigkeit aufweist. So würde die mit GPS gemessene Strecke ziemlich sicher länger ausfallen als die effektive Strecke.

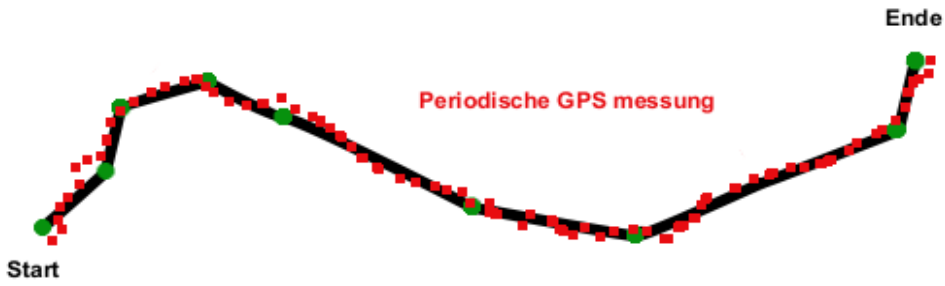


Abbildung 30 GPS Messpunkte auf geplanter Route

Dies könnte noch verbessert werden, wenn die gemessenen GPS Positionen nach der Methode, wie sie in (Douglas et al. 1973:112-122) beschrieben ist, geglättet würden. Die Schwierigkeit ist hier die richtige Toleranz zu finden, um die Route optimal zu glätten. Dies könnte jedoch abhängig von der Qualität des GPS-Signales gemacht werden (vgl. Thurston et al. 2003:144-147).

Hier stellt sich zudem die Frage, ob und wie die geplante Route geglättet werden soll. Diese Frage steht in starker Abhängigkeit zu der Genauigkeit der verwendeten Grundlagedaten. Die Methode, wie sie in (Douglas et al. 1973:112-122) beschrieben ist, könnte aber auch hier zum Zuge kommen.

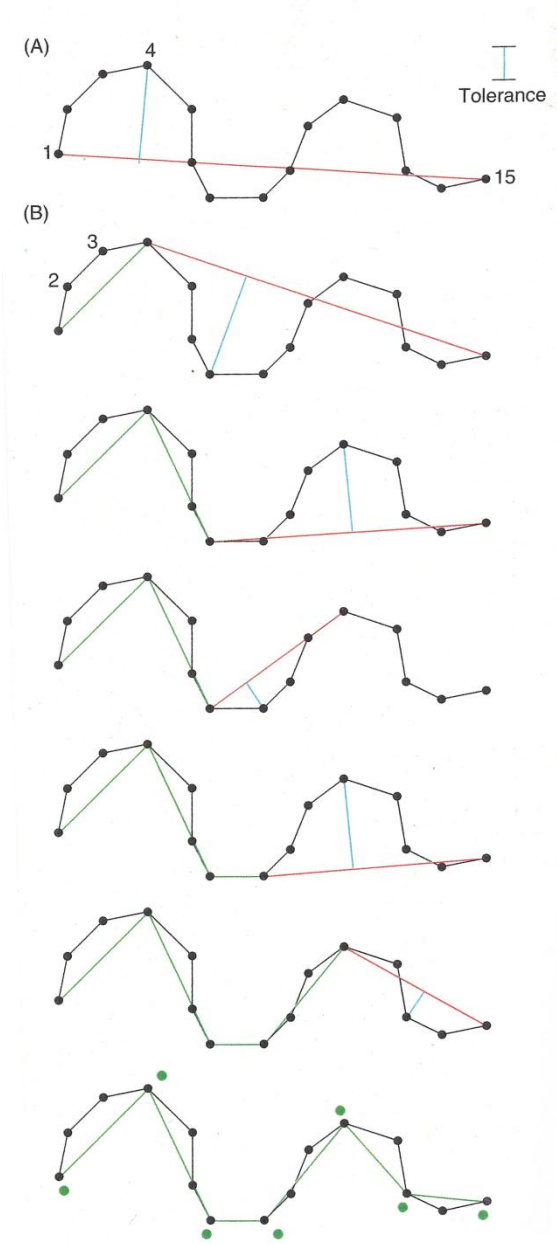


Abbildung 31 Glättung nach Douglas-Poiker (Quelle: Longley et al. 2007:82)

Der Linien – Vereinfachungs-Algorithmus von Douglas – Poiker funktioniert folgendermassen: Die Originallinie hat 15 Punkte. In (A) werden Punkt 1 und Punkt 15 verbunden (rot) und der am weitest entfernte Punkt (4) von dieser Verbindung wurde identifiziert (blau). Die Distanz zu Punkt 4 ist grösser als die vom Benutzer definierte Toleranz. In (B) wird Punkt 1 mit Punkt 4 verbunden (grün). Punkt 2 und 3 liegen innerhalb der definierten Toleranz. Punkt 4 und 15 werden verbunden und der Identifizierungsprozess zum entferntesten Punkt wiederholt sich. Im endgültigen Schritt verbleiben nur noch 7 Punkte (grün markiert). Die grüne Linie ist nie mehr als die benutzer-definierte Toleranz von der schwarzen Linie entfernt.

Der Douglas-Poiker Algorithmus gilt vor allem in der Ebene mit 2 Dimensionen. Um die Höhe auch mit einzubeziehen, könnte der Algorithmus wie er in (Lifan et al. 2009: 703–718) beschrieben ist, verwendet werden.

ERD

Um diverse Werte aus der Applikation dauerhaft zu speichern, ist es sehr hilfreich, die interne Datenbank von Android „SQLite“ zu verwenden. Das Datenmodell ist sehr einfach und gut überschaubar (s. Abbildung 32). Die einzelnen Tabellen haben untereinander keine Beziehung und dienen in erster Linie als Datenablage.

Die beiden Tabellen „Formel“ und „Erfahrung“ sind von der Struktur her identisch. Die Tabelle „Formel“ beinhaltet pro Fortbewegungstyp und Steigung eine Geschwindigkeit. Die Tabelle „Erfahrung“ beinhaltet pro Fortbewegungstyp alle Erfahrungswerte (Geschwindigkeiten) für die entsprechenden Steigungen. Damit die Erfahrungswerte einfach selektiert und auf Wunsch auch wieder gelöscht werden können, wurden diese in einer eigenen Tabelle realisiert.

Die Routentabelle beinhaltet alle Angaben zur Routenverwaltung. Hier werden neue Routen addiert und verwaltet.

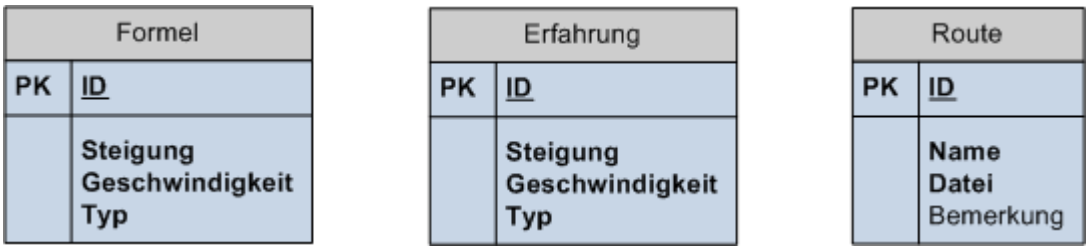


Abbildung 32 ERD

Tabelle Formel		
Feld	**Typ**	**Beschreibung**
ID	INTEGER (PRIMARY KEY; AUTOINCREMENT)	Automatisch hochzählender Primärschlüssel.
Steigung	INTEGER NOT NULL	Prozentuale Steigung.
Geschwindigkeit	REAL NOT NULL	Zugehörige Geschwindigkeit.
Typ	TEXT NOT NULL	HMP Typ (ZUFUSS, FAHRRAD, u.s.w)

Tabelle 22 Beschreibung Tabelle Formel

Tabelle Erfahrung		
Feld	**Typ**	**Beschreibung**
ID	INTEGER (PRIMARY KEY; AUTOINCREMENT)	Automatisch hochzählender Primärschlüssel.
Steigung	INTEGER NOT NULL	Prozentuale Steigung.
Geschwindigkeit	REAL NOT NULL	Zugehörige Geschwindigkeit.
Typ	TEXT NOT NULL	HMP Typ (ZUFUSS, FAHRRAD, u.s.w)

Tabelle 23 Beschreibung Tabelle Erfahrung

Tabelle Route		
Feld	**Typ**	**Beschreibung**
ID	INTEGER (PRIMARY KEY; AUTOINCREMENT)	Automatisch hochzählender Primärschlüssel.
Name	TEXT NOT NULL	Name der Route.
Datei	TEXT NOT NULL	Namen der Routendatei.
Bemerkung	TEXT	Zusätzliche Bemerkungen zur Route.

Tabelle 24 Beschreibung Tabelle Route

Aus architektonischer Sicht macht es vor allem auch Sinn, eine eigene Datenhaltung in der Datenbank anzulegen, damit die Erweiterbarkeit und der Zugriff via standardisierte Schnittstelle gegeben sind. Auch der Zugriff aus der Applikation kann einfacher implementiert werden. Zudem können die abgelegten Daten mit SQL in der Applikation abgefragt werden.

Klassendiagramm

Das Klassendiagramm (s. Abbildung 33) zeigt die implementierten Klassen und ihre Zugehörigkeit (Farben). Grundsätzlich bietet das Android Framework schon sehr viel, um einfache Aktivitäten zu integrieren. So war es nicht nötig (möglich), mit dem Android Framework ein vollständiges MVC Muster zu implementieren. Der Grundgedanke von Model, View und Controller wurde aber dennoch eingehalten. Es wurde versucht, die Klassen so zu implementieren, dass sie eine der Aufgaben vom MVC Muster übernehmen, damit der wichtigste Bestandteil vom MVC Muster - die Erweiterbarkeit und möglichst einfache Pflege – dennoch gewährleistet ist. Die Farben der Klassen bilden 4 Grundkategorien. Die gelb markierten Klassen werden zur Darstellung der Datenbank Daten (model) verwendet. Die violetten Klassen bilden die Geschäftslogik (model) ab. Hier sind alle Berechnungen für die individuelle Wegzeitberechnung implementiert. Die grün markierten Klassen werden für die Steuerung (controller) verwendet. Die roten Klassen sind die Präsentationsschicht (view), welche für die Darstellung der benötigten Daten aus dem Modell zuständig sind.

Datenbank-Klassen (model)

Geschäftslogik (model)

Präsentationsschicht (view)

Steuerung (controller)

Das vollständige Klassendiagramm ist im Anhang aufgeführt (A: Klassendiagramm vollständig).

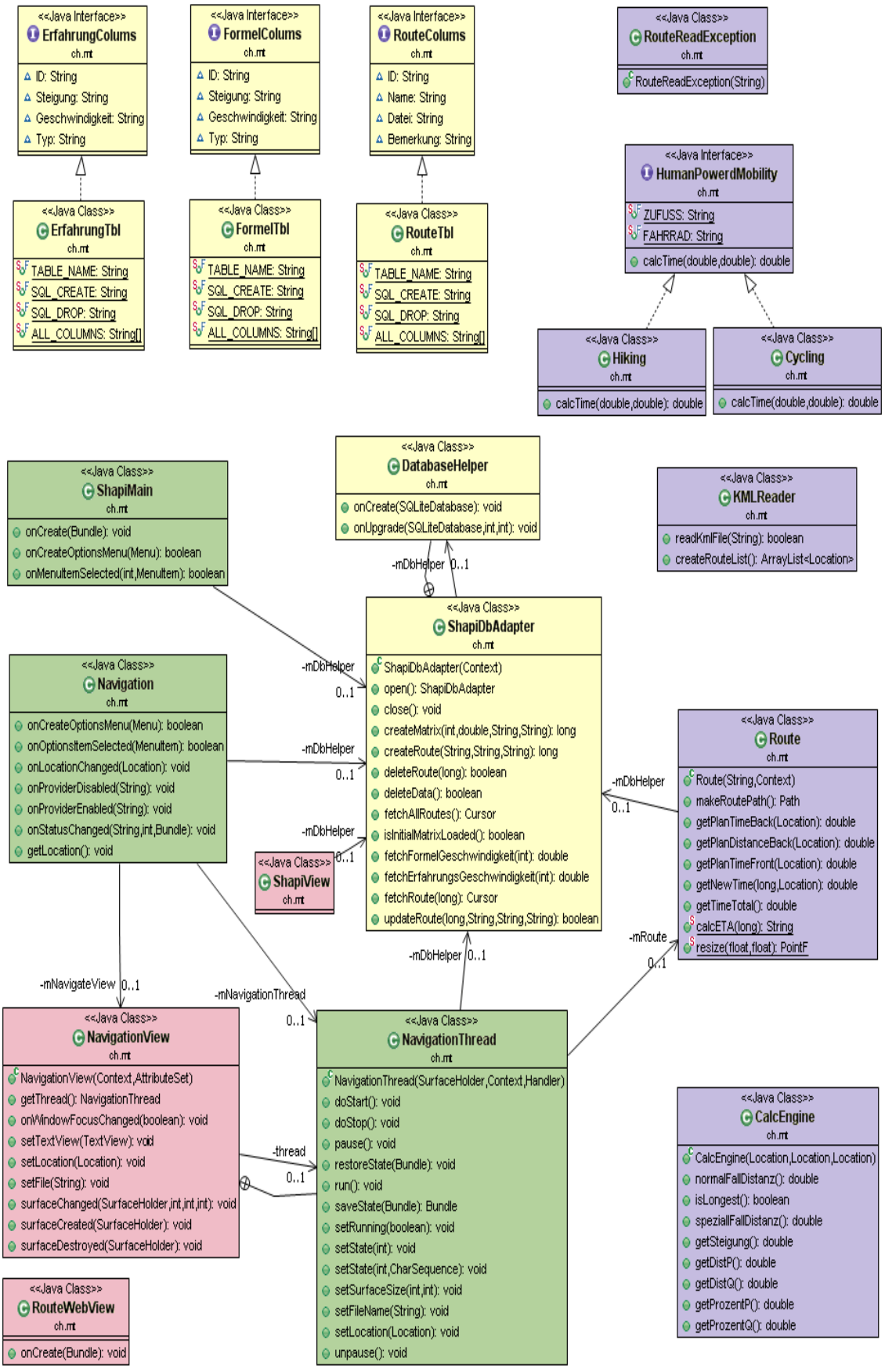


Abbildung 33 Klassendiagramm mit allen öffentlichen Funktionen

Sequenzdiagramm

Das folgende Sequenzdiagramm zeigt einen der komplexesten Abläufe des Prototyps.

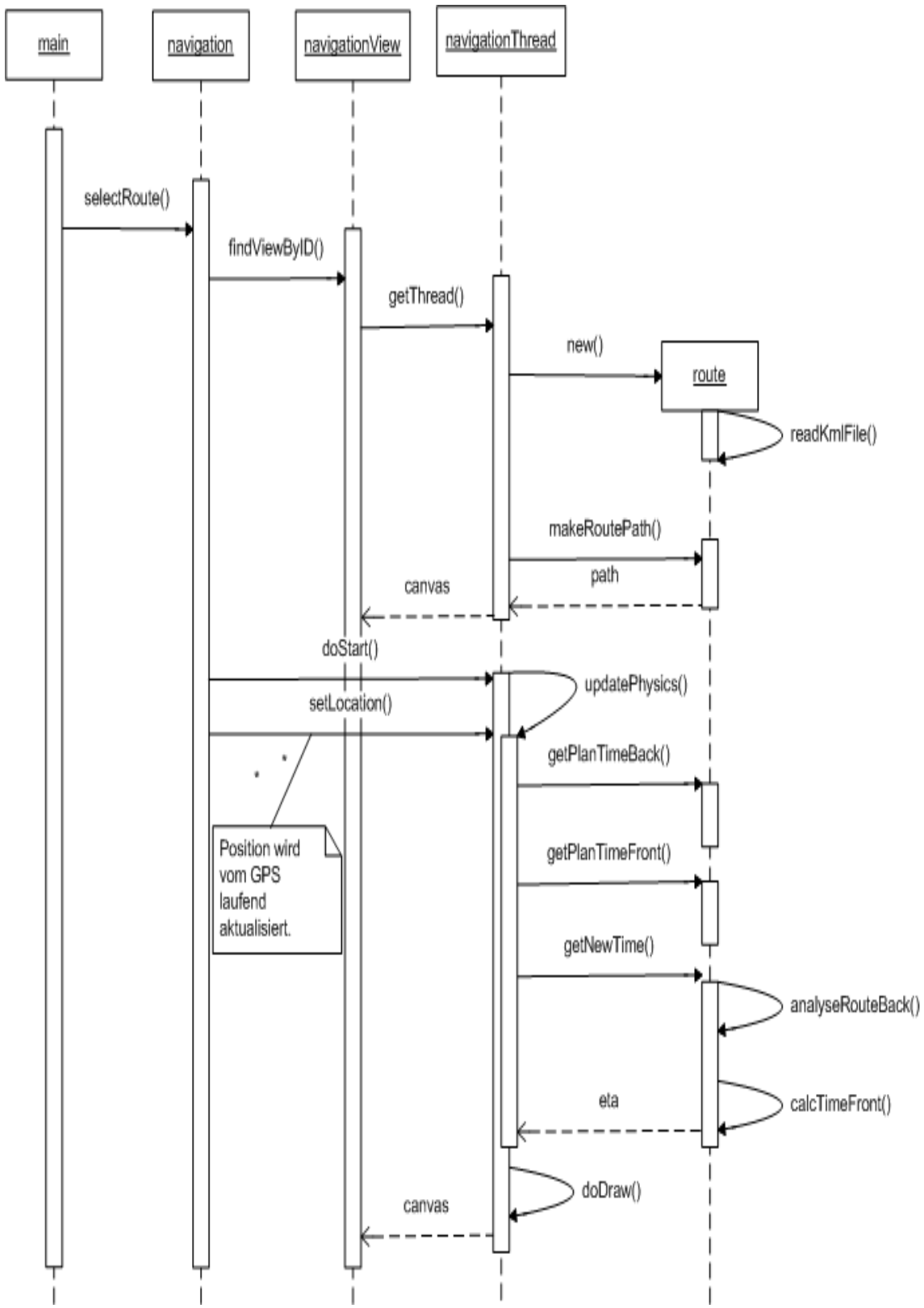


Abbildung 34 Sequenzdiagramm ETA Aktualisierung

Damit die Wegzeit berechnet werden kann, muss ein eigener Thread mit einer Laufzeit erstellt werden. Der Thread muss unterbrochen (pause) und wieder aktiviert (resume) werden können. Zudem muss die aktuelle Position immer wieder erneuert werden. In der Hauptklasse (ShapiMain) wird eine Route selektiert und an die Navigation weitergegeben. Dort wird die Klasse NavigationView und der dazugehörende Thread (NavigationThread) initialisiert. Anschliessend wird ein Routenobjekt anhand des dazugehörigen KML's erstellt. Sobald die KML Datei richtig gelesen wurde, kann der zurückzulegende Pfad als Höhenprofil dargestellt werden. Hier wird auch die Dauer nach der allgemeinen Wegzeitformel berechnet und dargestellt (ohne Erfahrungswerte). Wenn sich der Benutzer am Anfangspunkt der Route befindet und bereit ist, sich auf der Route fortzubewegen, kann Start gedrückt werden und der Thread wird gestartet. Innerhalb des Threads werden nun alle 10 Sekunden die physikalischen Änderungen aktualisiert. Das heisst vor allem die Zeit und der Standort. Damit werden die geplante Zeit, die bereits zurückgelegt wurde, die geplante Zeit, die noch zurückgelegt werden muss und die neue Dauer mit der vorausberechneten Ankunftszeit (mit Erfahrungswerten) berechnet und anschliessend dargestellt. Auch die aktuelle Position wird in dem dargestellten Höhenprofil dargestellt.

Aktivitätsdiagramm

Im folgenden Aktivitätsdiagramm werden die verschiedenen Zustände und Aktivitäten des Prototyps bei der Fortbewegung aufgezeigt. Das Diagramm soll den komplexen Ablauf vereinfacht darstellen.

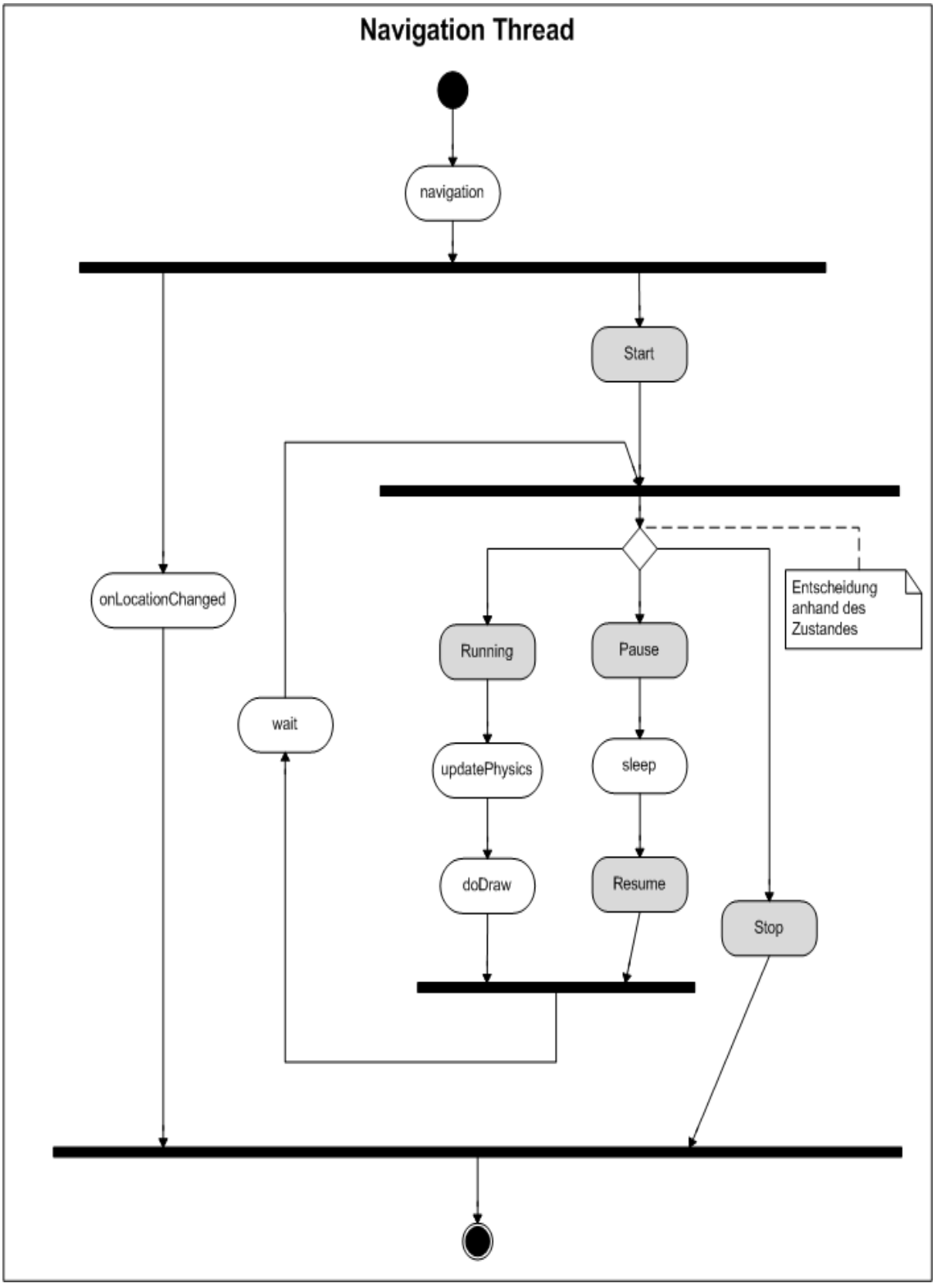


Abbildung 35 Aktivitätsdiagramm Navigation Thread

Sobald der Benutzer die Fortbewegung gestartet hat, wird der Navigation Thread gestartet. Der Hauptprozess bleibt weiterhin bestehen und aktualisiert laufend die Position. Innerhalb des Threads wird zwischen dem Zustand „Running“, „Pause“ und „Stop“ unterschieden. Solange der Zustand „Running“ ist, werden die Funktionen updatePhysiks() und doDraw() aufgerufen. Damit werden die aktuelle Position und Zeit aktualisiert und auf dem Display dargestellt. Anschliessend wartet der Thread, bis die Interrupttime verstrichen ist und prüft erneut den Zustand. Wenn der Zustand auf „Pause“ gesetzt worden ist, wird die Fortbewegungszeit angehalten (die Position sollte sich bei einer Pause sowieso nicht mehr ändern). Sobald der Zustand auf „Resume“ gesetzt worden ist, wird die Fortbewegungszeit wieder aktualisiert und der Zustand wieder auf „Running“ gesetzt. Der Zustand „Running“ kann nur durch den Zustand „Stop“ verlassen werden. Dadurch werden alle zur Route gesammelten Daten gelöscht und der Initialzustand wird wieder hergestellt.

Diskussion zum Prototyp

Durch die bedingten Startprobleme mit Symbian und zusätzlichen Einarbeitungsaufwendungen für Android, stellte sich die Entwicklung des Prototyps als aufwändiger als geplant heraus. Dennoch konnten die wichtigsten Anforderungen vollumfänglich implementiert werden. Aus zeitlichen Gründen wurde jedoch auf einen ausführlichen Test verzichtet. Allgemein wurden aber während der Implementierung und den ersten Tests sehr wichtige Erkenntnisse gewonnen, die für die Erarbeitung der Grundalgorithmen und eine allfällige Weiterentwicklung wertvolle Hinweise erbringen. Während der Implementierung gab es verschiedene Stellen, die sich als aufwändig oder aber auch als eher einfach herausstellten. Hier eine kleine Zusammenfassung:

Für die Routen-Basisdaten wurde KML als Standardformat definiert. Grundsätzlich wird das Format KML beim Versuch, via mobilen Browser aus dem Internet zu laden, gesperrt. Der Inhalt wird auf dem Telefon nicht unterstützt. Damit nicht zu viel Zeit in dieses Problem investiert wurde, sind die KML Dateien direkt via USB auf das mobile Endgerät geladen worden. Leider muss die KML Datei auch selber geparsed werden, da zum Zeitpunkt der Implementierung noch keine brauchbare Bibliothek für Android gefunden wurde.

Die meisten heutigen mobilen Endgeräte mit Android haben diverse Sensoren integriert. So auch ein Sensor, der beim Querstellen des mobilen Endgerätes das Display quer stellt. Dieses schöne Feature wurde bei der Implementierung des Höhenprofils zu einem Problem, dem leider aus Zeitgründen nicht mehr nachgegangen werden konnte. Bei der Darstellung des Höhenprofils gibt es in der aktuellen Version keine „resize“ Funktion, die das Höhenprofil in allen Situationen optimal darstellt. Als Umgehung kann unter Einstellungen „Ausrichtung bei Drehen des Telefons automatisch ändern“, ausgeschaltet werden.

Auch das Einschlafen des mobilen Endgerätes (Bildschirm-Timeout) bot einige Probleme, da der Navigationthread in den Hintergrund geschoben wurde und beim wieder Aufwecken keinen gültigen Zustand mehr hatte. Auch dieses Problem könnte mit dem entsprechenden Zeitaufwand sicher gelöst werden. In der aktuellen Implementierung muss es umgangen werden, indem unter Einstellungen „Verzögerung bis zum automatischen Ausschalten des Bildschirms anpassen“ auf „Kein Timeout“ gestellt wird.

Es gibt aber auch ganz erfreuliche Stellen die sehr einfach umgesetzt werden konnten wie die Datenbank- oder GPS-Schnittstelle.

Die Verbindung zur Datenbank wird ganz einfach mit der Funktion getWritableDatabase() hergestellt. Anschliessend können mit dem Datenbankobjekt und der Funktion query() SQL-Datenbankabfragen gemacht werden.

```
mDbHelper = new DatabaseHelper(mCtx);
mDb = mDbHelper.getWritableDatabase();
```

Um auf die GPS-Schnittstelle zugreifen zu können, muss nach der Initialisierung ein LocationManager Service geholt werden. Auf dem Objekt werden anschliessend der entsprechende Provider, die minimale Zeit und die minimale Distanz für die Erneuerung der Position angegeben.

```
public void getLocation() {
      this.mLM = (LocationManager)getSystemService(
            Context.LOCATION_SERVICE
      this.mLM.requestLocationUpdates(
            LocationManager.GPS_PROVIDER,
            NavigationThread.INTERRUPTTIME * 100, 10, this);
}
```

Wird eine neue Position ermittelt, so wird das Event onLocationChange(Location loc) mit der aktuellen Position als Parameter ausgelöst (vgl. Android (Location) - Android Developers).

```
public void onLocationChanged(Location loc) {
      mNavigateView.setLocation(loc);

}
```

Leider gibt es bei der Berechnung der relativen Distanzen ein kleines Problem. Um die Ankunftszeit zu berechnen wird versucht, alle Berechnungen mit relativen Distanzen durchzuführen. Mit der Funktion distanceTo() kann die Distanz von loc1 zu loc2 berechnet werden. Anhand der Hilfe von Android (vgl. Android (Location) - Android Developers) ist nicht ganz klar, ob nun die 3D Distanz oder nur die 2D Distanz berechnet wird. Mit einem kleinen Test konnte bewiesen werden, dass nur die 2D Distanz berechnet wird.

Der Test wurde folgendermassen durchgeführt. Mit dem WGS-84 Calculator (WGS-84 Calculator 2010) wurde die Distanz von zwei Positionen nur zweidimensional berechnet (s. Abbildung 36).

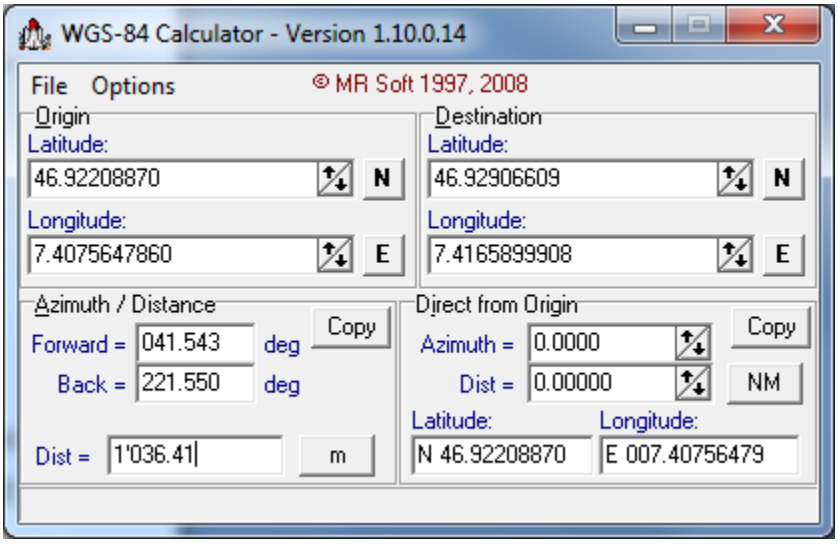


Abbildung 36 Berechnung der 2D Distanz

Anschliessend wurden mit dem Prototyp die genau gleichen zwei Positionen mit folgender Funktion berechnet.

```
float dist = loc1.distanceTo(loc2);
```

Der Eintrag in der Logdatei (s. Abbildung 37) gibt Auskunft über die loc1, loc2, Steigung und Distanz:

```
7.407564786081946,46.92208870227822,598.0000000000001
Location[mProvider=Weg,mTime=0,mLatitude=46.92208870227822,mLongitude=7.407564786081946,mHasAltitude=true,mAltitude=598.0
7.416589990819402,46.92906609375455,568
Location[mProvider=Weg,mTime=0,mLatitude=46.92906609375455,mLongitude=7.416589990819402,mHasAltitude=true,mAltitude=568.0
Dist= 1036.412109375
Steigung= -2.8946014552156645
```

Abbildung 37 Auszug der Positionen aus der Logdatei

Da die Distanz aus der Logdatei genau gleich ist wie die berechnete 2D Distanz, kann gesagt werden, dass die Funktion distanceTo() nur 2D Distanzen berechnet. Nun kann mit dem Satz des Pythagoras (s. Formel 15) die Differenz von der 2D Distanz (d_2) zur 3D Distanz (d_3) berechnet werden.

$$d_3 = \sqrt{{d_2}^2 + h^2} \quad (15)$$

Mit den Zahlen aus dem vorhergehenden Test gibt das eine 3D Distanz von 1036.84 Meter. Der Unterschied von der 2D Distanz zur 3D Distanz ist also nur 0.43 Meter und kann somit vernachlässigt werden.

Die ersten Tests ergaben relativ ungenaue GPS-Höhen. Mit der Korrektur von 50m (gilt für die Schweiz) aufgrund der Differenz vom WGS-84 Ellipsoid zur Route, die mit dem darunterliegenden SRTM Höhenmodell (Geoid EGM96) (SRTM) digitalisiert wurde, waren die Resultate besser. Jedoch springt der rote Standortpunkt immer noch ab und zu neben dem Höhenprofil hoch und runter. Der Effekt wird natürlich durch die überhöhte Darstellung verstärkt. Dies könnte sicher noch verbessert werden, um den Benutzer nicht zu stark zu irritieren.

Die ersten Erfahrungen haben ergeben, dass die Ankunftszeit relativ genau ist. Auch alle anderen Angaben sind sehr realitätsnah. Nun müsste jedoch ein ausführlicher Test durchgeführt werden, welcher die Zeit exakt misst und vergleicht. Die gesammelten Erfahrungswerte könnten aus der Datenbank gelesen und ausgewertet werden. So könnte ein persönliches Profil erstellt werden. Dies würde jedoch den Rahmen dieser Arbeit sprengen.

Allgemein kann gesagt werden, dass es im realisierten Prototyp sicher noch viel Verbesserungspotenzial gibt. Das Hauptziel, den Prototyp für die Erarbeitung des Grundalgorithmus für die individuelle Wegzeitberechnung zu verwenden, wurde aber vollumfänglich erreicht. Zudem wurden wichtige Erkenntnisse im ganzen Umfeld gesammelt.

Bedienung

- Nach der Installation von Shapi.apk kann die Applikation Shapi gestartet werden.

- Die Routenliste erscheint.
- Durch direktes Drücken auf die Route wird die Route selektiert.
- Durch Drücken auf die Menü Taste erscheint das eingeblendete Menü.
- Um eine neue Route herunterzuladen, das Eingabefeld „Route herunterladen“ drücken.

- Um eine Route herunterzuladen, benötigt man Zugriff auf das Internet. In der dargestellten Seite sind verschiedene Routenplanungsseiten aufgelistet, welche direkt angewählt werden können. Anschliessend wird ein Browserfenster mit der jeweiligen Seite geöffnet. Dort kann eine gewünschte Route angewählt und auf das mobile Endgerät geladen werden.

- Sobald die Route heruntergeladen worden ist, kann im Hauptmenü unter „Route bearbeiten“ die Route zur Routenliste hinzugefügt werden. Es muss ein Routenname und die Routendatei angegeben werden. Zudem kann eine Bemerkung zur Route eingegeben werden.

- Anschliessend kann im Hauptmenü unter „Auf Route navigieren“ auf der selektierten Route fortbewegt werden. Danach wird das Höhenprofil und die Zeit, die für das Zurücklegen der geplanten Strecke benötigt wird, dargestellt.
- Durch Drücken der Menütaste erscheint das dargestellte Menü.
- Wenn die Fortbewegung beginnen kann und der geplante Ausgangsort erreicht ist, kann „Start“ gedrückt werden.

- Sobald das GPS-Signal die Startposition erfasst hat, wird der nebenstehende Bildschirm angezeigt. Nun kann mit der Fortbewegung begonnen werden.

- Während der Fortbewegung aktualisiert das GPS die Position. Damit werden die neuberechnete geometrisch geplante zurückgelegte Zeit (<-), die geometrisch geplante noch zurückzulegende Zeit (->), die Dauer (d:) und die Ankunftszeit (ETA:) angezeigt.
- Durch Drücken der Menütaste und des Eingabefeldes „Pause“ werden das GPS und die Zeit angehalten.
- Durch Drücken der Menütaste und des Eingabefeldes „Resume“ werden das GPS und die Zeit wieder aktiviert.

- Zum Beenden der Fortbewegung die Menütaste und das Eingabefeld „Stop“ drücken. Anschliessend mit der „Zurück“ Taste in die Routenliste springen. Dort kann wieder aus allen möglichen Optionen ausgewählt werden.
- Um die Erfahrungswerte zu löschen, im Hauptmenü das Eingabefeld „Daten löschen“ auswählen. Um eine vorhandene Route aus der Routenliste zu löschen, im Hauptmenü das Eingabefeld „Route löschen“ auswählen.
- Schlussendlich um die Applikation zu beenden, im Hauptmenü das Eingabefeld „Beenden“ auswählen.

7. Zusammenfassung

Die in der Einleitung erwähnten Hypothesen und Ziele werden in diesem Kapitel überprüft und das gewählte Vorgehen wird beleuchtet. Zudem wird ein Ausblick für zukünftige Weiterentwicklungen gegeben.

Schlussfolgerung

Hypothesen

1. **Mit den heutigen technischen und infrastrukturellen Mitteln lässt sich eine bessere Wegzeitberechnung als die vorberechneten Zeiten der Signalisierungstafeln realisieren.**

Mit dem heute wachsenden Markt an mobilen Endgeräten mit integriertem GPS ist es möglich, eine individuelle Wegzeitberechnung zu erstellen. Die in Kapitel 2 und 3 aufgeführten vorhandenen Lösungen und Grundlagen zeigen, dass die einzelnen Bausteine verfügbar sind. Mit dem erstellten Prototyp, der auf den Grundlagen von Kapitel 3 aufbaut, konnte gezeigt werden, dass es möglich ist, eine verbesserte Wegzeitberechnung zu erstellen. In Zukunft kann auch damit gerechnet werden, dass die Kartengrundlagen für HPM verbessert werden (genaueres DHM und digitalisierte Fussgängerwege) und dass der technische Fortschritt bei den mobilen Endgeräten weiter ausgebaut wird. Die bewusste Abgrenzung der genauen Positionsbestimmung und Startzeitberechnung bietet hier noch einige Verbesserungsmöglichkeiten. Mit der Fokussierung auf das Wandern konnten sicher die grundlegenden Überlegungen gemacht werden. Auch den anderen HPM Fortbewegungsmöglichkeiten steht nichts im Weg, um die erarbeiteten Überlegungen für die individuelle Wegzeitberechnung anwenden zu können. Grundsätzlich konnte gezeigt werden, dass es möglich ist, eine dem Individuum angepasstere Wegzeitberechnung zu realisieren.

2. **Es kann für (alle) muskelgetriebenen Fortbewegungsarten ein Algorithmus für die individuelle Wegzeitberechnung gefunden werden.**

Der in Kapitel 6. Prototyp beschriebene Algorithmus für die individuelle Wegzeitberechnung wurde in erster Linie für das Wandern erstellt. Er kann jedoch

mit der Adaption der jeweiligen Fortbewegungs-Startformel leicht angepasst werden.

Diese Hypothese kann im Allgemeinen bestätigt werden, jedoch muss der erstellte Algorithmus noch in einem grösseren Umfeld getestet werden, um eine klarere Aussage über die Genauigkeit machen zu können.

3. **Der Benutzer kann zu jedem Zeitpunkt der Fortbewegung die aktuelle individuell berechnete Ankunftszeit verfügbar haben.**

Mit dem realisierten Prototyp konnte gezeigt werden, dass es möglich ist, auf einem mobilen Endgerät mit GPS die individuell berechnete Ankunftszeit für jeden Moment der Fortbewegung darzustellen. Da eine HPM Aktivität auch aus Pausen besteht, wurde dies bei der Implementation ebenfalls berücksichtigt. Die Pause und das anschliessende wieder Fortbewegen muss aktiv beim mobilen Endgerät eingegeben werden. Somit wird die Fortbewegungszeit angehalten und alle Berechnungen sollten anschliessend wieder richtig durchgeführt werden können. Folglich konnte diese Hypothese bestätigt werden.

Im Umgang mit den Pausen gibt es sicher noch Verbesserungspotenzial. Es wäre z.B. vorstellbar, dass bei gleichbleibendem Standort über längere Zeit die Pause automatisch eingeschaltet würde. Sobald der Standort wieder ändert, könnte die Pause wieder aufgehoben werden.

Synthese

Das Hauptziel dieser Arbeit bestand darin, den Algorithmus zur individuellen Wegzeitberechnung in einem Prototyp zu realisieren. Die gewählte Vorgehensweise kann mit folgenden Punkten zusammengefasst werden.

Literatur- und Online - Recherchen

In diesem wichtigen Schritt konnte der aktuelle Stand des Themas und der Technik erfasst werden. Im Bereich der mobilen Endgeräte herrscht seit ca. 2 Jahren eine extreme Dynamik. Seit der Einführung von Apples iPhone und Android gibt es eine starke Tendenz zu immer verbundenen (online) mit GPS und diversen anderen Sensoren ausgestatteten mobilen Endgeräten. Somit ist es sehr einfach, aktuelle standortbezogene Informationen aufzubereiten. Zudem wurde die Verteilung von

Software für diese Endgeräte stark vereinfacht. In den sogenannten Markets kann online aus den unterschiedlichsten Applikationen (Apps) ausgewählt werden.

Im Bereich digitaler HPM Werkzeuge gibt es inzwischen gute Software, mit der eine Route geplant werden kann. Ein ähnlicher Ansatz zur individuellen Wegzeitberechnung, wie in dieser Arbeit beschrieben, wurde jedoch nicht gefunden.

Anforderungsanalyse

In der Anforderungsanalyse wurden die aus den Hypothesen und vorgegebenen Zielen definierten Anforderungen erarbeitet. Dabei wurden die fundamentale Architektur und die Schnittstellen festgelegt. Durch das vorgängige Recherchieren konnten die Bedürfnisse identifiziert werden, welche letztendlich das Aussehen des Prototyps bestimmen. Zudem wurden die Anforderungen nummeriert und nach Prioritäten aufgeteilt.

Anwendungsfälle

Das Identifizieren der Anwendungsfälle geschah auf der Grundlage der Anforderungsanalyse. Da es keinen effektiven Benutzer gibt, wurde mit Rücksprache des Betreuers versucht, möglichst wahrheitsgetreu die Sicht eines hypothetischen Benutzers einzunehmen. Effektive Benutzer hätten sicher die Anwendungsfälle noch erweitern können, da jedoch nur ein Prototyp geplant ist, um Erfahrungswerte zu sammeln, genügt vorerst dieser Stand. Die Akteure und Anwendungsfälle wurden mit UML in einem Anwendungsfalldiagramm veranschaulicht. Anschliessend wurden die identifizierten Anwendungsfälle nummeriert, genauer beschrieben und den Anforderungen zugewiesen. Zudem gibt ein Status den aktuellen Implementationszustand an.

Prototyp

Die Umsetzung des Prototyps nahm viel Zeit in Anspruch. Hier stellte sich heraus, ob die erarbeiteten Grundlagen, erstellten Anforderungen und identifizierten Anwendungsfälle in die Realität umgesetzt werden können. Zur Erstellung des Prototyps musste zuerst die Infrastruktur bereitgestellt werden. Hier gab es, nach

einem ersten Versuch mit Symbian, den Entscheid, den Prototyp auf Android zu implementieren. Schlussendlich kann gesagt werden, dass dieser Entscheid sicher richtig gewesen ist. Nachdem die Entwicklungsumgebung stand, mussten die Grundlagen der Applikationsentwicklung für Android erarbeitet werden. Nach einigen Tests mit Beispielprogrammen konnte anschliessend mit UML das Klassendiagramm und das ERD entwickelt werden. Auch die komplexen Bereiche wurden identifiziert und mit Sequenzdiagramm und Aktivitätsdiagramm geplant. Im iterativen Entwicklungsprozess wurden die gewonnenen Erfahrungen immer wieder in die UML Diagramme nachgeführt. Der entwickelte Quellcode wurde permanent mit der Versionsverwaltung Subversion verwaltet. Durch manuelle Refaktorierung (vgl. Wake 2003) konnte die Lesbarkeit, Verständlichkeit, Wartbarkeit und Erweiterbarkeit des Quellcodes verbessert werden. Anschliessend wurde die Applikation auf das bestehende, mobile Endgerät installiert und getestet. Nach dem Beheben einiger kleinerer Fehler wurde die Entwicklung abgeschlossen und die Dokumentationsarbeiten begannen. Die Hauptarbeit lag vor allem im detaillierten Wiedergeben der gesammelten Erfahrungen im vorliegenden Dokument. Zudem wurde eine Java Dokumentation mittels javadoc generiert.

Das gewählte Vorgehen führte mit einigen kleineren Umwegen schlussendlich zum Ziel der individuellen Wegzeitberechnung, implementiert als Prototyp auf einem mobilen Endgerät. Die iterative Entwicklung unter Zuhilfenahme von UML hat sich bewährt und kann hoffentlich für weiterführende Arbeiten, unabhängig von der aktuellen technischen Implementation, verwendet werden.

Ausblick

Der Ausblick muss im Gesamtkontext der mobilen Navigation mittels HMP angeschaut werden. Der entwickelte Prototyp könnte heute bereits sehr einfach als zusätzliche Navigationsseite in eine für HMP entwickelte Applikation integriert werden. Hauptseite ist eine 2D Karte mit der aktuellen Position und dem geplanten Weg. Um zusätzliche Informationen zu erhalten, kann eine Seite mit dem Höhenprofil angezeigt werden (der eigentliche Prototyp). Auch hier wird wieder die aktuelle Position markiert. Alle numerischen Informationen wie Ankunftszeit, Dauer, aktuelle Höhe uvm. könnten zusätzlich in die Benutzeroberfläche integriert werden. Somit hätte der HPM Benutzer alle Informationen, die ihn für seine Fortbewegung interessieren, beisammen.

Weiter denkbar wäre ein Ausbau in Richtung ökonomischster Weg. In Kombination mit einem Wegrouting könnte pro Fortbewegungsart der effizienteste und ökonomischste Weg ermittelt werden. Dabei könnten auch die individuellen Erfahrungswerte in die Berechnung einfliessen. Somit hätte man die auf seine Person zugeschnittene ökonomischste Route.

Auch sehr praktisch wäre eine Schnittstelle zu den Standorten, Ankunfts- und Abfahrtszeiten des öffentlichen Verkehrs. Sobald die Ankunftszeit der geplanten Route feststeht, könnte die Wegzeit zur nächstgelegenen Haltestelle berechnet und der nächste Anschluss des öffentlichen Verkehrs ermittelt werden. Hier müsste berücksichtigt werden, dass die Route von einer öV-Haltestelle und bis zu einer öV-Haltestelle geplant wird.

Wenn der Markt in diesem Gebiet in Zukunft weiter so dynamisch und aktiv bleibt wie bisher, werden sich sicher noch einige weitere spannende Ideen ergeben. Ausserdem wird das Orientieren und Navigieren auf dem Globus mit standortbezogener Information immer einfacher. Dies zeigen auch die Bestrebungen von Googles Navigationslösung mit Streetview, wo man das effektive Strassenbild für die Navigation verwendet.

Trotz all dieser technischen und infrastrukturellen Entwicklungen bieten die aktuellen Signalisationstafeln auch einige Vorteile wie zum Beispiel einfache Herstellbarkeit oder kein Energieverbrauch im Betrieb (uvm.). Somit bleiben sie hoffentlich auch in Zukunft

neben all den ergänzenden technischen Neuerungen erhalten, denn sie gehören nach wie vor zum Ortsbild wie die Wanderschuhe und der Rucksack zum Wandern.

Literaturverzeichnis

Android (Architecture) - Android Developers: Online verfügbar unter http://developer.android.com/guide/basics/what-is-android.html, zuletzt geprüft am 11.10.2010.

Android (Klassen und Schnittstellen) – Wikipedia: Online verfügbar unter http://de.wikipedia.org/wiki/Android_(Betriebssystem)#Architektur, zuletzt aktualisiert am 06.10.2010, zuletzt geprüft am 07.10.2010.

Android (Location) - Android Developers: Online verfügbar unter http://developer.android.com/reference/android/location/Location.html, zuletzt geprüft am 24.09.2010.

Android (SDK) - Android Developers: Online verfügbar unter http://developer.android.com/sdk/index.html, zuletzt geprüft am 12.10.2010.

Android (Versionsverlauf) – Wikipedia: Online verfügbar unter http://de.wikipedia.org/wiki/Android_(Betriebssystem)#Versionsverlauf, zuletzt aktualisiert am 06.10.2010, zuletzt geprüft am 07.10.2010.

Becker, Arno; Pant, Marcus; Müller, David (2010): Android 2. Grundlagen und Programmierung. 2., aktualisierte und erw. Aufl. Heidelberg: dpunkt-Verl.

Beier Andreas; Linke Andreas; Schulz Hajo (2010): Die eigene App. Entwickeln für Android, iPhone, WebOS, Symbian, Blackberry und Windows Mobile. In: c't, H. 16, S. 90–95. Online verfügbar unter http://www.heise.de/kiosk/archiv/ct/2010/16/90_kiosk, zuletzt geprüft am 13.10.2010.

Burnette, Ed; Staudemeyer, Jörg (2010): Eclipse IDE. Kurz & gut ; [für Java-Entwickler]. 2. Aufl. Bejing: O'Reilly

Buschmann, Frank; Löckenhoff, Christiane (2000): Pattern-orientierte Softwarearchitektur. Ein Pattern-System. 1., korr. Nachdr. München: Addison-Wesley.

Climbbybike difficulty index: Online verfügbar unter http://www.climbbybike.com/climb_difficulty.asp, zuletzt geprüft am 02.09.2010.

Date, C.J; Darwen, H. (1999): SQL - Der Standard. SQL/92 mit den Erweiterungen CLI und PSM. Bonn: Addison-Wesley-Longman.

Douglas, D. H. and Peucker T. K. (1973): Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. In: Canadian Cartographer, Volume 10, Jg. 73, H. No. 2, S. 112–122.

Eclipse: Online verfügbar unter http://www.eclipse.org/, zuletzt geprüft am 12.10.2010.

Einstein, Albert (1979): Aus meinen späten Jahren. Stuttgart: Deutsche Verlags-Anstalt.

Esri: Online verfügbar unter http://www.esri.com/, zuletzt geprüft am 12.10.2010.

Hurn, J. (1989): GPS –. A. Guide to the Next Utility. Sunnyvale: Trimble Navigation.

KML - Google Code: Online verfügbar unter http://code.google.com/intl/de-DE/apis/kml/documentation/kmlreference.html, zuletzt aktualisiert am 07.10.2010, zuletzt geprüft am 07.10.2010.

KML – OGC: Open Geospatial Consortium, Inc. (Version: 2.2.0). Online verfügbar unter http://www.opengeospatial.org/standards/kml, zuletzt geprüft am 23.08.2010.

Krüger, Guido; Stark, Thomas (2010): Handbuch der Java-Programmierung. Standard Edition Version 6. 6., aktualisierte Aufl., [Nachdr.]. München: Addison-Wesley

Lau Oliver (2008): Gut gespurt. GPS-Tracks aufzeichnen, bearbeiten und auswerten. In: c't, H. 19, S. 112–117. Online verfügbar unter http://www.heise.de/kiosk/archiv/ct/2008/19/112_kiosk, zuletzt geprüft am 12.10.2010.

Lifan Feia; Jin Heb (2009): A three-dimensional Douglas-Peucker algorithm and its application to automated generalization of DEMs. In: International Journal of Geographical Information Science, Jg. Volume 23, H. Issue 6, S. 703–718.

Longley, Paul A; Goodchild, Michael F.; Maguire, David J.; Rhind, David W. (2007): Geographical information systems and science. 2. ed., reprinted. Chichester: Wiley.

Meier, Reto (2010): Professional Android 2 application development. Indianapolis, Ind.: Wiley (Wrox programmer to programmer).

National Marine Electronic Association: Online verfügbar unter http://www.nmea.org/content/nmea_standards/nmea_standards.asp, zuletzt geprüft am 07.10.2010.

Oestereich, Bernd; Bremer, Stefan (2009): Analyse und Design mit UML 2.3. Objektorientierte Softwareentwicklung. 9., aktualisierte und erw. Aufl. München: Oldenbourg.

UML – OMG: Object Management Group, Online verfügbar unter http://www.omg.org/spec/UML/, zuletzt geprüft am 07.10.2010.

Owens, Michael (2006): The definitive guide to SQLite. Berkeley, Calif.: Apress

Pilone, Dan; Pitman, Neil; Heymann-Reder, Dorothea (2006): UML 2.0 in a Nutshell. 1. Aufl. Beijing: O'Reilly.

SQL-92: Online verfügbar unter http://www.contrib.andrew.cmu.edu/~shadow/sql/sql1992.txt, zuletzt aktualisiert am 02.04.1997, zuletzt geprüft am 04.10.2010.

SQLite: Online verfügbar unter http://www.sqlite.org/, zuletzt aktualisiert am 25.08.2010, zuletzt geprüft am 04.10.2010.

SRTM: Shuttle Radar Topography Mission, Online verfügbar unter http://www2.jpl.nasa.gov/srtm/, zuletzt aktualisiert am 18.06.2009, zuletzt geprüft am 11.10.2010.

St. Laurent, Simon; Fitzgerald, Michael (2006): XML. Kurz & gut. Dt. Ausg. der 3. Aufl. Beijing: O'Reilly

Swiss Map Mobile – Swisstopo: Swisstopo. Online verfügbar unter http://www.swisstopo.admin.ch/internet/swisstopo/de/home/products/maps/mobile.html, zuletzt aktualisiert am 09.04.2010, zuletzt geprüft am 10.08.2010.

Gehzeit Formel - Schweizer Wanderwege (2007): SWW. Online verfügbar unter http://www.wandern.ch/index.php?id=210, zuletzt geprüft am 03.08.2010.

Thurston, Jeff; Poiker, Thomas K.; Moore, J. Patrick (2003): Integrated geospatial technologies. A guide to GPS, GIS, and data logging. Hoboken, NJ: John Wiley & Sons.

U.S. Coast Guard Navigation Center: Online verfügbar unter http://www.navcen.uscg.gov/, zuletzt geprüft am 07.10.2010.

U.S. Naval Observatory: Online verfügbar unter http://www.usno.navy.mil/USNO/time/gps, zuletzt geprüft am 06.10.2010.

Wake, William C. (2004): Refactoring workbook. Boston, Mass.: Addison-Wesley.

Wegezeitberechnungen beim Wandern - Outdoor.de: Online verfügbar unter http://www.outdoor.de/sport/wandern-trekking/wegezeitberechnungen-beim-wandern-1331/, zuletzt geprüft am 03.08.2010.

WGS-84 Calculator (2010): Online verfügbar unter http://www.mrsoft.fi/ohj02en.htm, zuletzt aktualisiert am 05.04.2010, zuletzt geprüft am 13.10.2010.

Wilson, David Gordon (2004): Bicycling Science. 3rd Edition. The MIT Press (Hg.).

Zogg, Jean-Marie (2004): GPS Grundlagen. ublox. Online verfügbar unter http://www.akrempl.com/geohome/gps_doku/GPS_Grundlagen.pdf, zuletzt aktualisiert am 01.03.2004, zuletzt geprüft am 07.10.2010.

Anhang

A: Klassendiagramm vollständig

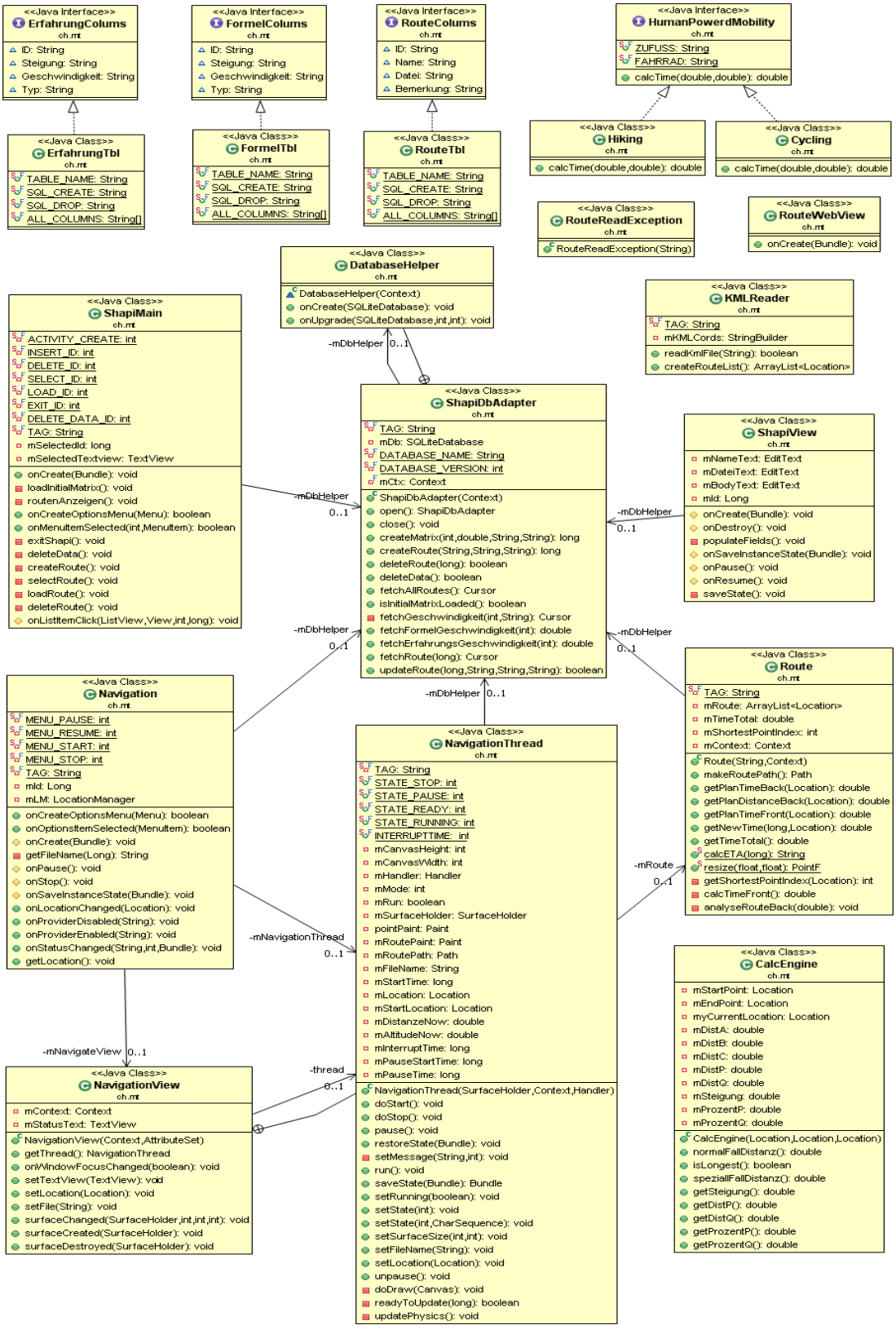


Abbildung 38 Klassendiagramm vollständig

B: AndriodManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest
xmlns:android="http://schemas.android.com/apk/res/android"
      package="ch.mt"
      android:versionCode="1"
      android:versionName="1.0">
    <application android:label="@string/app_name"
android:icon="@drawable/shapi">
        <activity android:name=".ShapiMain"
                  android:label="@string/app_name">
            <intent-filter>
                <action
android:name="android.intent.action.MAIN" />
                <category
android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity android:name=".Navigation" />
        <activity android:name=".ShapiView" />
        <activity android:name=".RouteWebView" />
    </application>
    <uses-sdk android:minSdkVersion="2" />
<uses-permission
android:name="android.permission.ACCESS_FINE_LOCATION"></us
es-permission>
<uses-permission
android:name="android.permission.ACCESS_LOCATION_EXTRA_COMM
ANDS"></uses-permission>
<uses-permission
android:name="android.permission.INTERNET"></uses-
permission>
<uses-permission
android:name="android.permission.WRITE_EXTERNAL_STORAGE"></
uses-permission>
</manifest>
```

C: Layout View edit.xml

```
<?xml version="1.0" encoding="utf-8"?>

<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
     android:orientation="vertical"
android:layout_width="fill_parent"
     android:layout_height="fill_parent">

     <LinearLayout android:orientation="vertical"
          android:layout_width="fill_parent"
          android:layout_height="wrap_content">


          <TextView android:layout_width="fill_parent"
               android:layout_height="wrap_content"
               android:text="@string/title" />
          <EditText android:id="@+id/title"
            android:layout_width="fill_parent"
               android:layout_height="wrap_content"
               android:layout_weight="1"/>
          <TextView android:layout_width="fill_parent"
               android:layout_height="wrap_content"
               android:text="@string/file" />
          <EditText android:id="@+id/file"
            android:layout_width="fill_parent"
               android:layout_height="wrap_content"
               android:layout_weight="1"/>
     </LinearLayout>

     <TextView android:layout_width="fill_parent"
          android:layout_height="wrap_content"
          android:text="@string/body" />
     <EditText android:id="@+id/body"
android:layout_width="fill_parent"
          android:layout_height="wrap_content"
          android:layout_weight="1"
          android:scrollbars="vertical" />

     <Button android:id="@+id/confirm"
       android:text="@string/confirm"
          android:layout_width="fill_parent"
          android:layout_height="wrap_content" />


</LinearLayout>
```