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

„Plattformübergreifende, standortbezogene Mapping Apps mit freien Geodaten, offenen Standards und Open Source-Komponenten“

vorgelegt von

Dipl.-Ing. (BA) Hilmar Klink
U1381, UNIGIS MSc Jahrgang 2008

Zur Erlangung des Grades
„Master of Science (Geographical Information Science & Systems) – MSc(GIS)”

Gutachter:
Ao. Univ. Prof. Dr. Josef Strobl

München, 31. Dezember 2010

Danksagung

Hiermit danke ich Ao. Univ. Prof. Dr. Josef Strobl und dem gesamten UNIGIS-Team, insbesondere Karl Atzmanstorfer und Michael Fally, für die hervorragende Betreuung während des gesamten Studiums.

Ein besonderer Dank gebührt auch meiner Lerngruppe, bestehend aus Anne Hebsaker, Christian Jepsen, Gernot Ofner, Birgit Schwabe und Philipp Uebele, die mich während der letzten drei Jahre immer wieder aufs Neue ermunterten und motivierten. Schade, dass wir uns meistens nur per Skype getroffen haben!

Schließlich danke ich meiner Familie und meinen Freunden, insbesondere meiner Freundin Manuela Horlamus, für ihr Verständnis und ihre Unterstützung während der gesamten letzten drei Jahre.

Erklärung

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.

München, 31. Dezember 2010

Hilmar Klink

Kurzfassung

Seit den im Jahr 2005 gestarteten Kartendiensten Google Maps und Bing Maps haben digitale, interaktive Karten den Massenmarkt erobert und sind aus dem Alltag kaum mehr wegzudenken. Durch die Ausstattung von Mobiltelefonen mit GPS-Empfängern wurden Mapping-Applikationen auch für den Mobilfunkmarkt interessant. Schwierig gestaltet sich hierbei die hohe Fragmentierung durch die zahlreichen Plattformen (Android, Blackberry, iOS, Symbian, WebOS, Windows usw.). Um eine App einem möglichst großen Nutzerkreis zugänglich zu machen, muss sie aufwendig auf möglichst viele dieser Plattformen portiert werden.

Eine Lösung dieses Problems versprechen Widgets – installierbare Web-Applikationen, innerhalb derer per JavaScript auf Gerätefunktionen wie z.B. den GPS-Empfänger zugegriffen werden kann. Das World Wide Web Consortium arbeitet derzeit an der Standardisierung von Widgets und Geräteschnittstellen, so dass der Entwicklung von plattformübergreifenden Apps künftig nichts mehr im Wege steht.

Um die Komponenten einer solchen Web Mapping-Applikation möglichst interoperabel und austauschbar zu machen, sollten offene Standards eingesetzt werden. Hierfür existieren zahlreiche Open Source-Komponenten, die diese Standards implementieren und die Entwicklung einer App ohne großen Programmieraufwand ermöglichen. Mit OpenStreetMap und SRTM (Shuttle Radar Topography Mission) stehen zudem freie Geodaten zur Verfügung, die sich mittlerweile nicht mehr hinter den Daten kommerzieller Anbieter verstecken müssen.

Die vorliegende Arbeit zeigt, wie mit freien Geodaten, offenen Standards und Open Source-Komponenten Mapping Apps entwickelt werden können. Hierfür werden mögliche Quellen freier Geodaten hinsichtlich ihrer Eignung untersucht, relevante offene Standards identifiziert, Open Source-Software, die diese Standards implementiert, evaluiert und aus den gewählten Komponenten abschließend eine Konzeption für plattformübergreifende, standortbezogene Mapping Apps entwickelt.

Abstract

Since the beginning of the mapping services Google Maps und Bing Maps in 2005, digital, interactive maps have found their way into the mass market and everyday life is hard to imagine without them. Through the equipment of mobile phones with GPS receivers mapping applications became interesting for the mobile market, too. But the high fragmentation caused by the numerous platforms (Android, Blackberry, iOS, Symbian, WebOS, Windows etc.) is a big handicap for developers. If an app should be available for many users, it has to be ported time-consuming to as many platforms as possible.

Widgets, installable web applications, which provide JavaScript access to device features like the GPS receiver, promise to be a solution for this problem. The World Wide Web Consortium is already working on the standardisation of Widgets and device interfaces, so that the developing of cross-platform apps will be possible in the near future.

To make the componentes of such a web application as interoperable and compatible as possible, open standards should be used. Therefore, numerous open source components exist, which implement these standards and enable the development of an app without noteworthy programming effort. With OpenStreetMap and SRTM (Shuttle Radar Topography Mission) there exist free geodata, that can compete with the data of commercial providers.

This work shows the development of mapping apps with free geodata, open standards and open source components. Therefore, the suitability of possible sources of free geodata is examined, relevant open standards are identified, open source software implementing these standards is evaluated and a conception for cross-platform location based mapping apps is developed from the selected components.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Verzeichnis der Listings

Abkürzungsverzeichnis

ADT	**A**bstract **D**ata **T**ype
AGPL	GNU **A**ffero **G**eneral **P**ublic **L**icense
A-GPS	**A**ssisted **GPS**
AJAX	**A**synchronous **J**avaScript **A**nd **X**ML
API	**A**pplication **P**rogramming **I**nterface
AR	**A**ugmented **R**eality
ASTER	**A**dvanced **S**paceborne **T**hermal **E**mission and **R**eflection Radiometer
BSD	**B**erkeley **S**oftware **D**istribution
C/A	**C**oarse/**A**quisition
CC	**C**reative **C**ommons
CGI	**C**ommon **G**ateway **I**nterface
CGIAR-CSI	**C**onsultive **G**roup on **I**nternational **A**gricultural **R**esearch – **C**onsortium for **S**patial **I**nformation
COO	**C**ell **O**f **O**rigin
CRS	**C**oordinate **R**eference **S**ystem
CSS	**C**ascading **S**tyle **S**heets
DbCL	**D**ata**b**ase **C**ontents **L**icense
DEM	**D**igital **E**levation **M**odel
DOM	**D**ocument **O**bject **M**odel
ECMA	**E**uropean **C**omputer **M**anufacturers **A**ssociation
EPSG	**E**uropean **P**etroleum **S**urvey **G**roup
ESRI	**E**nvironmental **S**ystems **R**esearch **I**nstitute
FE	**F**ilter **E**ncoding
FSFE	**F**ree **S**oftware **F**oundation **E**urope
GDAL	**G**eospatial **D**ata **A**bstraction **L**ibrary
GDEM	**G**lobal **D**igital **E**levation **M**odel
GeoTIFF	**Geo**referenced **T**agged **I**mage **F**ile **F**ormat
GIS	**G**eographisches **I**nformations**S**ystem
GML	**G**eography **M**arkup **L**anguage
GPL	GNU **G**eneral **P**ublic **L**icense

GPS	**G**lobal **P**ositioning **S**ystem
GPX	**G**PS E**X**change **F**ormat
GSM	**G**lobal **S**ystem for **M**obile Communication
GSMA	**GSM A**ssociation
IDE	**I**ntegrated **D**evelopment **E**nvironment
HTML	**H**yper**T**ext **M**arkup **L**anguage
HTTP	**H**yper**T**ext **T**ransfer **P**rotocol
JIL	**J**oint **I**nnovation **L**ab
JME	**J**ava **M**icro **E**dition
JRE	**J**ava **R**untime **E**nvironment
JSON	**J**ava**S**cript **O**bject **N**otation
JVM	**J**ava **V**irtual **M**achine
KML	**K**eyhole **M**arkup **L**anguage
LGPL	**L**esser GNU **G**eneral **P**ublic **L**icense
LiMo	**Li**nux **Mo**bile
LoD	**L**evel **o**f **D**etail
MBR	**M**inimum **B**ounding **R**ectangle
MIME	**M**ultipurpose **I**nternet **M**ail **E**xtensions
MIT	**M**assachusetts **I**nstitute of **T**echnology
MWC	**M**obile **W**orld **C**ongress
NASA	**N**ational **A**eronautics and **S**pace **A**dministration
ODbL	**O**pen **D**ata**b**ase **L**icense
OGC	**O**pen **G**eospatial **C**onsortium
OMTP	**O**pen **M**obile **T**erminal **P**latform
OpenLS	**Open L**ocation **S**ervices
ORS	**O**pen **R**oute **S**ervice
OSGeo	**O**pen **S**ource **Geo**spatial Foundation
OSI	**O**pen **S**ource **I**nitiative
PDA	**P**ersonal **D**igital **A**ssistant
POI	**P**oint **O**f **I**nterest
RAM	**R**andom **A**ccess **M**emory
REST	**RE**presentational **S**tate **T**ransfer
RGB	**R**ot **G**rün **B**lau
RSSI	**R**eceived **S**ignal **S**trength **I**ndication
SA	**S**hare **A**like
SDK	**S**oftware **D**evelopment **K**it

SE	**S**ymbology **E**ncoding
SFS	**S**imple **F**eatures **S**QL
SLD	**S**tyled **L**ayer **D**escriptor
SOS	**S**ensor **O**bservation **S**ervice
SQL	**S**tructured **Q**uery **L**anguage
SVG	**S**calable **V**ector **G**raphics
TA	**T**iming **A**dvance
TDMA	**T**ime **D**ivision **M**ultiple **A**ccess
TIGER	**T**opologically **I**ntegrated **G**eographic **E**ncoding and **R**eferencing
TMC	**T**raffic **M**essage **C**hannel
TMS	**T**ile **M**ap **S**ervice
TTFF	**T**ime **T**o **F**irst **F**ix
UDD	**U**pdate **D**escription **D**ocument
UMTS	**U**niversal **M**obile **T**elecommunications **S**ystem
URI	**U**niform **R**esource **I**dentifier
URL	**U**niform **R**esource **L**ocator
USGS	**U**nited **S**tates **G**eological **S**urvey
VGI	**V**olunteered **G**eographic **I**nformation
VML	**V**ector **M**arkup **L**anguage
W3C	**W**orld **W**ide **W**eb **C**onsortium
W3DS	**W**eb **3D** **S**ervice
WAC	**W**holesale **A**pplications **C**ommunity
WCS	**W**eb **C**overage **S**ervice
WFS	**W**eb **F**eature **S**ervice
WFS-T	**W**eb **F**eature **S**ervice **-** **T**ransactional
WGS 84	**W**orld **G**eodetic **S**ystem 19**84**
WKB	**W**ell-**K**nown **B**inary
WKT	**W**ell-**K**nown **T**ext
WLAN	**W**ireless **L**ocal **A**rea **N**etwork
WMS	**W**eb **M**ap **S**ervice
WMS-C	**W**eb **M**ap **S**ervice **-** **C**ached
WMTS	**W**eb **M**ap **T**ile **S**ervice
WPS	**W**LAN **P**ositioning **S**ystem
WRT	**W**eb **R**un**T**ime
WWW	**W**orld **W**ide **W**eb
XAPI	e**X**tended **A**pplication **P**rogramming **I**nterface

XLS	**X**ML for **L**ocation **S**ervices
XML	e**X**tended **M**arkup **L**anguage
XSLT	e**X**tensible **S**tylesheet **L**anguage **T**ransformation

1 Einführung

1.1 Motivation

Spätestens seit den im Jahr 2005 gestarteten Kartendiensten Google Maps und Microsoft Virtual Earth (mittlerweile Bing Maps) sind interaktive, digitale Karten in aller Munde. Millionen Menschen planen ihre Ausflüge seitdem im Web oder lassen sich von einem Navigationssystem zu ihrem Ziel leiten. Kartenanwendungen sind damit nicht mehr nur GIS-Spezialisten vorbehalten, sondern für die breite Masse interessant.

Durch die Ausstattung von Mobiltelefonen mit GPS (Global Positioning System)-Empfängern zur Positionsbestimmung, erobern digitale Karten mittlerweile auch den Mobilfunkmarkt. Schwierig gestaltet sich dabei die hohe Fragmentierung durch die zahlreichen unterschiedlichen auf dem Markt befindlichen Betriebssysteme (Android, Blackberry, iOS, Symbian, Windows Mobile usw.). Um eine App[1] einem breiten Nutzerkreis zugänglich zu machen, müssen Entwickler die Software aufwändig für möglichst viele Plattformen portieren. Obwohl der Trend hin zu offenen Betriebssystemen (Android, Symbian, LiMo, MeeGo) geht und die Marktforscher von Gartner davon ausgehen, dass im Jahr 2015 nur noch drei Betriebssysteme eine Rolle spielen [Graf 2009], nimmt die Vielfalt derzeit sogar noch zu. Jüngstes Beispiel hierfür ist bada von Samsung. Erste Geräte mit diesem Betriebssystem kamen erst im Laufe des Jahres 2010 auf den Markt.

Abhilfe gegen diese Fragmentierung versprach lange Zeit die JME (Java Micro Edition), eine virtuelle Maschine für Handys und PDAs (Personal Digital Assistants), die es ermöglicht eine Applikation zu entwickeln, die auf mehreren Plattformen lauffähig ist. Leider gibt es nach wie vor einige Plattformen, die die JME nicht unterstützen. Dazu gehören mit dem linuxbasierten Android von Google, dessen virtuelle Maschine Dalvik im Gegensatz zur JVM (Java Virtual Machine) registerbasiert ist, und Apples iOS Plattformen, denen für die Zukunft ein hoher Marktanteil prognostiziert wird [Graf 2009]. Für Windows Mobile existieren zwar Laufzeitumgebungen; diese sind jedoch oftmals nicht vorinstalliert oder nur teilweise kompatibel.

[1] Bei einer App handelt es sich im allgemeinen Sprachgebrauch um eine auf einem Smartphone installierbare Anwendung.

Eine Lösung dieses Problems versprechen nun die sogenannten Widgets, kleine installierbare Web-Applikationen, die in einer Widget Engine ausgeführt werden, die JavaScript-Schnittstellen für den Zugriff auf verschiedene Telefonfunktionen (z.B. GPS-Empfänger, Adressbuch, Kamera, Kalender usw.) bietet. Das W3C (World Wide Web Consortium) arbeitet derzeit an einer ganzen Reihe von Spezifikationen in diesem Bereich. Bis diese verabschiedet und von den verschiedenen Herstellern implementiert sind wird zwar noch einige Zeit vergehen, dennoch versprechen Widgets das lang ersehnte Ende der Fragmentierung im Bereich der mobilen Applikationen.

Doch keine Mapping App ohne die nötigen Geodaten. Mit dem Google Maps API (Application Programming Interface) bzw. dem Bing Maps API ist es zwar möglich binnen kürzester Zeit eine eigene Web-Applikation mit kostenlosem Kartenmaterial, Routing und Geocoding zu erstellen, allerdings bekommt man hier keinen direkten Zugriff auf die Geodaten, sondern kann diese nur über nutzungsbeschränkte Dienste einbinden. Doch mit OpenStreetMap gibt es eine freie Alternative, die es mittlerweile in vielerlei Hinsicht bereits mit der proprietären Konkurrenz aufnehmen kann.

Um die verschiedene Komponenten eines verteilten Mapping-Systems interoperabel und austauschbar zu machen, sollten die Schnittstellen offenen Standards entsprechen. Für deren Spezifikation zeichnet sich im Geoinformatik-Bereich in erster Linie das OGC (Open Geospatial Consortium) verantwortlich, das eine ganze Reihe von für das Web Mapping relevanten Standards wie beispielsweise den vielfach eingesetzten WMS (Web Map Service) veröffentlicht hat.

Viele dieser offenen Standards werden mittlerweile von diversen Software-Produkten unterstützt. Während Software im GIS-Bereich früher fest in proprietärer Hand wie beispielsweise ESRI (Environmental Systems Research Institute) oder GeoMedia war, gibt es mittlerweile eine große Open Source-Community, die leistungsfähige, quelloffenen Software implementiert.

Damit sollten mittlerweile alle Voraussetzungen erfüllt sein, um plattformübergreifende, positionsbezogene Mapping Apps mit freien Geodaten, offenen Standards und quelloffener Software realisieren zu können.

1.2 Ziele

Die vorliegende Arbeit soll zeigen, ob und wie mit Hilfe freier Geodaten und quelloffener Software positionssensitive Mapping-Applikationen entwickelt werden können, die aufgrund der konsequenten Verwendung offener Standards auf zahlreichen Plattformen lauffähig sind und deren Komponenten sich durch eine hohe Interoperabilität auszeichnen.

Hierfür müssen die folgenden Teilziele erreicht werden:

- **Analyse des aktuellen Forschungsstandes**
  Dieser Komplex beinhaltet eine gründliche Literaturrecherche, eine detaillierte Untersuchung vorhandener Mapping-Applikationen bezüglich ihrer implementierten Funktionalitäten und deren technischer Umsetzung sowie eine genaue Betrachtung der derzeit verfügbaren Technologien zur Positionsbestimmung eines Mobiltelefons.
- **Auswahl von geeigneten freien Geodaten**
  Zum Erreichen dieses Teilzieles müssen die verschiedenen freien Datenquellen in Bezug auf ihre Lizenzbestimmungen, Formate, Genauigkeit usw. untersucht und miteinander verglichen werden.
- **Identifizierung relevanter offener Standards**
  Hier sollten neben den Widget-Standards des W3C auch verfügbare Spezifikationen für den Abruf und das Styling von Karten, die Abfrage und die Editierung von Geodaten, die Anforderung von Routen und Navigationsinformationen, die Geokodierung von geographischen Namen und Adressen sowie die Ermittlung der eigenen Position hinsichtlich ihrer Eignung analysiert werden.
- **Vergleich von Open Source-Komponenten, die diese Standards implementieren**
  Aufbauend auf dem vorherigen Teilziel gilt es nun die wichtigsten quelloffenen Implementierungen der relevanten Standards zu ermitteln und bezüglich ihrer Features miteinander zu vergleichen.
- **Konzeption einer Architektur sowie der einzelnen Komponenten**
  Hier werden die in den vorherigen Schritten ausgewählten Ressourcen und Technologien zu einer Anwendung kombiniert. Besonderes Augenmerk sollte dabei auf der Architektur und den Schnittstellen des Systems liegen.

1.3 Abgrenzung und beabsichtigtes Publikum

Die Implementierung einer konkreten Mapping App oder eines Prototyps ist nicht das Ziel dieser Arbeit. Viel mehr geht es darum, einen Überblick über freie Geodaten, offene Standards und Open Source-Software zu geben, deren Eignung zu untersuchen und Einsatzmöglichkeiten aufzuzeigen. Dies bedeutet jedoch nicht, dass das Zusammenspiel der in der Konzeption vorgeschlagenen Komponenten nicht in der Praxis überprüft wurde. Dennoch ist die resultierende Konzeption nicht als die ultimative Vorlage für konkrete Applikationen zu sehen, sondern dient als Anhaltspunkt und Ideengeber für potenzielle Entwickler solcher Anwendungen. Das beabsichtige Publikum umfasst neben Entwicklern auch Personen, die ein allgemeines Interesse an freien Geodaten, offenen Standards und Open Source-Software im Bereich des Web Mapping haben.

1.4 Gliederung

Nach dieser kurzen Einführung in die Thematik (Kapitel 1) folgt im Kapitel 2 eine Analyse des aktuellen Forschungsstandes in Form eines Literaturüberblicks (Kapitel 2.1), einer Betrachtung bereits bestehender, innovativer Mapping-Applikationen (Kapitel 2.2) sowie einer Vorstellung der für Mobiltelefone verfügbaren Positionierungstechnologien (Kapitel 2.3).

Kapitel 3 widmet sich den verfügbaren freien Geodaten. Hierzu gehören neben dem OpenStreetMap-Projekt (Kapitel 3.1) und der GeoNames-Datenbank (Kapitel 3.2) auch zwei Datenquellen für flächendeckende Höhenwerte (Kapitel 3.3).

Das Kapitel 4 steht ganz im Zeichen offener Standards. Neben den relevanten Standards des W3C (Kapitel 4.1) werden vor allem Spezifikationen des OGC (Kapitel 4.2) beleuchtet. Obwohl die OSGeo (Open Source Geospatial Foundation) sich nicht als Standardisierungsgremium sieht, hat sie einige wichtige Empfehlungen (Quasistandards) geprägt (Kapitel 4.3), die abschließend betrachtet werden.

In Kapitel 5 folgt eine Evaluation der relevanten quelloffenen Software-Komponenten. Kapitel 5.1 klärt hier zunächst die rechtlichen Aspekte der verschiedenen Lizenzen, bevor die wichtigsten räumlichen Datenbanken (Kapitel 5.2), Web Mapping Server (Kapitel 5.3), Tile Cache Server (Kapitel 5.4), Web Mapping Clients (Kapitel 5.5), Mobile Web Frameworks (Kapitel 5.6), Routing Server (Kapitel 5.7) und Geocoder (Kapitel 5.8) betrachtet werden. Den

Abschluss bildet ein kurzer Überblick über kleine, aber wichtige Hilfswerkzeuge zur Konvertierung zwischen den benötigten Formaten (Kapitel 5.9).

Nach diesen wichtigen Bausteinen kann im Kapitel 6 schließlich eine Konzeption für plattformübergreifende Mapping Apps entworfen werden. Nach der Erläuterung der Architektur (Kapitel 6.1) werden die einzelnen Schichten dieser Architektur genauer beleuchtet: Von der Datenhaltungsschicht (Kapitel 6.2) über die Logikschicht (Kapitel 6.3) bis hin zur Präsentationsschicht (Kapitel 6.4).

Den Abschluss der Arbeit bildet schließlich Kapitel 7 mit einer Zusammenfassung der Ergebnisse (Kapitel 7.1) sowie deren Bewertung und einem kurzen Ausblick (Kapitel 7.2).

2 Stand der Technik

2.1 Literaturüberblick

Einen umfassenden Überblick über Technologien im Bereich internetgestützter GIS (Geoinformationssysteme) liefern [Korduan/Zehner 2008], während [Blankenbach 2007] sich speziell auf die Architektur und Umsetzung von Location Based Services unter Berücksichtigung von Interoperabilität fokussiert.

Interoperabilität wird durch den Einsatz von offenen Standards erreicht, deren Spezifikationen wichtige Quellen der vorliegenden Arbeit darstellen, aber an dieser Stelle nicht explizit genannt werden. Speziell mit den Widgets-Standards setzen sich [Kriesing 2009] und [Sachse 2010] auseinander, wobei letzterer auch die neuesten Entwicklungen der Industrie berücksichtigt und Alternativen vorstellt, um der Fragmentierung Herr zu werden. Hier setzen auch [Spiering/Haiges 2010] an, die die praktische Anwendung von Mobile Web Frameworks und HTML5-Schnittstellen wie dem Geolocation API zur Positionsbestimmung erläutern. Mit den verschiedenen Möglichkeiten zur Ermittlung der Position befassen sich [Ingensand/Bitzi 2001] und [Köhne/Wössner 2007].

Die wichtigste Quelle für freie Geodaten stellt sicherlich OpenStreetMap dar, für das [Ramm/Topf 2009] das deutsche Standardwerk liefern. Da sich das Projekt jedoch schnell weiterentwickelt, sind die aktuellsten Informationen nur im [OpenStreetMap-Wiki 2010] zu finden. Die Daten werden bei OpenStreetMap auf freiwilliger Basis durch die Community erfasst. Mit den Risiken und Potenzialen dieses Phänomens befasst sich [Goodchild 2007], während [Zielstra/Zipf 2010] die Qualität der auf diese Weise erfassten Daten durch einen Vergleich mit proprietären Geodaten ermitteln. [Jacobsen 2004] beschäftigt sich hingegen mit der Generierung und Validierung von Höhenmodellen aus freien Weltrauminformationen.

Im Bereich der Open Source-Komponenten bilden [Mitchell 2008] mit dem Fokus auf Serverseite und [Jansen/Adams 2010] mit Client-Schwerpunkt wertvolle Quellen, wohingegen [Grassmuck 2004] und [Fraunhofer-Gesellschaft 2006] rechtliche Aspekte der verschiedenen Lizenzen klären.

2.2 Innovative Mapping-Applikationen

2.2.1 OpenRouteService

Bei dem OpenRouteService[2] (ORS), der im Rahmen der Masterarbeit von Pascal Neis [NEIS 2008] entstand, handelt es sich um einen webbasierten Routenplaner auf OpenStreetMap-Basis, bei dem besonderes Augenmerk auf die Implementierung offener Standards gelegt wurde. Er berechnet nicht nur die kürzeste und schnellste Strecke für Autos, sondern auch den kürzesten Weg für Fahrräder und Fußgänger sowie besonders sichere oder für Mountainbikes und Rennräder geeignete Routen. Daneben unterstützt der ORS auch Echtzeit-Verkehrsdaten (TMC) sowie die Umfahrung von benutzerdefinierten Bereichen (so genannte Avoid Areas), die der Nutzer in der Karte markieren kann. Neben dem Routing unterstützt der ORS auch die Geokodierung von Adressen und die Suche von POIs (Points Of Interest) – nach Namen oder in einem definierten Umkreis um einen markierten Punkt oder die aktuelle Route. Ein besonders interessantes Feature ist die Erreichbarkeitsanalyse, die berechnet wie weit man sich in einem bestimmten Zeitfenster von einem Ausgangspunkt wegbewegen kann. Dies ist zum Beispiel interessant, um herauszufinden, welche Orte innerhalb einer halben Stunde von einer spezifischen Adresse erreichbar sind (siehe Abbildung 1).

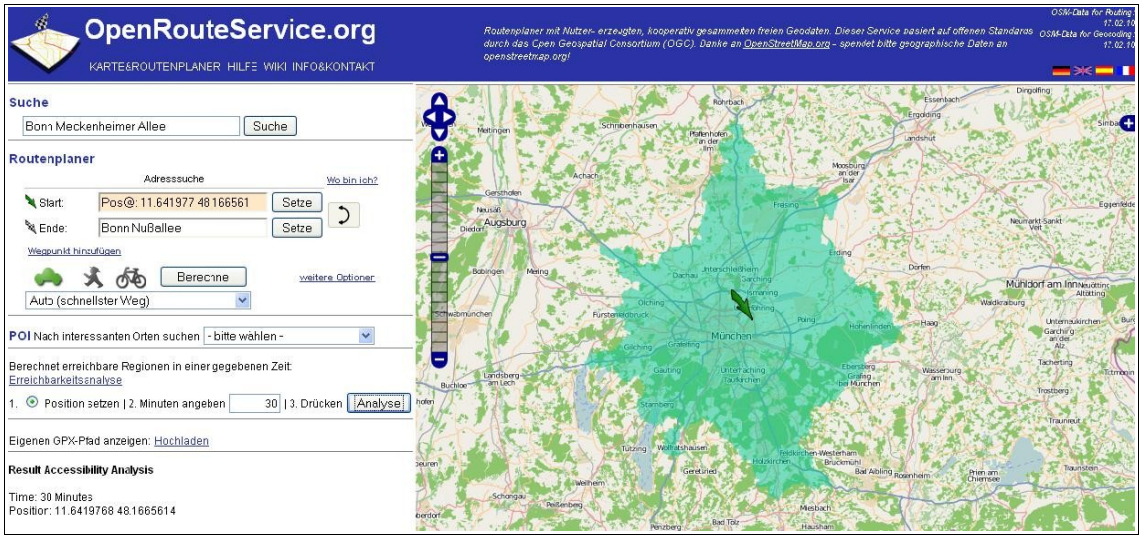

Abbildung 1: Erreichbarkeitsanalyse mit dem OpenRouteService
Quelle: http://www.openrouteservice.org/

[2] http://www.openrouteservice.org/

Der ORS setzt sich aus einer Vielzahl von Komponenten zusammen (siehe Abbildung 2): Die OSM- und TMC-Daten werden in zwei PostGIS-Datenbanken (siehe Kapitel 5.2.2) gehalten und sind über zwei GeoServer-Instanzen (siehe Kapitel 5.3.2) als WMS (siehe Kapitel 4.2.1) bzw. WFS (Web Feature Service) (siehe Kapitel 4.2.4) verfügbar. Aus Performance-Gründen greift der OpenLayers-Viewer (siehe Kapitel 5.5.1) jedoch nicht direkt auf den OSM-WMS zu, sondern über einen TileCache (siehe Kapitel 5.4.1), der die gerenderten Kacheln zwischenspeichert. Neben der selbst gerenderten Kartenansicht, werden auch noch die drei OSM-Renderer Mapnik, Osmarender und CycleMap (siehe Kapitel 3.1.6) eingebunden. Den Kern des ORS bildet die Implementierung der OpenLS (Open Location Services) Route Service, Location Utility Service, Directory Service und Presentation Service (siehe Kapitel 4.2.5). Die „mit * gekennzeichneten Dienste des OpenLS-Frameworks zählen nicht zu den Core Services bauen aber durch deren Implementierung auf dem Framework auf“ [Neis 2008].

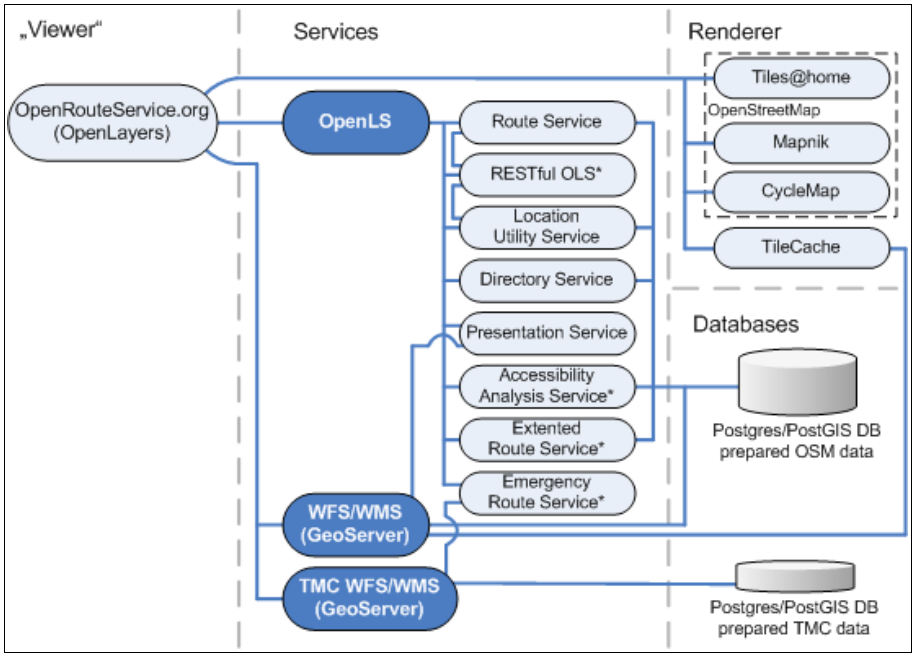

Abbildung 2: Komponenten des OpenRouteService
Quelle: http://wiki.openstreetmap.org/wiki/DE:OpenRouteService

Für den OSM-WMS gibt es mittlerweile die separate Webseite „OSM WMS Europe“[3], die auch eine Schummerungsebene basierend auf Höhendaten der Shuttle Radar Topography Mission (SRTM) (siehe Kapitel 3.3.1) umfasst, die die Karte wesentlich plastischer erscheinen lässt.

Die quelloffene Navigationssoftware AndNav2[4] bringt die Dienste des ORS auf Android-Handys. Um auch offline navigieren zu können, besteht die Möglichkeit, die Route vor Fahrtantritt berechnen zu lassen und das benötigte Kartenmaterial in Form eines so genannten MapTilePacks auf der Speicherkarte des Mobiltelefons zu installieren.

2.2.2 OpenStreetMap 3D

Das Projekt OpenStreetMap 3D[5] der Universität Heidelberg macht aus OpenStreetMap- und SRTM-Daten einen 3D-Szenengraph und liefert diesen in Form des OGC-Entwurfs Web 3D Service (W3DS) an den Client aus. Um ein realistisches Stadtbild zu erhalten, werden auch Gebäude, deren Grundrisse in OpenStreetMap erfasst sind, in 3D modelliert. Da in OpenStreetMap bisher kaum die Höhe oder die Anzahl der Stockwerke eines Gebäudes erfasst sind, wird die Höhe vielfach noch per Zufallsgenerator vergeben.

Für die Routenberechnung und die Suche nach POIs nutzt auch dieses Projekt die OpenLS-Services des ORS (siehe Kapitel 2.2.1). Daneben unterstützt OpenStreetMap 3D auch die Integration von Echtzeit-Sensordaten wie Flusspegelstände oder Luftschadstoffdaten per SOS (Sensor Observation Service), einem weiteren OGC-Standard. Für die Visualisierung wurde mit dem XNavigator ein eigener Java-Viewer entwickelt, der eine freie clientseitige Navigation ermöglicht. Damit die Navigation flüssig läuft, benötigt man allerdings einen leistungsstarken Computer (Dual Core Prozessor, 2 GB RAM, separate Grafikkarte mit 256 MB RAM, Internetverbindung mit 4000 kbit/s). Derzeit liegt ein erster Prototyp für ganz Deutschland vor. Abbildung 3 zeigt die Altstadt von Heidelberg in OpenStreetMap 3D.

[3] http://www.osm-wms.de/
[4] http://www.andnav.org/
[5] http://www.osm-3d.org/

Abbildung 3: Heidelberg in OpenStreetMap 3D
Quelle: http://www.osm-3d.org/

2.2.3 Wikitude

Wikitude[6] nutzt neben Kartenmaterial auch die Kamera von Mobiltelefonen und blendet in deren Live-Bild POIs ein. Diese Erweiterung der Realität wird als Augmented Reality (AR) bezeichnet. Ein Großteil der verfügbaren POI-Daten stammt von georeferenzierten Artikeln der Online-Enzyklopädie Wikipedia[7]. Damit die POIs lagerichtig eingezeichnet werden können, muss neben der Position auch die Orientierung des Handys bekannt sein. Deren Bestimmung ist in aktuellen Smartphones über integrierte Orientierungssensoren und Kompasse möglich. Abbildung 4 zeigt New York in der AR-Ansicht von Wikitude. Mittlerweile ist mit Wikitude Drive auch ein Navigationssystem auf AR-Basis verfügbar.

6 http://www.wikitude.org
7 http://www.wikipedia.org

Abbildung 4: New York in der AR-Ansicht von Wikitude
Quelle: http://www.mobilizy.com/de/category/02_wikitude/world-browser

2.2.4 Google Maps

Auch Google Maps[8] als eines der populärsten Beispiele im Bereich des Web Mapping entwickelt sich stetig weiter. Für immer mehr Regionen stehen mittlerweile die 360-Grad-Panoramabilder von Street View[9] zur Verfügung, die es ermöglichen sich virtuell durch Straßen zu bewegen. Abbildung 5 zeigt das Kolosseum in Rom in Google Street View.

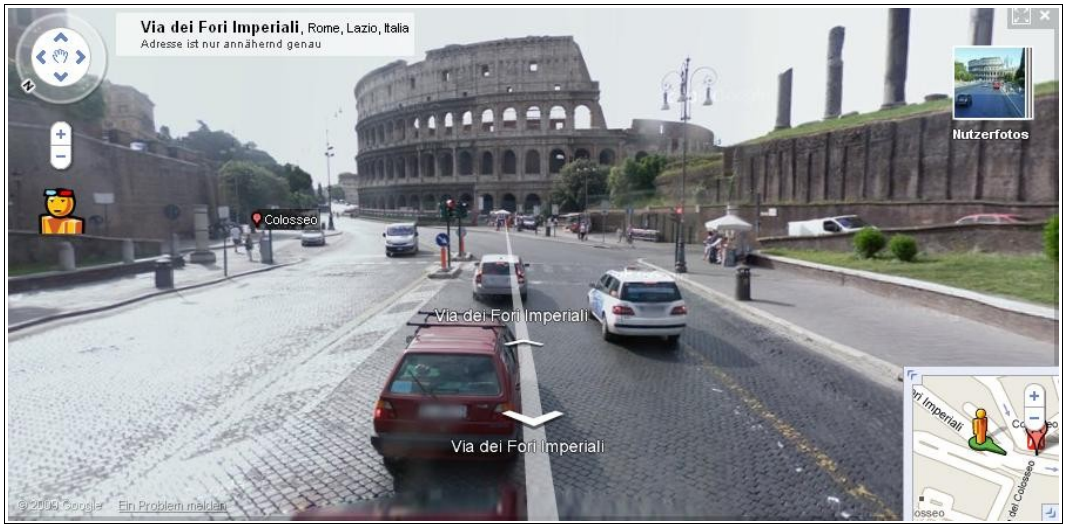

Abbildung 5: Das Kolosseum in Google Street View
Quelle: http://maps.google.de/

[8] http://maps.google.de
[9] http://maps.google.de/help/maps/streetview/

In Deutschland war dieses Feature aufgrund von Kontroversen um den Datenschutz lange Zeit nicht freigeschaltet; mittlerweile sind die größten Städte verfügbar. Seit April 2010 gibt es in Street View auch einen Farbanaglyphen-Modus, wodurch die Bilder mit Hilfe einer passenden Brille dreidimensional werden.

Ein interessantes Feature ist auch die Darstellung der Verkehrssituation. Wahlweise kann die aktuelle Verkehrslage oder die zu bestimmten Zeitpunkten übliche Verkehrslage angezeigt werden.

Mit Latitude[10] besteht die Möglichkeit, sich den derzeitigen Aufenthaltsort seiner Freunde auf der Karte anzeigen zu lassen. Bedingung hierfür ist allerdings, dass die Freunde die Übermittlung ihrer aktuellen Position an Google aktiviert haben. Auf Wunsch kann man sich mittels Standort-Alerts auch benachrichtigen lassen, wenn sich ein Freund in der Nähe befindet. Damit diese Benachrichtigung aber nicht an Orten erfolgt, an denen man sich ohnehin häufig zu bestimmten Zeiten trifft (z.B. an der Arbeitsstätte), müssen die üblichen Aufenthaltsorte bekannt sein. Hierfür muss die Funktion Standortverlauf aktiviert werden, die datenschutzrechtlich jedoch äußerst fragwürdig ist.

2.2.5 Bing Maps

Bing Maps[11], das Pendant von Microsoft, bietet ebenfalls eine 3D-Ansicht. Für ausgewählte Städte in den Vereinigten Staaten enthält diese sogar detaillierte, texturierte Modelle der Gebäude und Bäume. Abbildung 6 zeigt einen Ausschnitt von Las Vegas in 3D. Für diese Ansicht muss allerdings zunächst ein Browser-Plugin installiert werden und auch die Anforderungen an Hardware und Internetverbindung sind hoch.

Optisch ansprechend aber hardwareschonend ist die Vogelperspektive, bei der die Luftbilder aus einem schrägen Winkel aufgenommen wurden. Dadurch wirkt die Darstellung wesentlich plastischer. Leider sind die Schrägluftbilder derzeit noch auf größere Städte begrenzt.

[10] http://www.google.com/intl/de_de/latitude/intro.html
[11] http://www.bing.com/maps

Abbildung 6: Las Vegas in der 3D-Ansicht von Bing Maps
Quelle: http://www.bing.com/maps/

2.3 Positionierungstechnologien

Um eine Anwendung standortbezogen zu machen, muss die Position des Mobiltelefons automatisiert möglichst genau bestimmt und laufend aktualisiert werden. Hierfür stehen verschiedene Technologien zur Verfügung. Die wichtigsten dieser Technologien werden im Folgenden genauer untersucht:

2.3.1 GPS

Das vom amerikanischen Verteidigungsministerium initiierte GPS (Global Positioning System) ist ein globales Navigationssatellitensystem zur Positionsbestimmung. Es besteht aus mindestens 24 Satelliten, die sich auf sechs Umlaufbahnen in einer Höhe von 20200 km bewegen. Auf jedem Punkt der Erde sind damit zu jeder Zeit mindestens vier Satelliten sichtbar, die permanent ihre aktuelle Position und die genaue Uhrzeit senden. Um die eigene Position ermitteln zu können, berechnet ein GPS-Empfänger die Differenz zwischen der aktuellen Uhrzeit und der vom jeweiligen Satelliten gesendeten Uhrzeit und erhält auf diese Weise die Laufzeit des Signals, aus der die Entfernung zu dem Satelliten bestimmt werden kann.

Theoretisch reichen drei Satelliten aus, um die genaue dreidimensionale Position mittels Trilateration zu bestimmen. Da GPS-Empfänger im Gegensatz zu den Satelliten jedoch nicht mit Atomuhren ausgestattet sind, benötigt man noch die Daten eines vierten Satelliten, mit dessen Hilfe der Zeitfehler ermittelt und korrigiert werden kann. Auf diese Weise ist laut [Köhne/Wössner 2007] eine auf etwa fünfzehn Meter genaue Positionsbestimmung möglich.

GPS hat jedoch den Nachteil, dass innerhalb von Gebäuden, in tiefen Häuserschluchten oder in dichten Wäldern die Signale zu stark gedämpft werden, so dass „mit herkömmlicher Empfangs- und Auswertetechnik keine Positionsbestimmung möglich ist“ [Blankenbach 2007]. Hinzu kommt, dass GPS-Empfänger nach dem Einschalten einige Minuten benötigen, bis sie zum ersten Mal die Position ermittelt haben.

2.3.2 GSM/UMTS

Mit dem Cell Of Origin-Verfahren (COO) kann in Mobilfunknetzen, wie GSM (Global System for Mobile Communications) und UMTS (Universal Mobile Telecommunications System), auf einfache Art und Weise die grobe Position eines Handys bestimmt werden, da ein Mobiltelefon immer die ID der Funkzelle kennt, in die es gerade eingebucht ist. Anhand dieser so genannten Cell-ID, können die Koordinaten der entsprechenden Antenne ermittelt werden. Die Genauigkeit der Position hängt somit „stark von der Zellgröße und damit von der Antennendichte ab“ [Blankenbach 2007]. Laut [Ingensand/Bitzi 2001] bewegt sich der Radius von GSM-Zellen zwischen 100 m in Innenstädten und 35 km in unbewohnten Gebieten. In UMTS-Netzen ist die Genauigkeit bedingt durch die kleinere Zellgröße etwas höher.

In GSM-Netzen lässt sich die Genauigkeit durch Berücksichtigung des Timing Advance-Wertes (TA) noch steigern. Da GSM mit TDMA (Time Division Mulitple Access) – also Zeitmultiplex – arbeitet, „ist es notwendig, die Daten vom Mobiltelefon aus etwas früher abzuschicken, damit sie beim Öffnen des Zeitfensters auch bei der Basisantenne angelangt sind“ [Ingensand/Bitzi 2001]. Diesen Sendevorlauf teilt die Basisstation dem Mobiltelefon als Vielfaches der Bitdauer (3,7 µs) in Form des Parameters TA mit, der je nach Entfernung Werte zwischen 0 und 63 annehmen kann. Da das Licht in 3,7 µs etwa 1,1 km zurücklegt und die Signallaufzeit zwischen Basisstation und Mobiltelefon und zurück kompensiert werden muss, erhöht sich der TA-Wert je 550 m Abstand um eins. Durch Berücksichtigung dieses Parameters erhöht sich die Ortungsgenauigkeit bzgl. des Abstandes zur Antenne somit auf 550 m. Da die Signallaufzeit „durch Reflexion, Beugung und Mehrwegeeffekte jedoch stark beeinflusst ist, wird die theoretische Genauigkeit des Verfahrens insbesondere im urbanen Bereich in der Praxis jedoch deutlich herabgesetzt“ [Blankenbach 2007].

Eine weitere Präzisierung der Position erhält man durch sektorisierte Antennen, d.h. ein Mobilfunkmast besteht aus mehreren Antennen, die unterschiedliche Winkelbereiche abdecken. Reflexionen können jedoch dazuführen, dass sich ein Mobiltelefon gar nicht in dem vermuteten Sektor befindet.

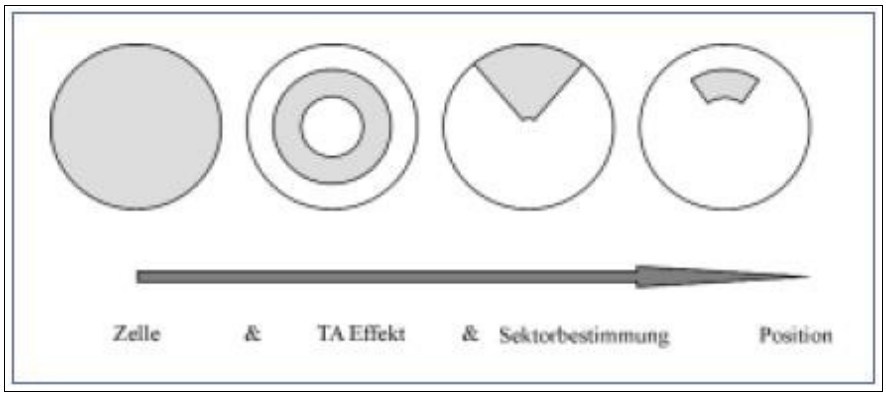

Abbildung 7: Ortung in Mobilfunknetzen
Quelle: [Ingensand/Bitzi 2001]

Abbildung 7 zeigt wie die Kombination aus Cell-ID, TA und sektorisierten Antennen die Position des Mobiltelefons auf einen Ringsektor innerhalb einer Zelle einschränkt.

2.3.3 WPS

Ein WPS (WLAN Position System) bestimmt die Position anhand der Empfangsfeldstärken (RSSI) der in Reichweite befindlichen WLANs (Wireless Local Area Networks). Voraussetzung hierfür ist eine vorherige Messung der Signalstärken an Referenzpunkten. Anhand dieser Fingerprints können WLAN-fähige Endgeräte dann autark ihre Position ermitteln. Abhängig von der Umgebung ist mit dieser Methode eine Genauigkeit von wenigen Metern erreichbar.

Diese Positionierungsmethode ist nur in urbanen Bereichen geeignet, da nur hier eine ausreichende Dichte an WLAN Access Points herrscht, kann dort aber auch hervorragend zur Indoor-Ortung eingesetzt werden. Das Erfassen der Fingerprints für ein großes Gebiet ist allerdings äußerst aufwendig. Die amerikanische Firma Skyhook Wireless[12] hat mittlerweile jedoch eine sehr große Datenbank aufgebaut, die die Metropolregionen Nordamerikas sowie West- und Mitteleuropas umfasst. Das WPS von Skyhook Wireless kommt bei der Lokalisierung im iPhone zum Einsatz. Auch das Fraunhofer-Institut für Integrierte Schaltungen[13] hat ein WPS entwickelt, dass derzeit in mehreren deutschen Großstädten (u.a. Berlin, Hamburg, München und Nürnberg) intensiv getestet wird.

[12] http://www.skyhookwireless.com/
[13] http://www.iis.fraunhofer.de

2.3.4 A-GPS

Wie bereits erwähnt hat GPS trotz seiner relativ hohen Genauigkeit zwei große Nachteile: Die mangelnde Verfügbarkeit innerhalb von Gebäuden, tiefen Häuserschluchten und dichter Vegetation und die große Zeitspanne vom Einschalten des Empfängers bis zur ersten Positionsbestimmung, die so genannte TTFF (Time To First Fix).

Neben dem Course/Aquisition-Code (C/A-Code), der dem GPS-Empfänger zur Laufzeitmessung dient, werden die Navigationsdaten auf das Trägersignal moduliert. Diese umfassen Uhrenkorrekturen, die eigenen hochgenauen Bahndaten für die nächsten Stunden (Ephemeriden) sowie die gröberen Bahndaten aller GPS-Satelliten für die nächsten Tage (Almanach). Während sich der C/A-Code nach einer Millisekunde wiederholt, benötigt die vollständige Übertragung der Navigationsnachricht 12,5 Minuten, wobei die Ephemeriden mehrfach enthalten sind und somit alle 30 Sekunden empfangen werden können. [KÖHNE/WÖSSNER 2007]

Die TTFF hängt nun davon ab, ob die vom GPS-Empfänger gespeicherten Ephemeriden- und Almanach-Daten noch aktuell sind und der Empfänger seit dem letzten Fix mehrere 100 Kilometer bewegt wurde. Kennt der GPS-Empfänger seine ungefähre Position und Ephemeriden und Almanach sind noch aktuell, spricht man von einem Heißstart und die TTFF beträgt etwa 15 Sekunden. Sind die Ephemeriden hingegen veraltet, handelt es sich um einen Warmstart. Durch die benötigte Aktualisierung der Ephemeriden verlängert sich die TTFF auf etwa 45 Sekunden. Wenn auch der Almanach nicht mehr aktuell ist, weiß der GPS-Empfänger nicht mehr welche Satelliten sich in seinem Sichtfeld befinden und man spricht von einem Kaltstart. In diesem Fall steigt die TTFF entsprechend auf über zehn Minuten. [KÖHNE/WÖSSNER 2007]

Mit dem Hybridansatz A-GPS (Assisted GPS) kann man die TTFF jedoch stark verkürzen, indem dem GPS-Empfänger die Ephemeriden, der Almanach, die aktuelle GPS-Zeit oder auch die per COO (siehe Kapitel 2.3.2) grob ermittelte Position über das Internet oder das Mobilfunknetz zur Verfügung gestellt werden. Auf diese Weise kann die TTFF „auf bis zu 1 Sekunde“ [KÖHNE/WÖSSNER 2007] verkürzt werden. Neben der schnelleren Positionsbestimmung gelingt es dank der Assistenzinformationen auch „stark abgeschwächte Signale (bspw. nach Durchdringung einer Betonwand) auszuwerten und selbst Signalfetzen zur Positionierung nutzen zu können“

[BLANKENBACH 2007]. Somit kann die Position mit A-GPS auch an Orten ermittelt werden, die für reine GPS-Positionierung nicht in Frage kommen. A-GPS wird mittlerweile von vielen aktuellen Mobiltelefonen unterstützt.

3 Freie Geodaten

Die eingangs erwähnten Kartendienste (Google Maps, Bing Maps usw.) bieten zwar kostenlose APIs an, um eigene Karten auf Basis ihrer Geodaten, so genannte Mashups, zu erstellen. Die Möglichkeiten solcher Mashups beschränken sich jedoch im Allgemeinen auf die Überlagerung der vorgefertigten Karten mit eigenen Markern (siehe Abbildung 8), Polylinien und Polygonen. Der Nutzer bekommt keinen direkten Zugriff auf die Geodaten, sondern lediglich auf die daraus gerenderten Karten. Es ist demnach unmöglich das Kartenbild des Hintergrunds nach eigenen Vorlieben zu modifizieren. Eine weitere Einschränkung stellt die Abhängigkeit vom Anbieter dar, der das API jederzeit verändern, kostenpflichtig machen oder gar einstellen kann. [Ramm/Topf 2009]

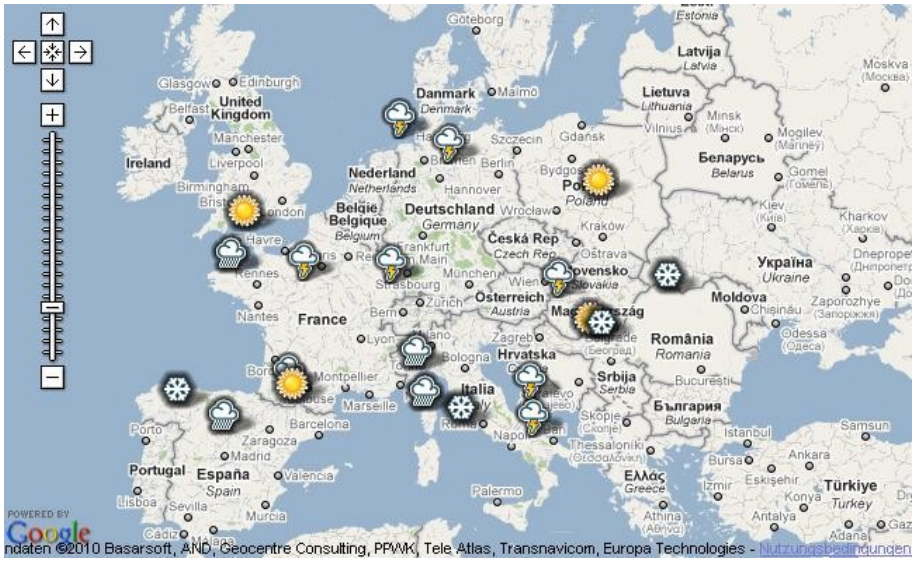

Abbildung 8: Beispiel eines Google Maps Mashups
Quelle: http://code.google.com/intl/de-DE/apis/maps/documentation/examples/

Um diesen Limitierungen zu entgehen, benötigt man freie Geodaten, die von jedem bezogen und beliebig verwendet werden können, um daraus beispielsweise eine Statistik zu erstellen oder eine selbst gestaltete Karte zu rendern.

3.1 OpenStreetMap

Das OpenStreetMap-Projekt wurde 2004 von Steve Coast gegründet mit dem Ziel „freie geographische Daten über Straßen, Eisenbahnen, Flüsse, Wälder, Häuser und alles andere, was gemeinhin auf Karten zu sehen ist, zu erfassen“ [OpenStreetMap-Wiki 2010]. Die Erfassung der Geodaten erfolgt im Allgemeinen mittels GPS-Aufzeichnung durch die Mitglieder der Community, an der sich jeder beteiligen kann. Diese Form der Geoinformation bezeichnet Goodchild als Volunteered Geographic Information (VGI) und bescheinigt ihr ein hohes Potenzial [Goodchild 2007]. Teilweise wurden dem Projekt jedoch auch bereits vorhandene Daten zur Verfügung gestellt, wie beispielsweise Luftbilder von Yahoo oder der TIGER (Topographically Integrated Geographic Encoding and Referencing)-Datensatz des United States Census Bureau.

3.1.1 Lizenz

Die OpenStreetMap-Daten stehen unter der Lizenz CC-BY-SA 2.0 (Creative Commons Attribution-Share Alike Generic 2.0)[14]. Diese Lizenz erlaubt die Daten beliebig zu vervielfältigen, zu verbreiten und zu bearbeiten. Zu beachten ist hierbei allerdings, dass der Name des Rechteinhabers zu nennen ist und Werke, die die Daten verwenden, ebenfalls unter diese Lizenz zu stellen sind. [Creative Commons 2004]

Problematisch ist die Tatsache, dass theoretisch alle Nutzer, die Daten beigesteuert haben und das sind bereits über 200.000 (Stand: März 2010)[15], Rechteinhaber sind und damit genannt werden müssten. Es reicht jedoch OpenStreetMap als Rechteinhaber zu nennen. Abgeleitete Werke können gemäß der CC-BY-SA wiederum beliebig weiter vervielfältigt, verbreitet und modifiziert werden.

In Zukunft soll diese Lizenz allerdings von der speziell für Daten entwickelten ODbL (Open Database License) abgelöst werden, da diese im Gegensatz zur CC-BY-SA nicht nur auf dem Urheberrecht sondern auch auf dem europäischen Datenbankrecht aufbaut. „Dadurch soll erreicht werden, dass die Lizenz auch dann den Zugriff auf OpenStreetMap regelt, wenn die einzelnen in der Datenbank enthaltenen Fakten nicht schutzwürdig sind“ [Ramm/Topf 2009]. Die Daten selbst wiederum werden unter die zugehörige DbCL (Database Contents License) gestellt.

[14] http://creativecommons.org/licenses/by-sa/2.0/
[15] http://www.openstreetmap.org/stats/data_stats.html

Diese OdBL/DbCL-Kombination fordert neben der Namensnennung des Rechteinhabers auch weiterhin die Weitergabe unter gleichen Bedingungen, allerdings sind davon nur abgeleitete Daten betroffen. Dadurch ist es mit der neuen Lizenz möglich, Karten mit Ebenen aus vorher inkompatiblen Datenquellen (siehe Beispiel in Kapitel 3.3.2) zu erstellen und beliebig zu lizensieren. Die Umstellung wird allerdings noch einige Zeit in Anspruch nehmen. Derzeit müssen neue Benutzer, die neue Daten erfassen oder bestehende Daten editieren, bereits beiden Lizenzen zustimmen, während bestehende Mitglieder die Relizensierung ihrer Daten derzeit freiwillig vornehmen können. [OpenStreetMap-Wiki 2010]

3.1.2 Datenmodell

Das Datenmodell von OpenStreetMap umfasst die drei Objekttypen Node, Way und Relation.

Nodes dienen der Abbildung von geometrischen Punkten, deren Position durch die geographische Länge und Breite spezifiziert wird. Beide Werte werden in Dezimalgrad mit sieben Nachkommastellen gespeichert. Dies „entspricht einer Genauigkeit von (im ungünstigsten Falle) ±1 cm und reicht daher für alle derzeit denkbaren Anwendungsgebiete aus – insbesondere, wenn man die Tatsache berücksichtigt, dass die beste derzeit mit handelsüblichen GPS erzielbare Genauigkeit im Bereich von ±5 m liegt“ [Ramm/Topf 2009]. Ein Node kann sowohl ein Stützpunkt eines übergeordneten Objektes (Polylinie oder Polygon) sein als auch ein alleinstehender POI wie beispielsweise eine Sehenswürdigkeit.

Ways wiederum können linienförmige Objekte wie Straßen und Flüsse abbilden. Ein Way besteht aus einer geordneten Liste von mindestens zwei bis maximal 2000 Nodes und verfügen somit automatisch über eine Richtung, die beispielsweise als Fahrrichtung von Einbahnstraßen oder als Fließrichtung von Gewässern Verwendung findet. Ein Node kann dabei von beliebig vielen Ways – beispielsweise allen Wegen einer Gabelung – referenziert werden.

Neben Linienzügen werden auch Flächen durch Ways abgebildet. Hierfür müssen zwei Bedingungen erfüllt sein: Der erste und letzte Stützpunkt des Ways müssen identisch sein und der Way muss passende Tags (siehe unten) enthalten. Nur mit Hilfe dieser Tags kann entschieden werden, ob es sich um eine Fläche oder einen geschlossenen Linienzug handelt.

Relationen ermöglichen die Abbildung von Beziehungen zwischen Objekten und bestehen daher aus einer geordneten Liste von Mitglieder-Objekten, den so genannten Members. Diese Members können Nodes, Ways oder auch andere Relationen sein, die mittels ihrer ID referenziert werden. Jedem Member kann eine Rolle zugewiesen werden. Ein typisches Beispiel hierfür ist eine Abbiegevorschrift, die aus drei Members besteht: Ausgangsstraße (Way mit der Rolle „from“), Zielstraße (Way mit der Rolle „to“) und Kreuzungspunkt (Node mit der Rolle „via“).

Allen drei genannten Objekttypen Node, Way und Relation können beliebig viele Eigenschaften, so genannte Tags, zugewiesen werden. Ein Tag besteht aus einem Schlüssel und einem Wert, die beide beliebig gesetzt werden können. Dies mag zunächst paradox klingen, hat aber den großen Vorteil, dass neu benötigte Tags „unkompliziert und ohne eine zentrale Autorität oder aufwändige Abstimmungsprozesse“ [Ramm/Topf 2009] eingeführt werden können. Damit die OpenStreetMap-Daten jedoch korrekt gerendert werden können und jeder Bach auch wirklich als Bach dargestellt wird, ist es natürlich unumgänglich, sich auf allgemeingültige Schlüssel und Werte zu einigen. Diese sind im OpenStreetMap-Wiki zu finden[16].

3.1.3 Abdeckung

Die Abdeckung der OpenStreetMap-Daten ist äußerst heterogen und lässt sich daher nicht pauschalisieren. Durch den Vergleich mit amtlichen oder kommerziellen Geodaten kann die Verteilung der Abdeckung für einen definierten Bereich ermittelt werden [Zielstra/Zipf 2010]. Abbildung 9 vergleicht die Länge des deutschen Straßennetzes von OpenStreetMap mit dem des kommerziellen Datenlieferanten TeleAtlas, dessen Daten u.a. bei Google Maps eingesetzt werden. Hierbei ist deutlich zu erkennen, dass bei OpenStreetMap in Ballungsräumen mehr Straßen und Wege als bei TeleAtlas erfasst wurden (grüne Bereiche), während sich in ländlichen Gegenden ein umgekehrtes Bild ergibt (orangene Bereiche). Diese räumlichen Disparitäten lassen sich durch die Anzahl der potenziell verfügbaren Mapper[17] erklären. Die Vergleiche von [Zielstra/Zipf 2010] zeigen ferner, dass die Abdeckung auch von der Art des Fortbewegungsmittels abhängt, da Mapper – besonders in Großstädten – auch kleine Fußgängerwege erfassen, die für den Autoverkehr nicht relevant sind. In Anbetracht der schnellen Entwicklung von

[16] http://wiki.openstreetmap.org/wiki/DE:Map_Features
[17] Als Mapper werden im OpenStreetMap-Umfeld jene Personen bezeichnet, die Daten erfassen.

OpenStreetMap, ist jedoch bereits abzusehen, dass der Rückstand zu den kommerziellen Anbietern auch in den ländlichen Bereichen zügig verkürzt wird.

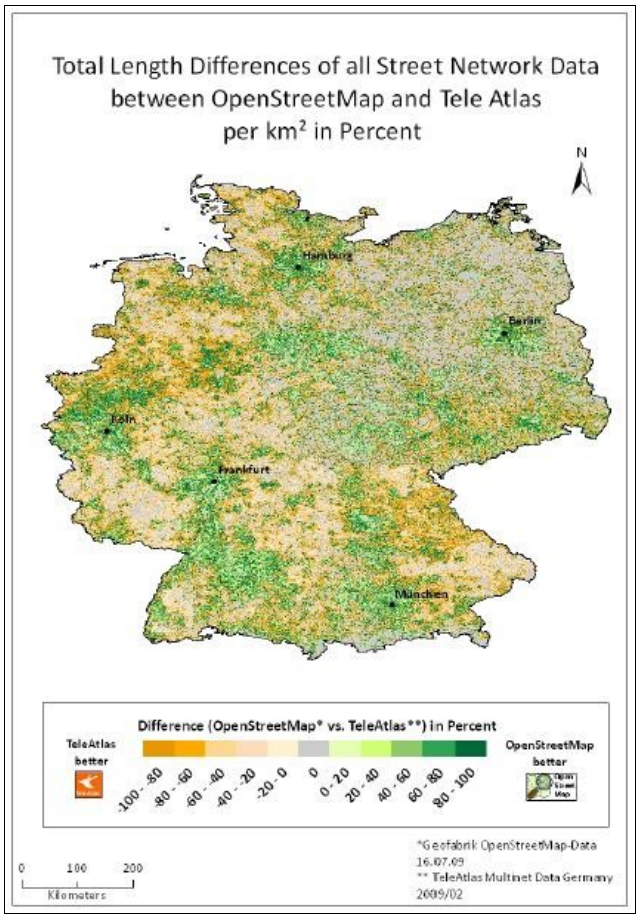

Abbildung 9: OpenStreetMap und TeleAtlas im Vergleich
Quelle: [Zielstra/Zipf 2010]

3.1.4 Dateiformate

Der Austausch der OSM-Daten erfolgt in dem XML (Extensible Markup Language)-basierten OSM-XML-Format (siehe Listing 1). Das Wurzelelement <osm> enthält neben dem optionalen <bounds>-Element, das das umschreibende Rechteck spezifiziert, beliebig viele Nodes, Ways und Relationen, die neben den IDs noch weitere obligatorische Attribute wie Nutzerkennung und -name des letzten Bearbeiters, Versionsnummer, Zeitpunkt der letzten Änderung und Sichtbarkeit enthalten. Jeder Node enthält zusätzlich noch eine Angabe des Breiten- und Längengrads sowie optional beliebig viele Tags mit den Attributen k (für key) und v (für value) als Kindelemente. Neben Tags verfügen Ways über eine geordnete Liste von mindestens zwei <nd>-

Elementen, deren ref-Attribut auf die ID des entsprechenden Nodes verweist. Die Relationen enthalten eine geordnete Liste von member-Elementen mit den Attributen type (Node oder Way), ref (ID des entsprechenden Nodes oder Ways) und role (entsprechende Rolle).

```
<osm version="..." generator="...">
  <bounds minlat="..." minlon="..." maxlat="..." maxlon="..."/>
  <node id="..." lat="..." lon="..." ...>
    <tag k="..." v="..."/>
    ...
  </node>
  ...
  <way id="..." ...>
    <nd ref="...">
    ...
    <tag k="..." v="..."/>
    ...
  </way>
  ...
  <relation id="..." ...>
    <member type="..." ref="..." role="..."/>
    <tag k="..." v="..."/>
    ...
  </relation>
  ...
</osm>
```

Listing 1: Aufbau einer OSM-XML-Datei

Das so genannte Plane File, ein wöchentlich neu erzeugter Abzug der OSM-Datenbank, wird im OSM-XML-Format zum Download[18] angeboten und umfasst trotz bzip2-Komprimierung im Frühjahr 2010 bereits 8,5 GB.

Neben dem OSM-XML-Format gibt es das OsmChange-Format (zu erkennen an der Endung osc), das der Beschreibung von Änderungen dient und ebenfalls XML-basiert ist. Als Wurzelelement dient hier <osmChange>, das beliebig viele <create>-, <modify>- und <delete>-Elemente umschließt, mit denen die darin enthaltenen Nodes, Ways und Relationen entsprechend hinzugefügt, ersetzt und gelöscht werden können. Die täglich, stündlich und minütlich erzeugten Änderungsdateien, die ebenfalls zum Download angeboten werden, liegen in diesem Dateiformat vor und ermöglichen somit die Daten aktuell zu halten, ohne permanent das komplette Planet File herunterladen zu müssen.

[18] http://planet.openstreetmap.org/

3.1.5 APIs

Der Zugriff auf die OpenStreetMap-Daten erfolgt nicht direkt per SQL (Structured Query Language) auf die Datenbank, sondern über ein definiertes API (Application Programming Interface). Diese REST (Represenational State Transfer)-basierte Schnittstelle verwendet HTTP (Hypertext Transfer Protocol) als Protokoll in der Anwendungsschicht. Jede Operation auf den Daten wird somit mittels eines HTTP-Requests abgebildet. Derzeit aktuell ist die API in der Version 0.6. Um die Daten eines bestimmten rechteckigen Bereichs in Form einer OSM-XML-Datei zu erhalten, stellt man eine HTTP-GET-Anfrage der folgenden Form, wobei die Platzhalter *W*, *S*, *O* und *N* durch die entsprechenden Koordinaten in Dezimalgrad ersetzt werden müssen:

```
http://www.openstreetmap.org/api/0.6/map?bbox=W,S,O,N
```

Allerdings darf der auf diese Weise spezifizierte geographische Ausschnitt derzeit nur eine Größe von maximal 0,25 Quadrat-Graden haben, was in Deutschland einem Bereich von etwa 50x50 km entspricht, oder nicht mehr als 50.000 Nodes im Ergebnis enthalten. [Ramm/Topf 2009]

Weniger Einschränkungen bzgl. der Größe bietet XAPI, das eXtended Application Programming Interface, das lesenden Zugriff auf eine Kopie der OSM-Datenbank bietet und in folgender Form angesprochen werden kann:

```
http://www.informationfreeway.org/api/0.6/*[bbox=W,S,O,N]
```

Statt dem Stern, der dazu führt, dass alle Nodes, Ways und Relationen des angegeben Bereichs enthalten sind, kann auch nur ein einzelner Objekttyp spezifiziert werden. Des Weiteren bietet XAPI die Möglichkeit nach Tags zu filtern. Die folgende Anfrage liefert beispielsweise alle Biergärten und Pubs in Island:

```
http://www.informationfreeway.org/api/0.6/node[bbox=-25,63,-13,67]
                        [amenity=biergarten|pub]
```

3.1.6 Visualisierung

Es gibt viele Möglichkeiten, um die OpenStreetMap-Daten in Form einer Karte zu visualisieren. Den einfachsten Weg bietet das Rendering-Tool Osmarender[19], das eine OSM-XML-Datei mittels XSLT (Extensible Stylesheet Language Transformation) in das XML-basierte Vektorgrafikformat SVG (Scalable Vector Graphics) umwandelt. Hierfür benötigt man lediglich einen XSLT-Prozessor, das Osmarender-Stylesheet sowie eine Regeldatei im XML-Format, „in der angegeben ist, welche Map Features wie gezeichnet werden sollen“ [Ramm/Topf 2009]. Die resultierende Vektorgrafik kann dann problemlos in eine Rastergrafik konvertiert werden.

Einen völlig anderen Weg geht Mapnik[20], der Default-Renderer des OpenStreetMap-Projektes, der in C++ geschrieben wurde. Da Mapnik unabhängig von OpenStreetMap entwickelt wurde, kann es nicht mit OSM-XML-Dateien umgehen, so dass diese zunächst in eine PostGIS-Datenbank (siehe Kapitel 5.2.2) importiert werden müssen. Anhand einer XML-basierten Map-Datei – ganz ähnlich der Regeldatei bei Osmarender – werden die OpenStreetMap-Daten dann gezeichnet. Im Gegensatz zu Osmarender erzeugt Mapnik allerdings keine Vektor- sondern Rastergrafiken.

Abbildung 10 zeigt München gerendert von Osmarender (links) und Mapnik (rechts).

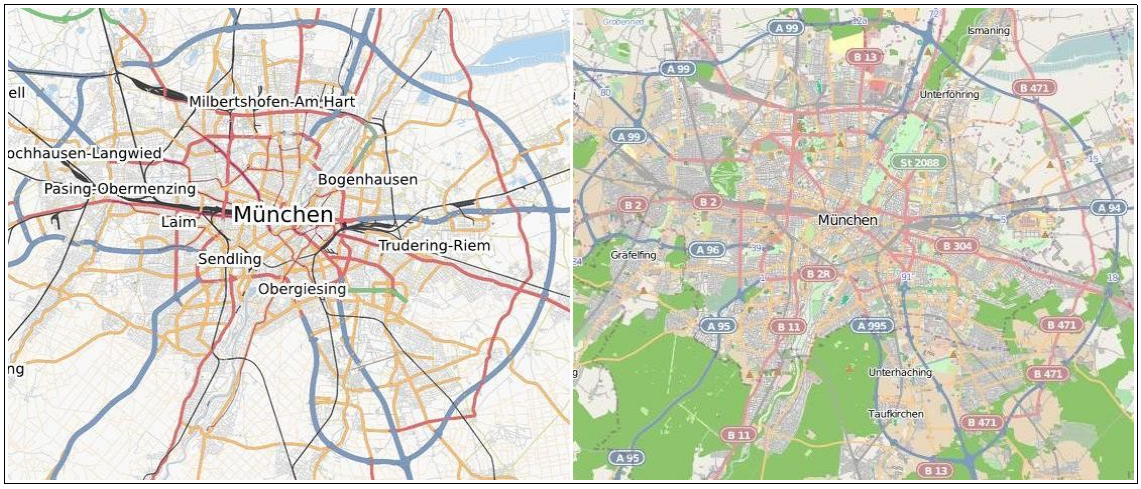

Abbildung 10: Osmarender und Mapnik im Vergleich
Quelle: http://www.openstreetmap.org

[19] http://wiki.openstreetmap.org/wiki/DE:Osmarender
[20] http://www.mapnik.org

3.1.7 Gazetteer-Dienste

Ein Gazetteer ist ein Ortslexikon, das geographische Bezeichnungen und deren jeweilige räumliche Lage auflistet. Es beinhaltet „ausschließlich räumliche Begriffe, also solche, die mit Koordinaten in Verbindung gebracht werden können“ [Korduan/Zehner 2008]. Die verzeichneten Geometrien sind neben Punkten oftmals auch Polylinien (bei Flüssen, Straßen, Wegen usw.) und Polygone (bei Stadt- und Gemeindegrenzen, Naturschutzgebieten, Mittelgebirgen usw.). Vielfach werden statt Polygonen vereinfachend auch umschreibende Rechtecke verwendet. Die Verknüpfung zwischen den Begriffen (z.B. Landkreis und zugehörige Gemeinden) muss – im Gegensatz zu einem Thesaurus – nicht explizit gespeichert werden, sondern kann mittels räumlicher Operationen berechnet werden. [Korduan/Zehner 2008]

Der Name Finder[21] war der erste Gazetteer-Dienst im Umfeld des OpenStreetMap-Projektes und durchsucht entsprechend dessen Datenbasis. Unterstützt werden neben einer klassischen nominalen Suche auch eine kategoriebasierte Suche nach POIs sowie eine koordinatenbasierte Suche. Der Dienst arbeitet mit Kacheln, die eine Fläche von etwa 111 km^2 aufweisen[22]. Wenn eine Suche auf den Umkreis einer Ortschaft eingeschränkt wird, wird nur die betreffende Kachel durchsucht. Fällt die Ortschaft jedoch in die Nähe einer Kachelgrenze, wird die benachbarte Kachel ebenfalls in die Suche miteinbezogen. Die möglichen Kategorien der POIs entsprechen den allgemein akzeptierten Werten der OpenStreetMap-Tags. Der Name Finder verfügt über eine XML-Schnittstelle, die es ermöglicht ihn in eigene Applikationen einzubinden. Diese wird im Anhang A näher beschrieben.

Nominatim[23] entstammt ebenfalls dem OpenStreetMap-Projekt und hat den Name Finder als Geocoder auf der OpenStreetMap-Homepage abgelöst. Im Gegensatz zu seinem Vorgänger berücksichtigt Nominatim auch Hausnummern, sofern diese erfasst wurden. Statt Kacheln verwendet Nominatim die entsprechenden Polygone der Verwaltungseinheiten (Länder, Bundesländer, Regierungsbezirke, Landkreise usw.) um die Suche einzuschränken und liefert damit präzisere Ergebnisse. Alternativ kann die Suche auch mittels „viewbox“ auf ein definiertes Rechteck (beispielsweise den aktuell sichtbaren Kartenausschnitt) eingegrenzt werden. Nominatim verfügt ebenfalls über ein API, das im Anhang B genauer beschrieben wird.

[21] http://gazetteer.openstreetmap.org
[22] http://wiki.openstreetmap.org/wiki/Name_finder
[23] http://nominatim.openstreetmap.org

3.2 GeoNames

GeoNames[24] ist völlig unabhängig vom OpenStreetMap-Projekt, wird jedoch neben Nominatim ebenfalls von der Suche auf der OpenStreetMap-Webseite eingebunden. GeoNames steht mit CC-BY 3.0[25] unter einer ähnlichen Lizenz, die jedoch nicht vorschreibt, die abgeleiteten eigenen Werke ebenso zu lizenzieren. Die Datenbank des Projektes enthält mittlerweile über acht Millionen geographische Bezeichnungen für 6,5 Millionen Objekte aus verschiedensten Quellen, die jeweils einer von 645 unterschiedlichen Objektklassen zugeordnet sind [GeoNames 2010]. Diese Daten können über verschiedene REST-basierte Web Services im XML- oder JSON (JavaScript Object Notation)-Format abgefragt werden[26]. Neben gewöhnlichem Geocoding und Reverse Geocoding können auch Höhendaten, Postleitzahlen, das aktuelle Wetter, georeferenzierte Wikipedia-Artikel oder Zeitzonen abgefragt werden. Im Gegensatz zu Name Finder und Nominatim kann bei GeoNames der Suchradius festgelegt werden, so dass die zurückgelieferten Suchtreffer der spezifizierten Position wirklich am nächsten sind. Straßennamen und Hausnummern unterstützt GeoNames bisher allerdings nur in den Vereinigten Staaten. Eine detailliertere Beschreibung des GeoNames-API ist im Anhang C zu finden.

[24] http://www.geonames.org/
[25] http://creativecommons.org/licenses/by/3.0/
[26] http://www.geonames.org/export/ws-overview.html

3.3 Höhendaten

Da OpenStreetMap bislang nur zweidimensionale Daten liefert, werden für Höhendaten zusätzliche Datenquellen benötigt. Aus diesen Höhendaten können dann die in topografischen Karten obligatorischen Isohypsen und Schummerungen erzeugt werden, die die Erdoberfläche wesentlich plastischer erscheinen lassen und dadurch die Orientierung im Gelände vereinfachen.

3.3.1 SRTM

Die Shuttle Radar Topography Mission (SRTM) startete im Jahr 2000, um die Topographie der Erde zwischen dem 60. Breitengrad der nördlichen Hemisphäre und dem 54. Breitengrad der südlichen Hemisphäre mittels Radar zu erfassen. Die Messungen erfolgten durch zwei räumlich getrennte Antennen. Während sich eine Antenne im Laderaum des Space Shuttles befand, war die zweite Antenne an einem 60 Meter langen Ausleger angebracht. Auf diese Weise erhielt man Radarbilder aus minimal unterschiedlichen Perspektiven, aus denen durch Vergleich der Phaseninformation ein Interferogramm erstellt werden kann, in dem die Topographie des Geländes bereits zu erkennen ist. Unter Berücksichtigung der genauen Position des Space Shuttles im Orbit kann daraus schließlich die exakte Höhe über Grund ermittelt werden und man erhält schließlich ein DEM (Digital Elevation Model) der Erde. [DLR 2009]

Die USGS (United States Geological Survey), die geowissenschaftliche Behörde des US-amerikanischen Innenministeriums, bietet SRTM-Daten mit einer Auflösung von einer Bogensekunde (etwa 30 Meter) für die Vereinigten Staaten sowie drei Bogensekunden (etwa 90 Metern) für den restlichen Bereich als Kacheln mit einer Größe von 1 x 1 Grad und damit 1201 x 1201 Pixeln im binären hgt-Format frei zum Download an[27]. Sie weisen eine vertikale Genauigkeit von ±6 Metern auf und dürfen uneingeschränkt verwendet werden. Aufgrund von Abschattungen weisen die Daten in steilen Gebieten allerdings Lücken auf. Während die Datenlücken im Bereich des Mount Everest 9% ausmachen, fehlen weltweit nur 0,15% [Jacobsen 2004].

Das CGIAR-CSI (Consultive Group on International Agricultural Research – Consortium on Spatial Information) bietet nachbearbeitete SRTM-Daten im

[27] http://dds.cr.usgs.gov/srtm/

Georeferenced Tagged Image File Format (GeoTIFF) an[28], bei denen die Datenlücken durch Interpolation eliminiert wurden, für die nicht-kommerzielle Verwendung an. Diese Einschränkung hat allerdings zur Folge, dass die Daten des CGIAR-CSI nicht mit OpenStreetMap-Daten kombiniert werden können, da die CC-BY-SA 2.0 solch eine Limitierung nicht erlaubt.

3.3.2 ASTER

Seit Juni 2009 bieten die US-amerikanische Luft- und Raumfahrtbehörde NASA (National Aeronautics and Space Administration) und das japanische Ministerium METI (Ministry of Economy, Trade and Industry) ein GDEM (Global Digital Elevation Model) aus Daten des ASTER (Advanced Spaceborn Thermal Emission and Reflection Radiometer)-Sensors kostenlos zum Download[29] an. Das ASTER-GDEM bietet im Vergleich zu SRTM eine größere Abdeckung bis zum 83. Breitengrad Nord/Süd und eine durchgehende horizontale Auflösung von einer Bogensekunde, also 30 Metern, sowie eine vertikale Genauigkeit von 20 Metern[30]. Trotz der niedrigeren Genauigkeit der Höhenwerte können laut [Jacobsen 2004] in Gebirgen mehr Details aus dem ASTER-GDEM als aus den SRTM-Resultaten entnommen werden. Aufgrund der lizenzbedingten Einschränkungen, die eine Veröffentlichung von abgeleiteten Werken nur erlaubt, wenn die ursprünglichen ASTER-Höhenwerte nicht mehr rekonstruierbar sind[31], sind SRTM-Rohdaten jedoch vorzuziehen.

3.3.3 Visualisierung

Höhendaten können als Isohypsen oder in Form einer Schummerung in zweidimensionalen Karten visualisiert werden (siehe Abbildung 11). Während Isohypsen die relativ genaue Ermittlung der Höhe an jedem Punkt der Karte ermöglichen, dient die Schummerung lediglich dazu die Geländeformen zu veranschaulichen. Die Schatten entstehen durch die Beleuchtung des Geländes mit einer imaginären Lichtquelle, die sich meist im Nordwesten befindet. Auf diese Weise erscheinen die Nordwesthänge einer Erhebung hell, während die Südosthänge in

[28] http://srtm.csi.cgiar.org/
[29] http://www.gdem.aster.ersdac.or.jp/index.jsp
[30] https://lpdaac.usgs.gov/lpdaac/products/aster_products_table/routine/global_digital_elevation_model/v1/astgtm
[31] https://lpdaac.usgs.gov/lpdaac/products/aster_policies

dunklen Farbtönen dargestellt werden. Je dunkler der Schatten, umso stärker ist die Hangneigung an der entsprechenden Stelle.

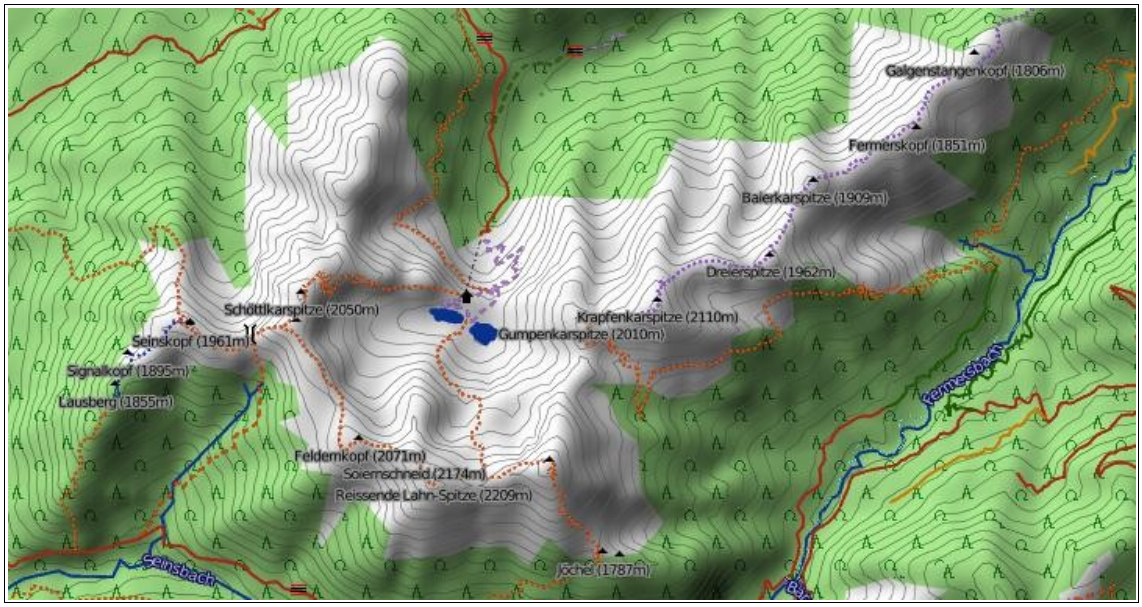

Abbildung 11: Visualisierung von Höhendaten
Quelle: http://topo.openstreetmap.de/

4 Offene Standards

Für offene Standards existieren zahlreiche Definitionen. Die folgende Definition wurde von zahlreichen Organisationen darunter der FSFE (Free Software Foundation Europe) in der Genfer Erklärung 2008 unterstützt [FSFE ET AL. 2008]:

Ein Offener Standard bezieht sich auf ein Format oder Protokoll, das

1. *einer vollständig öffentlichen Bewertung und Nutzung ohne Hemmnisse auf eine für alle Beteiligten gleichermaßen zugänglichen Weise unterliegt,*
2. *ohne jegliche Komponenten oder Erweiterungen ist, die von Formaten oder Protokollen abhängen, die selbst nicht der Definition eines Offenen Standards entsprechen,*
3. *frei ist von juristischen oder technischen Klauseln, die seine Verwendung von jeglicher Seite oder jeglichem Geschäftsmodell einschränken,*
4. *unabhängig von einem einzelnen Anbieter geleitet und weiterentwickelt wird in einem Prozess, der einer gleichberechtigten Teilnahme von Wettbewerbern und Dritten offen steht,*
5. *verfügbar ist in verschiedenen vollständigen Implementierungen von verschiedenen Anbietern oder als vollständige Implementierung gleichermaßen für alle Beteiligten.*

Um eine Anwendung möglichst plattformübergreifend und interoperabel zu gestalten, müssen die Schnittstellen offenen Standards genügen. Dies ermöglicht die einfache Austauschbarkeit einzelner Komponenten des Gesamtsystems und macht die Applikation auf vielen Plattformen lauffähig. Im Folgenden werden die wichtigsten Standards im GIS-Umfeld vorgestellt.

4.1 W3C-Standards

Das W3C wurde 1994 von Tim Berners-Lee, dem Erfinder des World Wide Web (WWW), gegründet mit dem Ziel die Technologien rund um das WWW zu standardisieren. Da das W3C jedoch keine zwischenstaatlich anerkannte Organisation ist, sind die von ihm verabschiedeten Standards keine offiziellen Standards, sondern nur De-facto-Standards. Zu diesen so genannten W3C Recommendations zählen allerdings mit HTML (Hypertext Markup Language), CSS (Cascading Stylesheets) und XML die wesentlichen Grundlagen heutiger Webseiten. Jede W3C-Empfehlung durchläuft zuvor die Status Working Draft, Last Call Working Draft, Candidate Recommendation und Proposed Recommendation.

4.1.1 Geolocation API

Das Geolocation API ist eine Schnittstelle zur Ermittlung des aktuellen Benutzerstandorts per JavaScript im Browser und ermöglicht somit positionsbezogene Webapplikationen. Dabei hängt es von der jeweiligen Implementierung ab, welche Technologie (GPS, GSM/UMTS, WPS usw.) letztendlich zur Lokalisierung eingesetzt wird. Die Schnittstelle unterstützt neben der einmaligen Ermittlung des Standorts auch permanente Updates der Position. Um die Anwendung nicht zu blockieren erfolgt die Lokalisierung immer asynchron. Mittels der PositionOptions kann die erforderliche Genauigkeit und ein Timeout für die Ortsbestimmung festgelegt werden. Um den Stromverbrauch und die Lokalisierungszeit zu reduzieren können auch explizit alte Aufenthaltsorte aus dem Cache angefordert werden. Hierfür kann ein maximales Alter der Position in Sekunden spezifiziert werden. Zurückgeliefert werden neben der geographischen Länge und Breite (jeweils WGS 84) die Genauigkeit in Metern sowie optional die Höhe, der Kurs und die Geschwindigkeit. Um die Privatsphäre des Nutzers nicht zu verletzen, muss vor einer Bestimmung des Aufenthaltsortes allerdings zunächst dessen Erlaubnis eingeholt werden. Die Spezifikation liegt seit September 2010 als Candidate Recommendation [Popescu 2010] vor.

Bislang gibt es mit Opera 11 erst eine spezifikationskonforme Implementierung, die die Test Suite erfolgreich absolviert hat. Dennoch wird die Schnittstelle mittlerweile von zahlreichen Browsern – wenn auch nicht vollkommen konform – unterstützt. Die Browser-Erweiterung Gears[32] von Google, das federführend bei der Spezifikation beteiligt ist, lieferte bereits 2008 eine erste Implementierung und brachte die Schnittstelle somit nach und nach in die wichtigsten Desktop-Browser (Chrome, Firefox, Internet Explorer, Opera und Safari) sowie auf die Mobilplattformen Windows Mobile (mittels der Browser Internet Explorer Mobile oder Opera Mobile) und Android. Mittlerweile unterstützen die meisten Desktop-Browser und viele Mobilplattformen in ihren aktuellsten Versionen das Geolocation API nativ und benötigen daher kein Plugin mehr. Da der Mobilfunkmarkt aber schon viel früher nach einer JavaScript-Schnittstelle zur Positionsbestimmung strebte, gibt es dort viele ähnliche APIs. Abhilfe gegen die Fragmentierung schafft hier das Open Source-Projekt geo-location-javascript[33], das einen am Geolocation API orientierten Wrapper um diese plattformspezifischen Implementierungen bildet. Berücksichtigt werden u.a. iOS ab Version 3.0, BlackBerry

[32] http://gears.google.com/
[33] http://code.google.com/p/geo-location-javascript/

ab Version 4.1, webOS sowie die Web Runtime (WRT), eine Browser-Erweiterung der S60-Plattform.

4.1.2 Widgets

Das W3C definiert Widgets folgendermaßen [Caceres/Priestley 2009]:

A widget is an interactive single purpose application for displaying and/or updating local data or data on the Web, packaged in a way to allow a single download and installation on a user's machine or mobile device.

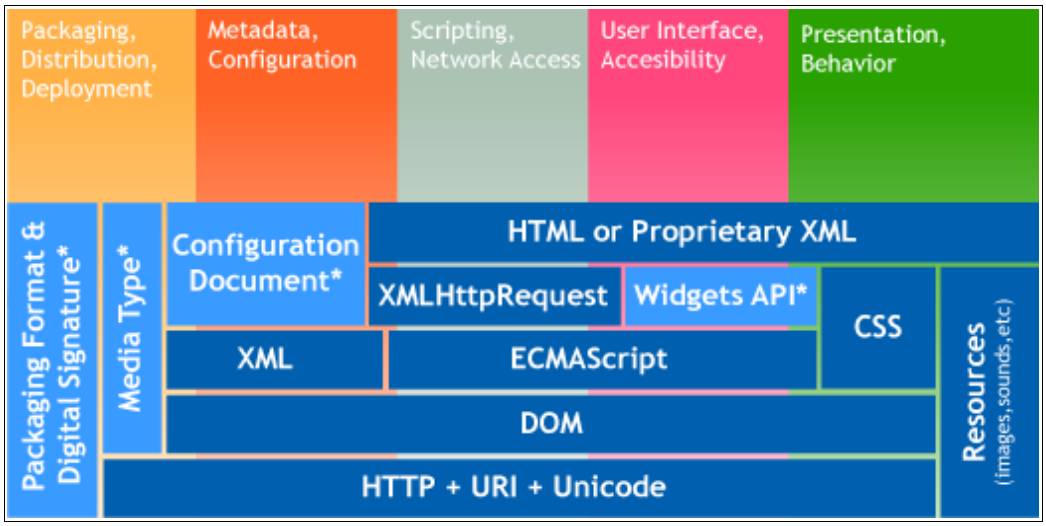

Abbildung 12: Architektur eines typischen Widgets
Quelle: http://www.w3.org/TR/2008/WD-widgets-land-20080414/

Abbildung 12 zeigt die Architektur eines typischen Widgets: Aufbauend auf bekannten Webstandards wie dem Übertragungsprotokoll HTTP, URIs (Uniform Resource Identifier) zur Referenzierung von Ressourcen (Grafiken, Audio, Video usw.) und der Unicode-Kodierung nutzen typische Widgets das DOM (Document Object Model), ECMAScript (der durch die European Computer Manufacturers Association standardisierte Kern von JavaScript) sowie die Auszeichnungssprachen XML, CSS (Cascading Style Sheets) und HTML. Des Weiteren verwenden Widgets üblicherweise auch die für das Web 2.0 charakteristische, asynchrone Datenübertragung über HTTP mittels des XMLHttpRequest-Objekts, das so genannte AJAX (Asynchronous JavaScript and XML). Alle bisher genannten Technologien finden auch in üblichen Webseiten Verwendung und sind größtenteils bereits durch das W3C oder andere Organisationen standardisiert. Hinzukommt ein

Paketformat, das alle Ressourcen eines Widgets in eine Datei verpackt, eine digitale Signatur zur Sicherstellung der Authentizität, ein eigener MIME-Type sowie eine Widget-spezifische JavaScript-Schnittstelle. Diese letztgenannten Elemente (hellblau hinterlegt) waren bisher noch nicht standardisiert und sind für die derzeitige Fragmentierung im Widgets-Bereich verantwortlich.

Diese Fragmentierung zeigt sich in der Inkompatibilität der auf dem Markt befindlichen proprietären Widget Runtimes, die sich darin äußert, dass ein Widget, das für eine Laufzeitumgebung entwickelt wurde, nicht auf einer anderen Laufzeitumgebung lauffähig ist. In der non-normativen Spezifikation „Widgets 1.0: The Widget Landscape“ [Caceres 2008] wurden die marktführenden Laufzeitumgebungen bezüglich ihrer Gemeinsamkeiten und Unterschiede analysiert, um die zu standardisierenden Aspekte eines Widgets zu identifizieren. Diese werden in der ebenfalls non-normativen Spezifikation „Widgets 1.0: Requirements“ [Caceres/Priestley 2009] in 54 konkreten Anforderungen formuliert. Die eigentliche Standardisierung dieser Anforderungen erfolgt dann in mehreren normativen Spezifikationen.

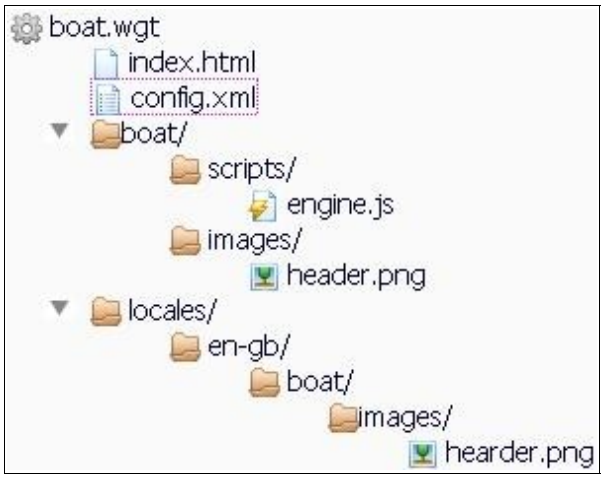

Abbildung 13: Dateistruktur eines Widgets
Quelle: http://www.w3.org/TR/2009/CR-widgets-20091201/

Die Spezifikation „Widget Packaging and Configuration“ [Caceres 2009], die seit Dezember 2009 als Candidate Recommendation vorliegt, beschriebt, wie ein Widget formal aufgebaut ist und welche Schritte eine Laufzeitumgebung durchführen muss, um ein Widget standardkonform zu starten. Formal ist ein Widget ein Zip-Archiv mit der Endung wgt und dem neu geschaffenen MIME-Type application/widget. Abbildung 13 zeigt beispielhaft die Dateistruktur eines Widgets, das die Startdatei index.html sowie

die Konfigurationsdatei config.xml im Wurzelverzeichnis enthält. Die Konfigurationsdatei ist eine XML-Datei mit dem Wurzelelement <widget>, das optional Metainformationen und Konfigurationsdaten enthält. Widgets, die mehrere Sprachen unterstützen, enthalten im Wurzelverzeichnis ein Unterverzeichnis locales, in dem die lokalisierten Ressourcen für jede unterstützte Sprache abgelegt werden. Die Test Suite zu dieser Spezifikation konnten bis Mai 2010 bereits drei Implementierungen (BONDI, Apache Wookie und Aplix Web runtime) erfolgreich absolvieren[34].

In der Spezifikation „The Widget Interface“ [Caceres et al. 2009] wird das Widget API definiert, das es ermöglicht auf die Metadaten des Widgets zuzugreifen, Nutzereinstellungen persistent zu speichern und auszulesen sowie Breite und Höhe des Darstellungsbereichs zu ermitteln. Daneben verfügt die Widget-Schnittstelle noch über eine Methode, die beliebige URIs an den passenden Handler delegiert. Auf diese Weise können aus einem Widget heraus beispielsweise E-Mails und Kurznachrichten verschickt, Telefonanrufe initiiert und Webseiten im Browser geöffnet werden. Die Spezifikation liegt ebenfalls seit Dezember 2009 als Candidate Recommendation vor und wies mit der Opera Widget Runtime bis Mai 2010 eine konforme Implementierung auf[35].

Daneben gibt es noch fünf weitere Spezifikationen in der Widgets-Spezifikationsfamilie: „Digital Signatures for Widgets“ [Caceres et al. 2010] spezifiziert die optionale digitale Signierung von Widgets durch Autoren und Distributoren, die die Überprüfung der Integrität und Authentizität eines Widgets ermöglicht. Automatisierte Updates eines Widgets über HTTP beschreibt die Spezifikation „Widgets Updates“ [Caceres/Berjon 2010], die der Konfigurationsdatei config.xml ein weiteres Element hinzufügt, das auf das UDD (Update Description Document) verweist. Dieses XML-Dokument enthält die aktuelle Versionsnummer des Widgets, einen Link, unter dem die aktuelle Version verfügbar ist sowie optional eine Beschreibung des Updates. [Berjon 2010] definiert in „Widget Access Request Policy“ ein weiteres Element in der Konfigurationsdatei, das es dem Widget-Autor ermöglicht, Netzwerkressourcen zu deklarieren, auf die das Widget zugreifen möchte. Auf alle anderen Netzwerkressourcen darf aus Sicherheitsgründen nicht zugegriffen werden. „Widgets 1.0: Widgets URIs“ [Berjon 2009] spezifiziert ein URI-Schema, um Ressourcen innerhalb eines Widgets zu adressieren. Den Abschluss bildet die

[34] http://dev.w3.org/2006/waf/widgets/imp-report/
[35] http://dev.w3.org/2006/waf/widgets-api/imp-report/

Spezifikation „View Mode Media Feature“ [Berjon et al. 2010], die es ermöglicht das Styling dem aktuellen Ansichtsmodus (minimiert, maximiert, Vollbild usw.) anzupassen.

Damit Widgets mehr Funktionalitäten bieten als Webseiten, werden JavaScript-Schnittstellen benötigt, die es ermöglichen auf verschiedene Gerätefunktionalitäten (Adressbuch, Kalender, Kamera, GPS usw.) zuzugreifen. Für diese Schnittstellen ist die Device APIs and Policy Working Group des W3C[36] zuständig, die im Juli 2009 ihre Arbeit aufnahm und plant bis Ende Juli 2011 die Arbeit an allen Spezifikationen abgeschlossen zu haben. Von den insgesamt 15 Spezifikationen sind derzeit allerdings erst sechs als Working Draft veröffentlicht. Wesentlich weiter fortgeschritten sind die Spezifikationen von BONDI und JIL (Joint Innovation Lab).

BONDI[37] ist eine Initiative der OMTP (Open Mobile Terminal Platform)[38], die 2004 von den Mobilfunknetzbetreibern AT&T, Deutsche Telekom AG, Orange, Smart Communications, Telecom Italia, Telefonica und Vodafone gegründet wurde. BONDI wurde 2008 ins Leben gerufen, um unter dem Slogan „Write once, deploy anywhere“ Device APIs zu definieren, die es Webapplikationen und Widgets ermöglichen, die Funktionalitäten eines Mobiltelefons unabhängig von dessen Hersteller oder dessen Plattform zu nutzen. Um den Benutzer vor den Risiken zu schützen, die sich durch die Verfügbarkeit der sensitiven APIs ergeben, arbeitet BONDI daneben auch an einem Sicherheitsframework, das den Zugriff auf diese Schnittstellen reguliert. Im Februar 2010 wurde die BONDI-Spezifikation in der Version 1.1 veröffentlicht. Dieses Release verfügt über 14 Geräteschnittstellen, darunter auch ein Geolocation API, das sich strikt an den Entwurf des W3C (siehe Kapitel 4.1.1) hält. Neben der Spezifikation veröffentlichte BONDI bereits ein quelloffenes Software Development Kit (SDK) samt Emulator sowie eine Referenzimplementierung für Windows Mobile, die als erste Implementierung die Test Suite der W3C-Spezifikation „Widgets Packaging and Configuration“ erfolgreich absolvierte. Im Februar 2010 wurde auf dem Mobile World Congress (MWC) in Barcelona mit dem Samsung Wave (bada) das erste Mobiltelefon vorgestellt, das ab Werk über eine BONDI-Implementierung verfügt.

JIL[39] ist ein vergleichbares Joint Venture, das 2008 von China Mobile Limited, SOFTBANK und Vodafone gegründet und 2009 um Verizon Wireless erweitert wurde, mit dem Ziel, die Fragmentierung auf dem Mobilfunkmarkt durch die Entwicklung einer

[36] http://www.w3.org/2009/dap/
[37] http://bondi.omtp.org/
[38] http://www.omtp.org/
[39] http://www.jil.org/

standardisierten Plattform für mobile Widgets zu reduzieren. JIL spezifiziert für diese Plattform Device APIs samt Sicherheitsframework sowie den formalen Aufbau von Widgets, der sich stark an der W3C-Spezifikation „Widgets Packaging and Configuration“ orientiert und diese erweitert. Daneben folgt JIL auch den W3C-Entwürfen für digitale Signaturen und Widget-Updates. Aktuell liegt die JIL-Spezifikation in der Version 1.2 vor, die derzeit von zwei auf dem Markt befindlichen Mobiltelefonen unterstützt wird. Für Entwickler steht ein SDK samt Emulator zur Verfügung.

Da beide Initiativen das gleiche Ziel verfolgen wurde auf dem MWC 2010 die Zusammenarbeit in der neu gegründeten Wholesale Applications Community (WAC)[40] beschlossen. Die WAC – gebildet aus mehr als 40 Mobilfunkanbietern – hat es sich zum Ziel gesetzt innerhalb von zwölf Monaten eine auf BONDI und JIL aufbauende Spezifikation zu verabschieden und diese ins W3C einzubringen. Daneben soll auch der Zugriff auf verschiedene Netzdienste wie Nachrichtenvermittlung, Positionierung (siehe Kapitel 2.3.2) und Bezahlung aufbauend auf dem OneAPI[41] der GSM Association (GSMA) standardisiert werden. Mit 3 Milliarden Kunden weltweit besitzt die WAC laut [Sachse 2010] das Potenzial die Fragmentierung im Bereich mobiler Widgets zu überwinden.

Neben BONDI und JIL hat auch Marktführer Nokia APIs seiner Web Runtime (WRT) dem W3C zur Verfügung gestellt. Die WRT ist eine Widget-Erweiterung des Browsers der S60-Plattform, die zahlreiche JavaScript-Schnittstellen für Gerätefunktionalitäten bietet. WRT Widgets werden zwar von allen neueren S60-Handys unterstützt, sind aber nicht auf anderen Plattformen lauffähig.

Aktuell ist es demnach noch nicht möglich, ein Widget ohne Portierung auf viele Geräte zu bringen. Abhilfe schaffen hier so genannte Mobile Web Frameworks (siehe Kapitel 5.6), die Widgets in native Anwendungen für verschiedene Plattformen konvertieren.

[40] http://www.wholesaleappcommunity.com/
[41] https://gsma.securespsite.com/access/default.aspx

4.2 OGC-Standards

Das OGC wurde 1994 von Universitäten und Software-Herstellern aus dem GIS-Bereich gegründet, dem mittlerweile mehrere Hundert Firmen, Behörden und Forschungseinrichtungen angehören. Ziel des OGC ist die Schaffung von Schnittstellenstandards, um den interoperablen Austausch von Geodaten zu ermöglichen. Analog zum W3C durchläuft ein künftiger Standard einen konsensorientierten Prozess vom ersten Entwurf bis zur OGC-Empfehlung. Das OGC gilt als das mit Abstand bedeutendste Standardisierungsgremium im GIS-Bereich. [Blankenbach 2007]

4.2.1 WMS

Ein WMS bietet die Möglichkeit dynamisch gerenderte Kartenbilder von georeferenzierten Daten per HTTP-Request anzufordern. Die WMS-Spezifikation liegt seit März 2006 in der Version 1.3.0 vor und definiert die drei Methoden GetCapabilities, GetMap und GetFeatureInfo, wobei letztere lediglich optional ist. [de la Beaujardiere 2006]

Mit GetCapabilities kann der Client Metadaten über den WMS abfragen. Dies ist nötig, um herauszufinden, welche Parameterwerte in den GetMap-Anfragen valide sind. Der WMS antwortet mit einem XML-Dokument, das u.a. Auskunft darüber gibt, welche Ausgabeformate er unterstützt, welche verschiedenen Layer er anbietet sowie deren jeweilige Abdeckung, unterstützten Koordinatenreferenzsysteme und Kartenstile. Daneben enthält das Dokument auch Informationen, auf welche Layer die Methode GetFeatureInfo angewandt werden kann.

Das Herzstück eines WMS ist der GetMap-Request, mit dem ein Kartenbild angefordert wird. Tabelle 1 listet die wichtigsten Parameter sowie deren Bedeutung auf, wobei die letzten drei Parameter (TRANSPARENT, BGCOLOR und EXCEPTIONS) nicht verpflichtend gesetzt werden müssen:

Parameter	Bedeutung
LAYERS	Kommaseparierte Liste der gewünschten Kartenlayer in der gewünschten Zeichenreihenfolge
STYLES	Kommaseparierte Liste der gewünschten Kartenstile, Reihenfolge korrespondierend zu LAYERS
CRS	Koordinatenreferenzsystem
BBOX	Kommaseparierte Liste der Eckkoordinaten der Bounding Box in CRS-Einheiten (minX, minY, maxX, maxY)
WIDTH	Breite der Karte in Pixeln
HEIGHT	Höhe der Karte in Pixeln
FORMAT	Ausgabeformat der Karte (MIME-Type)
TRANSPARENT	Transparenz des Hintergrundes (TRUE oder FALSE)
BGCOLOR	Hintergrundfarbe als hexadezimaler RGB-Wert
EXCEPTIONS	Ausgabeformat im Fehlerfall

Tabelle 1: Parameter des WMS-GetMap-Requests
Quelle: [de la Beaujardiere 2006]

Die optionale Methode GetFeatureInfo bietet die Möglichkeit, genauere Informationen zu den Features an einem Punkt einer vorher angeforderten Karte zu erhalten. Ein solcher Request ist jedoch nur für diejenigen Layer möglich, die in der GetCapabilities-Antwort entsprechend attributiert sind. Da ein GetFeatureInfo-Request immer auf einem vorherigen GetMap-Request basiert, müssen die Parameter der GetMap-Anfrage wieder mitangegeben werden. Tabelle 1 listet die zusätzlichen, spezifischen Parameter sowie deren Bedeutung auf, wobei die letzten zwei Parameter (FEATURE_COUNT und EXCEPTIONS) nur optional sind:

Parameter	Bedeutung
QUERY_LAYERS	Kommaseparierte Liste der Kartenlayer, zu denen mehr Informationen erwünscht sind
INFO_FORMAT	Ausgabeformat der Feature-Informationen (MIME-Type)
I	Spalte des abzufragenden Pixels (zwischen 0 und WIDTH-1)
J	Zeile des abzufragenden Pixels (zwischen 0 und HEIGHT-1)
FEATURE_COUNT	Maximale Anzahl der Features pro Layer, für die Informationen zurückgeliefert werden sollen
EXCEPTIONS	Ausgabeformat im Fehlerfall

Tabelle 2: Parameter des WMS-GetFeatureInfo-Requests
Quelle: [de la Beaujardiere 2006]

4.2.2 SLD-WMS

Der WMS bietet dem Nutzer zwar viele Freiheiten (Grafikformat, Kartenausschnitt, Höhe und Breite, Auswahl und Reihenfolge der Layer usw.), aber nur wenige Möglichkeiten bzgl. der Ausgestaltung der Karte. Es können zwar durchaus für jeden einzelnen Layer mehrere Kartenstile angeboten werden, aber von jedem dieser Stile ist lediglich der Name bekannt und dieser sagt wenig über das resultierende Kartenbild aus. Sollte dem Nutzer keiner der angebotenen Kartenstile zusagen, besteht keine Möglichkeit einen eigenen zu definieren.

Diese Lücke füllt das Styled Layer Descriptor (SLD)-Profil, eine Erweiterung des WMS, die benutzerdefiniertes Styling ermöglicht. SLD ist eine XML-basierte Auszeichnungssprache, die es ermöglicht den einzelnen Layer eines WMS beliebige Kartenstile zuzuordnen. Das GetCapabilities-Antwortdokument gibt Auskunft darüber, ob ein WMS SLD-fähig ist oder nicht. Um eine Karte mit benutzerdefiniertem Styling zu erhalten, wird dem GetMap-Request entweder eine URL (Uniform Resource Locator), unter der das betreffende SLD-Dokument abgerufen werden kann oder gleich der Inhalt jenes Dokuments angehängt.

Seit der derzeit aktuellen Version 1.1.0 [Lupp 2007] umfasst die SLD-Spezifikation nur noch die WMS-spezifischen Teile. Das eigentliche Styling wurde in die Symbology Encoding (SE)-Spezifikation [Müller 2006] ausgelagert, um auch in anderen Diensten wiederverwendet werden zu können. Mittels Symbology Encoding können Punkte, Linien, Polygone, Texte und Raster abhängig vom Maßstab und Objektattributen symbolisiert werden. Die Unabhängigkeit vom zugrunde liegenden Datenformat, „ermöglicht den Austausch von Visualisierungsvorschriften zwischen Systemen verschiedener Hersteller“ [Lendholt 2009].

Listing 2 zeigt ein einfaches SLD-Dokument, das die Strichstärke mit dem Maßstab variiert:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<StyledLayerDescriptor version="1.1.0"
xsi:schemaLocation="http://www.opengis.net/sld
StyledLayerDescriptor.xsd"
xmlns="http://www.opengis.net/sld"
xmlns:ogc="http://www.opengis.net/ogc"
xmlns:se="http://www.opengis.net/se"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <NamedLayer>
  <se:Name>Strassen</se:Name>
  <UserStyle>
   <se:FeatureTypeStyle>
    <se:Rule>
```

```
     <se:MinScaleDenominator>10000</se:MinScaleDenominator>
     <se:MaxScaleDenominator>100000</se:MaxScaleDenominator>
     <se:LineSymbolizer>
      <se:Stroke>
       <se:SvgParameter name="stroke">#FF0000</se:SvgParameter>
       <se:SvgParameter name="stroke-width">4</se:SvgParameter>
      </se:Stroke>
     </se:LineSymbolizer>
    </se:Rule>
    <se:Rule>
     <se:MinScaleDenominator>100000</se:MinScaleDenominator>
     <se:MaxScaleDenominator>1000000</se:MaxScaleDenominator>
     <se:LineSymbolizer>
      <se:Stroke>
       <se:SvgParameter name="stroke">#FF0000</se:SvgParameter>
       <se:SvgParameter name="stroke-width">2</se:SvgParameter>
      </se:Stroke>
     </se:LineSymbolizer>
    </se:Rule>
   </se:FeatureTypeStyle>
  </UserStyle>
 </NamedLayer>
</StyledLayerDescriptor>
```

Listing 2: Beispiel eines SLD-Dokuments

Das Filtern nach Objektattributen geschieht mittels Filter Encoding (FE), einem weiteren Standard des OGC [VRETANOS 2005], der auch räumliches Filtern per XML unterstützt. Dadurch ist es beispielsweise möglich nur die Autobahn- und Bundesstraßenobjekte eines Straßen-Layers zu visualisieren wie der Attributfilter in Listing 3 zeigt, der innerhalb eines Rule-Elements (siehe Listing 2) eingesetzt werden kann:

```
<ogc:Filter>
  <ogc:Or>
    <ogc:PropertyIsEqualTo>
      <ogc:PropertyName>Kategorie</ogc:PropertyName>
      <ogc:Literal>Autobahn</ogc:Literal>
    </ogc:PropertyIsEqualTo>
    <ogc:PropertyIsEqualTo>
      <ogc:PropertyName>Kategorie</ogc:PropertyName>
      <ogc:Literal>Bundesstrasse</ogc:Literal>
    </ogc:PropertyIsEqualTo>
  </ogc:Or>
</ogc:Filter>
```

Listing 3: Beispiel eines FE-Attributfilters

4.2.3 WMTS

Die hohe Flexibilität eines WMS hat jedoch auch einen gravierenden Performance-Nachteil, da jede Karte erst noch gerendert werden muss. Diesem Manko versucht der neue OGC-Standard Web Map Tile Service (WMTS) zu begegnen, der seit April 2010 in der Version 1.0.0 vorliegt [Masó et al. 2010]. Ein WMTS bietet für jeden Layer vordefinierte Kacheln an, die entweder bereits gerendert sind oder on-the-fly generiert und für spätere Anfragen in einem Cache vorgehalten werden. Der Client muss die entsprechenden Kacheln vom WMTS anfordern, selbstständig zusammensetzen und die Layer in der gewünschten Reihenfolge übereinanderlegen. Dieser Kachel-Ansatz wird schon seit Jahren von den allseits bekannten Kartendiensten wie Google Maps, Bing Maps oder auch OpenStreetMap praktiziert. Der Vorteil der Performance geht aber zur Lasten der Flexibilität. Zwar stehen im Regelfall genügend unterschiedliche Maßstäbe zur Verfügung, aber im Gegensatz zu einem SLD-fähigen WMS kann das Styling nicht beeinflusst werden. Hinzukommt das die generierten Kacheln meist nicht den aktuellen Datenstand widerspiegeln, abhängig davon wie oft sie neu gerendert werden.

Die WMTS-Spezifikation definiert analog zum WMS die drei Methoden GetCapabilities, GetTile und GetFeatureInfo, wobei letztere wiederum nicht obligatorisch implementiert sein muss. Das GetCapabilities-Anwortdokument gibt Auskunft über die verfügbaren Layer, Grafikformate, Styles und die so genannten TileMatrixSets. Ein TileMatrixSet (siehe Abbildung 14) besteht aus mehreren Kachelmatrizen. Jede Kachelmatrix wiederum repräsentiert die verfügbaren Kacheln eines Maßstabs im Koordinatenreferenzsystem des TileMatrixSets. Damit ein Client die Kacheln unterschiedlicher WMTS-Server zu einer Karte kombinieren kann ohne aufwendige Transformationen oder Skalierungen vornehmen zu müssen, definiert die Spezifikation so genannte Well-Known Scale Sets, eine Kombination aus einem Koordinatenreferenzsystem und mehreren Maßstäben und empfiehlt zur Steigerung der Interoperabilität deren Unterstützung. [Masó et al. 2010]

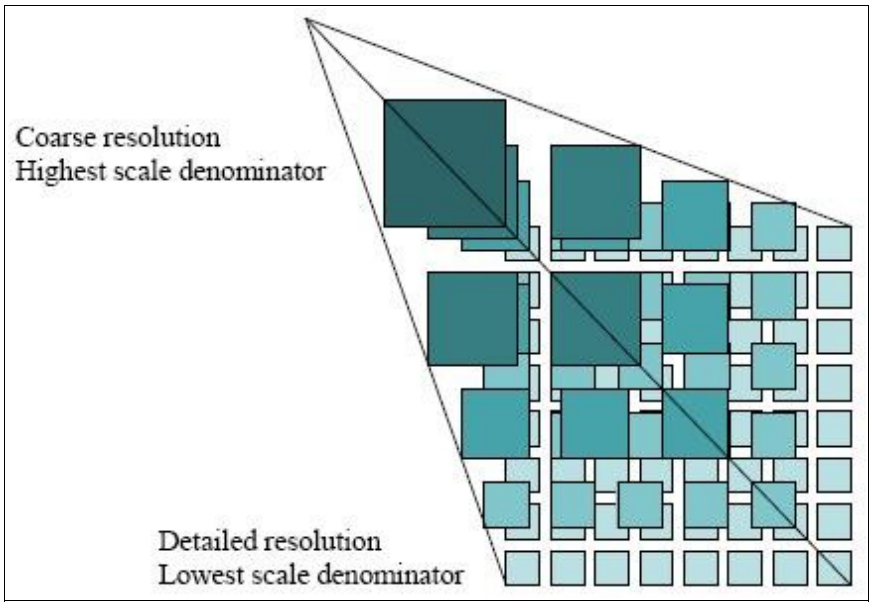

Abbildung 14: Struktur eines TileMatrixSets
Quelle: [Masó et al. 2010]

Tabelle 3 listet die wichtigsten Parameter des GetTile-Requests sowie deren Bedeutung auf:

Parameter	Bedeutung
Layer	Gewünschter Kartenlayer
Style	Gewünschter Kartenstil
Format	Ausgabeformat der Kachel (MIME-Type)
TileMatrixSet	Bezeichner des TileMatrixSets
TileMatrix	Bezeichner der Kachelmatrix
TileRow	Zeilenindex der gewünschten Kachel
TileCol	Spaltenindex der gewünschten Kachel

Tabelle 3: Parameter des WMTS-GetTile-Requests
Quelle: [Masó et al. 2010]

4.2.4 WFS

Der Web Feature Service (WFS) ermöglicht den lesenden und optional auch den schreibenden Zugriff auf vektorielle Geodaten per HTTP-Request. Die Spezifikation liegt seit Mai 2005 in der Version 1.1.0 vor und definiert die sechs Methoden GetCapabilities, DescribeFeatureType, GetFeature, GetGmlObject, LockFeature und Transaction wobei die letzten drei lediglich optional sind. [Vretanos 2005-2]

GetCapabilities gibt Auskunft über die unterstützten Methoden, Formate und Koordinatenreferenzsysteme sowie die angebotenen Feature-Klassen. Über

DescribeFeatureType können die XML-Schemata für jede Feature-Klasse abgefragt werden. Diese benötigt der Client, um mittels GetFeature konkrete Instanzen einer solchen Klasse abfragen zu können. Hierbei kann nach bestimmten Kriterien – attributbasiert oder räumlich – gefiltert werden. Tabelle 4 listet die wichtigsten Parameter sowie deren möglichen Werte und Bedeutung auf, wobei nur die ersten drei Parameter obligatorisch sind und nur einer der letzten drei Parameter verwendet werden darf:

Parameter	Bedeutung
TYPENAME	Kommaseparierte Liste der gewünschten Feature-Klassen
OUTPUTFORMAT	Ausgabeformat (MIME-Type)
PROPERTYNAME	Kommaseparierte Liste der gewünschten Attribute
SRSNAME	Gewünschtes Koordinatenreferenzsystem
FEATUREID	Kommaseparierte Liste der Identifier der gewünschten Instanzen
FILTER	Filter in FE-Notation
BBOX	Kommaseparierte Liste der Eckkoordinaten der Bounding Box in CRS-Einheiten (minX, minY, maxX, maxY)

Tabelle 4: Parameter des WFS-GetFeature-Requests
Quelle: [Vretanos 2005-2]

Als Ausgabeformat für einen solchen Request muss mindestens GML (Geography Markup Language), ein XML-basiertes Format, das ebenfalls vom OGC zum Austausch raumbezogener Objekte spezifiziert wurde, als kleinster gemeinsamer Nenner unterstützt werden.

Die optionale Methode GetGmlObject dient dazu, einzelne Elemente eines solchen GML-Dokuments per XLink zu erhalten.

Die beiden übrigen Methoden LockFeature und Transaction werden nur von einem transaktionalen WFS, einem so genannten WFS-T (Web Feature Service – Transactional) unterstützt, also ein WFS, der auch schreibenden Zugriff auf seine Geodaten bietet. Da es sich bei HTTP um ein zustandsloses Protokoll handelt, muss durch Locking sichergestellt werden, dass ein Objekt zu einem bestimmten Zeitpunkt nur von einem Client bearbeitet werden kann. Mittels LockFeature kann ein Client eine solche Sperre für ein oder mehrere Features anfordern, bevor mittels Transaction die Änderung der Daten (Anlegen, Editieren oder Löschen) vorgenommen und die Sperre wieder aufgehoben werden kann.

4.2.5 OpenLS

Die Open Location Services (OpenLS)-Spezifikation umfasst die folgenden sieben positionsabhängigen Dienste (siehe [Blankenbach 2007] und [Mabrouk 2008]):

- **Directory Service** zum Auffinden eines spezifischen oder nächstgelegenen Ortes, Produktes oder Dienstes im Sinne eines Gelbe-Seiten-Dienstes
- **Gateway Service** zum Ermitteln der Position eines Handys innerhalb des Mobilfunknetzes
- **Location Utility Service** zur Geokodierung semantischer Ortsbezeichnungen wie Adressen in Geokoordinaten und die reverse Geokodierung von Geokoordinaten in semantische Positionsangaben
- **Presentation Service** zum Erstellen von Kartendarstellungen ausgehend von beliebigen Geodaten und abstrakten Datentypen
- **Route Service** zum Ermitteln von Routen und Navigationsinformationen zwischen einem Ausgangs- und Zielpunkt
- **Navigation Service** (Erweiterung des Route Service) zur Neuberechnung von Routen on-the-fly im Falle von Abstechern und Umleitungen
- **Tracking Service** zur Positionsverfolgung von Personen und Gütern

Der Datenaustausch zwischen diesen Diensten und den Clients erfolgt mittels ADTs (Abstract Data Types). Die ADTs sowie die Anfragen an die Dienste und deren Antworten werden in der XML-basierten Auszeichnungssprache XLS (XML for Location Services) kodiert.

Obwohl Version 1.0 bereits im Januar 2004 verabschiedet wurde und kein konkurrierender Standard in diesem Bereich existiert, führen die OpenLS bislang ein regelrechtes Schattendasein. Da bislang auch keine Open Source-Implementierungen verfügbar sind, wird auf eine vertiefende Betrachtung an dieser Stelle verzichtet.

4.3 OSGeo-Spezifikationen

Die OSGeo ist eine „unabhängige Non-Profit-Organisation zur Unterstützung von Open Source Software sowie deren Anwendergemeinschaft in der Geoinformationsverarbeitung“ [Jansen/Adams 2010]. Unter den OSGeo-Projekten befinden sich Desktop-GIS wie GRASS GIS und gvSIG, Web Mapping-Komponenten wie MapServer (siehe Kapitel 5.3.1) und OpenLayers (siehe Kapitel 5.5.1) sowie raumbezogene Bibliotheken wie GDAL/OGR (siehe Kapitel 5.9.1) und GeoTools. Ein weiterer Schwerpunkt neben der Entwicklung, Pflege und Verbreitung der Softwareprojekte ist die Förderung der freien Verfügbarkeit von öffentlichen Geodaten sowie die Kooperation mit Universitäten und Bildungseinreichtungen [Mitchell 2008]. Die Definition von Standards im Bereich der Geoinformation hingegen sieht die OSGeo nicht in ihrem Aufgabenbereich. Da von Seiten des OGC aber lange Zeit keine Bestrebungen bzgl. Kachelung und Caching von WMS-Requests zu erkennen waren, wurden die folgenden beiden inoffiziellen Empfehlungen entwickelt, die im Gegensatz zum noch jungen WMTS-Standard von wesentlich mehr Softwareprojekten implementiert wurden und daher noch oft eingesetzt werden.

4.3.1 WMS-C

Die WMS Tiling Client Recommendation[42] definiert mit WMS-C (Web Map Service - Cached) eine Erweiterung des WMS der Version 1.1.1, die das Caching von WMS-Anfragen durch die Einschränkung von GetMap-Requests auf vordefinierte Werte ermöglicht. Listing 4 zeigt die Erweiterung des GetCapabilities-Antwortdokuments um das TileSet-Element, das einen Kachelsatz repräsentiert. Innerhalb eines Kachelsatzes wird das Koordinatenreferenzsystem, der Wertebereich, die verfügbaren Auflösungen in Karteneinheiten pro Pixel, die Kachelbreite und -höhe, das Grafikformat sowie die Layer und Kartenstile spezifiziert. Anhand dieser Angaben kann ein Client dann gültige Werte für die GetMap-Anfragen ermitteln, an die noch der Parameter `tiled=true` angehängt werden muss. [OSGeo-Wiki 2010]

[42] http://wiki.osgeo.org/wiki/WMS_Tiling_Client_Recommendation

```
<WMT_MS_Capabilities>
  ...
  <VendorSpecificCapabilities>
    <TileSet>
      <SRS>EPSG:4326</SRS>
      <BoundingBox srs="EPSG:4326" minx="-180" miny="-90"
       maxx="180" maxy="90"/>
      <Resolutions>
        0.703125 0.3515625 0.17578125 0.087890625 0.04394531250
      </Resolutions>
      <Width>256</Width>
      <Height>256</Height>
      <Format>image/png</Format>
      <Layers>coastline</Layers>
      <Styles></Styles>
    </TileSet>
    <TileSet>
      ...
    </TileSet>
  </VendorSpecificCapabilities>
  ...
</WMT_MS_Capabilities>
```

Listing 4: WMS-C-Erweiterung der GetCapabilities-Antwort
Quelle: [OSGeo-Wiki 2010]

4.3.2 TMS

Die TMS (Tile Map Service)-Empfehlung[43] geht noch einen Schritt weiter und definiert eine eigene REST-basierte Schnittstelle für Kartenkacheln. Ausgehend von der XML-Wurzelressource, die alle Dienste unter dieser URL referenziert, kann ein Client die angebotenen TileMaps eines TMS ermitteln. Eine TileMap ist eine kartographisch vollständige Kartenrepräsentation, die genau ein Koordinatenreferenzsystem und ein Grafikformat unterstützt. Sie besteht aus mehreren TileSets, wobei jedes TileSet die Kacheln eines Maßstabs, also eine Zoomstufe, repräsentiert. Um die Kacheln verschiedener TMS übereinander legen zu können wurden drei Profile definiert, die die möglichen Auflösungen einschränken. Um nun eine bestimme Kachel anzufordern, muss an die Wurzel-URL, der Service, die TileMap, das TileSet sowie die x- und y-Koordinate und das Grafikformat der Kachel angehängt werden. Die folgende URL (vgl. [OSGeo-Wiki 2010]) liefert die Kachel mit der y-Koordinate `4` und der x-Koordinate `3` des TileSets `2` der TileMap `vmap0` des TMS `1.0.0`:

```
http://tms.osgeo.org/1.0.0/vmap0/2/3/4.jpg
```

[43] http://wiki.osgeo.org/wiki/Tile_Map_Service_Specification

5 Quelloffene Software

Die OSI (Open Source Initiative) definiert die folgenden neun Kriterien, denen die Lizenzbestimmungen quelloffener Software entsprechen muss [Ronneburg 2010]:

1. Die Lizenz darf niemanden in seinem Recht einschränken, die Software als Teil eines Software-Paketes, das Programme unterschiedlichen Ursprungs enthält, zu verschenken oder zu verkaufen.
2. Das Programm muss den Quellcode beinhalten.
3. Die Lizenz muss Veränderungen und Derivate zulassen.
4. Die Lizenz muss die Weitergabe von Software, die aus verändertem Quellcode entstanden ist, ausdrücklich erlauben.
5. Die Lizenz darf niemanden benachteiligen.
6. Die Lizenz darf niemanden daran hindern, das Programm in einem bestimmten Bereich einzusetzen.
7. Die Rechte an einem Programm müssen auf alle Personen übergehen, die diese Software erhalten, ohne dass für diese die Notwendigkeit bestünde, eine eigene, zusätzliche Lizenz zu erwerben.
8. Die Rechte an dem Programm dürfen nicht davon abhängig sein, ob das Programm Teil eines bestimmten Software-Paketes ist.
9. Die Lizenz darf keine Einschränkungen enthalten bezüglich anderer Software, die zusammen mit der lizenzierten Software weitergegeben wird.

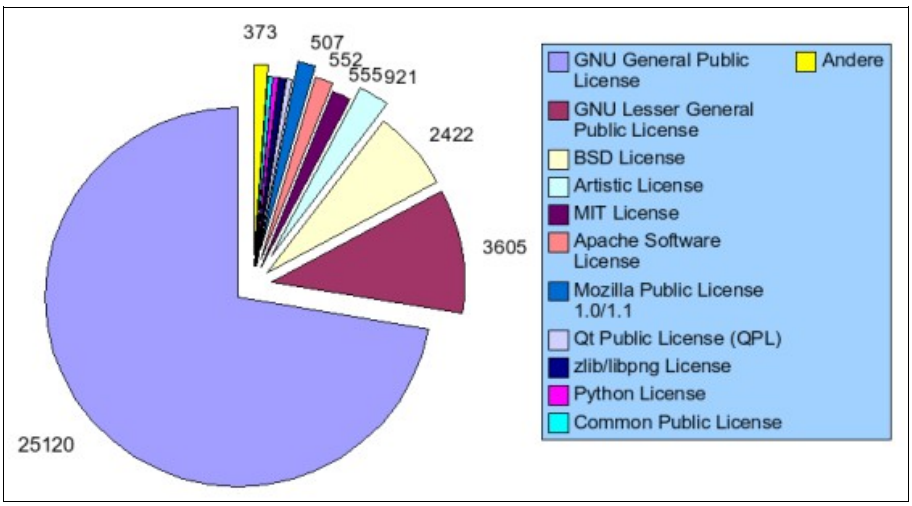

Abbildung 15: Verbreitung der OSI-Lizenzen bei SourceForge 2003
Quelle: http://openfacts.berlios.de/index.phtml?title=Open-Source-Lizenzen

Es gibt eine Vielzahl unterschiedlicher, von der OSI anerkannter Lizenzen. Abbildung 15 zeigt die Verbreitung dieser Lizenzen bei SourceForge, dem größten Hosting-Anbieter für quelloffene Software, im Jahr 2003. Im Folgenden werden die wichtigsten von der OSI anerkannten Lizenzen vorgestellt.

5.1 Lizenzen

5.1.1 GPL

Die GPL (GNU General Public License) ist die mit Abstand bekannteste und am meisten verbreitete Open Source-Lizenz. Sie erfüllt alle neun genannten Kriterien der OSI. Kerneigenschaft der GPL ist die Einschränkung, dass „Modifikationen an einer der GPL unterliegenden Software im Falle der Weiterverbreitung wiederum der GPL unterliegen müssen. Es ist also nicht möglich Teile von GPL-Software in kommerziellen (Closed Source) Produkten zu verwenden.“ [Fraunhofer-Gesellschaft 2006]

5.1.2 LGPL

Diese Einschränkung der GPL kennt die LGPL (Lesser GNU General Public License) nicht. Sie wird daher häufig für Software-Bibliotheken genutzt, die auch kommerziellen Produkten zur Verfügung gestellt werden sollen. Wird die Bibliothek allerdings nicht nur eingebunden sondern auch modifiziert, muss sie wiederum unter der LGPL lizensiert werden. [Fraunhofer-Gesellschaft 2006]

5.1.3 AGPL

Die AGPL (Affero GNU General Public License) ist eine Erweiterung der GPL. Sie fordert die Veröffentlichung des Quellcodes abgeleiteter Werke auch dann, wenn diese als Dienst über ein Netzwerk zur Verfügung gestellt werden[44]. Da Benutzer in diesem Fall nur auf die Ausgabe der Server-Software und nicht auf die ausführbaren Dateien zugreifen, liegt gemäß der GPL keine Weitergabe vor. Diese Lücke wird von der AGPL geschlossen.

[44] http://www.opensource.org/licenses/agpl-v3.html

5.1.4 BSD

Die BSD (Berkeley Software Distribution)-Lizenz „erlaubt die Verwendung und Weiterverbreitung der Software in Quell- und Binärform, mit oder ohne Veränderungen, solange der Copyright-Vermerk und der Lizenztext mitverbreitet werden“ [GRASSMUCK 2004]. Daneben enthält die ursprüngliche BSD-Lizenz noch eine Werbeklausel, die die Nennung der Universität und ihrer Kontributoren in Werbematerialien nach vorheriger schriftlicher Genehmigung vorschreibt. Diese Klausel ist wegen ihrer Inkompatibilität mit der GPL in der aktuellen Version der Lizenz jedoch nicht mehr vorhanden [FRAUNHOFER-GESELLSCHAFT 2006]. Im Gegensatz zu den GPL-artigen Lizenzen, besteht bei der BSD-Lizenz keine Einschränkung bezüglich der Lizenzierung abgeleiteter Werke.

5.1.5 MIT

Diese Lizenz ist nach dem Massachusetts Institute of Technology (MIT) benannt, aus dem sie stammt und ähnelt stark der BSD-Lizenz ohne Werbeklausel. Die einzigen Einschränkungen der MIT-Lizenz bestehen darin, dass der Urheberrechtsvermerk und der Lizenztext allen Kopien der Software beizulegen sind. Ansonsten darf unter der MIT-Lizenz stehende Software beliebig verwendet, kopiert, verändert, fusioniert, veröffentlicht, verbreitet, unterlizensiert und verkauft werden ohne dass der Quelltext veröffentlicht werden muss[45].

5.1.6 Apache

Die Apache-Lizenz stammt von der Apache Software Foundation und liegt mittlerweile in der Version 2.0[46] vor. Sie bietet die gleichen Freiheiten wie die BSD-Lizenz. Allerdings muss in jeder modifizierten Datei an auffälliger Stelle angegeben werden, dass die Datei geändert wurde. Außerdem ist es verboten, Markennamen der Apache Software Foundation für andere Zwecke als den Urheberrechtsvermerk zu verwenden.

[45] http://www.opensource.org/licenses/mit-license.php
[46] http://www.opensource.org/licenses/apache2.0.php

5.2 Räumliche Datenbanken

Datenbanken ermöglichen die persistente Speicherung und effiziente Abfrage von Daten. Am weitesten verbreitet sind relationale Datenbanken, bei denen die Daten logisch in Tabellen, den so genannten Relationen, abgelegt werden. Die Abfrage, Definition und Manipulation der Daten erfolgt mittels der Structured Query Language (SQL), die von allen gängigen Datenbanksystemen unterstützt wird. Für positionssensitive Anwendungen werden zusätzlich raumbezogene Erweiterungen benötigt, die neben der Möglichkeit Geometrien abzuspeichern und zu indizieren oftmals auch raumbezogene Analysen unterstützen. [Blankenbach 2007]

5.2.1 MySQL

MySQL[47] ist die am weitesten verbreitete Open Source-Datenbank und zeichnet sich durch eine hohe Abfragegeschwindigkeit aus. Eine Besonderheit der unter der GPL lizensierten Datenbank ist die Unterstützung verschiedener Storage Engines, die sich stark bezüglich ihres Transaktionsverhaltens und der Indexunterstützung unterscheiden. [Blankenbach 2007]

Seit der Version 4.1 verfügt MySQL über raumbezogene Erweiterungen, die allerdings nur von wenigen Storage Engines unterstützt werden. Die raumbezogenen Erweiterungen orientieren sich an dem OGC-Standard SFS (Simple Features SQL)[48] und unterstützen die Repräsentation von Geometrien im WKT (Well-Known Text)-Format und WKB (Well-Known Binary)-Format. Daneben unterstützt MySQL auch eine Vielzahl von Funktionen zur raumbezogenen Analyse wie Verschneidungen, Puffer usw.. Die Funktionen zur Überprüfung der räumlichen Beziehungen zwischen zwei Geometrien implementiert MySQL allerdings nicht gemäß des OGC-Standards, sondern lediglich unpräzise mittels der MBRs (Minimum Bounding Rectangles) der Geometrien. Um die raumbezogenen Analysen effizienter zu gestalten bietet MySQL räumliche Indizes in Form von R-Bäumen. [MySQL 2010]

[47] http://www.mysql.de
[48] http://www.opengeospatial.org/standards/sfs

5.2.2 PostgreSQL/PostGIS

PostgreSQL[49] ist eine objektrelationale Datenbank unter einer BSD-Lizenz. Mit PostGIS[50] steht für PostgreSQL eine raumbezogene Erweiterung zur Verfügung, die unter der GPL lizensiert ist. PostGIS ermöglicht die Erzeugung, Indizierung, Manipulation und Analyse räumlicher Daten. Im Gegensatz zu MySQL ist PostGIS vollständig konform zu dem oben genannten SFS-Standard und bietet zusätzlich Transaktionsintegrität für Geometrien. In der GIS-Welt kommt PostGIS deswegen sehr häufig auf der Datenbankseite zum Einsatz. Dementsprechend ist auch reichlich Dokumentation vorhanden. Zudem sind mittlerweile viele GIS-Programme in der Lage PostGIS-Daten zu lesen oder zu schreiben. Mit pgRouting (siehe Kapitel 5.7.1) existiert sogar eine PostGIS-Erweiterung für die Routenplanung. [Mitchell 2008]

5.3 Web Mapping Server

Web Mapping Server sind Server, deren Hauptfunktionalität „in der (dynamischen) Visualisierung von Geodaten in Form von kartenähnlichen Darstellungen besteht“ [Blankenbach 2007]. Die Geodaten für die verschiedenen Layer können dabei aus völlig unterschiedlichen Quellen kommen, weshalb Web Mapping Server eine Vielzahl an Dateiformaten und SQL-Dialekten für den Zugriff auf verschiedene Datenbanken unterstützen sollten. Ein Großteil der Web Mapping Server implementiert aus Gründen der Interoperabilität den OGC-Standard WMS. Viele Web Mapping Server beschränken sich jedoch nicht nur auf die Generierung von Karten sondern implementieren weitere OGC-Standards wie WFS und WCS (Web Coverage Service), dem WFS-Pendant für Rasterdaten.

5.3.1 MapServer

Der unter einer MIT-Lizenz stehende MapServer[51], mit dessen Entwicklung bereits 1994 an der University of Minnesota begonnen wurde, war lange Zeit konkurrenzlos im Open Source-Bereich und ist deshalb sehr weit verbreitet. In den meisten Fällen läuft der MapServer als CGI-Programm in einem Webserver (siehe Abbildung 16). Das CGI

[49] http://www.postgresql.org/
[50] http://postgis.refractions.net/
[51] http://mapserver.org/

(Common Gateway Interface) leitet entsprechende Anfragen an den MapServer weiter und liefert das generierte Kartenbild an den anfragenden Client zurück. Die Konfiguration der verschiedenen WMS-Layer erfolgt innerhalb der so genannten Mapdatei. Hier werden für jeden Layer die Datenquelle, Projektion, Klassen und deren Symbolik in einer MapServer-eigenen Syntax festgelegt. Die Daten können dabei datei-, datenbank- oder netzwerkbasiert sein. MapServer unterstützt durch die Einbindung der GDAL/OGR-Bibliothek (siehe Kapitel 5.9.1) eine Vielzahl unterschiedlicher Vektor- und Rasterformate. Neben dem WMS implementiert der MapServer auch die OGC-Standards WFS und WCS. [MITCHELL 2008]

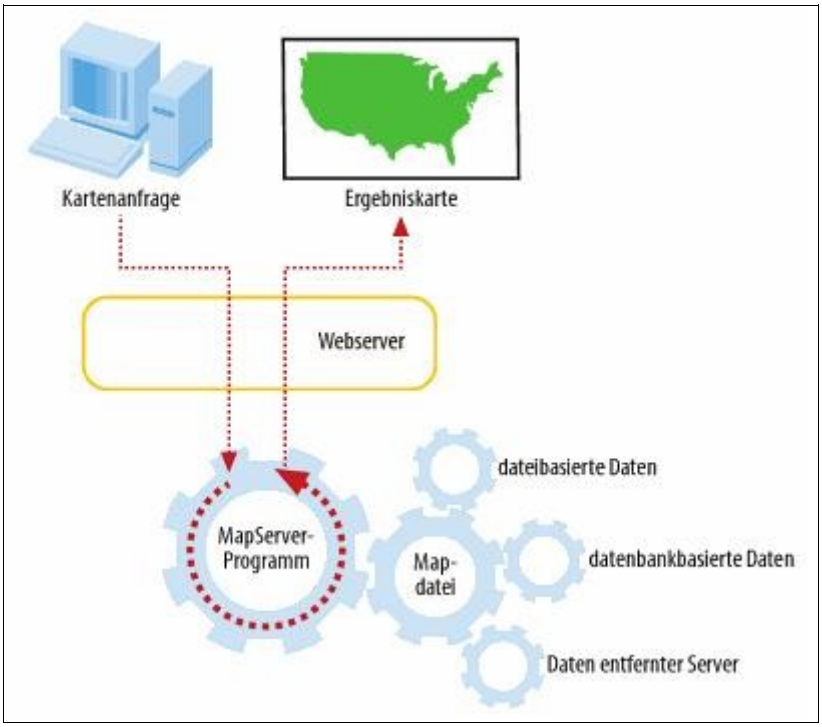

Abbildung 16: Funktionsweise des MapServer
Quelle: [Mitchell 2008]

5.3.2 GeoServer

Der unter der GPL lizensierte GeoServer[52] ist in Java geschrieben und benötigt deshalb eine entsprechende Laufzeitumgebung. Bei seiner Entwicklung „wird viel Wert auf die Einhaltung offener Standards gelegt, um die Interoperabilität mit anderen Anwendungen zu ermöglichen“ [JANSEN/ADAMS 2010]. Ebenso wie der MapServer implementiert auch der GeoServer die OGC-Standards WMS, WFS und WCS und kann

[52] http://geoserver.org/

zahlreiche Vektor- und Rasterformate lesen und schreiben. Darüber hinaus unterstützt er auch WFS-T, die Erweiterung des WFS um Transaktionen, die das Editieren der Vektordaten ermöglicht. Für die Integration mit MySQL wird allerdings eine Erweiterung benötigt, die derzeit nicht gewartet wird. Die Konfiguration der Dienste erfolgt wahlweise über eine benutzerfreundliche Weboberfläche (siehe Abbildung 17) oder über eine REST-Schnittstelle, die durch eine Erweiterung verfügbar ist. Für das Styling der einzelnen WMS-Layer wird im GeoServer der SLD-Standard (siehe Kapitel 4.2.2) verwendet. Um die mühsame Erstellung von SLD-Dokumenten zu vereinfachen, existiert jedoch eine Extension, die die grafische Erstellung von Stilen ermöglicht.

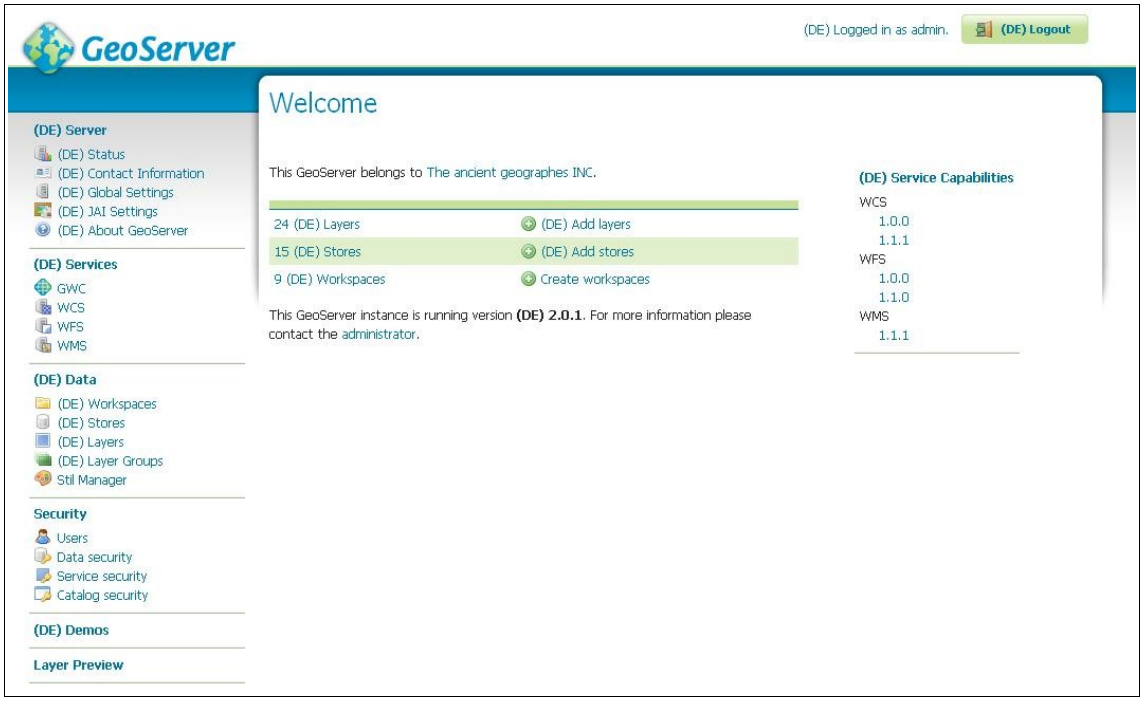

Abbildung 17: Konfigurationsoberfläche des GeoServer

5.4 Tile Cache Server

Tile Cache Server dienen dazu, die Performance der Web Mapping Server durch Kachelung und Caching zu steigern und werden daher zwischen Clients und Server geschaltet. Sie unterstützen Kachelungsstandards wie WMTS, WMS-C oder TMS und setzen solche Kachel-Requests in entsprechenden WMS-Anfragen an den Web Mapping Server um. Kommt eine Anfrage zum ersten Mal, wird sie einfach an den betreffenden WMS weitergeleitet und die zurückgelieferte Karte abgespeichert. Kommt

eine Anfrage ein zweites Mal kann der Tile Cache Server den Request ohne Hilfe des WMS beantworten, da die Kachel nicht erneut gerendert werden muss. Um die Performance-Vorteile nicht erst bei der zweiten Anfrage zu haben, bieten Tile Cache Server die Möglichkeit, den Cache automatisch zu befüllen (Seeding).

5.4.1 TileCache

Der pythonbasierte TileCache[53] ist der älteste Tile Cache Server und wird von MetaCarta unter einer BSD-Lizenz zur Verfügung gestellt. Er implementiert die Kachelungsstandards WMS-C und TMS und läuft z.B. als CGI-Programm in einem Webserver. Meist wird er im Zusammenspiel mit einem MapServer eingesetzt.

5.4.2 GeoWebCache

Der unter der GPL lizensierte GeoWebCache[54] ist ein Servlet und benötigt daher wie der GeoServer eine Java-Laufzeitumgebung. Seit der Version 1.7 ist er fest in den GeoServer integriert. Er kann jedoch auch standalone mit beliebigen anderen WMS-Servern eingesetzt werden. Im Gegensatz zum TileCache unterstützt der GeoWebCache neben WMS-C und TMS auch den neuen WMTS-Standard. Seit der Version 1.2.2 ist er zudem in der Lage beliebige WMS-Anfragen selbstständig zu beantworten, indem er die angeforderte Karte aus einzelnen Kacheln bzw. Kachelteilen zusammensetzt.

5.4.3 MapProxy

Wesentlich jünger als der TileCache und der GeoWebCache ist der pythonbasierte MapProxy[55], der von Omniscale im März 2010 unter der AGPL freigegeben wurde. Neben dem WMS unterstützt er mit dem TMS nur einen einzigen Kachelungsstandard. Dafür können neben WMS-Servern auch TMS-Server wie die bereits vorgestellten TileCache und GeoWebCache als Server eingebunden werden (siehe Abbildung 18). Zudem bietet er die Möglichkeit Karten on-the-fly in eine andere Projektion zu transformieren und eigene Layer aus verschiedenen Web Mapping Servern zusammenzusetzen.

[53] http://tilecache.org/
[54] http://geowebcache.org/
[55] http://mapproxy.org/

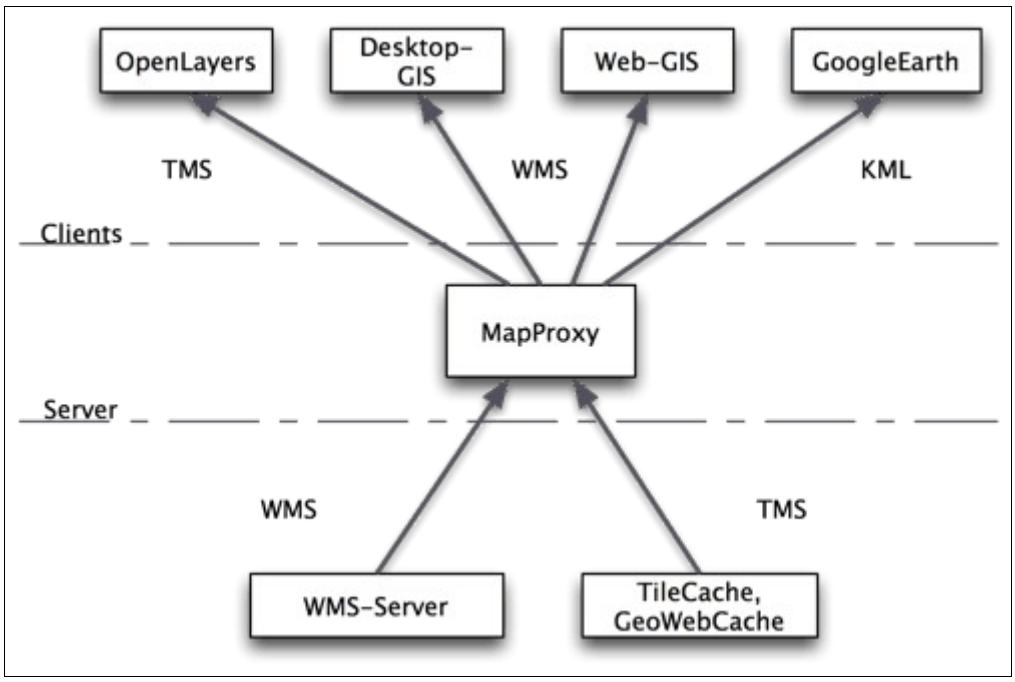

Abbildung 18: Einsatzmöglichkeiten des MapProxy
Quelle: http://mapproxy.org/

5.5 Web Mapping Clients

Web Mapping Clients erlauben es „Geodaten und Geodatendienste in einer mit Bedienelementen ausgestatteten Oberfläche in einem Internetbrowser anzuzeigen“ [Jansen/Adams 2010]. Die Bedienelemente ermöglichen beispielsweise die Navigation und das Zoomen in der dargestellten Karte, die Auswahl der Layer, die Anzeige von weiterführenden Informationen zu einzelnen Objekten, die Messung von Entfernungen und Flächen sowie oftmals auch die Bearbeitung der Geodaten. Aus Gründen der Interoperabilität sollten Web Mapping Clients möglichst viele offene Standards wie WMS, WMTS, WMS-C oder TMS beherrschen. Daneben werden häufig auch proprietäre Anbieter von Kartenkacheln wie Google Maps oder Bing Maps unterstützt. Auf diese Weise können die Datenquellen ohne viel Programmieraufwand gewechselt werden.

5.5.1 OpenLayers

OpenLayers[56] ist eine JavaScript-Bibliothek, die 2006 von MetaCarta unter einer BSD-Lizenz veröffentlicht wurde. Da bei der Entwicklung besonderer Wert auf die Trennung von Daten und Interaktionen gelegt wurde, können Datenquellen einfach ausgetauscht oder überlagert werden. Neben den offenen Standards WMS, SLD-WMS, WMTS, WMS-C, TMS, WFS und WFS-T werden auch zahlreiche proprietäre Formate (ESRI, Bing Maps, Google Maps, Yahoo Maps usw.) unterstützt. Vektordaten werden abhängig vom Browser mittels Canvas (HTML 5), SVG oder VML (Vector Markup Language) gerendert. Die Bibliothek stellt eine Vielzahl nützlicher Steuerelemente zum Erkunden und Bearbeiten von Karten zur Verfügung. Abbildung 19 zeigt einen Ausschnitt der Startseite von OpenStreetMap mit Steuerelementen zum Navigieren und Zoomen sowie zur Auswahl der Layer. Mittlerweile ist OpenLayers im Bereich der freien Web Mapping Clients beinahe konkurrenzlos. So entschlossen sich 2008 die Entwickler des freien Clients Mapbuilder, ihr Projekt zu Gunsten von OpenLayers aufzugeben. [JANSEN/ADAMS 2010]

Abbildung 19: OpenLayers im Einsatz
Quelle: http://www.openstreetmap.org/

[56] http://www.openlayers.org/

5.5.2 GeoExt

GeoExt[57] ist ein unter der BSD-Lizenz veröffentlichtes JavaScript Toolkit für Web Mapping-Anwendungen, „die vom Look & Feel sehr nahe an Desktop-Oberflächen herankommen“ [JANSEN/ADAMS 2010]. Es verknüpft die räumlichen Fähigkeiten von OpenLayers mit den Steuerelementen des JavaScript-Framework ExtJS und ermöglicht auf diese Weise komplexe Seitenlayouts mit Menüs, klappbaren Bäumen, sortierbaren Listen, verschiebbaren Fenstern und Legenden. Dabei ist es keinesfalls als Konkurrenz zu OpenLayers zu betrachten. Während das nackte OpenLayers für einfache Anwendungen wie klassische Routenplaner vollkommen ausreicht, kommt GeoExt bei aufwendigeren Applikationen mit vielen Funktionen (Rich Clients) zum Einsatz. Abbildung 20 zeigt eine Beispielanwendung von GeoExt bestehend aus einer Vielzahl hierarchisch organisierter, thematischer Overlays in einer klappbaren Baumstruktur, einer beliebig sortierbaren Attributtabelle und einer Klassifikationslegende.

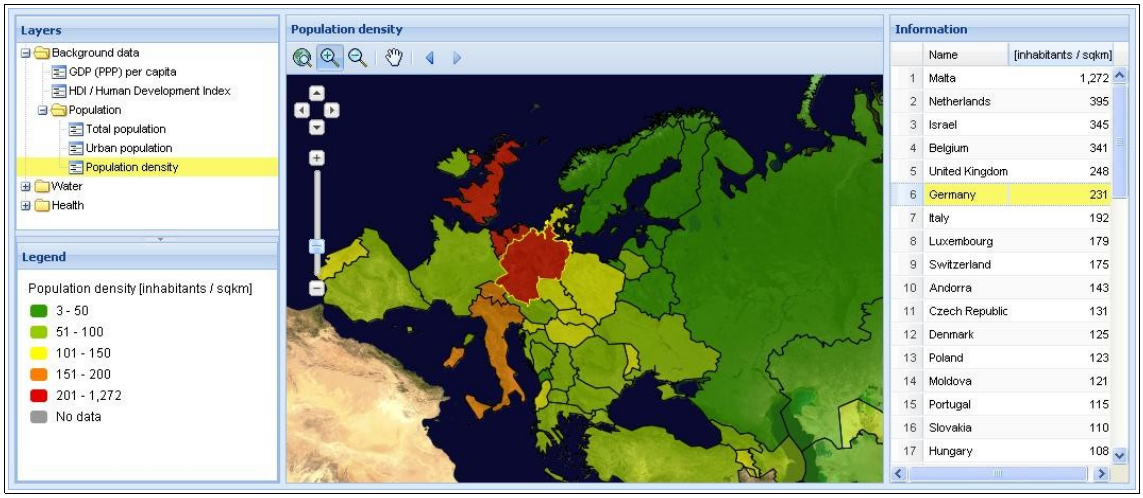

Abbildung 20: GeoExt im Einsatz
Quelle: http://www.waterandhealth.eu/

5.6 Mobile Web Frameworks

Da die Standardisierung im Bereich der Mobiltelefone noch nicht weit genug fortgeschritten ist (siehe Kapitel 4.1.1 und 4.1.2), werden Mobile Web Frameworks benötigt, um plattformübergreifende Anwendungen gemäß dem Credo „Write once, run anywhere“ entwickeln zu können. Sie ermöglichen es, native Applikationen mittels der

[57] http://geoext.org/

Web-Techniken HTML, CSS und JavaScript zu erstellen und dabei auch auf Gerätefunktionen wie GPS-Empfänger, Beschleunigungsensoren oder das Adressbuch zuzugreifen. Durch geeignete Konvertierung, Kompilierung oder Verpackung in Web-Container entstehen daraus schließlich plattformspezifische Anwendungen.

5.6.1 PhoneGap

Das MIT-lizensierte PhoneGap[58] unterstützt mit Android, Blackberry, iOS, Symbian und WebOS die meisten mobilen Plattformen und verfügt wohl über den höchsten Bekanntheitsgrad im Bereich der Mobile Web Frameworks. Nach anfänglichen Problemen werden mit PhoneGap entwickelte Anwendungen mittlerweile im App Store von Apple akzeptiert. PhoneGap sieht sich selbst nur als vorübergehende Lösung bis zur Standardisierung der mobilen Browser: „An express goal of the PhoneGap project is for the project to not exist“ [Sachse 2010]. Für jede der unterstützen Plattformen stellt PhoneGap ein Template zur Verfügung, das mit den Web-Dokumenten befüllt wird. Daraus generiert das Framework nativen Code, „allerdings größtenteils nur für die Komponente, die die eigenetliche Webanwendung zeigt“ [Spiering/Haiges 2010]. Hierfür werden allerdings auch die jeweiligen plattformspezifischen SDKs oder IDEs (Integrated Devlopment Environments) benötigt. Das Javascript-API, das Zugriff auf die Gerätefunktionen ermöglicht, orientiert sich sehr stark den den jeweiligen Drafts des W3C (siehe Kapitel 4.1.2). Mittlerweile wurde PhoneGap sogar als Erweiterung in die Symbian WRT integriert.

5.6.2 Appcelerator Titanium

Einen ähnlichen Ansatz verfolgt auch Appcelerator mit seinem unter einer Apache-Lizenz veröffentlichten Titanium Mobile Framework[59]. Im Gegensatz zu PhoneGap verfügt Titanium über eine eigene graphische Benutzeroberfläche, die unabhängig von der Zielplattform ist. Allerdings unterstützt Titanium derzeit nur Android und iOS. Eine Unterstützung für Blackberry wurde jedoch bereits angekündigt. Das JavaScript-API orientiert sich nicht an den W3C-Standards, ist aber deutlich umfangreicher als das PhoneGap-Äquivalent und ermöglicht beispielsweise die Erstellung von nativen Bedienelementen via JavaScript, so dass die resultierenden Anwendungen das

[58] http://www.phonegap.com/
[59] http://www.appcelerator.com/products/titanium-mobile-application-development/

Look & Feel einer nativen Anwendung aufweisen. Im Gegensatz zu PhoneGap generiert Titanium „wesentlich mehr nativen Code und produziert Anwendungen, die zu großen Teilen auch wirklich native Anwendungen sind“ [Spiering/Haiges 2010].

5.7 Routing Server

Es gibt eine Vielzahl an Open Source-Projekten, die sich mit Routenberechnung beschäftigen, aber keines implementiert den Routing Service des OpenLS-Standards. Einzige Ausnahme ist der bereits erwähnte OpenRouteService, von dem allerdings nur der Quellcode der Routing Engine, eine modifizierte Version der Java-Bibliothek GeoTools[60], veröffentlicht wurde. Von Georepublic wurde für 2010 mit WebSI-Cola[61] ein vielversprechendes Java-Framework angekündigt, das diese Lücke schließen könnte. Leider ist dieses Projekt bislang noch nicht unter einer Open Source-Lizenz freigegeben. Es bleibt also derzeit nur die Möglichkeit, auf andere Routing Server zurückzugreifen, die ihre eigene Schnittstelle definieren und somit nicht problemlos ausgetauscht werden können.

5.7.1 pgRouting

pgRouting[62] ist eine Routing-Erweiterung von PostGIS, die unter der GPL veröffentlicht wurde. Sie ermöglicht die Erstellung von Topologien und implementiert mit Dijkstra, A* und Shooting Star drei verschiedene Algorithmen zur Berechnung des kürzesten Weges. Im Gegensatz zum klassischen Dijkstra-Algorithmus verwenden der A* und der Shooting Star eine Schätzfunktion (Heuristik), um die Laufzeit der Routenberechnung zu verkürzen. Der Shooting Star beachtet zudem Abbiegevorschriften. Daneben kann pgRouting auch das Problem des Handlungsreisenden lösen und Isodistanzlinien berechnen.

Mit dem Web Routing Service[63] existiert bereits ein REST-basierter Web Service unter GPL-Lizenz, der Routing-Anfragen über HTTP in SQL-Statements für pgRouting umsetzt. Der in Java geschriebene Service unterstützt eine Vielzahl geographischer Ausgabeformate wie GML, KML (Keyhole Markup Language), GeoJSON, GPX

[60] http://geotools.org/
[61] http://georepublic.de/de/technology/websi-cola/
[62] http://pgrouting.postlbs.org/
[63] http://pgrouting.postlbs.org/wiki/tools/WebRoutingService

(GPS Exchange Format) und XLS. Einziges unterstütztes Eingabeformat ist allerdings JSON.

5.7.2 Gosmore

Gosmore[64] ist eine unter einer BSD-Lizenz veröffentlichte Routing-Anwendung für OpenStreetMap-Daten, die die Berechnung des kürzesten und schnellsten Weges für Autos, Fahrräder und Fußgänger samt Abbiegebeschränkungen unterstützt und u.a. auf Windows- und Unix-Plattformen lauffähig ist. Die OpenStreetMap-Daten müssen hierfür als pak-Datei, einem Gosmore-eigenen binären Format, vorliegen. Der Download und die Konvertierung der Daten ist jedoch problemlos über die Anwendung möglich. Gosmore verfügt zwar über eine graphische Benutzerschnittstelle, kann jedoch auch per CGI angesteuert werden und ist damit für den Einsatz als Routing Engine in Webanwendungen prädestiniert.

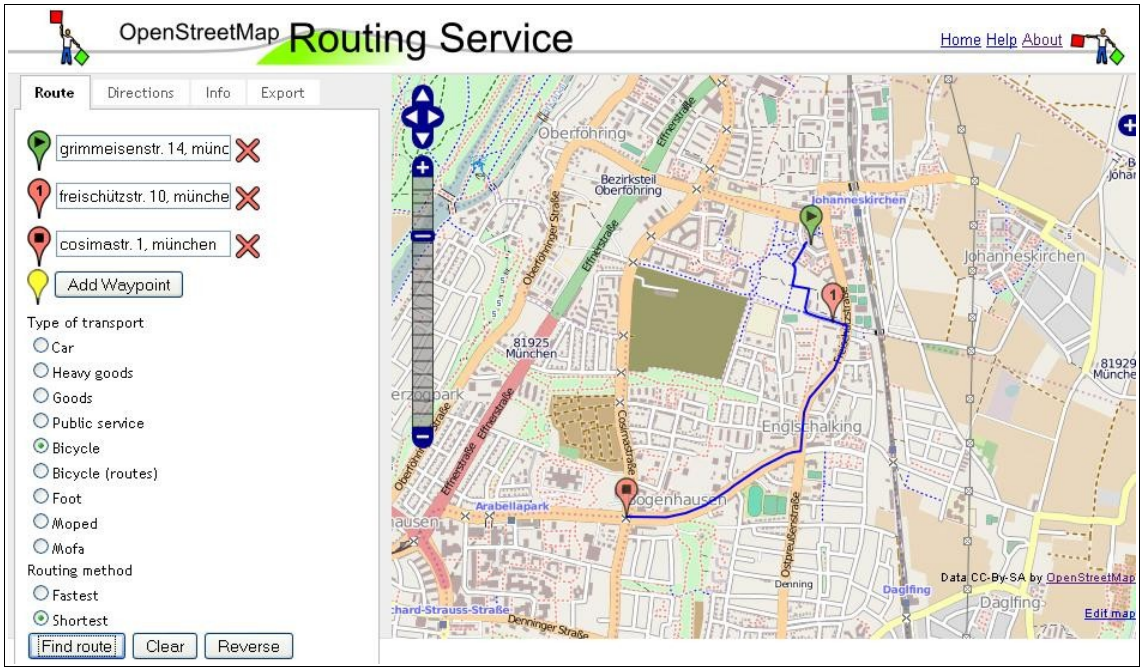

Abbildung 21: YOURS-Webseite
Quelle: http://yournavigation.org/

Dies macht sich der unter einer BSD-Lizenz stehende Routing Service YOURS[65] zunutze, der eine graphische Web-Schnittstelle für Gosmore bildet. Abbildung 21 zeigt

[64] http://wiki.openstreetmap.org/wiki/Gosmore
[65] http://wiki.openstreetmap.org/wiki/YOURS

einen Screenshot der YOURS-Webseite. Die Geokodierung der Nutzereingaben erfolgt mittels Name Finder (siehe Kapitel 3.1.7), während die Kartenkacheln und die vektorbasierte Route mittels OpenLayers dargestellt wird. Neben der Routenberechnung ermöglicht YOURS auch den Export der Routen in das weit verbreitete GPX-Format.

Neben der Webseite stellt YOURS auch ein API für Routing- und Export-Anfragen zur Verfügung. Das Routing API ist unter der folgenden URL erreichbar:

```
http://www.yournavigation.org/api/1.0/gosmore.php?
```

Tabelle 5 listet die möglichen Parameter des Routing-Requests sowie deren möglichen Werte und Bedeutung auf:

Parameter	Mögliche Werte	Bedeutung
flat	-90 bis +90	Breitengrad des Ausgangspunktes
flon	-180 bis +180	Längengrad des Ausgangspunktes
tlat	-90 bis +90	Breitengrad des Zielpunktes
tlon	-180 bis +180	Längengrad des Zielpunktes
v	motorcar, bicycle oder foot	Fortbewegungsmittel
fast	0 oder 1	Kürzeste (0) oder schnellste (1) Route
layer	mapnik oder cn	Normales Netzwerk (mapnik) oder Fahrrad-wegenetz (cn)
format	kml oder geojson	Ausgabeformat

Tabelle 5: Parameter für Routing mittels YOURS
Quelle: http://wiki.openstreetmap.org/wiki/YOURS

5.8 Geocoder

Auch für den Location Utility Service der OpenLS-Spezifikation gibt es bislang keine Open Source-Implementierung. Deshalb muss hier wie auch schon im Bereich der Routing Server auf Implementierungen zurückgegriffen werden, die eigene Schnittstellen definieren.

Mit Name Finder und Nominatim (siehe Kapitel 3.1.7) wurden schon zwei Geocoder vorgestellt, deren Quellcode im OpenStreetMap-Repository[66] zur Verfügung steht. Während Name Finder eigene Skripte für den Import der OpenStreetMap-Daten in eine MySQL-Datenbank mitbringt, verwendet Nominatim osm2pgsql (siehe Kapitel 5.9.2)

[66] http://svn.openstreetmap.org/

für den Import in eine PostGIS-Datenbank. In beiden Fällen übernehmen PHP-Skripte die Umsetzung der HTTP-Requests in SQL-Anfragen.

5.8.1 Gisgraphy

Gisgraphy[67] ist ein unter der LGPL lizensiertes Java-Framework, das Geocoding über REST-basierte Web Services ermöglicht. Daneben bietet Gisgraphy auch Reverse Geocoding sowie eine Volltextsuche nach Orten. Während Geocoding und Reverse Geocoding auf Basis der OpenStreetMap-Daten erfolgen, werden bei der Volltextsuche nur die GeoNames-Daten (siehe Kapitel 3.2) einbezogen. Über den Importer können die benötigten OpenStreetMap- und/oder GeoNames-Daten komfortabel heruntergeladen und in eine PostGIS-Datenbank importiert werden. Als Ausgabeformat werden neben XML auch JSON und GeoRSS unterstützt. Die Administration des Gisgraphy Servers erfolgt über eine komfortable Webschnittstelle. Für den nichtkommerziellen Einsatz besteht die Möglichkeit statt einer eigenen Installation die Web Services unter services.gisgraphy.com zu nutzen.

Bei genauerer Betrachtung von Gisgraphy stellte sich allerdings heraus, dass der Geocoding Service derzeit eine Koordinatenangabe erfordert, die vorgibt in welchem Bereich nach der angegebenen Straße gesucht werden soll. Es muss demnach vorher eine Volltextsuche nach der gewünschten Stadt durchgeführt werden, bevor in deren Umkreis dann die gewünschte Straße georeferenziert werden kann. Der Entwickler arbeitet jedoch bereits an der Implementierung eines Adressparsers, der eine direkte Geokodierung von Adressen ermöglichen wird.

Der Reverse Geocoding Service ist unter der folgenden URL verfügbar:

```
http://services.gisgraphy.com/street/search?
```

Tabelle 6 listet die wichtigsten Parameter des Reverse Geocoding-Requests sowie deren möglichen Werte und Bedeutung auf:

[67] http://www.gisgraphy.com/

Parameter	Mögliche Werte	Bedeutung
lat	-90 bis +90	Geographische Breite in Dezimalgrad
lng	-180 bis +180	Geographische Länge in Dezimalgrad
radius	*beliebig*	Suchradius in Metern
format	XML, JSON, ATOM oder GEORSS	Ausgabeformat
streettype	*siehe Dokumentation*	Straßenart
oneway	true oder false	Einbahnstraße

Tabelle 6: Parameter für Reverse Geocoding mittels Gisgraphy
Quelle: http://www.gisgraphy.com/documentation/index.htm

5.9 Konverter

5.9.1 GDAL/OGR

Die Geospatial Data Abstraction Library (GDAL)[68] ist eine unter der MIT-Lizenz stehende Programmbibliothek, die über 60 verschiedene räumliche Rasterdatenformate unterstützt und zwischen diesen konvertieren kann. Daneben sind noch verschiedene Hilfsprogramme für die Kommandozeile enthalten, die es ermöglichen ohne eigenen Programmieraufwand mit räumlichen Rasterdaten zu arbeiten. Beispielsweise können mit gdal_countour[69] Isohypsen und mit gdaldem[70] Schummerungen aus einem Höhendaten-Raster erzeugt werden. Letzteres und ein Treiber für das binäre hgt-Format, in dem die SRTM-Daten (siehe Kapitel 3.3.1) vorliegen, sind allerdings erst seit der Version 1.7 integriert.

Die OGR Simple Feature Library[71] ist das vektorielle Äquivalent zu GDAL und unterstützt über 25 verschiedene Vektorformate, die allerdings teilweise nur gelesen und nicht geschrieben werden können. Die Vektorformate umfassen dabei nicht nur Dateiformate, sondern auch räumliche Datenbanken wie MySQL und PostGIS. [Mitchell 2008]

[68] http://www.gdal.org/
[69] http://www.gdal.org/gdal_contour.html
[70] http://www.gdal.org/gdaldem.html
[71] http://www.gdal.org/ogr/index.html

5.9.2 osm2pgsql

osm2pgsql[72] ist ein unter der GPL lizensiertes Tool, um die Daten aus einer OSM-XML-Datei (siehe Kapitel 3.1.4) in eine PostGIS-Datenbank zu importieren. Da osm2pgsql speziell für das Rendering mit Mapnik (siehe Kapitel 3.1.6) entwickelt wurde, werden auch nur für das Rendern benötigte Daten importiert, die mit Hilfe einer style-Datei angepasst werden können. Die Nodes, Ways und Relationen werden dabei in Punkte, Polylinien und Polygone konvertiert. Nach dem Importvorgang enthält die angegebene Datenbank die vier Tabellen planet_osm_point, planet_osm_line, planet_osm_polygon und planet_osm_roads, wobei letztere im Vergleich zu planet_osm_line nur die größeren Straßen enthält.

osm2pgsql kann die OpenStreetMap-Daten auch speziell für den Geocoding-Einsatz importieren. Hierfür muss die Ausgabeoption gazetteer gesetzt werden.

Im so genannten Slim-Mode, werden alle temporären Daten (Nodes, Ways und Relationen) statt im Arbeitsspeicher in der Datenbank in den Tabellen planet_osm_nodes, planet_osm_ways und planet_osm_rels abgelegt. Neben den geringeren RAM-Anforderungen, hat dieser Modus den Vorteil, dass die Datenbank mittels der OsmChange-Dateien (siehe Kapitel 3.1.4) ohne großen Aufwand aktuell gehalten werden kann. Hierfür muss allerdings vor dem initialen Import das intarray-Modul von PostgreSQL aktiviert werden.

5.9.3 osm2pgrouting

osm2pgrouting[73] ist ein unter der GPL lizensiertes Tool, um OpenStreetMap-Daten im OSM-XML-Format in eine pgRouting-Datenbank zu importieren. Im Gegensatz zu osm2pgsql erzeugt es einen routingfähigen Graphen. Mittels einer XML-Konfigurationsdatei kann vorgegeben werden, welche Tags beim Importvorgang berücksichtigt werden sollen.

Abbildung 22 erläutert die Erzeugung eines topologisch korrekten Graphen mit Knoten und Kanten, die die Kreuzungen und Straßen bzw. Wege des Verkehrsnetzes repräsentieren. Die beiden Ways müssen hierfür jeweils an Knoten 3 geteilt werden, während überflüssige Nodes, die keinen Knoten im Sinne einer Kreuzung bilden (2, 4, 44 und 45), entfernt werden.

[72] http://wiki.openstreetmap.org/wiki/Osm2pgsql
[73] http://pgrouting.postlbs.org/wiki/tools/osm2pgrouting

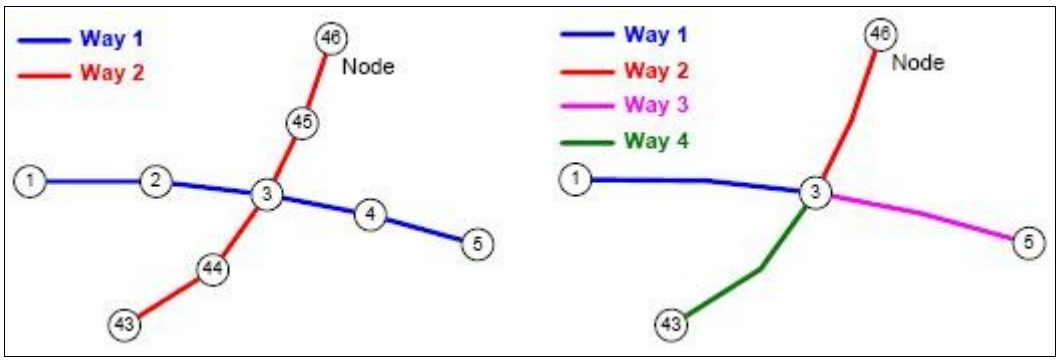

Abbildung 22: Erzeugung eines routingfähigen Graphen
Quelle: [Neis 2008]

5.9.4 OpenStreetMap-in-a-Box

OpenStreetMap-in-a-Box[74] ist einfach zu installierender WMS/WFS-Server auf OpenStreetMap-Basis, der von Studenten der Hochschule für Technik in Rapperswil entwickelt wurde. Abbildung 23 zeigt die Komponenten von OpenStreetMap-in-a-box und die Schnittstellen der Software. Herzstück des Projekts ist osm2gis, das den Import von OpenStreetMap-Daten in eine PostGIS-Datenbank ermöglicht. Im Gegensatz zu osm2pgsql kann hier das Datenbankschema und das Mapping der OpenStreetMap-Entitäten auf dieses mittels einer XML-Konfigurationsdatei frei definiert werden. osm2gis unterstützt auch vollautomatische tägliche, stündliche oder minütliche Updates der Datenbank auf Basis der OsmChange-Dateien. Das Rendern der auf diese Weise importierten Geodaten übernimmt ein GeoServer mit integriertem GeoWebCache, der von OpenStreetMap-in-a-Box installiert und konfiguriert wird. Voraussetzung hierfür ist eine Installation eines PostgreSQL/PostGIS-Servers und des Servlet-Containers Apache Tomcat[75]. OpenStreetMap-in-a-Box bringt neben einem vordefinierten Datenbankschema und passenden SLD-Styles zwei Webanwendungen mit, die ebenfalls im Servlet-Container installiert werden und die vom GeoServer gerenderte OpenStreetMap-Daten mittels OpenLayers darstellen.

[74] http://dev.ifs.hsr.ch/osminabox/
[75] http://tomcat.apache.org/

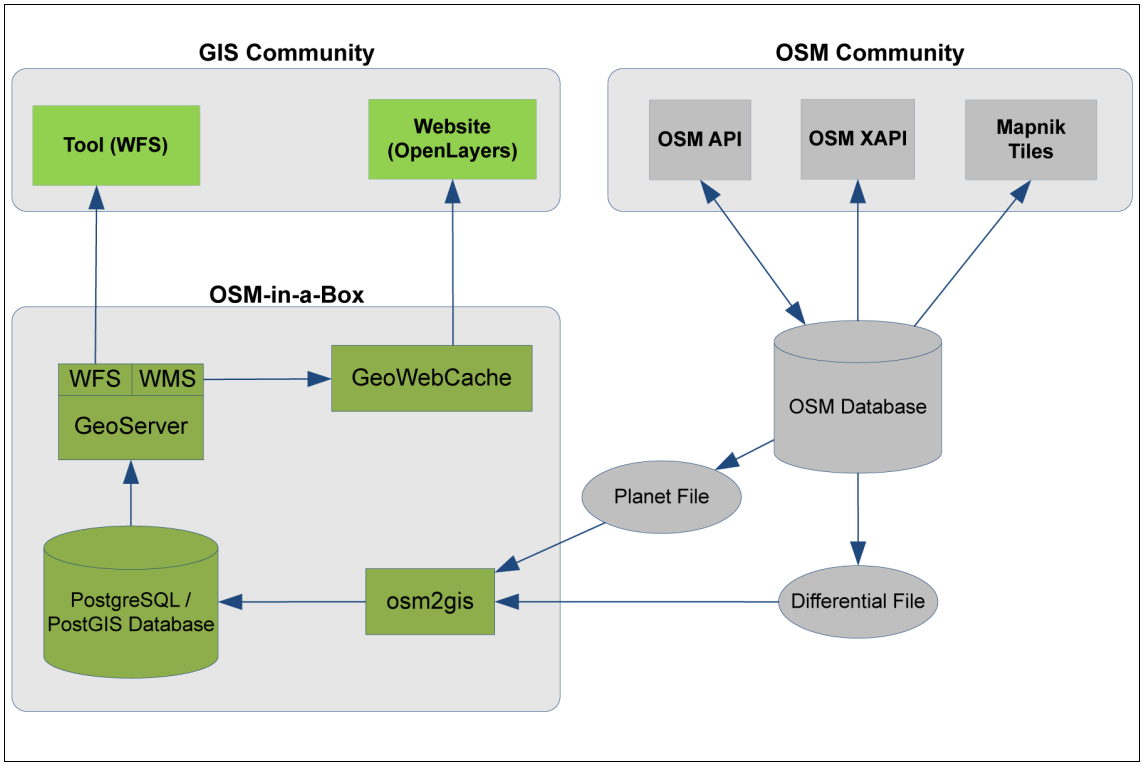

Abbildung 23: Schnittstellen von OpenStreetMap-in-a-Box
Quelle: http://dev.ifs.hsr.ch/osminabox/browser

5.9.5 mapnik2geotools

mapnik2geotools[76] ist ein in Scala geschriebener Konverter, der die Verwendung von Mapnik-Styles in Projekten wie dem GeoServer, die auf der Geotools-Bibliothek aufbauen, ermöglicht. Da Mapnik gerade im OpenStreetMap-Bereich sehr stark verbreitet ist und viele Map-Dateien mit kartographisch hochwertigen Styles (z.B. von MapQuest[77]) frei verfügbar sind, kann auf diese Weise die zeitaufwendige Erstellung von eigenen SLD-Styles von Grund auf umgangen werden. mapnik2geotools erzeugt neben den benötigten SLD-Dateien auch ein SQL-Skript, dass die benötigten Views in der PostGIS-Datenbank erzeugt. Dieses Skript setzt den vorherigen Import von OpenStreetMap-Daten mittels osm2pgsql voraus. Daneben übernimmt das Tool auch die Konfiguration der einzelnen Layer im GeoServer über dessen REST-Schnittstelle.

[76] https://github.com/dwins/mapnik2geotools
[77] https://github.com/MapQuest/MapQuest-Mapnik-Style

6 Konzeption

Die vorgestellten freien Geodaten, offenen Schnittstellen und quelloffenen Software-Komponenten sollen nun zu einer plattformübergreifenden Anwendung kombiniert werden. Hauptbestandteil einer solchen Mapping App ist eine Karte, die sich positionsabhängig verschiebt und mit verschiedenen thematischen Daten in Vektorform überlagert werden kann, zu denen der User detaillierte Informationen abrufen kann. Des Weiteren soll die Anwendung Routing zu POIs beherrschen, wobei die gewünschten Ziele über eine ortsabhängige Suchfunktion ermittelbar sein sollen.

6.1 Architektur

Abbildung 24 zeigt die grobe Architektur der Anwendung: Auf der rechten Seite sind die benötigten Daten aufgelistet. Dazu gehören die anwendungsneutralen Geobasis-Daten für die Hintergrundkarte, das Routing und das Geocoding, Höhendaten für Isohypsen und Schummerung sowie die anwendungsspezifischen Fachdaten. Der Web Mapping Server erzeugt aus diesen Daten die verschiedenen Kartenlayer in Form von Raster- oder Vektorgrafiken, während der Routing Server Routingrequests beantwortet und der Geocoding Server für die Durchführung der Suchen nach POIs in den Geobasis- und Fachdaten zuständig ist. Damit die Karten nicht für jede Anfrage gerendert werden müssen, wird dem Web Mapping Server aus Performance-Gründen ein Tile Cache Server vorgeschaltet. Alle genannten Server können auf einer physischen Maschine laufen oder bei Bedarf (z.B. hoher Last) auf mehrere Maschinen verteilt werden. Hierbei ist jedoch zu beachten, dass unter Umständen ein Proxy-Server davor geschaltet werden muss, um der Same Origin Policy gerecht zu werden. Auf der Client-Seite wird entweder ein Thin Client mit den nötigsten Funktionen oder ein komfortabler Rich Client mit Zusatzfunktionen wie beispielsweise der Editierung von Fachdaten ausgeliefert. Es handelt sich also um eine klassische 3-Schichten-Architektur bestehend aus Datenhaltungsschicht, Logikschicht und Präsentationsschicht.

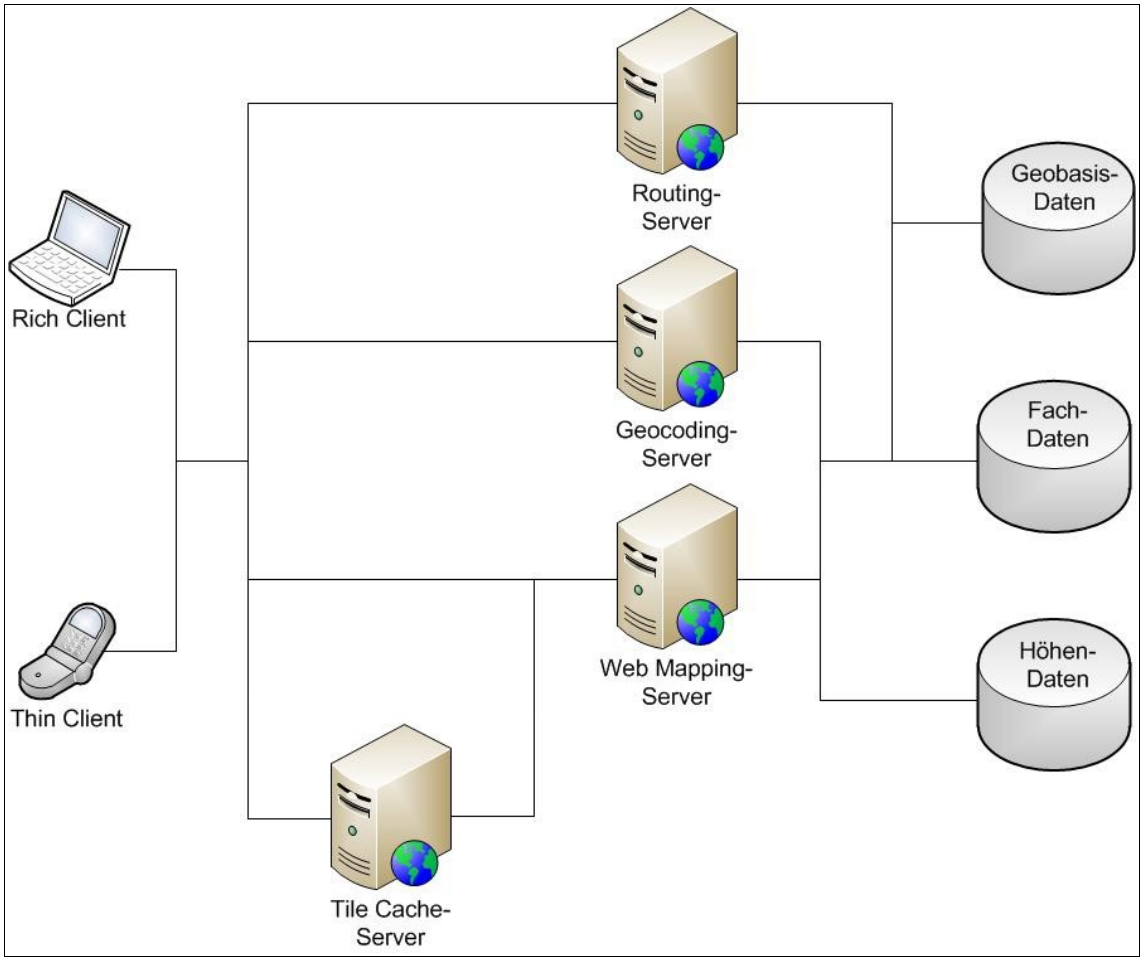

Abbildung 24: Architektur

6.2 Datenhaltungsschicht

6.2.1 Datenquellen

Die Geobasis-Daten liefert das OpenStreetMap-Projekt. Sie können über das bereits in Kapitel 3.1.5 beschriebene XAPI bis zu einer Größe von 10 Quadrat-Grad im OSM-XML-Format heruntergeladen werden. Auf den Einsatz von GeoNames-Daten wird auf Grund des fehlenden Mehrwerts verzichtet. Für die Quelle der Höhen-Daten fällt die Wahl lizenzbedingt auf SRTM statt das höher aufgelöste ASTER. Da SRTM-Daten schon wesentlich länger verfügbar sind als ASTER-Daten, sind auch wesentlich mehr Tools für die Konvertierung in andere Formate vorhanden. Die anwendungsspezifischen Fachdaten liegen im Regelfall bereits vor und müssen unter Umständen geeignet migriert bzw. konvertiert werden.

6.2.2 Datenbank-Server

Als Datenbank-Server ist PostgreSQL MySQL aufgrund seiner bewährten räumlichen Erweiterung PostGIS vorzuziehen. Diese Kombination bietet wesentlich mehr räumliche Funktionalitäten als MySQL und hält sich zudem konsequent an den SFS-Standard des OGC. Hinzu kommt die wesentlich breitere Tool-Unterstützung und Dokumentation im GIS-Bereich. Sollten die anwendungsspezifischen Fachdaten jedoch bereits in einer MySQL-Datenbank gehalten werden, kann es durchaus sinnvoll sein MySQL einzusetzen. Denkbar ist in diesem Fall aber auch, die Fachdaten in MySQL beizubehalten und die Geobasis- und Höhen-Daten in PostGIS zu importieren.

6.2.3 Datenimport

Die OpenStreetMap- und SRTM-Daten müssen zunächst in verschiedene PostgreSQL-Datenbanken importiert bzw. in geeignete Dateiformate konvertiert werden. Dies hängt jedoch davon ab, welche Web Mapping-, Routing- und Geocoding-Software letztendlich eingesetzt wird. Im Folgenden wird nur auf den Import für die in den folgenden Kapiteln vorgeschlagenen Tools eingegangen.

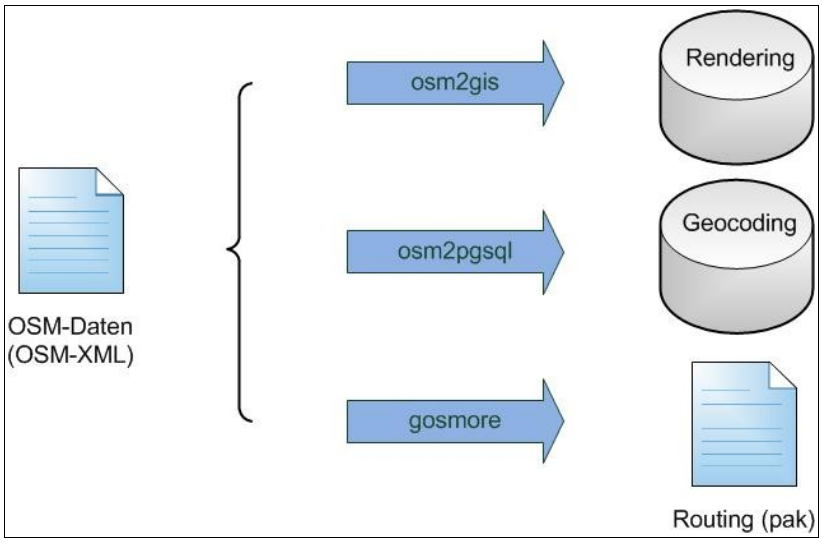

Abbildung 25: Import der OpenStreetMap-Daten

Abbildung 25 zeigt den Import der OpenStreetMap-Daten, die nach dem Entpacken in Form einer OSM-XML-Datei vorliegen. Da die OpenStreetMap-Daten nicht nur zum Rendern der Hintergrundkarte benötigt werden, sondern auch als Basis für die

Routenberechnung und das Geocoding dienen, müssen sie für jeden dieser Zwecke separat importiert bzw. konvertiert werden. Für das Rendering erfolgt der Import mit osm2gis, das im Vergleich mit osm2pgsql deutlich flexibler ist und die Möglichkeit vollautomatischer Updates bietet. Letzteres ist mit der Ausgabeoption gazetteer jedoch prädestiniert für den Import in die Geocoding-Datenbank. Für das Routing mit Gosmore müssen die OpenStreetMap-Daten hingegen in das binäre pak-Format konvertiert werden, das speziell für die Routenberechnung auf OpenStreetMap-Daten konzipiert wurde.

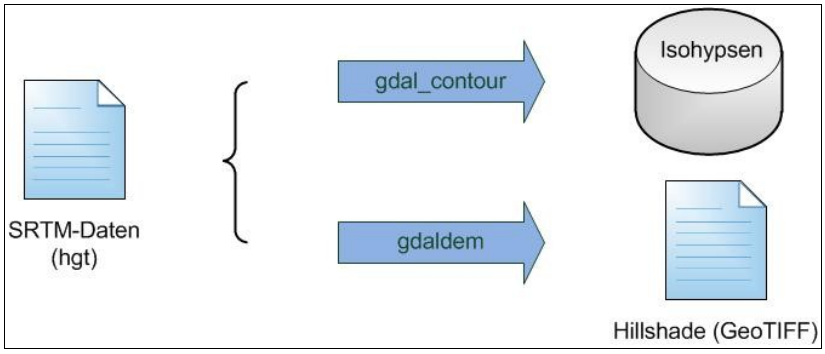

Abbildung 26: Konvertierung der SRTM-Daten

Abbildung 26 erläutert, wie mit den im hgt-Format vorliegenden SRTM-Daten verfahren wird um die Isohypsen und die Schummerung in einer für Web Mapping-Server geeigneten Form zu erzeugen. Mittels gdal-contour aus der GDAL/OGR-Bibliothek können die Isohypsen in einem Schritt erzeugt und in eine PostgreSQL-Datenbank importiert werden. Die Erzeugung der Schummerung im Rasterformat GeoTIFF erfolgt hingegen mit gdaldem aus der gleichen Bibliothek.

6.3 Logikschicht

6.3.1 Web Mapping

Als Web Mapping Server empfiehlt sich der GeoServer, der sich gegenüber dem MapServer durch die zusätzliche Unterstützung des transaktionalen WFS auszeichnet. Zudem erfolgt das Styling der Layer im GeoServer per SLD-Standard während der MapServer ein eigenes Format verwendet. Auch die Administration und Konfiguration gestaltet sich beim GeoServer mittels der Weboberfläche deutlich einfacher und

komfortabler. Allerdings erfordert der GeoServer eine Java-Laufzeitumgebung und einen Servlet-Container und ist daher nicht so schlank wie der als CGI-Programm betriebene MapServer.

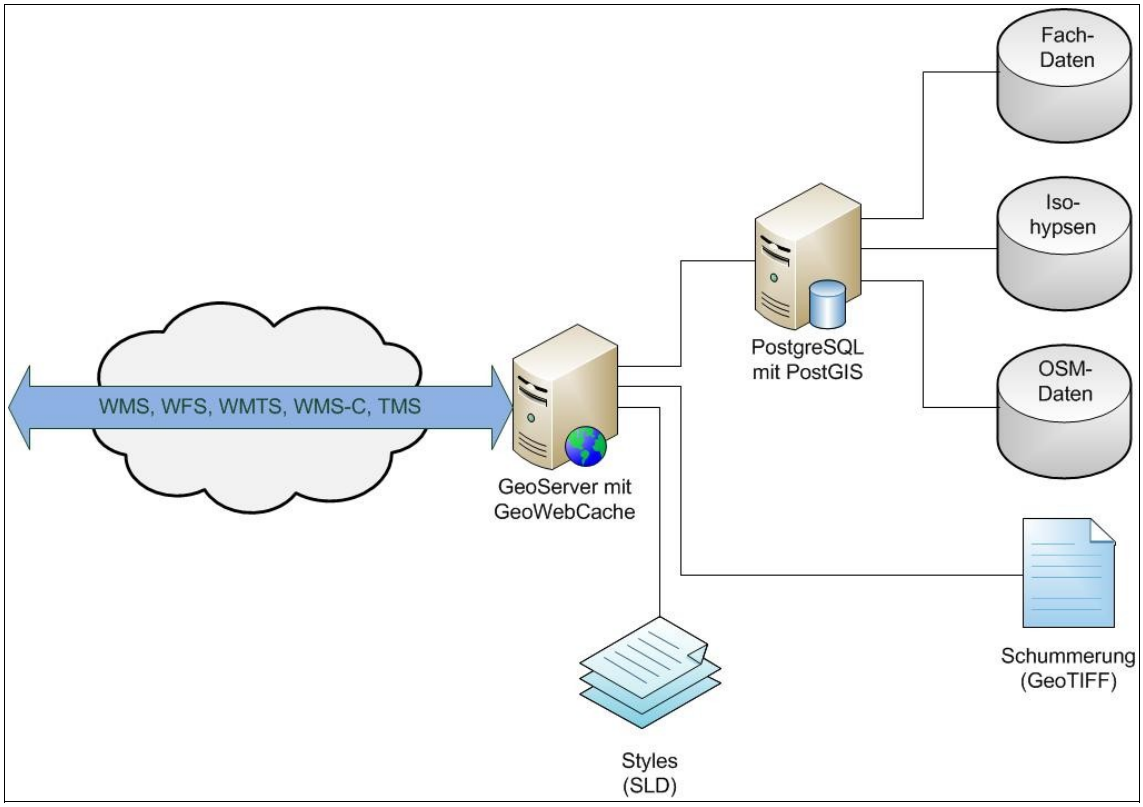

Abbildung 27: Web Mapping-Komponenten

Abbildung 25 illustriert die Verwendung des GeoServers, der die OpenStreetMap-Daten, die Fachdaten, die Isohypsen und die Schummerung als WMS-Layer zur Verfügung stellt. Hierfür müssen für alle Datenquellen entsprechende SLDs erstellt werden. Wurde bei dem Import mit osm2gis die Konfiguration nicht verändert, können für die OpenStreetMap-Daten die ansprechenden Styles von OpenStreetMap-in-a-Box verwendet und bei Bedarf angepasst werden. Hierbei ist es problemlos möglich, für einzelne Featureklassen separate Layer zu erstellen. Das Filtern geschieht mittels Filter Encoding innerhalb der SLDs. Auf diese Weise haben Benutzer alle Freiheiten sich eine Karten gemäß ihren Wünschen zu gestalten. Analog werden auch die Isohypsen, die Schummerung und die Fachdaten gestylt. Falls letztere in einer MySQL-Datenbank gehalten werden, muss hierfür zunächst das MySQL-Plugin installiert werden.

Damit die Geometrien der Fachdaten editiert werden können, werden diese per WFS-T verfügbar gemacht werden. Dies geschieht bei dem GeoServer automatisch. Damit jedoch nur ausgewählte Benutzer die Daten editieren können, müssen die Sicherheitseinstellungen entsprechend angepasst werden.

Als Tile Cache Server bietet sich der GeoWebCache an, der im Gegensatz zu TileCache und MapProxy bereits den offiziellen OGC-Kachelungsstandard WMTS unterstützt. Darüber hinaus beherrscht er jedoch auch die beiden OSGeo-Standards WMS-C und TMS und kann auch gewöhnliche WMS-Requests beantworten. Der GeoWebCache ist bereits in den GeoServer integriert und muss deshalb nicht zusätzlich installiert werden. Alle im GeoServer veröffentlichten Layer stellt der GeoWebCache automatisch ohne zusätzliche Konfiguration als Kacheln in den beiden Projektionen WGS 84 und Spherical Mercator zur Verfügung. Letztere ist eine Mercatorprojektion, bei der die Erde als Kugel betrachtet wird. Sie wird von den proprietären Kartendiensten wie Google Maps und Bing Maps aber auch von OpenStreetMap eingesetzt und wird – trotz ihrer durch die Vereinfachung bedingten Ungenauigkeit – im Web Mapping-Bereich häufig eingesetzt. Die automatische Konfiguration des GeoWebCache kann bei Bedarf problemlos mittels einer XML-Konfigurationsdatei überschrieben werden.

Standardmäßig werden nur Kacheln gecached, die von Clients angefordert wurden. Dies hat zur Folge, dass Anfragen an den GeoWebCache zunächst keine großen Performance-Vorteile mit sich bringen. Über die Web-Schnittstelle des GeoWebCache, kann jedoch das automatische Befüllen des Caches für jeden Layer angestoßen werden. Hierbei sollte jedoch beachtet werden, dass der Speicherbedarf für hohe Zoomstufen immens ist.

6.3.2 Routing

Da wie bereits erwähnt keine quelloffene Implementierung des OpenLS-Routing Service existiert, muss an dieser Stelle auf den Einsatz eines offenen Standards verzichtet werden. Als äußerst flexibel erweist sich allerdings die Kombination aus der PostGIS-Erweiterung pgRouting und dem Web Routing Service, die mehrere Routing-Algorithmen und eine Vielzahl an Ausgabeformaten u.a. sogar XLS bietet. Leider ist der Quellcode des Web Routing Service derzeit (Dezember 2010) nicht mehr verfügbar. Bis dahin kann auf die Kombination Gosmore/YOURS zurückgegriffen werden, die

allerdings nur für kürzere Routen (bis zu 200 km in dicht besiedelten Gebieten wie in Deutschland) designed wurde.

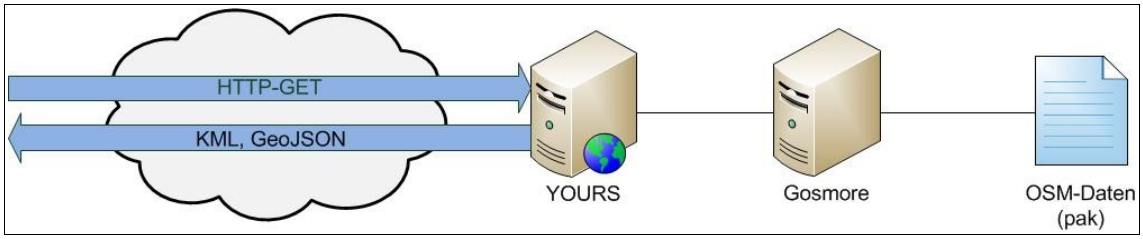

Abbildung 28: Routing-Komponenten

Abbildung 28 veranschaulicht den Ablauf der Routenberechnung mit dem YOURS-API und Gosmore als Routing Engine. Die HTTP-GET-Requests werden von YOURS an Gosmore weitergeleitet und dessen Antwort in das angeforderte Format (KML oder GeoJSON) konvertiert.

6.3.3 Geocoding

Da auch für den Location Utility Service der OpenLS-Spezifikation keine Open Source-Implementierung existiert, muss auch hier auf offene Standards verzichtet werden. Gisgraphy stellt dank seines Webinterfaces zwar die komfortabelste Lösung dar, erfordert aber eine vorherige Auswahl des Ortes und beantwortet Anfragen bislang nur auf Straßen- und noch nicht auf Hausnummernebene. Letzteres beherrscht auch der Name Finder nicht, weshalb die Wahl auf dessen ausgereifteren Nachfolger Nominatim fällt. Dieser muss allerdings dahingegen erweitert werden, dass neben den OpenStreetMap-Daten auch die Fachdaten durchsucht werden.

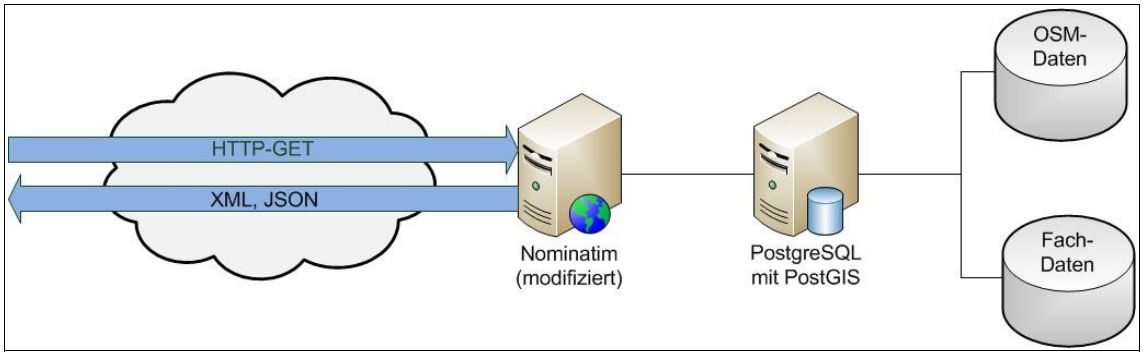

Abbildung 29: Geocoding-Komponenten

Abbildung 29 illustriert den Ablauf einer Geocoding-Anfrage an Nominatim, das den HTTP-GET-Request in eine SQL-Anfrage an den Datenbank-Server umsetzt. Dessen Antwort verpackt Nominatim schließlich in das angeforderte Ausgabeformat (XML oder JSON).

6.4 Präsentationsschicht

In der Präsentationsschicht werden die Anfragen an die verschiedenen Dienste gestellt und deren Antworten verarbeitet. Dabei hilft die JavaScript-Bibliothek OpenLayers, die alle verwendeten OGC-Protokolle unterstützt und in beiden Clients (Thin Client und Rich Client) eingesetzt wird.

Abbildung 30 zeigt die in der Präsentationsschicht verwendeten Schnittstellen:

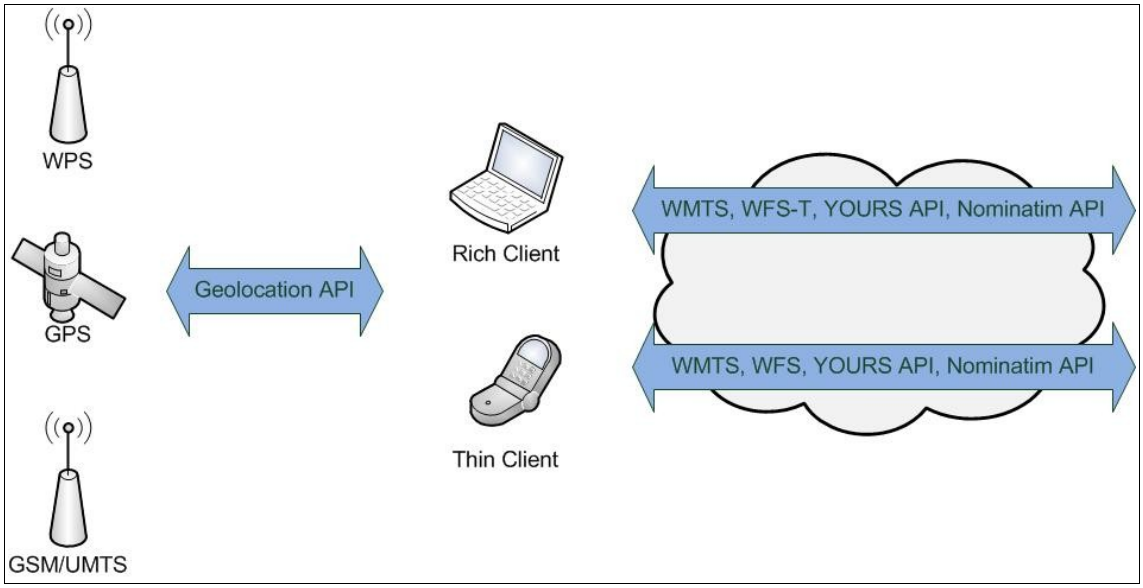

Abbildung 30: Schnittstellen der Präsentationsschicht

Standardmäßig sollen beide Clients die Karte immer entsprechend des aktuellen Standorts zentrieren. Dieser wird mittels der watchPosition-Funktion des Geolocation API ermittelt und fortlaufend aktualisiert. Sinnvoll ist hier der von [Spiering/Haiges 2010] vorgeschlagene Ansatz, anhand der PositionOptions (siehe Kapitel 4.1.1) zunächst möglichst schnell eine unter Umständen ungenaue Position (z.B. per GSM/UMTS) zu ermitteln, um diese dann mit einer zweiten Anfrage zu verfeinern (z.B. via GPS). Auf welche Positionierungstechnologie letztendlich zurückgegriffen wird, entscheidet aber die Implementierung des Geolocation API.

Während die Hintergrundkarte, bestehend aus den Geobasisdaten von OpenStreetMap und den SRTM-Höhendaten in Form von Isohypsen oder einer Schummerung als Kacheln per WMTS-Requests vom GeoWebCache angefordert werden, werden die thematischen Daten per WFS vom GeoServer geladen. Auf diese Weise ist kein zusätzlicher WMTS-Request (GetFeatureInfo) nötig, um bei Bedarf Details zu den einzelnen Features in einem Popup anzeigen zu können, da der Client bereits über alle benötigten Informationen verfügt. Welche thematischen Layer in der Karte angezeigt werden, kann der jeweilige Benutzer per Mausklick entscheiden.

Daneben bieten die Clients auch eine POI-Suchfunktion, die neben der Geokodierung einer Adresse auch die Suche nach bestimmten Feature-Kategorien unterstützt. Bei Letzterer kann der Umkreis auf ein definiertes Rechteck eingeschränkt werden (siehe Kapitel 3.1.7). War solch eine Anfrage an den Geocoding-Server (Nominatim-API) erfolgreich, wird die Karte entsprechend der Koordinaten zentriert und ein entsprechender Marker gesetzt.

Zu einem auf diese Weise ermittelten POI kann der Client im Anschluss eine Routenberechnung vom aktuellen Standort aus anstoßen. Hierfür stellt er eine entsprechende Anfrage an den Routing-Dienst (YOURS-API) und erhält die Route im KML-Format. Da die eigene Position laufend überwacht wird, kann eine Abweichung von der Route erkannt und eine Anpassung vom Server angefordert werden.

Beide Clients sind im Idealfall ein Widget gemäß der W3C-Spezifikation, dass auf dem Endgerät – in der Regel einem GPS-fähigen Handy oder Tablet – installiert werden kann. Hierfür muss allerdings eine entsprechende Laufzeitumgebung vorhanden sein, die derzeit nur in Form von Opera Mobile für Symbian-Handys frei verfügbar ist. Durch den Einsatz von PhoneGap kann der Client jedoch in native Apps für die fünf Plattformen iOS, Android, Blackberry, Symbian und WebOS konvertiert werden. Andere Geräte müssen auf den Komfort einer installierbaren App verzichten und auf den Browser zurückgreifen.

Der Rich Client dient der Bearbeitung der thematischen Daten und greift im Gegensatz zum Thin Client daher auch schreibend per WFS-T auf den Web Mapping Server zu. Authorisierte Personen haben so die Möglichkeit neue Features zu erfassen und bestehende Objekte zu editieren oder zu löschen. Für die Gestaltung der Oberfläche des Rich Clients wird GeoExt als OpenLayers-Aufsatz eingesetzt, um den Bearbeitern das Look & Feel einer Desktop-Anwendung geben zu können.

7 Fazit

7.1 Ergebnisse

Ziel der vorliegenden Arbeit war die Entwicklung einer Konzeption für plattformübergreifende, standortbezogene Mapping Apps basierend auf freien Geodaten, offenen Standards und Open Source-Komponenten.

Hierfür wurden in einem ersten Schritt freie Geodaten hinsichtlich ihrer Eignung untersucht. In diesem Bereich ist das rasant wachsende OpenStreetMap das Maß aller Dinge und kann es in vielerlei Hinsicht mit der kommerziellen Konkurrenz aufnehmen. Besonders in Deutschland ist die Abdeckung mittlerweile außerordentlich hoch und es werden vielerorts bereits Details wie Briefkästen oder Altglascontainer, die in den populären Geodatendiensten Google Maps und Bing Maps schlichtweg nicht vorhanden sind, erfasst. Die große Stärke von OpenStreetMap – das Crowdsourcing – ist allerdings auch gleichzeitig die große Schwäche des Projekts. Da jeder die Möglichkeit hat, bei der Datenerfassung mitzuwirken, können sich sehr leicht Fehler einschleichen und die Genauigkeit leidet darunter und wird mitunter sehr heterogen. Das einfache Datenmodell – bestehend aus Nodes, Ways, Relationen und Tags – bietet die nötigen Freiheiten, um jederzeit neue Feature-Typen hinzufügen zu können. Die derzeitige CC-BY-SA-Lizenz soll in Zukunft durch die besser geeignete OdBL ersetzt werden, die mehr Flexibilität bei der Nutzung von OpenStreetMap-Daten in Kombination mit anderen Datenquellen bietet.

Allenfalls eine Ergänzung zu OpenStreetMap stellt GeoNames, eine umfangreiche Datenbank für geographische Namen, die unter einer CC-BY-Lizenz steht, dar. Für Höhendaten stehen mit SRTM und ASTER zwei freie Datenquellen zur Verfügung, die einen Großteil der Landmasse der Erde abdecken. Obwohl ASTER eine höhere durchgehende horizontale Auflösung und eine größere Abdeckung als SRTM bietet, können aufgrund der derzeitigen OpenStreetMap-Lizenz nur SRTM-Daten mit OpenStreetMap-Daten kombiniert werden.

Im zweiten Schritt wurden die relevanten offenen Standards von W3C, OGC und OSGeo identifiziert und genauer betrachtet. Das Geolocation API des W3C dient der plattformunabhängigen Positionsbestimmung mittels JavaScript. Obwohl die

Schnittstelle noch kein offizieller Standard ist, existieren bereits zahlreiche Implementierungen für Desktop- und Mobilbrowser, so dass sie durchaus schon eingesetzt werden kann. Ein Test mit Opera Mobile 10 auf einem S60-Gerät mit A-GPS konnte diese Einschätzung bestätigen.

Noch nicht so weit fortgeschritten ist der Standardisierungsprozess im Bereich der Widgets. Hier handelt es sich um eine ganze Reihe von Spezifikationen, die den formalen Aufbau, APIs, die Signierung, automatisierte Updates, den Zugriff auf Netzwerkressourcen, ein URI-Schema und unterschiedliche Ansichtsmodi beschreiben. Um innerhalb von Widgets auf Gerätefunktionen wie Adressbuch, Kalender oder Kamera zugreifen zu können, werden allerdings weitere APIs benötigt, mit deren Spezifizierung das W3C bereits begonnen hat. Da es bis zur Verabschiedung der Standards noch eine ganze Weile dauern wird, haben sich Industriekonsortien wie die WAC gebildet, deren Ziel es ist, möglichst bald plattformübergreifende Widgets mit erweiterten Funktionalitäten zu ermöglichen.

Die relevanten Standards des OGC sind hingegen schon alle verabschiedet und für die meisten existieren schon zahlreiche Implementierungen. Vielfach bewährt und weit verbreitet ist der WMS zum Abrufen dynamisch gerenderter Kartenbilder. Um diese Karten flexibler gestalten zu können, existiert mit dem SLD-Profil eine Erweiterung des WMS, die benutzerdefiniertes Styling ermöglicht. Weniger Flexibilität jedoch einen deutlichen Performancegewinn verspricht der neue WMTS-Standard, mit dem das OGC endlich auf die weite Verbreitung des Kachelungsansatzes reagiert hat. Der WFS hingegen bietet die Möglichkeit die vektoriellen Geodaten, aus denen der WMS bzw. WMTS die Karten bzw. Kacheln rendern, abzurufen und zu bearbeiten. Speziell für Location Based Services wurde beim OGC der OpenLS-Standard entwickelt, der insgesamt sieben Dienste definiert, u.a. den Route Service zum Ermitteln von Routen und Navigationsanweisungen und den Location Utility Service für Geocoding und Reverse Geocoding. Obwohl die erste Version von OpenLS bereits 2004 verabschiedet wurde, konnte sich der Standard bislang nicht durchsetzen.

Bei der OSGeo handelt es sich zwar nicht um ein Organisationsgremium, dennoch hat sie mit dem WMS-C und dem TMS zwei weit verbreitete Kachelungsstandards spezifiziert, die derzeit noch deutlich weiter verbreitet sind als der junge WMTS.

Nach der Identifizierung der relevanten offenen Standards wurden im dritten Schritt Open Source-Komponenten evaluiert, die diese implementieren. Im Bereich der räumlichen Datenbanken wurde MySQL und die Kombination PostgreSQL/PostGIS

näher betrachtet, wobei letztere aufgrund der deutlich größeren räumlichen Funktionalitäten und höheren Standardkonformität vorzuziehen ist. Als Web Mapping Server wurden der MapServer und der GeoServer untersucht, die beide zahlreiche OGC-Protokolle und GIS-Formate unterstützen. Im Gegensatz zum GeoServer ermöglicht der MapServer jedoch keine Bearbeitung von Vektordaten per WFS-T und beschränkt sich auf lesende Zugriffe. Im Bereich der Tile Cache Server weist der GeoWebCache die höchste Interoperabilität auf, da er anders als TileCache und MapProxy alle drei vorgestellten offenen Kachelungsstandards (WMTS, WMS-C und TMS) implementiert.

Auf der Client-Seite hat sich die JavaScript-Bibliothek OpenLayers als Standard im Open Source-Bereich etabliert. Mit GeoExt existiert zudem ein Aufsatz, der eine Desktop-ähnliche Benutzerschnittstelle ermöglicht. Da der Standardisierungsprozess der relevanten W3C-Spezifikationen (Geolocation API, Widgets) noch nicht abgeschlossen ist und erst wenige Implementierungen vorhanden sind, ist eine plattformübergreifende App-Entwicklung nur durch die Verwendung von Mobile Web Frameworks möglich, unter denen sich PhoneGap durch die Unterstützung der meisten mobilen Plattformen auszeichnet.

Leider stellte sich heraus, dass es bislang keine Open Source-Implementierung des OpenLS-Standards gibt, weshalb im Routing- und Geocoding-Bereich Alternativen mit eigenen Protokollen betrachtet wurden. Während pgRouting die PostgreSQL/PostGIS-Datenbank um Routing-Funktionalitäten erweitert, wurde Gosmore speziell für die Routenberechnung auf Basis von OpenStreetMap-Daten entwickelt und verfügt mit YOURS über eine komfortable Web-Schnittstelle. pgRouting ist zwar deutlich flexibler, erfordert aber Programmieraufwand, da derzeit kein quelloffener Web-Wrapper verfügbar ist. Als Geocoder kann auf den derzeitigen OpenStreetMap-Geocoder Nominatim zurückgegriffen werden, der im Gegensatz zu seinem Vorgänger Name Finder und dem Gisgraphy-Framework auch Hausnummern – sofern diese erfasst wurden – beachtet. Eine Betrachtung der Programmbibliothek GDAL/OGR, die den lesenden und schreibenden Zugriff auf viele Raster- und Vektorformate der GIS-Welt ermöglicht, und der Vergleich einiger OpenStreetMap-Import-Tools, unter denen sich osm2gis von OpenStreetMap-in-a-Box dank seiner Flexibilität hervorhob, beschließt die Evaluierung der Open Source-Komponenten.

Im letzten Schritt konnte nun aus den gewonnenen Erkenntnissen eine Konzeption für plattformübergreifende, positionsbezogene Mapping Apps erarbeitet werden. Hierbei

handelt es sich um eine klassische 3-Schicht-Architektur bestehend aus Persistenz-, Applikationslogik- und Präsentationsschicht.

In der Persistenz-Schicht empfiehlt sich der Einsatz einer PostgreSQL-Datenbank mit PostGIS-Erweiterung. In diese werden die OpenStreetMap-Daten mittels osm2gis für das Rendering und mittels osm2pgsql für das Geocoding importiert. Für das Routing hingegen werden die OpenStreetMap-Daten mittels Gosmore in das benötigte binäre Dateiformat konvertiert. Aus den SRTM-Daten können mittels GDAL/OGR Isohypsen erzeugt und in eine PostgreSQL-Datenbank importiert bzw. eine Schummerung im GeoTIFF-Format erzeugt werden.

Eine GeoServer-Instanz mit integrierten GeoWebCache stellt in der Logikschicht OpenStreetMap-Daten, Isohypsen, Schummerung und Fachdaten als einzelne Layer in Form von bereits gerenderten Kartenkacheln via WMTS und in Vektorform via WFS zur Verfügung. Das Styling der Layer erfolgt dabei mittels SLD, Symbology Encoding und Filter Encoding. Da der GeoServer auch WFS-T beherrscht, können die Daten über diese Schnittstelle auch bearbeitet werden, so dass die Sicherheitseinstellungen entsprechend angepasst werden müssen. Neben dem Web Mapping umfasst diese Schicht noch das Routing mittels Gosmore/YOURS und das Geocoding mittels Nominatim.

In der Präsentationsschicht werden die genannten Dienste entsprechend der Benutzereingaben aufgerufen und deren Ergebnisse in der Karte dargestellt, die sich mittels Geolocation API positionsabhängig verschiebt. Da OpenLayers alle verwendeten OGC-Protokolle unterstützt müssen nur Routing und Geocoding selbst implementiert werden. Um installierbare Apps für möglichst viele Mobilplattformen anbieten zu können, wird der mittels der Webtechniken HTML, CSS und JavaScript erstellte Client mit Hilfe des PhoneGap-Frameworks in native Anwendungen konvertiert.

7.2 Bewertung und Ausblick

Innerhalb der vorliegenden Arbeit konnte gezeigt werden, dass bereits heute plattformübergreifende, positionsbezogene Mapping Apps mit Routing- und Geocoding-Funktionalitäten basierend auf freien Geodaten, offenen Standards und Open Source-Komponenten möglich sind. Im Bereich der offenen Standards müssen allerdings noch einige Einschränkungen gemacht werden.

So stellte sich beispielsweise heraus, dass sich der einzige Standard im Bereich des Routing und Geocoding (OpenLS) bislang nicht durchsetzen konnte und an dieser Stelle deshalb auf den Einsatz eines offenen Standards verzichtet werden muss. Mit Ausnahme des von Georepublic angekündigten REST-Frameworks WebSI-Cola sind in diesem Bereich – zumindest kurzfristig – keine standardkonformen, quelloffenen Implementierungen zu erwarten.

Statt auf einen offenen Standard zu verzichten, könnte eine Geocoding-Anfrage jedoch auch als WFS-GetFeature-Request formuliert werden. Mittels Filter Encoding kann die Suche auf beliebige Eigenschaften eingeschränkt werden. Hierfür müssen allerdings zunächst die verfügbaren Feature-Klassen sowie deren Attribute via GetCapabilities und DescribeFeatureType ermittelt werden. Eine einfacher Geocoding-Request setzt demnach immer mehrere Anfragen voraus. Mit dem WFS-G [Fitzke/Atkinson 2006] wurde beim OGC bereits ein Gazetteer Service als WFS-Profil in Form eines Best Practices Documents spezifiziert. Seit Version 0.9.3 im Juni 2006 wurde dieser allerdings nicht mehr weiter entwickelt.

Deutlich besser sind hingegen die Aussichten im Widgets-Bereich. Hier ist zwar noch kein endgültiger W3C-Standard verabschiedet und insbesondere die Device APIs befinden sich noch in einem sehr frühen Stadium, aber das Interesse – insbesondere der Netzbetreiber – an plattformübergreifenden Widgets ist sehr groß. Dies führte bereits zur Bildung der WAC, die in weniger als einem Jahr auf über 50 Mitglieder anwuchs und mit WAC 1.0 bereits die erste Spezifikation veröffentlicht hat. Während WAC 1.0 sich noch stark an der JIL-Spezifikation orientiert, werden die kommenden Releases in Zusammenarbeit mit der Community entwickelt, mit dem Ziel die Ergebnisse in die W3C-Standards einfließen zu lassen. Bis die ersten WAC-fähigen Geräte bzw. Widget Runtimes auf den Markt kommen, kann jedoch auf Frameworks wie PhoneGap zurückgegriffen werden, die es erlauben aus plattformunabhängigen HTML, CSS und JavaScript native Apps für verschiedene Mobilplattformen zu erstellen.

Einen Schritt weiter ist das Geolocation API, das bereits von vielen aktuellen Browsern unterstützt wird. Hier ist allerdings zu beachten, dass nur die neuesten Mobiltelefone über solch einen aktuellen Browser verfügen. PhoneGap unterstützt jedoch das Geolocation API und konvertiert Positionsanfragen in das jeweilige plattformspezifische Äquivalent.

Die Entwicklung der letzten Jahre hat gezeigt, dass der Mobiltelefon-Markt in Zukunft – wie schon heute im Highend-Bereich – von kapazitiven Touchscreens dominiert sein wird. Diese können im Vergleich zu den älteren resistiven Touchscreens nur mit bloßen Fingern und Gesten (Multitouch) bedient werden. Dieser Umstand muss bei der Entwicklung von Apps berücksichtigt werden. Obwohl bei der Entwicklung von OpenLayers bislang keine Gestensteuerung berücksichtigt wurde, gibt es bereits erste vielversprechende Ansätze[78] OpenLayers multitouch-fähig zu machen.

Ein weiterer interessanter Aspekt, der in der vorliegenden Arbeit nicht berücksichtigt wurde, ist die Benutzbarkeit der Anwendung, wenn das entsprechende Endgerät keine Internetverbindung hat. Die Plattformunabhängigkeit wurde dadurch erkauft, möglichst viel Logik in den Server zu verlagern. Dies hat zur Folge, dass der Client ohne Server nicht in der Lage ist Routen zu ermitteln oder Adressen zu geokodieren. Durch Speichern der Kacheln auf dem Client, könnte aber zumindest die Karte auch dann noch angezeigt werden, wenn keine Verbindung zum Web Mapping Server mehr aufgebaut werden kann. Einen interessanten Ansatz könnte hierfür HTML 5 bieten, dass derzeit beim W3C entwickelt wird und viele neue Features unter anderem „Offline Web Applications“[79] mitbringt. Anhand einer so genannten Cache-Manifestdatei kann eine Webseite zukünftig festlegen, welche Ressourcen vom Client gecached werden sollen [Spiering/Haiges 2010]. Mit dem Einsatz eines TMS, der die einzelnen Kacheln im Gegensatz zum WMTS direkt und nicht über Parameter adressiert, könnte auf diese Weise ein Caching bestimmter Kartenausschnitte erzwungen werden.

[78] http://mobilegeo.wordpress.com/2010/01/05/testing-open-layers-with-iphone-and-android/
[79] http://www.w3.org/TR/html5/offline.html#offline

Literaturverzeichnis

[Berjon 2009] Berjon R.: *Widgets 1.0: Widget URIs*, 2009, http://www.w3.org/TR/widgets-uri/, Stand: 08.10.2009, letzter Zugriff: 15.05.2010

[Berjon 2010] Berjon R.: *Widget Access Request Policy*, 2010, http://www.w3.org/TR/2010/CR-widgets-access-20100420/, Stand: 20.04.2010, letzter Zugriff: 15.05.2010

[Berjon et al. 2010] Berjon R., Bersvendsen A., Caceres M., Hanclik M.: *View Mode Media Feature*, 2010, http://www.w3.org/TR/view-mode/, Stand: 20.04.2010, letzter Zugriff: 15.05.2010

[Blankenbach 2007] Blankenbach J.: *Handbuch der mobilen Geoinformation - Architektur und Umsetzung mobiler standortbezogener Anwendungen und Dienste unter Berücksichtigung von Interoperabilität*, Wichmann, 2007

[Caceres 2008] Caceres M.: *Widgets 1.0: The Widget Landscape*, 2008, http://www.w3.org/TR/2008/WD-widgets-land-20080414/, Stand: 14.04.2008, letzter Zugriff: 12.05.2010

[Caceres 2009] Caceres M.: *Widget Packaging and Configuration*, 2009, http://www.w3.org/TR/2009/CR-widgets-20091201/, Stand: 01.12.2009, letzter Zugriff: 12.05.2010

[Caceres et al. 2009] Caceres M., Berjon R., Bersvendsen A.: *The Widget Interface*, 2009, http://www.w3.org/TR/2009/CR-widgets-apis-20091222/, Stand: 22.12.09, letzter Zugriff: 12.05.2010

[Caceres et al. 2010] Caceres M., Hirsch F., Priestley M.: *Digital Signatures for Widgets*, 2010, http://www.w3.org/TR/widgets-digsig/, Stand: 11.05.2010, letzter Zugriff: 15.05.2010

[Caceres/Berjon 2010] Caceres M., Berjon R.: *Widget Updates*, 2010, http://www.w3.org/TR/widgets-updates/, Stand: 13.04.2010, letzter Zugriff: 15.05.2010

[Caceres/Priestley 2009] Caceres M., Priestley M.: *Widgets 1.0: Requirements*, 2009, http://www.w3.org/TR/2009/WD-widgets-reqs-20090430/, Stand: 30.04.2009, letzter Zugriff: 12.05.2010

[Creative Commons 2004] Creative Commons: *Namensnennung-Weitergabe unter gleichen Bedingungen*, 2004, http://creativecommons.org/licenses/by-sa/2.0/deed.de, Stand: 24.05.2004, letzter Zugriff: 03.03.2010

[de la Beaujardiere 2006] de la Beaujardiere J.: *OpenGIS Web Map Server Implementation Specification*, 2006, http://portal.opengeospatial.org/files/?artifact_id=14416, Stand: 15.03.2006, letzter Zugriff: 15.06.2010

[DLR 2009] Deutsches Zentrum für Luft- und Raumfahrt: *SRTM*, 2009, http://www.dlr.de/srtm/, Stand: 16.04.2009, letzter Zugriff: 01.05.2010

[Fitzke/Atkinson 2006] Fitzke J., Atkinson R.: *Gazetteer Service - Application Profile of the Web Feature Service Implementation Specification*, 2006, http://portal.opengeospatial.org/files/?artifact_id=15529, Stand: 05.06.2006, letzter Zugriff: 27.12.2010

[Fraunhofer-Gesellschaft 2006] Renner T., Vetter M., Rex S., Kett H.: *Open Source Software: Einsatzpotenziale und Wirtschaftlichkeit*, Fraunhofer IRB, 2006

[FSFE et al. 2008] Free Software Foundation Europe et al.: *Offene Standards*, 2008, http://fsfe.org/projects/os/def.de.html, Stand: 24.03.2010, letzter Zugriff: 05.05.2010

[GeoNames 2010] Wick M., Boutreux C.: *GeoNames*, 2010, http://www.geonames.org/, Stand: keine Angabe, letzter Zugriff: 01.05.2010

[Goodchild 2007] Goodchild M.: *Citizens as sensors: the world of volunteered geography*, GeoJournal, 69, S. 211-221

[Graf 2009] Graf A.: *Marktforscher: Offene Handy-Betriebssysteme werden wichtiger*, 2009, http://www.heise.de/newsticker/meldung/Marktforscher-Offene-Handy-Betriebssysteme-werden-wichtiger-195324.html, Stand: 12.02.2009, letzter Zugriff: 20.02.2010

[Grassmuck 2004] Grassmuck V.: *Freie Software - Zwischen Privat- und Gemeineigentum*, 2004, http://freie-software.bpb.de/Grassmuck.pdf, Stand: 30.11.2004, letzter Zugriff: 07.07.2010

[Ingensand/Bitzi 2001] Ingensand H., Bitzi P.: *Technologien der GSM-Positionierungsverfahren*, Allgemeine Vermessungs-Nachrichten, 8-9/2001, S. 286-294

[Jacobsen 2004] Jacobsen K.: *Generierung und Validierung von Höhenmodellen aus Weltrauminformationen*, Publikationen der DGPF, Band 13, S. 475-482

[Jansen/Adams 2010] Jansen M., Adams T.: *OpenLayers - Webentwicklung mit dynamischen Karten und Geodaten*, Open Source Press, 2010

[Köhne/Wössner 2007] Köhne A., Wößner M.: *GPS-System*, 2007, http://www.kowoma.de/gps/, Stand: 18.06.2007, letzter Zugriff: 06.04.2010

[Korduan/Zehner 2008] Korduan P., Zehner M.: *Geoinformation im Internet - Technologien zur Nutzung raumbezogener Informationen im WWW*, Wichmann, 2008

[Kriesing 2009] Kriesing W.: *W3C Widgets - mobile Applikationen werden plattformneutral*, iX - Magazin für professionelle Informationstechnik, 11/2009, S. 134-139

[Lendholt 2009] Lendholt M.: *Topografische Punkte - Geodaten visualisieren mit Symbology Encoding*, iX - Magazin für professionelle Informationstechnik, 3/2009, S. 108-110

[Lupp 2007] Lupp M.: *Styled Layer Descriptor profile of the Web Map Service Implementation Specification*, 2007, http://portal.opengeospatial.org/files/?artifact_id=22364, Stand: 29.06.2007, letzter Zugriff: 17.06.2010

[Mabrouk 2008] Mabrouk M.: *OpenGIS Location Services (OpenLS): Core Services*, 2008, http://portal.opengeospatial.org/files/?artifact_id=22122, Stand: 09.09.2008, letzter Zugriff: 19.06.2010

[Masó et al. 2010] Masó J., Pomakis K., Julià N.: *OpenGIS Web Map Tile Service Implementation Standard*, 2010, http://portal.opengeospatial.org/files/?artifact_id=35326, Stand: 06.04.2010, letzter Zugriff: 18.06.2010

[Mitchell 2008] Mitchell T.: *Web Mapping mit Open Source-GIS-Tools*, O'Reilly, 2008

[Müller 2006] Müller M.: *Symbology Encoding Implementation Specification*, 2006, http://portal.opengeospatial.org/files/?artifact_id=16700, Stand: 21.07.2006, letzter Zugriff: 17.06.2010

[MySQL 2010] MySQL: *MySQL 5.1 Referenzhandbuch*, 2010, http://dev.mysql.com/doc/refman/5.1/de/index.html, Stand: 05.06.2010, letzter Zugriff: 14.07.2010

[Neis 2008] Neis P.: *Location Based Services mit OpenStreetMap Daten*, Masterarbeit, Fachhochschule Mainz, 2008

[OpenStreetMap-Wiki 2010] Anonymus: *OpenStreetMap Wiki*, 2010, http://wiki.openstreetmap.org/, Stand: keine Angabe, letzter Zugriff: 03.03.2010

[OSGeo-Wiki 2010] Anonymus: *OSGeo-Wiki*, 2010, http://wiki.osgeo.org/, Stand: 03.06.2010, letzter Zugriff: 25.06.2010

[Popescu 2010] Popescu A.: *Geolocation API Specification*, 2010, http://www.w3.org/TR/2010/CR-geolocation-API-20100907/, Stand: 07.09.2010, letzter Zugriff: 18.09.2010

[Ramm/Topf 2009] Ramm F., Topf J.: *OpenStreetMap - Die freie Weltkarte nutzen und mitgestalten*, Lehmanns Media, 2009

[Ronneburg 2010] Ronneburg F.: *Freie Software / Open Source*, 2010, http://debiananwenderhandbuch.de/freiesoftware.html, Stand: keine Angabe, letzter Zugriff: 08.05.2010

[Sachse 2010] Sachse J.: *The standardization of Widget-APIs as an approach for overcoming device fragmentation*, Grin, 2010

[Spiering/Haiges 2010] Spiering M., Haiges S.: *HTML5-Apps für iPhone und Android*, Franzis, 2010

[Vretanos 2005] Vretanos P.: *OpenGIS Filter Encoding Implementation Specification*, 2005, http://portal.opengeospatial.org/files/?artifact_id=8340, Stand: 03.05.2005, letzter Zugriff: 17.06.2010

[Vretanos 2005-2] Vretanos P.: *Web Feature Service Implementation Specification*, 2005, http://portal.opengeospatial.org/files/?artifact_id=8339, Stand: 03.05.2005, letzter Zugriff: 04.11.2010

[Zielstra/Zipf 2010] Zielstra D., Zipf A.: *A Comparative Study of Proprietary Geodata and Volunteered Geographic Information for Germany*, AGILE 2010 - The 13th AGILE International Conference on Geographic Information Science, 2010

Anhang

A Name Finder-API

Der Name Finder ist unter der folgenden URL erreichbar:

`http://gazetteer.openstreetmap.org/namefinder/search.xml?find=`*`search`*

Der Platzhalter *search* muss hier durch den URL-encodierten Suchausdruck ersetzt werden.

Tabelle 7 listet einige Beispiele für mögliche Suchanfragen, die dieser Dienst unterstützt, auf:

Suchausdruck	Ergebnisse
Robert-Koch-Allee near München	• Robert-Koch-Allee in Pentenried im Münchner Umland • Robert-Koch-Allee im thüringischen Dorf München
Robert-Koch-Allee near 48.137,11.575	• Robert-Koch-Allee in Pentenried im Münchner Umland
Airports near 48.137,11.575	• Internationaler Flughafen München • Flughafen Moosburg/Kippe in der Nähe von München

Tabelle 7: Beispiele für Suchen mit dem Name Finder
Quelle: http://gazetteer.openstreetmap.org

Nach „near“ kann nur eine Ortschaft (Stadt, Dorf, Weiler usw.) oder eine gültige Koordinatenangabe stehen, wobei Letzteres zu bevorzugen ist, um Doppeldeutigkeiten zu vermeiden. Mittels des Parameters „max“ kann die Anzahl der Treffer eingeschränkt werden. Jeder Treffer im resultierenden XML-Dokument enthält eine Koordinatenangabe sowie Angaben zu den nächsten Ortschaften und deren jeweilige Distanzen.

B Nominatim-API

Um nominale Such-Anfragen an Nominatim zu stellen, müssen verschiedene Parameter (siehe Tabelle 8) an die folgende Basis-URL angehängt werden:

```
http://nominatim.openstreetmap.org/search?
```

Parameter	Mögliche Werte	Bedeutung
q	*beliebig*	Suchbegriff (Name bzw. POI-Kategorie) und evtl. Verwaltungseinheit
format	html, xml oder json	Ausgabeformat
viewbox	*links*,*oben*,*rechts*,*unten*	Suchgebiet als umschreibendes Rechteck in Dezimalgrad
polygon	0 oder 1	Polygonumriss für jeden Treffer
addressdetails	0 oder 1	Aufgliederung der Adresse jedes Treffers

Tabelle 8: Parameter für Geocoding mittels Nominatim
Quelle: http://wiki.openstreetmap.org/wiki/DE:Nominatim

Statt der Einschränkung auf eine Rechteck mittels „viewbox“, kann das Suchgebiet auch auf die exakten Grenzen einer Verwaltungseinheit beschränkt werden (Beispiel: „München, Bayern“).

Eine Suchanfrage liefert bis zu zehn Treffer, wobei jeder Treffer mindestens eine Koordinatenangabe und ein umschreibendes Rechteck umfasst. Sollte es mehr als zehn Treffer geben, enthält das Antwortdokument eine entsprechende URL für die nächsten zehn Treffer.

Für eine Reverse Geocoding-Anfrage müssen verschiedene Parameter (siehe Tabelle 9) an die folgende Basis-URL angehängt werden:

```
http://nominatim.openstreetmap.org/reverse?
```

Parameter	Mögliche Werte	Bedeutung
lat	-90 bis +90	Geographische Breite in Dezimalgrad
lon	-180 bis +180	Geographische Länge in Dezimalgrad
format	xml oder json	Ausgabeformat
zoom	zwischen 0 und 18	Level of Detail (LoD) von Land (0) bis Gebäude (18)
addressdetails	0 oder 1	Aufgliederung der Adresse

Tabelle 9: Parameter für Reverse Geocoding mittels Nominatim
Quelle: http://wiki.openstreetmap.org/wiki/DE:Nominatim

Je nach angegebenen Level of Detail (LoD) liefert eine Reverse Geocoding-Anfrage das entsprechende Land oder die nächste exakte Adresse.

C GeoNames-API

Um nominale Such-Anfragen an GeoNames zu stellen, müssen verschiedene Parameter (siehe Tabelle 10) an die folgende Basis-URL angehängt werden:

```
http://ws.geonames.org/search?
```

Parameter	Mögliche Werte	Bedeutung
q	*beliebig*	Name
type	xml, json oder rdf	Ausgabeformat
style	SHORT, MEDIUM, LONG oder FULL	Wortfülle der Ausgabe
lang	ISO 636-Sprachcodes	Ausgabesprache
country	ISO 3166-Ländercodes	Einschränkung der Suche auf einzelne Länder
featureCode	siehe Codes[80]	Einschränkung der Suche auf einzelne Objektklassen

Tabelle 10: Parameter für Geocoding mittels GeoNames
Quelle: http://www.geonames.org/export/geonames-search.html

Um Reverse Geocoding-Anfragen an GeoNames zu stellen, müssen verschiedene Parameter (siehe Tabelle 11) an die folgende Basis-URL angehängt werden:

```
http://ws.geonames.org/findNearby?
```

Parameter	Mögliche Werte	Bedeutung
lat	-90 bis +90	Geographische Breite in Dezimalgrad
lng	-180 bis +180	Geographische Länge in Dezimalgrad
radius	*beliebig*	Suchradius in Kilometern
type	xml oder json	Ausgabeformat
style	SHORT, MEDIUM, LONG oder FULL	Wortfülle der Ausgabe
lang	ISO 636-Sprachcodes	Ausgabesprache
featureCode	siehe Tabelle 10	Einschränkung der Suche auf einzelne Objektklassen

Tabelle 11: Parameter für Reverse Geocoding mittels GeoNames
Quelle: http://www.geonames.org/export/web-services.html

[80] http://www.geonames.org/export/codes.html