Entwicklung eines Radroutenplaners unter Verwendung von freien Geodaten und Open-Source-Software

Studienprojekt 2009/2010

Bachelor of Science Geoinformatik

an der

Universität Osnabrück

Erstgutachter: **Prof. Dr. Manfred Ehlers**
Zweitgutachter: **Kai Behncke, Dipl.-Geogr., Dipl.-Umweltwiss.**

Osnabrück, den 30. April 2010

Die Dokumentation des Projektes wurde unter Mitwirkung folgender Studenten des Studiengangs Bachelor of Science Geoinformatik an der Universität Osnabrück durchgeführt:

Mehmet Cakmak

Niels Giebel

Steffen Gräuler

Bryan Hempen

Henry Hurink

Lena Hütten

Stefan Lüttmann

Andreas Mescheder

Ines Schiller

Mareike Schoof

Dennis Ströer

Matthias Thielscher

Christina Tischer

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1 Einleitung und Motivation

Seit dem 4. Januar 2010 ist Osnabrück eine Umweltzone, ab dem 3. Januar 2012 dürfen sogar nur noch Fahrzeuge mit einer grünen Feinstaubplakette in diesen Bereich - ein guter Grund, den Bürgern in und um Osnabrück das Radfahren wieder näherzubringen. In Zeiten steigender Benzinpreise werden außerdem viele Leute lieber mit dem Fahrrad als mit dem Auto zur Arbeit oder auf den Sonntagsausflug fahren. Laut dem Statistischen Bundesamt besitzen knapp 80% der deutschen Haushalte mindestens ein Fahrrad. Bei Familien mit Kindern sind dies sogar 94%, bei Paaren ohne Kinder 82%. Auch die unter 25-jährigen sind gut versorgt, denn 76% besitzen ein eigenes Fahrrad und von den über 65-jährigen erfreuen sich noch 43% an der motorfreien Fortbewegung. [Sta09]
Das Fahrrad ist in Deutschland also sehr beliebt, unabhängig von Alter, Beziehungsstatus oder Anzahl der Kinder. Laut dem Masterplan Mobilität der Stadt Osnabrück, der sich mit zukünftigen Verkehrsentwicklungen beschäftigt, beträgt der Anteil der Fahrradfahrer am Gesamtverkehr in Osnabrück 11-13%. [SHPer, vgl. S. 28]
Aus diesen Gründen und der Nachhaltigkeit zuliebe haben sich die Studenten des Bachelorstudiengangs Geoinformatik an der Universität Osnabrück im Rahmen eines Studienprojektes entschieden, einen interaktiven Radroutenplaner für die Stadt Osnabrück und das Umland zu entwickeln, der außerdem besondere Informationen für Touristen und Bürger bereitstellt. Dass dieser nur auf Grundlage von Open-Source-Software und freien Geodaten entwickelt wird, stellt dabei eine weitere Herausforderung dar. Neben der Bedingung nur freie Software zu verwenden, wird auch der Quellcode des Radroutenplaners selbst frei zugänglich sein.
Ab März 2009 soll unter der Domain `http://www.fahrradies.net` der Planer für Osnabrück zur Vefügung stehen.
In Osnabrück ist bereits eine Vielzahl an Geodaten vorhanden und besonders durch die Frida-Daten, die in das OpenStreetMap-Projekt eingepflegt wurden, besteht eine gute Grundlage im Bereich der freien Geodaten, auf welcher die Studenten aufbauen und die sie erweitern können.
Die Verbreitung von Radroutenplanern ist noch nicht sehr weit vorangeschritten: Es existiert kein explizit auf die Region Osnabrück oder für das Bundesland Niedersachsen ausgelegter interaktiver Radroutenplaner. Auch dies stellt einen Grund für die Studenten dar, das vorliegende Projekt auszuarbeiten.
Als oberstes Ziel des Projektes gilt es, einen Radroutenplaner zu erstellen, der sich in vielerlei Hinsicht von den wenigen anderen Planern dieser Art abhebt und auf lange Sicht konkurrenzfähig bleibt. Trotz der vergleichsweise schlechten Streckenführung, die in einer Stadt für Radfahrer gegeben ist, wollen die Studenten ein möglichst optimales Ergebnis erzielen. Dazu wird es nicht nur nötig sein, bestehende Routenplaner zu analysieren, sondern auch herauszufinden, wie der „perfekte Radroutenplaner“ aussieht. Letztendlich soll der Planer für die Stadt und den Landkreis einen Mehrwert besitzen und langfristig eingesetzt werden.

Mit dieser Motivation startete das Projekt Radroutenplaner im April 2009.
Im folgenden Kapitel werden zunächst einige grundlegende Begriffe, besonders auch im Hinblick auf Open-Source-Software, freie Geodaten und OpenStreetMap sowie Interaktivität erläutert. Kapitel 3 widmet sich der Vorgehensweise zur Erstellung des Radroutenplaners. Dafür werden einerseits die OpenStreetMap-Daten auf ihre Eignung für dieses Projekt untersucht und andererseits wird eine Analyse bestehender Radroutenplaner durchgeführt. Im 4. Kapitel werden die eingesetzten Softwareprodukte vorgestellt, welche allesamt auf Open Source sind. In Kapitel 5 geht es um die technische Realisierung des Radroutenplaners und die Implementation der Funktionen. Das 6. Kapitel beschäftigt sich mit der Öffentlichkeitsarbeit. Schließlich wird in Kapitel 7 das Projekt zusammengefasst und ein Fazit gezogen.

2 Grundlagen

In diesem Kapitel werden verschiedene Begriffe erläutert, die von grundlegender Bedeutung für die Konzeption eines interaktiven Radroutenplaners sind, basierend auf Open Source und freien Geodaten. Es sollen Hintergründe vermittelt und die Relevanz für vorliegende Projektarbeit aufgezeigt werden.

2.1 Open-Source-Software

Der Begriff Open-Source-Software wurde 1998 von der Open Source Initiative (OSI) verwendet, „mit der Absicht, den missverständlichen und antikommerziell klingenden Begriff „Freie Software“ zu vermeiden“[Moo01, S. 166 ff.] zitiert nach [Hen07, S. 57]. Zu dieser Problematik heißt es bei Grassmuck: „“Free“ ist nicht nur zweideutig („Freibier“ und „Freie Rede“), sondern offensichtlich war es in *The Land of the Free* zu einem unanständigen, „konfrontationellen“, irgendwie kommunistisch klingenden *four-letter word* geworden“[Gra04, S. 230]. Durch den neuen Begriff sollten Missverständnisse und negative Assoziationen vermieden werden.

Open-Source-Software wird definiert als „Software, die jeder Nutzer in beliebiger Weise nutzen, analysieren, kopieren, verändern und weiterverbreiten darf“[Hen07, S. 58]. Dazu ist es notwendig, dass neben dem maschinenlesbaren kompilierten Binärcode auch der von Menschen lesbare Quellcode zur Verfügung steht. Open Source bedeutet nicht, dass die Software zwangsläufig kostenlos zu erwerben ist, sondern das wesentliche Merkmal ist die Freiheit der Nutzer im Umgang mit der Software. [Hen07, vgl. S. 58]

Der Begriff „Open Source“ wird auf Software angewendet, deren Lizenzverträge den Forderungen entsprechen, die in der Open Source Definition der OSI detailliert beschrieben sind. Dies bedeutet u. a., dass der Quellcode verfügbar sein muss, die Software frei weiterverbreitet werden darf und Veränderungen der Software gestattet werden, die wiederum den gleichen Lizenzbedingungen unterliegen müssen wie die Originalsoftware. [Bun07b]

Die bekannteste Lizenz in diesem Zusammenhang ist die General Public License (GPL). Software, die unter einer solchen Lizenz steht, lässt sich nicht kommerziell vermarkten.

Im Laufe der Entwicklung hat Open Source eine Renaissance erfahren. In den frühen Tagen des Computers war der Quellcode aller Software frei zugänglich. Lediglich an der Hardware wurde Geld verdient. Die Software wurde damals meist von den Nutzern selbst geschrieben, entwickelte sich aber Mitte der 1970er Jahre zu einem eigenen Markt. Unternehmen verkauften ihre Software und schützten den Quellcode als Geschäftsgeheimnis. Dabei wurden nur noch Lizenzen zur Nutzung verkauft. [Bun07a]

Das Konzept der proprietären Software weist allerdings deutliche Nachteile auf. Da der Quellcode nicht veröffentlicht wird, werden mögliche Fehler nicht bekannt und können auch nicht behoben werden. Des Weiteren sind Softwarehersteller in der Lage ihre Software jederzeit zu verändern, sodass ihre Kunden eine neue Version erwerben müssen. [Mit08, vgl. S. 6 f.]

1984 startete Richard Stallman das GNU-Projekt mit dem Ziel ein Betriebssystem zu entwickeln, das frei zur Verfügung steht. GNU steht dabei für „GNU's Not Unix". 1985 gründete er die Free Software Foundation und im Jahre 1994 wurde das Betriebssystem GNU/Linux veröffentlicht. [Hen07, vgl. S. 58 f.]
An dem Begriff „Open Source" kritisiert Stallman, dass ausschließlich pragmatische Aspekte, Nützlichkeit, Features, Zuverlässigkeit und Effizienz der Software im Vordergrund stehen und nicht in erster Linie die Freiheit [Gra04, vgl. S. 231]. Auch Grassmuck schreibt: „Der Versuch der OSI, das Warenzeichen „Open Source" als Gütesiegel für Software durchzusetzen, deren Lizenz der Open Source-Definition genügt, ist gescheitert. Heute führen viele Produkte das Label, die keine Modifikationsfreiheit gewähren - und genau sie ist der Sinn der Quelloffenheit"[Gra04, S. 231].
Dennoch hat sich Open Source mit der Zeit zu einer weltweiten sozialen Bewegung entwickelt und zehntausende Menschen sind gemeinschaftlich tätig, um neben Software auch Wissen frei über das Internet verfügbar zu machen (Beispiel: http://wikipedia.org). [Bun07b]
Aufgrund der zahlreichen Vorteile und nicht zuletzt auch wegen der kostenlosen Verfügbarkeit basiert vorliegendes Projekt vollständig auf Open Source und Open-Source-Software. Zum Einsatz kommen beispielsweise Softwareprodukte wie Ubuntu (vgl. Kap. 4.1.2), der UMN MapServer (vgl. Kap. 4.3), PostgreSQL/PostGIS (vgl. Kap. 4.4) oder OpenLayers (vgl. Kap. 4.8). Neben Open-Source-Software stellen die im nächsten Abschnitt erläuterten freien Geodaten - insbesondere OpenStreetMap - einen weiteren wesentlichen Bestandteil des Radroutenplaners dar.

2.2 Freie Geodaten

Seit den letzten Jahren nimmt die Entwicklung und Verbreitung freier Geodaten stetig zu und freie Geodaten gewinnen somit immer mehr an Bedeutung. Ein gutes Beispiel hierfür ist das OpenStreetMap-Projekt, an dem sich weltweit fast 180.000 Menschen beteiligen, indem sie mit GPS-Geräten Daten erheben und der Allgemeinheit zur freien Verfügung stellen.
Da das vorliegende Projekt auf freien Geodaten aus OpenStreetMap basiert, soll nachfolgend der Begriff „Freie Geodaten" definiert und das OpenStreetMap-Projekt sowie die dazugehörige XML-Datenstruktur vorgestellt werden.

2.2.1 Definition

Geodaten sind Informationen, die in der Regel in digitaler Form vorliegen und einem bestimmten Punkt auf der Erdoberfläche zugewiesen sind. Sie können entweder Primärdaten, z. B. Logfiles aus einem GPS-Gerät, oder auch Sekundärdaten, d. h. in irgendeiner Form bearbeitet, sein.
Sie bestehen aus Geobasisdaten, Geofachdaten und Metadaten. Geobasisdaten sind die von den Vermessungs- und Katasterverwaltungen erhobenen und zur Verfügung gestellten Basis-

informationen zur raumbezogenen Abbildung von Geofachdaten. Geofachdaten sind fachbezogen erhobene Daten und beziehen sich auf ein bestimmtes Thema, wie z. B. Demographie oder Umwelt. Metadaten sind Daten über Daten und liefern Informationen über vorhandene Geobasis- und Geofachdaten. [HKS08]
Geodaten sind allgemein Ursprungsdaten, die zur Erstellung von beispielsweise Karten nötig sind. Mit ihnen können aber auch statistische Analysen durchgeführt oder Routen berechnet werden, so wie in diesem Radroutenplaner. Dabei können Geodaten verschiedene Dimensionen haben, von 2D (zweidimensional) bis 4D (vierdimensional).
Wegen großer Qualitätsschwankungen bei Geodaten wurde die DIN EN ISO 19113 entwickelt. Sie wurde letztmalig im Jahre 2002 geändert und regelt die Qualität von Geodaten. Die Norm beinhaltet die Qualitätsparameter Vollständigkeit, logische Konsistenz, Positionsgenauigkeit, zeitliche Genauigkeit und thematische Genauigkeit. [Mül07]
Freie Geodaten stehen den Nutzern unentgeltlich zur Verfügung und sind nicht durch besondere Lizenzen geschützt. Noch vor wenigen Jahren war es unvorstellbar, dass freie Geodaten überhaupt solche „beachtlichen Dimensionen“[Mit08, S. 80] annehmen. Mit dem Namen „freie Geodaten“ werden wie bei Open-Source-Software bestimmte Anforderungen verbunden. Zum einen heißt es, dass die Daten für jeden kostenlos zur Verfügung stehen. Außerdem dürfen sie jederzeit weiterverbreitet und an die eigenen Bedürfnisse angepasst, sprich verändert werden. Man findet freie Geodaten im Internet z. B. bei `http://www.freegis.org` oder `http://www.opengeodb.org`. Auch das OpenStreetMap-Projekt (s. Kap. 2.2.2) wird aus freien Geodaten erstellt und stellt diese, d. h. den gesamten Geodatenbestand, allen Benutzern zur freien Verfügung.
In der Regel werden von den Vermessungsstellen die Geobasisdaten aber nicht als freie Daten zur Verfügung gestellt, oder sie sind sehr heterogen und nicht kompatibel zueinander. Deshalb kann es schwierig sein für ein Projekt ausreichend gute, freie Geodaten zu finden. In den europäischen DACH-Ländern (Deutschland, Österreich, Schweiz) werden freie Geodaten unterschiedlich organisiert und sind bislang wenig verfügbar. Durch die 2007 verabschiedete INSPIRE-Richtlinie sollen sie aber der Öffentlichkeit besser zugänglich gemacht werden.
In Deutschland werden hochwertige Basisgeometrien als Katastergrundkarte geführt, wobei die Daten nicht zentral, sondern dezentral (z. B. in Nordrhein-Westfalen) oder landesweit zentral (Bayern) organisiert werden und über Geodatendienste angeboten werden. In der Schweiz regelt das Bundesamt für Landestopographie Swisstopo die Verwaltung von Geodaten. Die Institution ist zentral organisiert und bietet qualitativ hochwertige Geodaten. In Österreich hingegen gibt es sowohl auf Bundes- als auch auf Landesebene Zuständigkeiten für Geodaten. Wien, um nur ein Beispiel zu nennen, organisiert sich vorwiegend selbst, so dass eine ähnlich diffuse Vielfalt wie in Deutschland entsteht. Immer mehr Kommunen bieten Daten in Portalen und Diensten an, wie es vergleichbar auch in der Schweiz und in Deutschland der Fall ist. [Mit08, vgl. S. 80 ff.]
Für Osnabrück ist vor allem das Projekt „Frida“ bekannt. Diese von der Stadt zur Verfügung

gestellten, freien Vektor-Geodaten wurden 2003 erhoben und werden seit September 2007 in OpenStreetMap eingepflegt. Der Ansporn zu diesem Projekt war, dass freie Geodaten nur in den USA flächendeckend zur Verfügung stehen und dies auch in einer europäischen Stadt ausprobiert werden sollte. Als Ziele wurden genannt, dass ein detaillierter Datensatz zur Verfügung stehen sollte und für Kunden von Unternehmen Demos oder Vorführungen bereitgestellt werden sollten. Weiterhin war es als ein Test gedacht, inwieweit dieses Projekt längerfristig erfolgreich ist und vielleicht auch für Europa zur Verfügung gestellt werden könnte. [Int07]
Freie Geodaten weisen viele Vorteile auf. Ein wichtiger Aspekt ist, dass sie kostenlos sind. Außerdem sind sie für jeden frei verfügbar und in jedem Projekt etc. frei verwendbar. Dadurch, dass man sie verändern und an eigene Bedürfnisse anpassen darf, können freie Geodaten vielseitig eingesetzt werden. Oftmals sind sie auf einem aktuelleren Stand als kostenpflichtige Daten, da es im Sinne aller Nutzer ist, eine möglichst genaue Darstellung der Erdoberfläche zu schaffen. Durch die immer größer werdende Zahl von Nutzern werden außerdem alle Verbesserungsvorschläge kritisch betrachtet, sodass am Ende ein sehr gutes Ergebnis entsteht. [Geo]
Trotzdem gibt es auch einen gravierenden Nachteil, der gegen freien Geodaten spricht. Da „frei“ hier mit „jeder kann helfen“ gleichgesetzt wird, heißt es nicht zwangsläufig, dass es eine Verbesserung gibt. Wie auch im OpenStreetMap-Projekt zu sehen ist, kommt es hin und wieder zu Fehlern beim Mappen oder Einpflegen der Daten. Die Kartengrundlage des Radroutenplaners wird zwar auf Richtigkeit überprüft (vgl. Kap. 3), aber eine optimale Routenführung kann trotzdem nie komplett garantiert werden. Beispielsweise tragen nicht verbundene Straßenteilstücke dazu bei, dass die Route über einen Umweg führt (vgl. Kap. 5.1). Die Fehleranfälligkeit ist also vergleichsweise höher.
Vor allem die gute Datengrundlage von Osnabrück und die Tatsache, dass OpenStreetMap ein lohnenswertes Projekt ist, überzeugte die Studenten, sich - wie in der Aufgabestellung gefordert - ausschließlich auf freie Geodaten zu beschränken.

2.2.2 OpenStreetMap

OpenStreetMap ist ein Projekt, welches im Jahre 2004 von Steve Coast in Großbritannien gestartet wurde. Das Ziel ist eine freie Weltkarte, die für jeden frei verfügbar ist. Dabei bedeutet „frei“ nicht, dass es sich ausschließlich um kostenlose Daten handelt, sondern vielmehr, dass die Daten von jedem genutzt werden dürfen, beispielsweise für eine Anfahrtsskizze - oder in diesem Fall als Datengrundlage für das Projekt Radroutenplaner. [Ram08, vgl. S. 1 ff.]
Grundlage für die Verwendung der Daten ist die „Creative Commons Attribution-Share-Alike 2.0“-Lizenz [Ram08, vgl. S. 161]. Diese beinhaltet, dass die Daten zwar bearbeitet und genutzt werden dürfen, aber alle daraus entstehenden Produkte wieder frei zur Verfügung gestellt werden müssen. Andere Dienste, wie z. B. Google Maps stellen ebenfalls Kartenmaterial kostenlos zur Ansicht ins Netz. Dieses darf aber nicht für private oder gewerbliche Zwecke

weiter genutzt werden.
Um das OpenStreetMap-Projekt umzusetzen, sind User aus der ganzen Welt aktiv und zeichnen mit GPS-Geräten Daten, wie beispielsweise Straßen, Gebäude, Spielplätze, Restaurants, Tankstellen u. v. a. m. in ihren Heimatstädten, auf. Die verwendeten GPS-Geräte haben dabei zumeist eine (Un-)Genauigkeit von ca. 5 m (vgl. Kap. 2.8). Bis November 2009 hatten sich bereits fast 180.000 Benutzer weltweit für das Projekt engagiert (s. Abb. 1) und wie in Abb. 2 zu erkennen ist, nimmt diese Zahl stetig zu.

OpenStreetMap stats report run at Sun Nov 01 00:00:06 +0000 2009

Number of users	179305
Number of uploaded GPS points	1257230910
Number of nodes	480298744
Number of ways	35369196
Number of relations	263281

Abbildung 1: *OpenStreetMap-Statistik vom 01.11.2009*
Quelle: `http://www.openstreetmap.org/stats/data_stats.html` Abruf: 01.11.2009

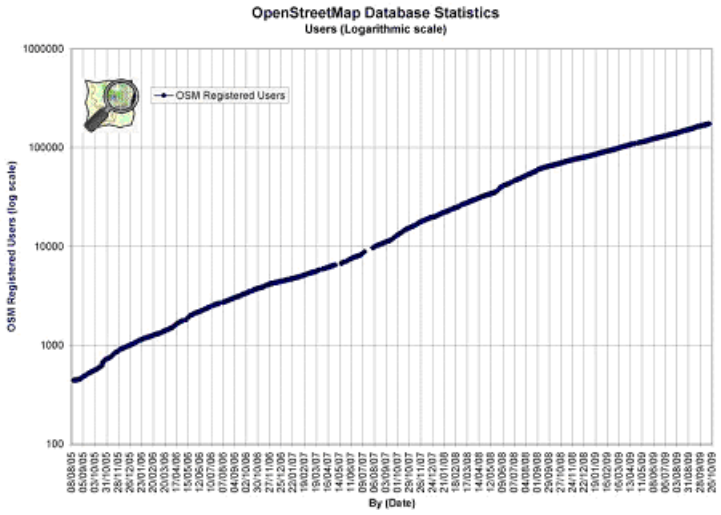

Abbildung 2: *Anstieg der Anzahl der User von 2005 bis 2009 (logarithmische Skala)*
Quelle:
`http://wiki.openstreetmap.org/wiki/Stats#OpenStreetMap_Statistics_Available`
Abruf: 01.11.2009

Auf diese Weise sind in vielen Gebieten bereits sehr detaillierte Karten vorhanden. Oftmals liegt die Detailgenauigkeit sogar über der von anderen Internetkartenwerken (z. B. Google oder Yahoo). Andere Regionen, vor allem die ländlichen, sind dagegen bisher nur rudimentär kartiert und können nicht mit proprietären Konkurrenten mithalten.
Allerdings birgt die Beteiligung von vielen Akteuren auch das Risiko, dass Daten „falsch“ erhoben oder geändert werden, in der Regel werden solche Verfälschungen aber zügig durch andere Nutzer oder automatische Skripte korrigiert. Neben der teilweise hohen Detaildichte

ist ein weiterer Vorteil von OSM, dass dem Nutzer nicht nur fertige Karten zur Verfügung stehen, wie z. B. bei Google Maps, sondern die zugrunde liegenden Geodaten auch für eigene Berechnungen verwendet werden können. Ob der Anspruch, eine freie Weltkarte zu erschaffen, umsetzbar ist, wird sich erst zeigen müssen. Bisher gibt es gute Datengrundlagen nur in Europa, Nordamerika und Teilen von Asien. Alle anderen Erdteile sind bis auf einige große Städte noch fast weiß. Das liegt hauptsächlich daran, dass die Projektteilnehmer mehrheitlich aus den erstgenannten Regionen stammen und daher vorwiegend dort GPS-Aufnahmen tätigen. Außerdem gibt es diverse Länder, die die GPS-Technik einzig dem Militär vorbehalten wollen (z. B. China). Hier sind legale GPS-Aufnahmen kaum möglich und diese Länder werden noch sehr lange weiße Flecken auf der OSM bleiben. Ein weiterer Faktor ist die schwache Verbreitung von Internetzugängen in den ärmeren Regionen der Erde.
Deshalb ist vor der Nutzung von OSM-Daten zuerst zu prüfen, ob die vorhandene Datendichte für den angestrebten Zweck ausreichend ist. Auch für vorliegende Projektarbeit ist eine Überprüfung, Vervollständigung und Aktualisierung der Daten unumgänglich, vor allem auch im Hinblick auf Attribute, die für ein auf Radfahrer zugeschnittenes Routing erforderlich sind und zumeist noch nicht vorhanden sind (vgl. Kap. 3). In diesem Zusammenhang ist auch die im nächsten Abschnitt beschriebene XML-Struktur der OSM-Daten von Bedeutung.

2.2.3 XML und die OpenStreetMap-XML-Struktur

Die eXtensible Markup Language (XML) [W3C04] ist ein vom World Wide Web Consortium vorgeschlagener Dokumentenverarbeitungsstandard, veröffentlich unter
`http://www.w3c.org/XML`.
XML ermöglicht dem Benutzer die Strukturierung seiner Daten mit Hilfe von selbstgewählten Tags. Mit Hilfe der eXtensible Stylesheet Language (XSL) [W3Cb] kann für die einzelnen Tags eine individuelle Darstellungsweise festgelegt werden, welche auf die zugehörigen Daten angewendet wird. Auf diese Weise wird eine Trennung zwischen Struktur, Inhalt und Layout erreicht. Typischerweise verteilen sich daher die Angaben zu den Daten auf drei Dateien:

- *.dtd: Document Type Definition mit der Strukturbeschreibung
- *.xml: XML-Datei mit den durch Tags markierten Daten
- *.xls: Stylesheet mit Angaben zum Rendern des Layouts

Ein XML-Parser kann zu einer vorliegenden XML-Datei ohne Angabe der zugehörigen DTD überprüfen, ob die XML-Datei wohlgeformt [W3Ca] ist, d. h. ob die grundsätzlichen Syntaxregeln eingehalten werden. Bei Vorlage der DTD kann der XML-Parser zusätzlich überprüfen, ob die Datei gültig ist, d. h. ob ihr Inhalt der Strukturbeschreibung gehorcht. Ein XSLT-Prozessor (eXtensible Stylesheet Language Transformation) rendert das Layout für die Daten der XML-Datei unter Anwendung des Stylesheets. Anstelle einer DTD kann die Struktur

auch durch ein XML Schema [W3Cd] beschrieben werden, welches selbst wiederum eine wohlgeformte XML-Datei darstellt.
Die OpenStreetMap-Daten werden in einer zentralen Datenbank gespeichert, welche die aktuelle Datenhaltung sowie eine History-Funktion beinhaltet. Beispiel für einen Auszug aus einer OSM-XML-Datei:

```
<?xml version='1.0' encoding='UTF-8'?>
<osm version='0.6' generator='JOSM'>
  ...
  <node id='408830040' ...>
    <tag k='highway' v='bus_stop' />
    ...
  </node>
  <way id='39498805' ...>
    <nd ref='56332865' />
    ...
    <tag k='highway' v='residential' />
    ...
  </way>
  <relation id='310502' ...>
<member type='way' ref='43103586' role='...' />
...
  </relation>
</osm>
```

Die OSM-XML-Struktur beinhaltet demnach vier große Ebenen:

- die Knoten (nodes), welche die einzigen Objekte sind, deren Koordinaten angegeben sind. Die anderen Objekte bestehen entweder direkt selbst aus Knoten (ways), oder aus Objekten, welche wiederum aus Knoten bestehen (relations).
- die Wege (ways), die aus einer Liste von Knoten bestehen, deren Reihenfolge die Richtung des Weges beschreibt.
- die Relationen (relations), welche aus Knoten und Wegen bestehen können, seine sogenannten `member`
- die Attribute zur Erfassung von Eigenschaften, die sogenannten Tags. Alle drei Objektklassen können durch Tags beschrieben sein.

OpenStreetMap lässt prinzipiell alle möglichen Schlüssel oder Werte zu. Trotzdem ist es natürlich für die Anwendung der OSM-Daten von Vorteil, wenn Einigkeit herrscht, welche

Basiseigenschaften durch welche Datenrepräsentation ausgedrückt werden. Deshalb sind die Map Features [Opef] ein Leitfaden (jedoch ausdrücklich keine Vorschrift) für solche Basiseigenschaften.
In Rahmen dieses Projektes spielt die OSM-XML-Struktur überall dort eine Rolle, wo die Eigenschaften von Straßen oder POIs berücksichtigt werden sollen. Das ist beispielsweise bei der Auswahl verschiedener Profile (vgl. Kap. 5.3) und bei der POI-Suche (vgl. Kap. 5.8) der Fall. Hierbei wird allerdings auf die zuvor in die Datenbank eingepflegten Daten zugegriffen.

2.3 Interaktivität

Da in diesem Projekt ein interaktiver Radroutenplaner entwickelt wird, ist es zunächst sinnvoll zu definieren, was Interaktivität bedeutet.
Der Begriff Interaktivität im Allgemeinen steht für eine Wechselbeziehung zwischen zwei oder mehr Objekten. Hier gilt es zwischen den Interaktivitätsbegriffen der Soziologie und der Informatik zu unterscheiden. Die Soziologie „[...] bezeichnet wechselseitig aufeinander bezogene menschliche Handlungen [...], also Beziehungen zwischen zwei oder mehreren Menschen. [...] Im Verständnis der Informatik bezeichnet *Interaktion* immer das Verhältnis von Mensch und Maschine, nicht aber die Kommunikation zwischen zwei Menschen mittels einer Maschine.“[BL04, S. 98 f.]
Nach diesen Definitionen wären also zwei verschiedene Interaktivitäten eines Routenplaners möglich. Zum einen könnte der Routenplaner mit dem Anwender interagieren, zum anderen könnten aber auch mehrere Anwender über den Routenplaner interagieren. Die bekannten Radroutenplaner sind in der Regel nur aus informatischer Sicht interaktiv. Sie „zeichnen sich dadurch aus, dass der Anwender individuelle Fahrradtouren mit ihnen erstellen kann. Die Tour wird dynamisch on the fly für den Nutzer erzeugt und ausgegeben.“[Str07]
Auch weitere Funktionen von Routenplanern wie z. B. die Suche von POIs (vgl. Kap. 5.8) sind hier denkbar. Es stehen dem Nutzer somit unterschiedliche Eingriffs- und Steuerungsmöglichkeiten zur Verfügung, um die individuellen Interessen bestmöglich umzusetzen.
Aber auch die soziologische Sichtweise der Interaktivität ist in einem Radroutenplaner zu realisieren. Soziologisch interaktive Radroutenplaner ermöglichen die Kommunikation zwischen verschiedenen Benutzern. Hier ist beispielsweise die Funktion der Routenbewertung zu erwähnen. Ein Anwender bewertet eine Route als besonders attraktiv, der Routenplaner empfiehlt diese spezielle Route daraufhin einem anderen Nutzer (vgl. Kap. 5.13). Ebenfalls hier zu nennen ist die Exportfunktion: Anwender können von ihnen erstellte Routen als GPX-Track exportieren und so direkt an andere Personen weitergeben.
Ziel eines interaktiven Radroutenplaners ist es also, beide Definitionen zu erfüllen und entsprechende Funktionen vorzuhalten.

2.4 Web 2.0

Im September 2005 beschrieb Tim O'Reilly in dem Artikel „What is Web 2.0: Design Patterns and Business Models for the Next Generation of Software“ Web 2.0 folgendermaßen: „Wie viele andere wichtige Konzepte, hat Web 2.0 keine genauen Begrenzungen, sondern vielmehr ein Gravitationszentrum. Man kann Web 2.0 als eine Ansammlung von Prinzipien und Praktiken visualisieren, die ein regelrechtes Sonnensystem von Seiten zusammenhalten, die einige oder alle dieser Prinzipien in unterschiedlicher Entfernung vom Zentrum demonstrieren“[Hol08].
Um das Verständnis von Web 2.0 visuell zu demonstrieren, sind in Abbildung 3 die Weiterentwicklungen von Web 1.0 zu Web 2.0 dargestellt. Die Abbildung 3 soll keine abgeschlossene

Web 1.0		Web 2.0
DoubleClick	->	Google AdSense
Ofoto	->	Flickr
Akamai	->	BitTorrent
mp3.com	->	Napster
Britannica Online	->	Wikipedia
personal websites	->	Blogging
evite	->	upcoming.org und EVDB
Domain-Namen Spekulation	->	Suchmaschinen-Optimierung
page views	->	Kosten pro Klick
Screen Scraping	->	Web-Services
Veröffentlichung	->	Teilnahme
Content Management Systeme	->	Wikis
Verzeichnisse (Taxonomie)	->	tagging ("folksonomy")
Klebrigkeit	->	Syndication

Abbildung 3: *Weiterentwicklungen von Web 1.0 zu Web 2.0*
Quelle: `http://oreilly.com/web2/archive/what-is-web-20.html` Abruf: 20.01.10

Aufzählung von Webanwendungen darstellen, zumal immer wieder interessante neue Anwendungsmöglichkeiten und Seiten entstehen, sondern vielmehr die rasante Weiterentwicklung des Webs veranschaulichen. In Zeiten von Web 1.0 waren die Nutzer des Internets eher darauf bedacht, Informationen passiv zu konsumieren, in der heutigen Zeit und vor dem Hintergrund des Web 2.0 besteht die Möglichkeit die neuen Interaktivitätsfunktionen des Netzes aktiv zu nutzen. Dadurch wird die Autonomie der Nutzer gestärkt, sie werden im weiteren Sinne nun selber zum Redakteur, Kommentator oder Networker und bestimmen selbst ihr Informations- und Beziehungsmanagement im Internet. Die weitläufige Verbreitung und aktuellen Funktionen haben einen höheren Grad an Nutzerengagement und (Meinungs-) Macht zur Folge. Dies wird die allgemeine soziale Vernetzung und das gesamte Konsumverhalten merklich beeinflussen. [Fra]
Um das Web 2.0 so genau wie möglich zu spezifizieren, fasste Tim O'Reilly bestimmte Schlüsselprinzipien von Anwendungen zusammen, dazu gehören unter anderem, dass das Web als Plattform fungiert und kein lokaler Rechner benutzt wird, dass Inhalte als wichtiger zu erachten sind als das Aussehen, dass die Architektur des Mitwirkens im Vordergrund steht, d. h. jeder kann sich aktiv beteiligen, dass der klassische Softwarelebenszyklus nicht mehr als

aktuell gilt, sondern dass sich alle Projekte fortlaufend im Beta-Stadium befinden. [Hol]
Diese und eine Reihe weiterer Prinzipien veranlassten Markus Angermeier im Jahre 2005 die Abbildung 4 zu veröffentlichen, um das Web 2.0 zu visualisieren. In der Abbildung 4 sieht

Abbildung 4: *Web 2.0 Mindmap*
Quelle: `http://kosmar.de/archives/2005/11/24/netz20-gedankenkarte/` Abruf: 20.01.10

man deutlich, dass Web 2.0 kein starrer Begriff des Internets ist, sondern dass eine Menge von Komponenten zusammenfließen muss, um den allumfassenden Begriff Web 2.0 mit Leben zu füllen. Zieht man darüber hinaus die erstaunliche Trefferquote von 470.000.000 (Stand 12.12.2009) bei der Google-Suche nach dem Begriff Web 2.0 in Betracht, so wird deutlich, dass eine neue Generation des Webs entstanden ist.
Auch im Rahmen des Studienprojektes besitzt Web 2.0 eine große Bedeutung, da besonders soziale Funktionen eine wichtige Rolle spielen und der Radroutenplaner diese in Form von Bewertungen und Weitergabe von Routen realisiert (vgl. Kap. 5.13). Letztlich bleibt die genaue Definition von Web 2.0 jedoch offen, da die Weiterentwicklung ständig voranschreitet. Dennoch spricht man in der heutigen Zeit von einer Neudefinition des sog. „alten Internets“.

2.5 WebMapping

Kartenanwendungen werden immer häufiger genutzt. Dabei gewinnen gerade interaktive Karten, d. h. Karten mit denen ein Nutzer interagieren kann, gegenüber statischen Karten immer mehr an Bedeutung. Wie bereits in den beiden vorherigen Abschnitten erläutert, stellt auch der zu entwickelnde Radroutenplaner eine interaktive WebMapping-Anwendung dar, die es dem Nutzer zum einen ermöglicht individuell zu bestimmen, welche Informationen er abrufen möchte und zum anderen die Möglichkeit bietet, diese mit anderen Personen zu teilen.

Im Folgenden werden die Bestandteile und die Funktionsweise einer WebMapping-Anwendung vorgestellt. Zu diesem Zweck wird zunächst der Begriff des Webservers erläutert und anschließend kurz das Zusammenspiel der einzelnen Komponenten in einer WebMapping-Anwendung dargestellt.

2.5.1 Webserver

Unter einem Webserver versteht man einen Computer mit einer Webserver-Software bzw. lediglich die Webserver-Software selbst. Über diesen werden Webseiten im Internet oder auch Intranet für alle Nutzer zur Verfügung gestellt. Im Grunde steht hinter jeder Internet-Adresse ein Webserver. Durch den Aufruf einer Website wird der Webserver der betreffenden Seite aufgerufen und schickt die Antwort in Form einer in der Hypertext Markup Language (HTML) (vgl. Kapitel 4.6) codierten Homepage zurück. Durch die Verarbeitung der gesendeten Texte und Grafiken durch einen Browser entsteht dann eine lesbare Seite.
Um die Übertragung sicher zu stellen, werden in der Regel Protokolle, sogenannte Übertragungsprotokolle (HTTP, HTTPS) bzw. Netzwerkprotokolle (TCP/IP) verwendet, üblicherweise über die Ports 80 (HTTP) und 443 (HTTPS). Unter einem Port versteht man den „Ein- oder Ausgang einer Einheit. Es kann sich um einen Verbindungspunkt für ein Peripheriegerät, für periphere Einheiten oder ein Anwendungsprogramm handeln. Ein Port kann logisch, physikalisch oder beides sein.“[IT-b]
Geschichtlich gesehen hatte der Webserver seine Anfänge im Jahre 1989 in Person von Tim Berners-Lee, der die Vermutungen aufstellte, dass miteinander verbundene Informationen nützlicher seien als eine einfache statische Hierarchie, und dass Anzeige- und Speichersoftware voneinander getrennt sein müssten. Um seine Visionen zu verwirklichen, entwickelte Berners-Lee den ersten Webserver, genannt CERN httpd, sowie den allerersten Webbrowser WorldWideWeb, der auf Unix und VMS bis 1996 weiterentwickelt wurde. Zwei Jahre zuvor hatte Berners-Lee das World Wide Web Consortium gegründet, um die Entwicklungen weiter voranzutreiben. [Nie]
Die bei einem Webserver dargestellten Dokumente können sowohl statische, d. h. unveränderliche Dateien (z. B. Bild-Dateien), als auch dynamisch erzeugte Dateien sein, die von einem Benutzer individuell verändert werden. Um eine komplette Webseite darzustellen bedarf es mehrerer Komponenten. Neben der HTML-Seite müssen auch Designbeschreibungen (CSS) und Bilddateien übertragen werden, jeweils als einzelne Dateien für sich. Oftmals sind für die Darstellung von komplexen Webseiten hunderte von Anfragen und Serverantworten nötig, da für jede einzelne Datei eine Anfrage vom Webbrowser an den Webserver gestellt werden muss. [Wika]
Üblicherweise werden alle Anfragen in einer sogenannten Logdatei protokolliert, in der alle wichtigen Statistiken geführt werden, beispielsweise die Anzahl der Zugriffe auf eine Webseite. Als Logfile bezeichnet man „eine Datei mit der Prozesse, die in Computern und Netzwerken ablaufen, aufgezeichnet werden. Logfiles sind wichtige Informationsquellen, um die aktuelle

Situation in einem Netzwerk zu erfassen. Anhand von Logfiles können zu einem späteren Zeitpunkt Fehler in der Datenübertragung, der Verlust von Daten oder der Absturz eines Rechners analysiert werden. Auch die Regeneration von verlorenen Daten ist über Logfiles möglich“[IT-a].
Um die Daten so genau wie möglich zu halten, werden Cookies und dynamische Seiten benutzt, da die Zuordnung von einem Nutzer mittels seiner IP-Adresse über LAN nicht immer eindeutig ist. Auf diese Weise werden eventuell verfälschte Angaben auf Webseiten bezüglich Hits oder Visits wieder genauer. [Wika]
Die bedeutendsten Webserver sind in der heutigen Zeit der Apache HTTP Server und Microsoft Internet Information Services (IIS). In vorliegendem Projekt wird der Apache Webserver verwendet, da er ein Open Source Produkt ist und somit jedem frei zur Verfügung steht. Nähere Informationen über den Apache Webserver sind in Kapitel 4.2 zu finden.

2.5.2 Architektur einer WebMapping-Anwendung

Bei WebMapping-Anwendungen wird eine serverseitige Architektur aufgebaut, in der die Komponenten Webserver, MapServer und Geodaten interagieren. Der Nutzer an einem Web-

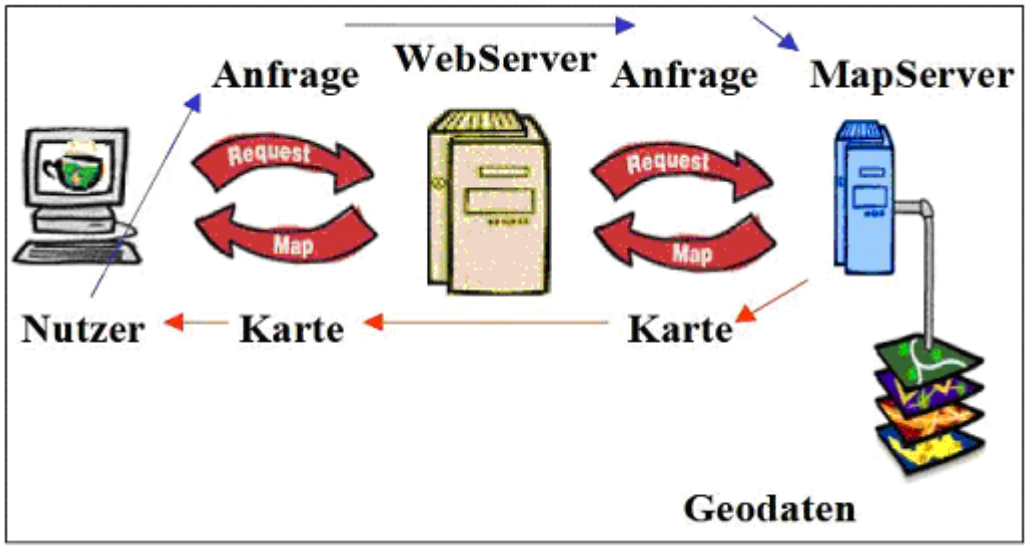

Abbildung 5: *Einfache WebGIS-Client-Server-Architektur*
Quelle: `http://www.mygeo.info/skripte/Praxishandbuch_WebGIS_Freie_Software.pdf` S. 41, Abruf: 18.02.2010

browser fordert eine Karte von einer Webseite an, indem er im Mapping-Client über das HTTP-Protokoll eine Anfrage an den Webserver stellt. Per Common Gateway Interface, kurz CGI, kommuniziert der Webserver im nächsten Schritt mit dem MapServer, der georeferenzierte Daten im Internet darstellen lassen kann. Die Geodaten können sowohl als Vektor- oder Rasterfile vorliegen oder aber auch in einer Datenbank (z. B. PostgreSQL/PostGIS) gespei-

chert sein. Es wird eine Karte des vom Nutzer gewünschten Ausschnitt des Bildes erzeugt, an den Webserver zurückgegeben und anschließend im Webbrowser dargestellt. [Tra04, vgl. S. 41 f.]
Die Vorteile der Speicherung von Geodaten in einer Datenbank liegen in der besseren Performance und strukturierterer Datenorganisation. In diesem Projekt wird die PostgreSQL-Erweiterung PostGIS verwendet (vgl. Kap. 4.4), die GIS-Funktionalität bietet. Als MapServer wird der UMN MapServer (vgl. Kap. 4.3) eingesetzt, der die Einbindung von PostGIS-Daten unterstützt. Durch das Zusammenspiel der einzelnen Komponenten ist es dem Nutzer möglich, eine dynamische Karte zu erhalten, auf welcher er frei navigieren kann und welche seinen Ansprüchen entsprechend gestaltet ist. [Tra04, vgl. S. 40]

2.6 Open Source Geospatial Consortium und OpenGIS Standards

Das Open Geospatial Consortium, Inc (OGC) ist ein internationales Konsortium von aktuell 386 Unternehmen, Regierungsbehörden und Universitäten, die gemeinsam einen offenen Schnittstellenstandard entwickeln. [Opea] Zu den Mitgliedern des Konsortiums, das seit 1994 existiert und bis 2004 unter dem Namen OpenGIS Consortium agierte, gehören namhafte Firmen wie Microsoft, Google, die NASA oder auch ESRI. [Opeb]
Ziel des OGC ist es, die Entwicklung von raumbezogener Informationsverarbeitung (insbesondere Geodaten) auf Basis allgemeingültiger Standards zum Zweck der Interoperabilität festzulegen. Durch die Definition von Schnittstellen ist es möglich, proprietäre Software im Verborgenen und für niemand Fremden einsehbar laufen zu lassen. Allerdings kann diese Software über ein Interface nach festgelegten Regeln mit anderen Programmen kommunizieren. Interoperabilität der Schnittstelle bedeutet hier die Möglichkeit des Datenaustauschs zwischen Systemen in Echtzeit, wodurch z. B. Client/Server-Anfragen mit Geodaten überhaupt erst möglich werden. Interoperabilität bezieht sich aber auch darauf, mit verschiedenen Betriebssystemen zu kommunizieren, meint aber ebenso, dass verschieden alte Versionen ein und derselben Schnittstelle miteinander kompatibel sein sollen, sowohl in die eine wie in die andere Richtung. Unter dem Markennamen OpenGIS® werden verschiedene offene Standards für Schnittstellen entwickelt.
Offene Standards werden auf der Basis frei verfügbarer Spezifikationen entwickelt, die von abstrakten Beschreibungen des Aufbaus, der Komponenten und der Funktionsweise eines dienstebasierten GIS im Sinne des OGC bis hin zu detaillierten Spezifikationen der Implementation der Dienste reichen. Es wird dabei jedoch nicht die konkrete Umsetzung der Software vorgeschrieben, sondern die verschiedenen Schnittstellen eines Dienstes, dessen Eigenschaften und Verhalten festgelegt. Wichtige durch das OGC entwickelten Standards sind:

- OpenGIS Implementation Specification for Geographic information
- OpenGIS Geography Markup Language (GML)
- OpenGIS Web Map Service (WMS) Implementation Specification

- OpenGIS Web Feature Service (WFS) Implementation Specification
- OpenGIS Location Service (OpenLS) Implementation Standard

Als eine Untermenge der *OpenGIS Implementation Specification for Geographic information* ist für dieses Projekt der Teil von Bedeutung, der als *Simple feature access* bezeichnet wird. In diesem Datenmodell geht es um eine einheitliche Spezifikation von maximal zweidimensionalen Vektorgeometrien, deren Stützpunkte geradlinig miteinander verbunden sind. Durch das Modell werden neben Datentypen auch räumliche Operationen für den Zugriff und die Anfrage und Verarbeitung von Geometrien definiert.
Das Modell besteht grundsätzlich aus zwei Teilen:

- Part 1: Common architecture
- Part 2: SQL Option

Es werden weiterhin zwei Standardwege definiert, um räumliche Objekte (Simple Features) zu repräsentieren:

- Das Well-Known Text (WKT) Format (Textrepräsentation)
- Das Well-Known Binary (WKB) Format (Binäre Speicherrepräsentation)

In beiden Versionen werden Informationen über den Geometrietyp und die Koordinaten des Objektes abgespeichert. Auf *fahrradies.net* wird dieses Modell speziell beim Routing (vgl. Kap. 5.1) und der Routenbeschreibung (vgl. Kap. 5.6) genutzt, hier wird die Route als WKT angegeben.
Außerdem wird der *OpenGIS Location Service (OpenLS) Implementation Standard* in diesem Projekt genutzt. *OpenGIS Location Services (OpenLS)* ermöglichen es Nutzern über ein mobiles Endgerät ortsbezogene Serviceleistungen wie Routeninformationen oder Umfeldangaben zur Verfügung zu stellen. Durch die Spezifikation des OpenLS wird versucht, ein störungsfreies Zusammenspiel der einzelnen Dienste zu ermöglichen. OpenLS sorgt dafür, dass der Austausch von Informationen zwischen den einzelnen Netzdiensten durch abstrakte Datentypen (ADT) geschieht, die ein OpenLS-konformer Netzdienst als Ergebnis auf eine Anfrage liefert und die für die Kommunikation innerhalb der Dienste verwendet werden. Die Netzdienste werden dabei als „Core Services“ bezeichnet. Core Services und ADTs zusammen beschreiben den „GeoMobility Server“, die OpenLS-Plattform.
Die OpenLS-Spezifikation ist als Bauplan für Netzdienste anzusehen. Inhalt ist eine detaillierte technische Beschreibung der Schnittstellen. Die Implementierung des Netzdienstes, der diese Schnittstellen anbietet, liegt aber in der Verantwortung des Dienstanbieters.
Auf *fahrradies.net* werden unter anderem OpenLS-konforme Routenlinks zur Verfügung gestellt.

2.7 Web Map Service und Web Feature Service

Der **Web Map Service (WMS)** ist ein Dienst zur Bereitstellung und Anforderung von dynamischen Karten. Es wird ein beliebiger rechteckiger zweidimensionaler Ausschnitt als Rasterbild zur Verfügung gestellt. Die Datengrundlage bilden Raster- und Vektordaten. Der WMS wurde, wie im vorherigen Kapitel erwähnt, durch das OGC standardisiert und kann über das Internet-Protokoll angesprochen werden. Für einen WMS werden verschiedene Requests definiert, über die die Kommunikation zwischen Client und Server stattfindet [Mit08, vgl. S.238].
Ein OGC-konformer WMS unterstützt drei Operationen, die über folgende Requests via Browser in der URL oder von GIS- Programmen angefragt werden können:

- GetCapabilities liefert als Antwort eine XML-Datei mit Metadaten zurück, die den WMS hinsichtlich seiner Funktionalitäten beschreibt. Dazu gehören neben allgemeinen Informationen die verfügbaren Operationen, Layer, Projektionen und Koordinatenreferenzsysteme.

- GetMap ist die Operation, um eine dynamisch erzeugte Karte als Rasterbild vom WMS anzufordern. Dazu müssen verschiedene Parameter angegeben werden, z. B. der Request-Typ, die WMS-Version, die Projektion, das Ausgabeformat, die Liste der Layernamen, die Höhe und Breite des Ausgabebildes, die Bounding Box sowie die Darstellungsoptionen für die einzelnen Kartenlayer.

- GetFeatureInfo liefert Informationen zu Objekten der Karte. Diese Operation ist optional, wird aber gegenwärtig von den meisten Servern unterstützt. Voraussetzung ist, dass der entsprechende Layer abfragbar ist [Mit08, vgl. S.239 ff.].

Der WMS wird in dieser Projektarbeit für die Druckfunktion (vgl. Kap. 5.12) eingesetzt. Der WFS wird zwar nicht verwendet, war aber ursprünglich für die Rückgabe der Ergebnisse der Points of Interest-Suche (vgl. Kap. 5.8) vorgesehen und soll an dieser Stelle der Vollständigkeit halber erwähnt werden.

Der **Web Feature Service (WFS)** ermöglicht die Recherche nach raumbezogenen Informationen aus beliebigen objektorientierten Vektordatenbeständen. Der WFS ist ebenfalls ein OGC-Standard, liefert im Gegensatz zum WMS aber kein Bild, sondern die Geodaten selbst als XML-Datei im Format GML (Geography Markup Language). Die Geodaten können räumlich oder thematisch selektiert werden. Ein Basis-WFS, wie beispielsweise der UMN MapServer, unterstützt die folgenden Operationen, die über einen Browser in der URL oder von GIS-Programmen angefragt werden können:

- GetCapabilities ist die Operation zur Abfrage der Fähigkeiten des WFS, die eine XML-Datei mit Metadaten zurückliefert, die u. a. die Feature Types enthält.

- DescribeFeatureType ist die Operation, die die Struktur der einzelnen Feature Types beschreibt.
- GetFeature fordert Feature Types an. Die Geometrien der einzelnen Features werden im GML-Format zurückgegeben.

Des Weiteren existiert ein Transactional-WFS, der den Basic-WFS um schreibenden Zugriff erweitert. Ein WFS bietet gegenüber einem WMS eine größere Flexibilität und kann für verschiedene Zwecke eingesetzt werden, wie z. B. Objektsuche, Zoomen und Markieren, Abfrage, Analyse sowie Verarbeitung und Weitergabe von Geometrien [Mit08, vgl. S.253 ff.]

2.8 Global Positioning System

Beim **Global Positioning System (GPS)** handelt es sich um ein „satellitengestütztes Navigationssystem, das vom US-Militär entwickelt, betrieben und kontrolliert wird, um die sofortige Positionierung eines beliebigen Objektes auf der Erdoberfläche zu ermöglichen“[Mit08, S. 216].
Es setzt sich aus einem Raum-, einem Kontroll- und einem Nutzersegment zusammen. Das Raumsegment besteht aus mindestens 24 Satelliten auf sechs Umlaufbahnen, welche in einem Winkel von 55 Grad gegen die Äquatorebene geneigt sind. Durch diese Anordnung wird gewährleistet, dass für jeden Punkt der Erde zwischen fünf und acht Satelliten gleichzeitig sichtbar sind, welche laufend die genaue Uhrzeit, Statusinformationen und Orbitaldaten senden. Das Kontrollsegment, bestehend aus fünf Bodenstationen, dient der Beobachtung und Kontrolle der Satelliten. Die GPS-Empfänger und die Anwender bilden das Nutzersegment [Opee]. Jeder Satellit sendet auf zwei Trägerwellen mittels Phasenmodulation einen Code an den GPS-Empfänger. Die L1-Trägerwelle ist zivil nutzbar, während die Nutzung der L2-Trägerwelle dem US-Militär vorbehalten ist. Durch Laufzeitmessung zwischen dem empfangenen Signal und dem im Empfänger erzeugten Signal wird die Entfernung zum Satelliten berechnet. Diese ist eine sog. Pseudoentfernung, da sich das Signal aufgrund des nicht vorhandenen Vakuums nicht exakt mit Lichtgeschwindigkeit ausbreitet. Zur Positionsbestimmung sind mindestens vier Satelliten notwendig. Da mit den x-, y- und z-Koordinaten der Satellitenposition sowie dem Zeitfehler, der durch den Zeitunterschied zwischen Atomuhr im Satelliten und Quarzuhr im Empfänger verursacht wird, insgesamt vier Unbekannte vorliegen, können diese nur mit Daten von vier Satelliten bestimmt werden. Die Koordinaten liegen im WGS84 vor, so dass Transformationen in verschiedene Benutzer-Koordinatensysteme notwendig werden. [Mit08, S. 219 f.]
Die Genauigkeit der Positionsbestimmung hängt u. a. von der Anzahl der verfügbaren Satelliten ab, da bei Überbestimmtheit Ausgleichsrechnungen erfolgen können [Opee, Kap. 3.3.3]. Mit Standard-GPS-Geräten lassen sich Genauigkeiten im Bereich bis ca. 5 Meter erzielen. Beim Differentiellen GPS finden Messungen mit zwei gleichzeitig betriebenen GPS-Empfängern statt. Von einer exakt vermessenen Referenzstation werden Korrekturdaten an

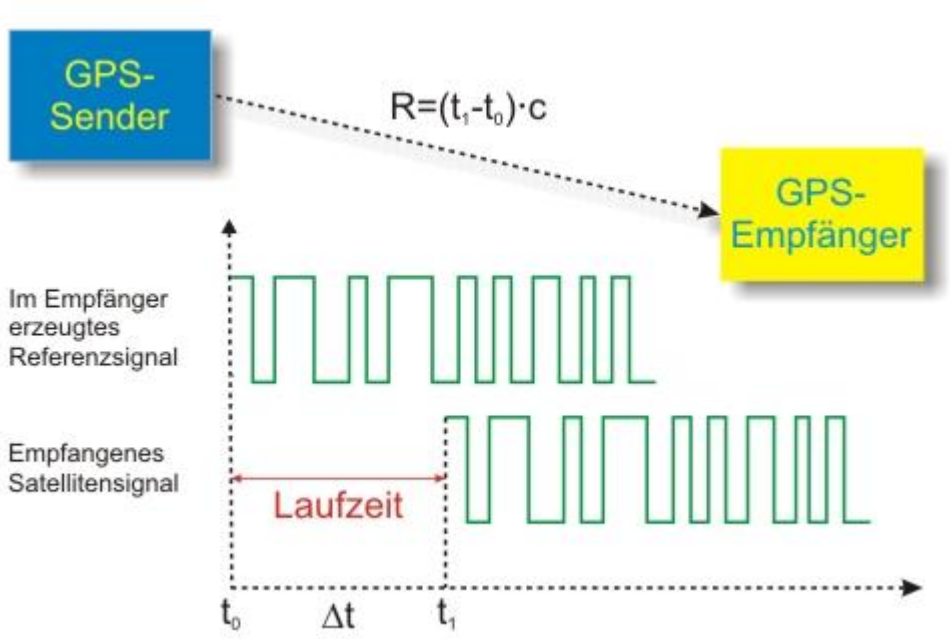

Abbildung 6: *Prinzip der Distanzbestimmung über Laufzeiten von Signalen*
Quelle: `http://www.fergi.uni-osnabrueck.de/module/gps1/` Fernstudienmaterialien Geoinformatik - FerGI-online, Modul: Koordinatensysteme und GPS

den Empfänger im Gelände übermittelt, so dass Genauigkeiten im Zentimeter- und sogar im Millimeterbereich erzielt werden können. [Mit08, vgl. S. 221 f.]
Die Genauigkeit der im Privatgebrauch eingesetzten GPS-Handgeräte lässt sich nicht mit den Genauigkeitsansprüchen der geodätischen Geräte vergleichen. Die mit den GPS-Handgeräten erzielte Genauigkeit von ca. 5 Meter ist für die Daten in OpenStreetMap ausreichend. Für eine bessere Genauigkeit ist es möglich die Strecken mehrfach abzufahren und die Daten anschließend (per Hand) zu „mitteln“. Für dieses Projekt wurden mit Hilfe von GPS-Handgeräten sowohl die Lagekoordinaten von POIs als auch der Verlauf von Straßen bzw. Routen erfasst. Die aufgenommen Daten wurden in den JOSM-Editor (vgl. Kap. 3.2) geladen und verarbeitet. Auf diese Weise konnten noch nicht vorhandene Straßen neu eingepflegt, ungenaue Verläufe korrigiert und die OSM-Daten somit aktualisiert werden (vgl. Kap. 3).

2.9 Geodätische Bezugssysteme und Höhenangaben

An dieser Stelle sollen einige grundsätzliche Informationen über Lage- und Höhenbezugssysteme geliefert werden, die für dieses Projekt relevant sind, um einen korrekten Raumbezug herzustellen.

Die Erde lässt sich mathematisch als ein an den Polen abgeplattetes Rotationsellipsoid beschreiben. Aufgrund der regional variierenden Oberflächengestalt der Erde erfolgt eine Anpassung durch verschiedene lokale Ellipsoide, die für die unterschiedlichen Regionen der Erde

jeweils die beste Annäherung darstellen [Mit08, vgl. S.181 ff.]. Auf diesen Referenzellipsoiden basieren wiederum verschiedene Kartenbezugssysteme.
Das WGS84-Ellipsoid ist die Bezugsfläche für das weltweit gültige World Geodetic System 1984 (WGS84). Dabei handelt es sich um ein globales geozentrisches Bezugssystem mit kartesischen X-, Y- und Z-Koordinaten. Der Ursprung des Koordinatensystems befindet sich im Massenmittelpunkt der Erde. Ein Anwendungsbeispiel ist die Positionsbestimmung und Höhenmessung mit GPS, die auf dem WGS84-Ellipsoid basiert.
Des Weiteren ist vor allem in Deutschland das Bessel-Ellipsoid gebräuchlich, welches als Bezugsfläche für Gauß-Krüger-Koordinaten dient.
Für interaktive Karten wird das Spherical-Mercator-System verwendet, das eine Mercator-Projektion benutzt, die auf einer Kugel statt auf einem Ellipsoid basiert. Diese Art der Kartenprojektion wird u. a. von Google Maps, Microsoft Virtual Earth und Yahoo! Maps verwendet und ist allgemein auch als „Google-Projektion“ bekannt. Auch alle Raster-Kacheln von OpenStreetMap liegen in der Spherical-Mercator-Projektion vor. Die Koordinaten werden in Metern angegeben [Opee].

EPSG-Codes

Die 1986 gebildete European Petroleum Survey Group (EPSG), seit 2005 OGP Surveying and Positioning Committee, pflegt und veröffentlicht einen Datensatz mit Parametern für Koordinatenbezugssysteme und Transformationen. Die Daten werden weltweit für eine Vielzahl von Bezugssystemen vorgehalten. Es werden sowohl Koordinatenbezugssysteme (Coordinate Reference Systems - CRS) definiert, um Koordinaten eindeutig zu identifizieren, als auch Operationen, um Koordinaten von einem Bezugssystem in ein anderes zu transformieren [OGPa][OGPb].
Jedem Bezugssystem ist ein EPSG-Code zugeordnet. Die folgenden EPSG-Codes werden im vorliegenden Projekt verwendet:

- 4326: WGS84 - Geographische Koordinaten basierend auf dem WGS84-Ellipsoid
- 31467: DHDN Zone 3 - Gauß-Krüger-Koordinaten basierend auf dem Bessel-Ellipsoid
- 900913: Google-Projektion - Spherical Mercator basierend auf einer Kugel

Höhenbezugsflächen

Die topographische Oberfläche der Erde besteht aus Hügeln, Bergen, Senken etc. Das Geoid berücksichtigt Unterschiede in der Gravitation und beschreibt eine Fläche gleicher Schwere. Es handelt sich um eine irreguläre Oberfläche, die der unter den Kontinenten fortgesetzten, ruhenden Meeresoberfläche nahe kommt. Das Geoid gibt sozusagen Normalnull für die gesamte Erde an. Dagegen ist ein Ellipsoid ein mathematischer Körper, der den Geoiden so gut

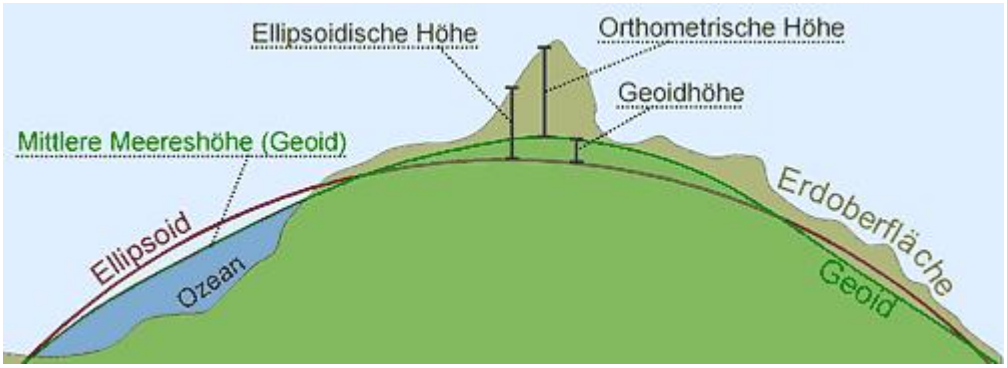

Abbildung 7: *Unterschiede zwischen Ellipsoid, Geoid (mittlere Meereshöhe) und der tatsächlichen Erdoberfläche.*

Quelle: `http://www.kowoma.de/gps/geo/mapdatum.htm` Abruf: 28.04.2010

wie möglich zu beschreiben versucht, aber jeweils nur für einen kleinen Bereich der Erde gut passt (s. Abb. 7). Die ellipsoidische Höhe gibt den Abstand eines Punktes vom Referenzellipsoid an, die orthometrische Höhe den Abstand eines Punktes über dem Geoid. Die Geoidhöhe bzw. Geoidundulation ist definiert als der Abstand des Geoids vom geodätischen Ellipsoid [DOR].
Höhen, die mittels GPS bestimmt werden, sind zunächst Höhen über einem Ellipsoid, werden aber umgerechnet in Höhen über dem Geoid.
Die Höhendaten der Shuttle Radar Topography Mission (SRTM)(vgl. Kap. 2.10) beziehen sich auf das WGS84-Ellipsoid und sind lagemäßig im WGS84-Datum bestimmt. Sie werden genutzt, um Steigungen zu berücksichtigen (vgl. Kap. 5.4) und um ein Höhenprofil (vgl. Kap. 5.5) zu erstellen. Da Höhen und Höhenunterschiede besonders für Fahrradfahrer eine wichtige Information darstellen, sind sie ein wesentlicher Bestandteil des Radroutenplaners.

2.10 Digitales Höhenmodell und SRTM-Höhendaten

Ein **Digitales Höhenmodell** (DHM) bzw. Digital Elevation Model (DEM) besteht aus digital gespeicherten Höhenwerten von Stützpunkten, die regelmäßig oder unregelmäßig über das Gelände verteilt sein können. Auf diese Weise wird die Geländeoberfläche lage- und höhenmäßig abgebildet. Die Stützpunkte können als regelmäßiges Gitter oder unregelmäßiges Dreiecksnetz (TIN) angeordnet werden. Da jedem Punkt in der Ebene exakt ein Höhenwert zugeordnet wird, spricht man bei dieser Art der Modellierung von einem 2,5D-Modell. [Pro03]
Die Begriffe Digitales Höhenmodell und Digitales Geländemodell (DGM) werden zwar häufig synonym verwendet, das DGM beinhaltet aber im Gegensatz zum DHM neben der Höheninformation zusätzlich noch weitere Parameter wie Neigung und Exposition [UNI08].
Ein DHM bzw. DGM lässt sich auf verschiedene Arten erfassen: z. B. durch dreidimensionale photogrammetrische Auswertung, mittels Laser Scanning oder durch terrestrische Vermessung. Nach Interpolation zwischen den gemessenen Punkten lassen sich Höhenlinien konstruieren. Weitere Repräsentationsformen sind Schummerungsmodelle, Höhenstufenkarten oder

Perspektivansichten. [DOR]

Im Februar 2000 wurde die **Shuttle Radar Topography Mission** (SRTM) durchgeführt mit dem Ziel ein hochauflösendes digitales Höhenmodell mit nahezu globaler Abdeckung zu erhalten. Die SRTM-Mission setzte die Single-Pass-Interferometrie erstmals weltraumgestützt ein. Diese Konfiguration wurde möglich, indem zusätzlich zur Hauptantenne an der Ladebucht des Spaceshuttles eine weitere Antenne am Ende eines 60 m langen Mastes angebracht wurde. Dabei kamen zwei unterschiedliche Wellenlängen zum Einsatz, neben dem amerikanischen C-Band-System (SIR-C) mit einer Wellenlänge von 6,0 cm auch das deutsch-italienische X-Band-System (X-SAR) mit einer Wellenlänge von 3,1 cm. Mit Hilfe des C-Bandes war es möglich die Erdoberfläche zwischen 60° Nord und 56° Süd abzudecken. Durch das X-Band konnte eine höhere relative vertikale Genauigkeit erreicht werden. [KHL02]
Aus den aufgenommenen Daten wurden Digitale Höhenmodelle erzeugt. Das X-SAR-Höhenmodell besitzt eine Auflösung von 1" x 1" (ca. 30 m x 30 m) und ist damit genauer als alle bisherigen globalen Höhendatensätze. Die X-Band-Daten werden vom Deutschen Zentrum für Luft- und Raumfahrt (DLR) verarbeitet und vertrieben [Deu]. Das C-Band-Höhenmodell mit einer Auflösung von 3" x 3" (ca. 90 m x 90 m) ist kostenfrei von der US Geological Survey (USGS) zu beziehen [USG].
Im Rahmen dieses Projektes werden, vor dem Hintergrund der Open Source, SRTM-Daten des frei verfügbaren C-Bandes verwendet. Sie bilden die Grundlage zur Einbindung von Steigung und Höhenprofilen in den Radroutenplaner (vgl. Kap. 5.4 und Kap. 5.5). Die Höhenangaben beziehen sich auf das WGS84-Ellipsoid, geben also keine orthometrischen Höhen an. Außerdem ist zu beachten, dass die Daten nicht unbedingt die Höhe des Erdbodens wiedergeben, sondern die Oberflächenstruktur der Erde mitsamt Bebauung und Bewuchs. Datenlücken, die u. a. durch das grobe Raster von 90 m x 90 m entstehen, werden linear interpoliert, was zu weiteren Ungenauigkeiten führt.

2.11 Geocoding

Unter dem Begriff „Geocoding" oder auch „Geokodierung" ist die Zuweisung von Koordinaten zu Geoobjekten zu verstehen. Beschreibungen über die Lage eines Objektes, wie beispielsweise ein Straßenname, der Name einer Sehenswürdigkeit oder eine Postleitzahl, werden genutzt, um daraus eine räumliche Darstellung abzuleiten. Diese Informationen können so für räumliche Anwendungen genutzt werden. [Ait09, Vgl. S. 157] Beispiele hierfür sind Umkreissuche oder Entfernungsbestimmung.
Im Internet werden verschiedene Geocoding-Dienste angeboten, z.B. auf den Seiten `http://www.geokodierung.net/` oder `http://travelgis.com/geocode/`, jeweils basierend auf der Google Maps API. Auf letzterer wird zusätzlich ein Reverse-Geocoding bereitgestellt, bei dem ausgehend von den Koordinatenangaben in Länge und Breite die zugehörige Adresse bzw. Position in der Karte bestimmt wird. Ein Beispiel für einen auf Yahoo! Maps basie-

renden Dienst ist auf der Seite `http://www.mapbuilder.net/demo/yahoo.geocode.php` zu finden.
Das Geocoding ist eine der grundlegendsten Funktionen des Radroutenplaners und eine Voraussetzung dafür, dass überhaupt eine Route berechnen werden kann. Speziell ist damit die Umwandlung des Namens eines Ortes, einer Straße oder eines POI in die zugehörigen Koordinaten gemeint, so dass diese in der Karte angezeigt oder für das Routing genutzt werden können. Bei der Straßensuche gibt der Nutzer den Namen der Straße des gewünschten Start- und Endpunktes in ein Formular ein. Aus dieser Eingabe wird jeweils die zu den Orten bzw. Straßen gehörige Koordinate ermittelt und in der Karte angezeigt. Zwischen den durch das Geocoding ermittelten Koordinaten kann anschließend das Routing durchgeführt werden. Ausführlich wird das Geocoding im Kapitel 5.7.1 beschrieben.

2.12 Dijkstra-Algorithmus zur Berechnung der kürzesten Route

Der von Edsger Wybe Dijkstra entwickelte und nach ihm benannte Algorithmus dient zur Lösung des Kürzesten-Wege-Problems in einem kantengewichteten Graphen. Ausgehend von einem gegebenen Startknoten findet er den „günstigsten“ Weg zu einem ausgewählten Zielpunkt. In der Tat werden alle kürzesten Distanzen zu allen Knoten ausfindig gemacht, sofern keine negativen Kanten zugelassen sind. Aus diesem Grunde ist dieser Algorithmus auch unter dem Namen „single-source-shortest-path“-Algorithmus [GT98, S. 401] bekannt. Nicht zuletzt wegen seiner einfachen Implementierung und Effizienz hat er einen großen Beliebtheitsgrad erreicht und findet Anwendung in verschiedenen Bereichen wie etwa der Spieleentwicklung, bei Flugstreckenplänen oder Routenplanern. Zum Zwecke dieser Projektarbeit muss die kostengünstigste Verbindung zwischen zwei Straßen oder anderen Geoobjekten gefunden werden. Dabei repräsentieren Straßen bzw. Straßenabschnitte gewichtete Kanten im Graphen (vgl. Kap. 5.1). Für die Berechnung der Verbindungen sind nicht nur die Längen der Wege von Relevanz, sondern auch andere Faktoren wie die mit in die Kostenfunktion einbezogene Straßenbeschaffenheit oder Steigung eines Streckenabschnitts.

```
Algorithm Dijkstra(G,s)
   Eingabe: Graph G mit Startknoten s
       for each Knoten u  V[G] - s do
            D[u] := infinity
       od
       D[s] := 0; PriorityQueue Q := V;
       While not isEmpty(Q) do
            U:= extractMinimal(Q);
            for each v  ZielknotenAusgehenderKanten(u) U Q do
                if Du] + y((u,v)) < D[v] then
                     D[v] := D[u] + y((u,v));+
```

```
                    adjustiere Q an neuen Wert D[v]
                fi
            od
        od
```

Pseudocode des Dijkstra-Algorithmus. Aus: Gunter Saake, Kai-Uwe Sattler: Algorithmen und Datenstrukturen

Anfangs werden alle durch Kanten verbundenen Knoten mit dem Wert unendlich versehen. Lediglich der Startknoten erhält den Wert 0. Von diesem Knoten aus breitet sich die Suche in allen Richtungen aus und terminiert, sobald alle verfügbaren Knoten besucht sind. Im nächstfolgenden Schritt werden alle von dem aktuell beobachteten Punkt direkt erreichbaren Knoten mit einem durch die Kanten festgelegten Gewicht versehen. Anschließend wird der Knoten mit der kleinsten Distanz, der in der Prioritätswarteschlange existiert, bearbeitet. Die Kosten aller an dem aktuell beobachteten Knoten eingehenden Pfade, einschließlich des ihm anfangs zugewiesenen Gewichts, unterliegen der Prüfung auf minimale Kosten. Letztlich wird der kostengünstigste Pfad ausgewählt und der Wert des Knotens ggf. aktualisiert bzw. aus der Warteschlange gelöscht. Zuletzt erfolgen die Neuordnung der Elemente in der Prioritätswarteschlange und ein erneutes Prüfen aller ausgehenden Kanten vom zuletzt gefundenen minimalen Knoten. [SS04, vgl. S. 439 f.]
Das Ende des Algorithmus tritt ein, sobald die Prioritätswarteschlange keine Elemente mehr enthält. Da der Dijkstra-Algorithmus ähnlich eines Spannbaumes alle existierenden Knoten besuchen würde, kann er für die Radrouten-Navigation abgebrochen werden, sobald der gewünschte Zielknoten erreicht ist.
Um eine weitestgehend optimale Laufzeit zu erreichen, sind geeignete Datenstrukturen auszuwählen. Für die das kleinste Element herauslesende Prioritätswarteschlange hat sich der Heap als eine geeignete Variante erwiesen. [GT98, vgl. S. 403 f.]
Im vorliegenden Projekt ist dieser Algorithmus in pgRouting (vgl. Kap. 4.5) integriert und findet unter anderem die kürzeste Verbindung zwischen zwei im Kartenfeld definierten Positionen bzw. Knoten im Graphen der importierten und mit einer Topologie versehenen OpenStreetMap-Daten.

2.13 Traveling Salesman Problem

Das Traveling Salesman Problem (TSP) oder auch Problem des Handlungsreisenden ist eines der bekanntesten und meist untersuchten Optimierungsprobleme. Es gehört zu der Klasse der NP-Probleme. NP steht für nichtdeterministisch polynomielle Zeit und sagt aus, dass es für dieses Problem keinen Algorithmus gibt, der es präzise löst. Genauer gesagt existiert kein deterministischer Algorithmus, der ohne exponentiellen Rechenaufwand auskommt. Verschiedene Algorithmen widmen sich diesem Problem, aber bei allen handelt es sich lediglich

um Näherungslösungen. Sie finden bei sehr kleinen Problemgrößen in annehmbarer Zeit eine Lösung, aber sobald die Zahl der Punkte sehr groß wird, benötigen sie einen sehr hohen Rechenaufwand und damit viel Zeit [uni].
Allgemein gesagt, beschreibt das Traveling Salesman Problem das Finden der kürzesten Strecke aus einer vorgegebenen Menge von Punkten unter Berücksichtigung, dass alle Punkte genau einmal besucht werden. Dazu startet man an einem Punkt und beginnt von dort die Rundreise zu allen Punkten bis man schließlich zum Ausgangspunkt zurückgekehrt ist. Ein wichtiger Faktor ist die Berücksichtigung der Kosten der Strecken. Eine Strecke wird über ihre Kosten definiert und nicht über ihre tatsächliche geometrischen Länge. Das Ziel ist es, einen geeigneten Weg zu finden, bei dem die insgesamt anfallenden Kosten minimiert werden [Näh]. Hierbei wird zwischen zwei verschiedenen Arten des TSP unterschieden. Auf der einen Seite gibt es das symmetrische TSP, bei dem die Kosten einer Kante gleich den Kosten der entgegengesetzten Kante sind, und auf der anderen Seite das asymmetrische TSP, bei dem sich die Kosten von entgegengesetzten Kanten unterscheiden können.
Die Brute-Force-Methode ist der einfachste Algorithmus zur Lösung dieses Problems. Die folgende Tabelle 1 zeigt die Laufzeitentwicklung mit steigender Anzahl von Punkten. Dabei wird deutlich, dass sich schon bei 16 Punkten (Städten) die Laufzeit im unannehmbaren Bereich befindet.

Städte	**mögliche Rundreisen**	**Laufzeit**
3	1	1 msec
4	3	3 msec
5	12	6 msec
6	60	60 msec
7	360	360 msec
8	2.520	2,5 sec
9	20.160	20 sec
10	181.440	3 min
11	1.814.400	0,5 Stunden
12	19.958.400	5,5 Stunden
13	239.500.800	2,8 Tage
14	3.113.510.400	36 Tage
15	43.589.145.600	1,3 Jahre
16	653.837.184.000	**20 Jahre**

Tabelle 1: *Anzahl der möglichen Rundreisen und die entsprechende Laufzeit*
Quelle: `http://www-i1.informatik.rwth-aachen.de/~algorithmus/algo40.php` Abruf: 28.04.2010

Die zur Zeit führenden Algorithmen, die sich der Lösung des Problems nähern, sind für das

symmetrische TSP der exakte „Branch-and-Cut-Algorithmus“ sowie die verbesserte „Kernighan-Heuristik“ und für das asymmetrische TSP der „Branch-and-Bound-Algorithmus“ [Mar].
Im Rahmen dieses Projektes tritt das TSP beim Routing über Zwischenpunkte in optimaler Reihenfolge auf. In Kapitel 5.2 wird die Lösung dieses Problems mit Hilfe des in pgRouting implementierten TSP-Algorithmus beschrieben.

3 Konzept

Auf Basis von Open Source und freien Geodaten soll ein interaktiver Radroutenplaner entwickelt werden, in welchem zudem besondere touristische Attraktionen integriert sind. Das Projektresultat vermarktet touristische Infrastruktur eines Teilgebietes des Osnabrücker Landes sowie der Stadt Osnabrück und stellt somit ein Informationsangebot für Osnabrücker Bürger und Touristen dar. Als Kartengrundlage wird OpenStreetMap verwendet und es ist zunächst erforderlich, eine umfassende Analyse der Daten von OpenStreetMap durchzuführen.
Wie bereits im Kapitel 2.2.2 Grundlagen erläutert, handelt es sich bei OpenStreetMap um ein freies Projekt, welches frei nutzbare Geodaten sammelt. Die OSM-Datenbank befindet sich fortlaufend im Aufbau. Gerade in größeren und insbesondere in Universitätsstädten herrscht in Deutschland eine sehr große Abdeckung, oft sogar detailreicher als bei manchen kommerziellen Anbietern. Ein Grund hierfür ist, dass kommerzielle Anbieter von Geodaten immer dem Grundsatz folgen müssen: „Was kostet es mich, und was bringt es mir ein?“ Wie viel detaillierter das Kartenangebot von OpenStreetMap gegenüber dem von Google Maps sein kann, zeigt der nachfolgende Vergleich mit transparentem Overlay zweier Bildausschnitte. Während das Osnabrücker „Heger Holz“ bei Google Maps nur eine mehr oder weniger grüne

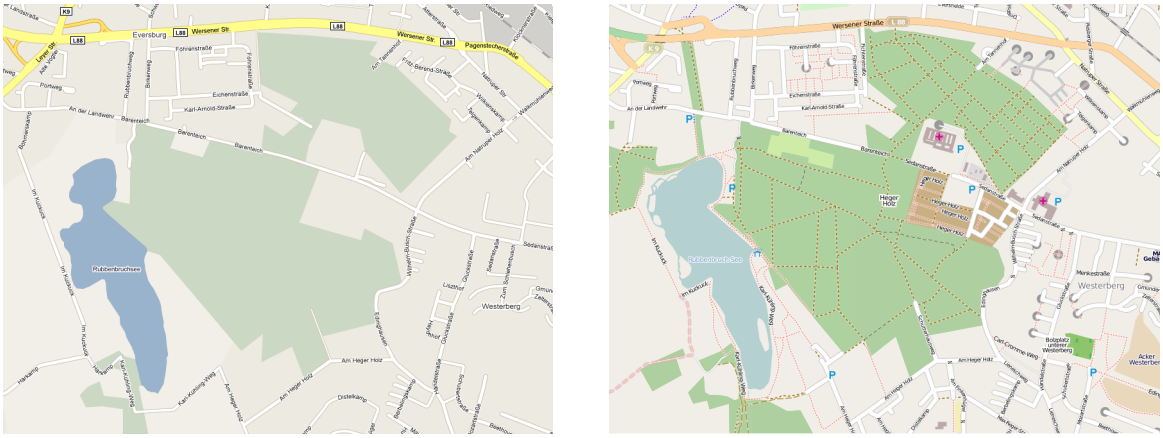

Abbildung 8: *Vergleich Google Maps und OpenStreetMap*
Quelle: `http://openstreetmap.org` und `http://maps.google.de` Abruf: 28.04.2010

Fläche darstellt, sind bei OpenStreetMap sämtliche Wege eingezeichnet, die quer durch den Wald führen.
Auch Fahrradwege in Großstädten sind bei OpenStreetMap mit hoher Genauigkeit erfasst, wohingegen ländliche Bereiche eher lückenhaft in ihrem Datenbestand sind. OSM-Daten für die Stadt Osnabrück sind bereits umfangreich vorhanden, doch ein Kontrollieren aller Gebiete ist zwingend erforderlich, da:

- Besonderheiten für Fahrradfahrer nicht oder als Grundlage für einen Radroutenplaner nur unzureichend bzw. falsch erfasst sind;
- diverse Radwege nicht verzeichnet sind, die Radfahrern erlauben auch dort ihre Wege

fortzusetzen;

- Faktoren, wie die Beschaffenheit der Straße und wie Radfahrer verkehrstechnisch auf dieser geführt werden sowie Schutz- und Fahrradstreifen im Stadtgebiet kaum erfasst sind.

Auch Meinungen von Mappern in der OSM-Community zeigen, dass umfangreiche Auswahlmasken zu Beginn nötig wären. Diese Auswahlmasken müssen vor allem Gewichtung auf einzelne Faktoren, wie straßenbegleitende Radwege, glatte Oberflächen, etc. legen. Eine weitere Meinung in der OSM-Community ist, dass zusätzlich zu dem Radrouting auch eine Auswahlmöglichkeit zwischen Rennrad, Trekking Bike, Mountain Bike, mit Kindern, mit Anhänger, etc. bestehen sollte. Der „allgemeine“ Fahrradfahrer würde besonders landschaftlich schöne Strecken und wenig befahrene Straßen bevorzugen, aber Steigungen ab 5%, Straßen ohne Radwege und unbefestigte Straßen vermeiden. In den Map Features von OpenStreetMap (`http://wiki.openstreetmap.org/wiki/DE:Map_Features`) sind genügend Eigenschaften für Radrouting vorhanden, wie z. B. von der Fahrbahn abgetrennte Fahrradspuren oder Schutzstreifen auf der Fahrbahn sowie Klassifizierungen von Straßengegebenheiten, die zur Attributierung der Daten in OSM genutzen werden können.
Damit es nicht zu Inkonsistenzen kommt, ist es wichtig, dass die Attributierung der OSM-Daten einheitlich erfolgt. Schwierigkeit hierbei ist, dass es selbst in der OSM-Community Streitigkeiten um die einheitliche Verwendung der Attribute gibt. Aus diesem Grund wird eine gemeinsame Grundlage der zu verwendenen Tags aufgebaut. Dieser sog. „Tagging-Standard“, mit dem die Gegebenheiten in Osnabrück und Umgebung überprüft werden, wird im nächsten Kapitel 3.1 näher erläutert. Das Osnabrücker Stadtgebiet wird dazu in 16 Gebiete eingeteilt und auf alle Projektteilnehmer verteilt. Diese kontrollieren alle Daten und pflegen diese anschließend über den Java OpenStreetMap-Editor (vgl. Kap. 3.2) in OpenStreetMap ein. In diesem Zusammenhang werden in Kapitel 3.3 in kurzer Form die Veränderungen in den OpenStreetMap-Daten beschrieben, die im Zeitraum des Projektes aufgetreten sind.
Ebenfalls ist es wichtig, bereits vorhandene Radroutenplaner näher zu betrachten, um einen Überblick über Möglichkeiten und Grenzen verschiedener Planer zu erhalten. Erst in der letzten Zeit sind einige Radroutenplaner entstanden, die sich in besonderer Weise aus der Menge der Radroutenplaner hervorheben. Hauptaugenmerk soll dabei auf folgende Radroutenplaner gelegt werden: OpenRouteService, RideTheCity und der Radroutenplaner NRW. Im Kapitel 3.4 werden diese drei Radroutenplaner näher betrachtet und miteinander verglichen, so dass sich im Anschluss daran die gewünschten Funktionen für den in diesem Projekt zu entwickelnden Radroutenplaner ergeben. Abschließend wird im Kapitel 3.5 die Konzeption und das Design der Benutzeroberfläche beschrieben, auf der die verschiedenen Funktionen und Elemente des Radroutenplaners, besonders im Hinblick auf Benutzerfreundlichkeit, angeordnet werden.

3.1 Bestandsaufnahme der OpenStreetMap-Daten

Die Kartengrundlage von *fahrradies.net* ist OpenStreetMap. Diese verfügt über eine Menge von Grundelementen in der Karte, denen verschiedene Schlüssel und Werte zugeordnet werden können (vgl. `http://wiki.openstreetmap.org/wiki/DE:Map_Features`). Dadurch ist eine sehr genaue und detaillierte Datenaufnahme möglich. Für *fahrradies.net* sind allerdings nur einige wenige Elemente wichtig. Damit keine Inkonsistenzen aufgrund der sehr großen Auswahl an Tags für ein und dasselbe Objekt entstehen, wird ein gemeinsamer „Tagging-Standard“ entworfen. Dieser stellt einen Katalog von Eigenschaften dar, anhand dessen die Eigenschaften der Straßen beim Abfahren des Gebietes eingestuft werden. Es muss ein umfangreiche Katalog ausgearbeitet werden, um alle Möglichkeiten der individuellen Routenplanung auszuschöpfen.
Das *highway*-Tag ist das Haupt-Tag für Straßen. Es ist recht allgemein und bestimmt in etwa die Verkehrsbedeutung der Straße. Über diese Definition werden in OpenStreetMap viele Straßen beschrieben, welches auch die Tagwatch-Statistik widerspiegelt.
Tagwatch (`http://tagwatch.stoecker.eu/Germany/De/index.html`) wertet die OSM-Datenbank aus und listet die Tags auf, die tatsächlich benutzt werden. Dort lässt sich beispielsweise verfolgen, welche Tags wie häufig verwendet werden. Tags mit großer Häufigkeit wurden bevorzugt in den den „Tagging-Standard“ mitaufgenommen. Deshalb ist auch *highway* einer der wichtigsten Schlüssel. Um zu bestimmen, um welche Art von Straße es sich handelt, wird dem Schlüssel ein genauer Wert zugeordnet. Handelt es sich z. B. um eine Land- oder Kreisstraße, wird sie durch *highway = secondary* beschrieben. Unter dem oben erwähnten Link können alle verschiedenen Straßentypen eingesehen und bezüglich ihrer genauen Definition untersucht werden. Bei der Art und Weise Straßen zu beschreiben, sind keine großen Veränderungen vorgenommen worden, sodass der „Tagging-Standard“ hier genau dem OpenStreetMap-Standard entspricht.
Besonders wichtig für die Radroutenplanung ist der Schlüssel *cycleway*. Mit *cycleway = ** werden Fahrradwege beschrieben, die gemeinsam mit der eigentlichen Straße verlaufen. Gerade bei diesem Tag tritt eine Problematik auf, die auch immer wieder in OSM-Foren diskutiert wird: Es exisitiert noch keine zufriedenstellende Lösung zum Erfassen und Attributieren von gebündelten linienhaften Objekten, wie straßenbegleitenden Rad- und Fußwegen. Es wird zwischen zwei verschiedenen Lösungsansätzen unterschieden:

1. Die jeweiligen Wege werden separat neben der Straße erfasst, welches den Vorteil mit sich bringt, dass ein genaueres Abbild der Realität erreicht wird. Außerdem werden komplexe geometrische Zusammenhänge einfacher erfassbar, was gerade für das Routing sehr wichtig ist. Beispielsweise werden bei komplexeren Kreuzungen PKWs völlig anders geführt als Radfahrer, da abhängig vom Verkehrsmittel andere Abbiegevorschriften gelten. Auf diese Weise wird ein besseres fahrzeugabhängiges Routen erreicht. Nachteile sind, dass eine Unübersichtlichkeit beim Editieren entsteht und an jedem Abzweig einer

Straße ein korrektes Verbinden der einzelnen Wege untereinander gegeben sein muss.

2. Die Fahrradwege werden als Attribut an die Straße angehängt, welches den Nachteil hat, dass eine starke Zersplitterung der Wege in kleine Abschnitte entsteht, da immer wieder eine Attributänderung entstehen kann. Von Vorteil ist, dass diese Informationen für das Routing gut auswertbar sind.

Beide beschriebenen Vorgehensweisen, ob separates Erfassen oder ein Attribut anhängen, sollten möglich sein, da es zwar für viele Straßen in Deutschland ausreicht ein Attribut anzuhängen, aber es dennoch möglich sein sollte komplexere Verkehrsführungen abzubilden. Das Tagwatch für die *cycleway*-Statistik zeigt, dass in Deutschland am häufigsten mit dem Schlüssel *cycleway* und den Werten *lane* und *track* getaggt wird. Außerdem werden die meisten Straßen im Raum Osnabrück so getaggt, dass Fahrradwege als Attribut an die Straße gehängt werden. In diesem Fall wird *highway = ** für die Straße benutzt und ergänzend *cycleway = **, um den Fahrradweg zu beschreiben.
„Lanes“ sind Radfahrstreifen, die sich auf der Straße befinden. Sie sind zumeist rot markiert und besitzen eindeutige Zeichen für Radfahrer. „Tracks“ hingegen sind baulich von der Straße getrennte Radwege. Sie verlaufen in Deutschland größtenteils auf dem Bürgersteig. Eine Unterscheidung zwischen diesen beiden Radwegetypen ist wichtig, weil so genauer auf die Bedürfnisse des Nutzers eingegangen werden kann. Ein Radweg, der mit dem Wert *track* belegt ist, bekommt eine höhere Priorität bei der Routenberechnung, weil er für Radfahrer neben erhöhter Sicherheit auch ein angenehmeres Fahrgefühl bietet. Zusätzlich muss eine Unterteilung vorgenommen werden, um zu definieren, auf welcher Straßenseite sich welcher Radwegetyp befindet. Dies geschieht durch die Schlüssel *cycleway:right* bzw. *cycleway:left*. Die Definition, welche Seite die linke und welche die rechte ist, entscheidet die Orientierungsrichtung der Straßen in OpenStreetMap. Befindet sich z.B. allein ein baulich abgetrennter Radweg auf der rechten Seite einer Straße, wird er durch *cycleway:right = track* eindeutig identifiziert. Auf diese Weise können alle verschiedenen Radwege eingetragen werden.
Des Weiteren existieren verschiedene Sonderfälle, wie z. B. Zufahrtsbeschränkungen für Radfahrer, die mit dem Wert *bicycle* angegeben werden. Ein Sonderfall ist z. B. ein Radwegetyp, der lediglich ein zusätzlich als Radweg zugelassener Bürgersteig ist. Dieser wird mit *bicycle = designated* beschrieben. Falls Fahrrad fahren auf einer bestimmten Strecke verboten ist, wird dies mit *bicycle = no* angekündigt. Ein weiterer Sonderfall sind Einbahnstraßen. Die meisten Straßen, die als Einbahnstraßen gekennzeichnet sind, dürfen mit dem Fahrrad in beiden Richtungen befahren werden, da das Verkehrszeichen „Fahrrad frei“ vorhanden ist. Die Umsetzung sieht wie folgt aus:

- *highway = **
- *oneway = yes*
- *bicycle = opposite*

Das Tag *opposite* zeigt hier eine Einbahnstraße ohne eigenen Radweg an, die für Radfahrer in Gegenrichtung geöffnet ist. Sollte *opposite* nicht attributiert sein, werden in Gegenrichtung dieser Einbahnstraße auch keine Routen berechnet.
Zwei weitere wichtige Schlüssel dienen zur Beschreibung der Oberfläche und der Oberflächenbeschaffenheit einer Straße. Mit dem Stichwort *surface* werden in OpenStreetMap die verschiedenen Oberflächentypen einer Straße beschrieben. Für den Radroutenplaner ist dies dringend notwendig, weil so eine Straße sehr genau beschrieben werden kann und somit die individuellen Bedürfnisse jedes Nutzers besser berücksichtigt werden können. Je nachdem welches Profil (vgl. Kap. 5.3) der User auf der Seite auswählt, werden Straßen bei der Routenberechnung entweder berücksichtigt oder nicht. Wählt der User z. B. das Profil „Offroad“, bekommen Straßen mit einer raueren Oberfläche eine höhere Priorität als Straßen, die standardmäßig eher glatt sind. In den Map Features von OpenStreetMap gibt es eine unüberschaubare Vielzahl verschiedener Werte, welche aber ähnliche Bedeutungen haben. Einige lassen sich nur schwer voneinander unterscheiden bzw. können anderen Werten zugeordnet werden. Um eine einfache gemeinsame Grundlage zu schaffen, werden folgende Werte unterschieden:

- *asphalt*: asphaltierte Straßen
- *cobblestone*: Kopfsteinpflaster
- *paved*: gepflasterte Straßen
- *unpaved*: unbefestigte Straßen
- *gravel*: Splitt, Schotter

Die Oberflächenbeschaffenheit *smoothness* wird nur bei extrem schlechten Straßen verwendet. Es wurde darauf verzichtet jede Straße hinsichtlich ihrer genauen Beschaffenheit aufzunehmen, weil es nur notwendig ist zu wissen, ob eine Straße befahrbar ist oder nicht. Befindet sich eine Straße in einem extrem schlechten Zustand, wird sie mit den Werten *bad* oder *very_bad* belegt. Diese Straßen werden dann bei entsprechender Auwahl nicht in die Routenberechnung einfließen.
In den Map Features gibt es auch ein Tag, das Steigungen wiedergibt. Straßen, die Gefälle oder Steigungen aufweisen, können mit *incline = *%* und zutreffendem Prozentwert getaggt werden. Da höchstens nur Schätzungen der Werte angeben werden können und die Alternativen, *incline = up* für Anstieg und *incline = down* für Gefälle, zu allgemein für den Steigungseinbezug und die Berechnung des Höhenprofil sind, werden die SRTM-Höhendaten (vgl. Kap. 2.10) verwendet. Um eine POI-/ Uni-Suche des Radroutenplaners zu ermöglichen (vgl. Kap. 5.8), ist es auch wichtig, Tags wie *amenity* (Bauten und Einrichtungen), *tourism* (touristische Einrichtungen), *leisure* (Freitzeiteinrichtungen) etc. mit in den Standard aufzunehmen, um bei dem Abfahren der Gebiete auf deren Vorhandensein zu achten.

3.2 Einpflegen der Daten mit dem Java OpenStreetMap-Editor

Im Zusammenhang mit der Datenerhebung ist es wichtig den Java OpenStreetMap - Editor (JOSM) zu nennen. JOSM ist ein Editor zum Einpflegen und Bearbeiten von OpenStreetMap-Daten. Es handelt sich dabei um eine plattformunabhängige Java-Anwendung, die im Rahmen des OpenStreetMap-Projektes entwickelt wurde. Er ist im Internet u.a. unter der Adresse `http://josm.openstreetmap.de` als Open-Source-Software in verschiedenen Sprachen zu beziehen. JOSM ist ein Offline-Editor, was bedeutet, dass die Geodaten zunächst aus dem Internet auf einen lokalen Rechner geladen, dort bearbeitet und später ins Netz hochgeladen werden können. Dazu ist keine dauerhafte Internetverbindung notwendig.
JOSM arbeitet mit drei Datenformaten. Zum einen sind das die beiden Formate, die von GPS-Geräten genutzt werden (Trackdaten, Pfaddaten), zum anderen das GPX- und NMEA-Format und außerdem Kartendaten von OpenStreetMap. Dabei dienen die GPX- oder NMEA-Daten nur als Vorlage, mit JOSM werden nur die OSM-Kartendaten editiert.
Der Editor ist baukastenmäßig aufgebaut, so ist es möglich, verschiedene Plug-Ins hinzuzufügen, die eine weitergehende und umfassendere Bearbeitung möglich machen [Ram08]. Ein wichtiges Plug-In ist das Validator-Plug-In, das fehlerhafte Daten erkennt und so der Qualitätssicherung dient.
Da es sich bei JOSM um das Standard-Tool zur Bearbeitung von OSM-Daten handelt, wurde es im Projekt bei der Korrektur und der Erweiterung der OSM-Daten eingesetzt. Die per GPS-Gerät erhobenen Daten können mittels JOSM in die OSM-Datenbank eingepflegt werden. Um die Daten zu bearbeiten, können verschiedene Layer in die Datenebene geladen werden, GPS-Tracks, Datensätze vom OpenStreetMap-Server und Luftbilder. Allerdings wird nur der Layer, der die OpenStreetMap-Daten enthält, verändert.
Im ersten Schritt werden dabei die neuen Daten mit den vorhandenen abgeglichen. Anhand des gemeinsamen „Tagging-Standards“ (vgl. Kap. 3.1) werden dann die nötigen Tags angeglichen und zudem fehlende Knoten bzw. Ways entsprechend hinzugefügt oder geändert. Im nächsten Schritt werden alle geänderten oder hinzugefügten Daten über einen eigens angelegten Account in der OSM-Community im Namen des Studienprojekts auf den OSM-Server geladen. Eine Aktualisierung der Daten erfolgt spätestens nach zwei Tagen, sodass die Ergebnisse schnell eingesehen werden können.
Die Vorteile von JOSM sind, dass viele Plug-Ins zur Verfügung gestellt werden, mit denen die verschiedensten Vorlagen für neue oder geänderte Punkte genutzt werden können. Außerdem wird beim Hochladen direkt eine Überprüfung durchgeführt, sodass schwerwiegende Fehler, wie z. B. das Benutzen einer falschen Version des Editors oder doppeltes Hochladen von Veränderungen am gleichen Ausschnittt, verhindert werden.

3.3 Analyse der Veränderungen der OpenStreetMap-Daten

An dieser Stelle war ursprünglich geplant, eine genaue Darstellung zu liefern, die darüber Aufschluss gibt, welches Projektmitglied in welchen Gebieten welche Straßen und Fahrradwege aufgenommen hat. Aus diversen Gründen, die im Folgenden erläutert werden, ist dies jedoch nicht möglich.
In OpenStreetMap wird zu jedem Objekt auch das Änderungsdatum sowie der OSM-Nutzer, der dieses Objekt zuletzt bearbeitet hat, abgespeichert. Anhand Daten dieser ist es möglich, diejenigen Objekte zu markieren, die von bestimmten Projektmitgliedern aufgenommen wurden. Hierfür gibt es mehrere Werkzeuge, speziell die Programme *osmdiff* [Gara] und *useractivity* [Garb]. Beide Programme wurden von Nutzern der OSM geschrieben und der OSM-Community zur Verfügung gestellt.
Das Programm *osmdiff* vergleicht zwei verschiedene OSM-Dateien und gibt die gefundenen Unterschiede zurück. Auf diese Weise können die Objekte herausgefiltert werden, die im Laufe des Projekts verändert worden sind. Hierzu wird ein OSM-Auszug, der vor Projektbeginn erstellt wurde, mit einem aktuellen verglichen.
Mit dem Programm *useractivity* ist es möglich, einen OSM-Auszug daraufhin zu untersuchen, welche Objekte von welchen Nutzern bearbeitet wurden. Es wird dann eine Rangfolge der aktivsten Nutzer zurückgegeben.
Beide Programme können die Ergebnisse auch in kartographischer Form darstellen, wobei die erstellten Karten zum Teil sehr unübersichtlich und daher nur bedingt geeignet sind, dem Betrachter eine Übersicht über die Leistungen einzelner Nutzer zu gewähren.
Mit dem OSM-Mapper [ITO] ist eine Onlineabfrage möglich, die alle in einem bestimmten Gebiet tätigen OSM-Nutzer nach der Menge der von ihnen bearbeiteten Objekte aufzählt und grafisch in einer Karte darstellt. Diese Karte ist aber zur Unterscheidung von vielen Nutzern ebenfalls nicht geeignet, da die einzelnen Nutzer durch verschiedene Farben dargestellt werden, welche sich zu stark ähneln.
Ein weiteres Problem des OSM-Datenvergleichs ist die Tatsache, dass im Rahmen des Projektes über den Zeitraum von fast einem Jahr OSM-Daten erhoben und verändert wurden. In diesem Zeitraum mussten Wege, die ursprünglich von einem Projektmitglied aufgenommen wurden, im Nachhinein nochmals durch andere Projektteilnehmer überarbeitet werden, da sich die genaue Bezeichnung einiger Attribute geändert hatte. Dies betraf mehrere tausend Objekte, die durch das *Validator-Tool* ermittelt und erneut bearbeitet werden mussten. Des Weiteren sind diverse Objekte nach ihrer Erfassung durch Projektmitglieder von OSM-Nutzern bearbeitet worden, die nicht am Projekt beteiligt waren.
Aus den genannten Gründen ist das Ergebnis des *useractivity*-Programms unbrauchbar, denn dort wird lediglich der User angezeigt, der das entsprechende Objekt als Letzter bearbeitet hat. In der Mehrzahl der Fälle ist dies aber inzwischen nicht mehr derjenige, der das Objekt ursprünglich aufgenommen hat.

Mit Hilfe der beschriebenen Programme und Internetanwendungen ist folglich keine sinnvolle graphische oder anders geartete Veranschaulichung der „Mapping-Tätigkeit“ der einzelnen Projektmitglieder möglich. Andere Hilfsmittel, mit denen eine solche Übersicht mit vertretbarem Aufwand zu erstellen wäre, stehen nicht zur Verfügung. Aus diesen Gründen wird auf die nutzergenaue Aufschlüsselung der Veränderung der OSM-Daten verzichtet und an dieser Stelle nur eine kurze statistische Übersicht gegeben, die den Umfang der Veränderungen in den OSM-Daten zwischen April und Oktober 2009 deutlich macht. Hierzu werden mit den oben genannten Programmen zwei OSM-Auszüge von Osnabrück vom 20.4.2009 und vom 20.10.2009 miteinander verglichen. Die Tabelle 2 zeigt diese Veränderungen.

Bezeichnung	20.04.2009	20.09.2009	Differenz
Anzahl Knoten	60723	73281	12558
Anzahl Wege	11029	13401	2372
Knoten Tags	4926	9432	4506
Wege Tags	28584	40138	11554
Wegeknoten	75383	92619	17236

Tabelle 2: *Veränderung der OSM-Daten (Stand: 2009)*
Quelle: Eigene Darstellung

Im fraglichen Zeitraum wurden also über 12500 Knoten, z. B. POIs, hinzugefügt und mehr als 2300 Wege neu erstellt. Diese Veränderungen sind allerdings nicht nur von Projektmitgliedern vorgenommen worden, sondern von allen OSM-Nutzern, die in Osnabrück und Umgebung aktiv waren.

3.4 Marktanalyse Radroutenplaner

Das Ziel des Projektes soll es sein, einen interaktiven Radroutenplaner zu erstellen, der möglichst benutzerfreundlich und intuitiv zu bedienen ist. Weiterhin ist es wünschenswert, dass die Routenplanung einen vergleichsweise hohen Grad an Individualität aufweisen kann. Somit sollte sich der Radroutenplaner letztlich durch eine Kombination von Funktionen und Benutzerfreundlichkeit von anderen Projekten positiv abgrenzen. Um ein Projekt wie einen interaktiven Radroutenplaner auf die Beine zu stellen, ist es am Anfang nötig, bereits vorhandene Projekte aus dem gleichen Themenfeld näher zu betrachten. So kann sich ein Überblick über die Möglichkeiten und Grenzen verschafft werden. Außerdem können Impressionen eingeholt werden. Diese können bereits den Grundstein dafür legen, welche Komponenten und Funktionen später im eigenen Projekt integriert werden sollen. Im Folgenden werden nun drei etablierte Radroutenplaner näher betrachtet, welche relativ verschieden sind, aber alle durch verschiedene Besonderheiten genauer betrachtet werden:

- OpenRouteService

- Radroutenplaner NRW
- RideTheCity

Um einen ersten Eindruck über die vorhandenen Funktionalitäten zu erhalten, wurde zunächst eine Vergleichsmatrix mit den vorhandenen Funktionen erstellt, siehe Tabelle 3.

	OpenRoute Service	Radroutenplaner NRW	RideTheCity
Ausgabe der Streckenlänge	ja	ja	ja
Routenbeschreibung	ja	ja	ja
Mehrsprachigkeit	ja	ja	ja
GPX-Download	ja	ja	nein
Höhenprofil	ja	ja	nein
Zwischenpunkte	ja	ja	nein
Themenrouten	nein	ja	nein
POI-Pop-Ups	nein	ja	ja
Profileditor	ja	nein	nein
Bereiche vermeiden	ja	nein	nein
Steigungseinbezug	nein	ja	nein

Tabelle 3: *Vergleichsmatrix der untersuchten Radroutenplaner (Stand 2009)*
Quelle: Eigene Darstellung

Die Auswahl der zu vergleichenden Radroutenplaner begründet sich durch folgende Kriterien: Mit OpenRouteService liegt ebenfalls ein Studentenprojekt vor, welches zudem auch auf die OpenStreetMap als Kartengrundlage und somit viele Parallelen aufweist. Der Radroutenplaner NRW hingegen ist einer der Marktführer in dieser Branche und besticht durch eine große Fülle von Funktionalitäten. RideTheCity wiederum stellt einen sehr einfachen, aber gerade deshalb auch sehr intuitiven Radroutenplaner dar.

3.4.1 Stärken und Schwächen der einzelnen Radroutenplaner

Der Routenplaner **OpenRouteService** ist ebenfalls aus einem Studentischen Projekt hervorgegangen. Der von Pascal Neis entwickelte Service hält sowohl für Fahrradfahrer als auch für Autofahrer und Fußgänger verschiedenste Funktionen und Features rund um die Gestaltung einer Route bereit. Da es sich bei diesem Vergleich konkret um die Stärken und Schwächen von fahrradtauglichen Funktionen handeln soll, wird der Fokus hauptsächlich auf jene Anwendungen gelegt, die sich auf die Planung einer Route für das Fahrrad konzentrieren. Online ist der OpenRouteService seit dem Jahr 2008, er hat eine momentane Kartenunterstützung für ganz Europa und hält bei der Unterstützung der Fahranweisungen sechs verschiedene Sprachen bereit. Neben einer deutschen Unterstützung sind auch English, Französisch, Spanisch,

Italienisch und Schwedisch im Repertoire vorhanden. OpenRouteService bietet eine Vielzahl von Diensten, welche die Angebotspalette für den Nutzer sehr umfangreich machen und auf Basis des OpenStreetMap Projektes realisiert sind. So bietet der so genannte Directory Ser-

Abbildung 9: *Benutzeroberfläche mit Funktionen des OpenRouteService*
Quelle: `http://www.openrouteservice.org` Abruf: 28.04.2010

vice einen Dienst, der einen Zugriff auf ein Online Verzeichnis zulässt, wobei über die Angabe einer Position und Distanz alle in der Nähe lokalisierten Orte, Dienste und Produkte gesucht werden können.
Neben dieser Suche ist es dem Nutzer darüber hinaus möglich, durch Setzen von Start- und Endpunkt eine einfache geplante Route zu definieren. Für die Berechnung einer fahrradtauglichen Route ist eine Auswahl an verschiedenen Profilen vorhanden, die es erlaubt, sowohl den kürzesten Weg als auch die Eigenschaften “Mountainbike“, “Rennrad“ oder “sicherste Route“ zu wählen.
Des Weiteren bietet OpenRouteService neben einer umfangreichen POI-Suche auch eine Erreichbarkeitsanalyse, die es dem Nutzer möglich macht, erreichbare Regionen in einer bestimmten Zeit zu berechnen. Bei einer individuellen Routenplanung bietet der Emergency Route Service eine Möglichkeit, Polygone in der Karte zu digitalisieren, die bei einer im Anschluss folgenden Routenberechnung vermieden werden. Selbiges gilt teilweise auch beim Ausschluss einzelner Straßen. Darüber hinaus bietet die Seite die Berechnung eines Höhenprofils und das Setzen von Zwischenpunkten bei der Planung einer Route an. Ebenfalls vorhanden ist die Möglichkeit eines GPX-Downloads, sowie die Erstellung eines Routenlinks, über den eine individuell gestaltete Route wieder aufzurufen bzw. weiterzureichen ist. Im Gegensatz zu den bereits erwähnten Funktionen, die zum größten Teil sehr ausgereift und anspruchs-

voll realisiert wurden, fehlt es dem OpenRouteService jedoch an einer klaren Gliederung und einer intuitiven Bedienbarkeit. Für den geschulten Nutzer eines Routenplaners wird es kein Problem sein, sich eine individuelle Radroute zusammenzustellen und alle Features der Seite zu nutzten. Für den Laien jedoch gibt es zu viele versteckte Funktionen, die auf den ersten Blick nicht sofort zu finden sind und deren Benutzung eventuell unbeachtet bleibt, wie z.B. die Erreichbarkeitsanalyse oder die Vermeidung bestimmter Flächen. Dennoch bietet der OpenRouteService insgesamt eine Fülle von Diensten an, die bei der Planung eines Radroutenplaners eine Menge von Inspirationen liefern.

Als Auftraggeber für den **Radroutenplaner NRW** ist das Ministerium für Bauen und Verkehr des Landes NRW verantwortlich. Schnell wird deutlich, dass für die Kartengrundlage vorzügliche Quellen vorhanden sind. So können hier z. B. auch die Hausnummern zu den Straßen eingebunden werden, was bei Planern, die auf der OpenStreetMap beruhen, bisher nicht realisiert werden konnte. Der Planer besticht zudem durch eine Vielzahl von weiteren Funktionalitäten. Es können bei der einfachen Routenberechnung entweder Ort und Straße des Start- bzw. Zielpunktes eingegeben oder diese direkt per Klick in die Karte festegelegt werden. Ferner hat der User die Möglichkeit, Zwischenpunkte in die Routenberechnung miteinzubeziehen. Über eine umfangreiche Kriterienfunktion besteht dazu die Möglichkeit, eine recht individuelle Route zu finden. Zur dieser erhält der User auch eine Fülle von Informationen und Zu-

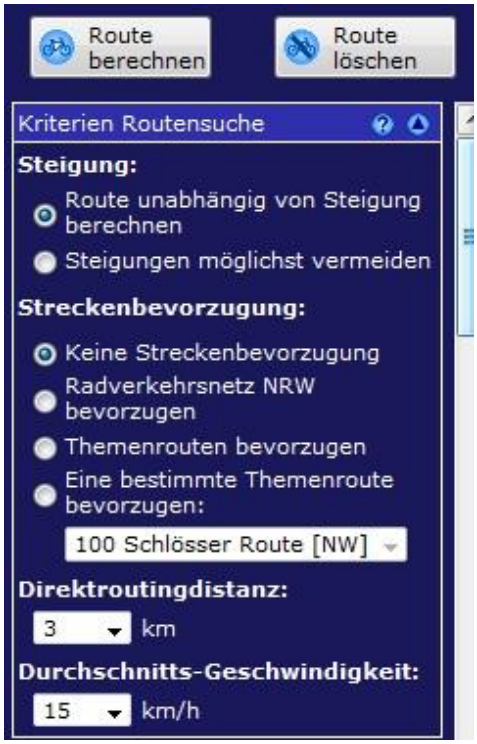

Abbildung 10: *Benutzeroberfläche mit den Kriterien zur Routensuche des Radroutenplaners NRW*
Quelle: `http://www.radroutenplaner.nrw.de` Abruf: 28.04.2010

satzmaterial, die in Form einer Statistik, eines Höhenprofils, Fahrtanweisungen, GPS-Tracks,

sowie Informationen zum Wetter und„Bike & Ride“ vorliegen. Die Route kann dann entweder auf einer Karte oder einem Luftbild dargestellt und auch abgespeichert werden. Außerdem besteht die Möglichkeit einer umfangreichen Suche nach Adressen, Sehenswürdigkeiten und Bahnhöfen in Nordrhein-Westfalen. Darüber hinaus ist eine Standortbestimmung möglich, die über die am Wegesrand des Streckennetzes stehenden Pfosten realisiert wird. Diese können über eine Nummer identifiziert werden und den User zu dessen Standort in der Karte führen. Als große Stärke des Radroutenplaners NRW kommen die Themenrouten hinzu. Hier kann der User aus einer sehr großen Zahl von lokalen oder überregionalen Routen wählen und bekommt zudem sehr viele Zusatzinformationen zu diesen geliefert. Die Seite enthält neben diesen auch ADFC-Touren, sowie detaillierte Tourentipps zu vielen Routen. Weiterhin kann der User noch Informationen zu den Themenpunkten „Radfahren und ÖPNV“, „Infos zum Radfahren in NRW“ und „Radfahren außerhalb von NRW“ beziehen, zu denen wiederum sehr umfangreiches Material zur Verfügung gestellt wird. Als Schwächen des Radroutenplaners NRW sind demgegenüber allerdings der komplizierte Aufbau der Seite, sowie die häufig recht wenig intuitive Benutzung zu erwähnen, welche sich z. B. beim Zoomen in der Karte äußert. Bei fast jedem Planer geschieht dies über das Scrollrad der Maus, hier müssen extra Buttons aktiviert werden. Insgesamt gesehen ist die Zahl von Funktionen und vor allem das umfangreiche Zusatzmaterial rund um das Radfahren aber als eine sehr positive Seite des Radroutenplaners NRW zu werten und stellen somit ein Vorbild in der Realisierung dar.

An dritter Stelle der Untersuchung steht der hierzulande wenig bekannte Radroutenplaner **RideTheCity**, der unter anderem für die Stadt New York ausgelegt ist und von Vaidila Kungys und Jordan Anderson entwickelt wurde. RideTheCity besticht vor allen Dingen durch seine äußerst benutzerfreundliche Oberfläche und der intuitiven Bedienbarkeit. Dem User wird auf den ersten Blick klar, wo er welche Funktionen findet und wie er vorgehen muss, um eine Route zu planen. Der Umfang der Funktionen, die dem Nutzer geboten werden, ist nicht so hoch wie bei den anderen Projekten. Die Seite ist eher einfach, aber dennoch gut strukturiert gehalten.

Die Hauptfunktion der Seite besteht in der Findung der kürzesten Entfernung zwischen zwei Punkten. Dabei werden für Fahrradfahrer untypische Straßen wie Autobahnen vermieden und fahrradfreundliche Strecken mit Fahrradwegen oder entlang besonderer Grünwege automatisch bevorzugt. Das Finden der immer sichersten Route hat bei RideTheCity dabei oberste Priorität. Darüber hinaus ist bei RideTheCity eine POI-Suche nach Bike Shops realisiert, mit der mittels Popups die nötigen Informationen eingeholt werden können. Sowohl Fahrradverleih, Fahrradverkauf und der Service rund ums Fahrrad werden dabei berücksichtigt. Im Anschluss an die Planung einer Route ist es dem User möglich, Kommentare zu den Strecken zu formulieren und an die Entwickler zu senden, um Verbesserungen direkt vornehmen zu können. Neben den bereits erwähnten Funktionen bietet sich dem User darüber hinaus die Möglichkeit, individuelle Routen über das eigene Benutzerkonto zu speichern bzw.

Abbildung 11: *Benutzeroberfläche RideTheCity New York*
Quelle: `http://www.ridethecity.com` Abruf: 28.04.2010

einzelne Straßensegmente durch konkrete Bewertungen zu bevorzugen oder aber nicht zu berücksichtigen. So ist möglich, seine individuellen Vorlieben selbst anzupassen und diese bei der nächsten Planung mit in die Routenfindung einzubeziehen. Über einen direkten Link, der von der Seite generiert wird, besteht des Weiteren die Möglichkeit, einen Standort direkt zu finden. Dazu muss nur der Name der Institution eingegeben werden und man erhält eine automatische Wegbeschreibung zu dem Geschäft. Ein weiteres Feature von RideTheCity ist die SMS Funktion, über welche man eine Route per SMS auf das eigene Handy überführen kann. Insgesamt bietet der mehrsprachige Radroutenplaner RideTheCity leicht zu bedienende Funktionen, welche auf das Wesentliche reduziert sind, aber vor allen Dingen durch gute Übersichtlichkeit und leichte Bedienung überzeugen.

3.4.2 Übersicht der Stärken und Schwächen der einzelnen Radroutenplaner

	OpenRoute Service	**Radroutenplaner NRW**	**RideTheCity**
Stärken	- Mehrsprachigkeit - Directory Service - Profilauswahl - Erreichbarkeits-analyse - Höhenprofil - Ausführliche Hilfe - Bereichs-vermeindung - Profileditor - API für externe Nutzung - Routenlink	- viele Kriterienfunktion (u.a. Steigungseinbezug) - ausgeprägte Suchfunktion - Themenrouten - POI-Popups - Zwischenpunkte - detailliertes Zusatzmaterial - Exportfunktion	- Benutzerfreundlich - intuitiv bedienbar - sicherste Routingfindung - POI-Popups - Kommentarfunktion - Standortfindung
Schwächen	- wenig intuitiv bedienbar - keine genaue Strukturierung - kein Steigungs-einbezug	- wenig intuitiv bedienbar - komplizierter Aufbau	- kein GPX-Download - keine Zwischenpunkte - kein Höhenprofil

Tabelle 4: *Übersicht der Stärken und Schwächen der einzelnen Radroutenplaner*
Quelle: Eigene Darstellung

Für die Entwicklung eines neuen interaktiven Radroutenplaners kann und sollte man sich an den Stärken marktführender Radroutenplaner zumindest orientieren und diese bei der Konzipierung des Funktionskatalogs einfließen zu lassen. Folgende Auflistung fasst Funktionen der zuvor betrachteten Radroutenplaner an, die als Ideengeber für das Projekt genutzt wurden:

- Profilauswahl
- Höhenprofil
- Profileditor
- Adresssuche
- Themenrouten

- POI-Popups
- Zwischenpunkte
- Mehrsprachigkeit
- Bereichsvermeidung
- Routenlink

Weiterhin müssen natürlich gewisse selbstverständliche Standardfunktionen implementiert werden, die praktisch in jedem Radroutenplaner zu finden sind bzw. zu finden sein sollten. Dieses Grundgerüst setzt sich aus folgenden Features zusammen:

- Berechnung einer Route zwischen zwei Punkten
- verbale Routenbeschreibung
- Druckfunktion
- Export auf mobile Endgeräte

Zusätzlich zu den als positiv bewerteten Funktionen der untersuchten Radroutenplaner und den Standardfunktionen wurde im Rahmen des Projektes eine weitere Reihe an Funktionen, die außerdem realisiert werden sollen, zur Diskussion gestellt. Diese Features erfüllen letztlich den Zweck, den Radroutenplaner von anderen Projekten abzuheben. Im Folgenden soll kurz beschrieben werden, in welche Richtung sich die Funktionen orientieren sollen.
Im Zeitalter des Web 2.0 ist es von besonderer Wichtigkeit, den User aktiv am Geschehen im Netz mitwirken zu lassen. Aus diesem Grund soll eine Funktion realisiert werden, welche dem User die Möglichkeit bietet, Bewertungen zu berechneten Routen abzugeben. Der Input des Users soll eine sinnvolle Verwendung finden und eine aktive Mitgestaltung der Webseite ermöglichen. Im Detail sollte die Routenbewertung zumindest nach verschiedenen Kriterien vorgenommen werden können. Zusätzlich ist eine individuelle Kommentarfunktion zu einzelnen Routen denkbar.
Viele Radroutenplaner bieten die Möglichkeit bei der Planung einer Route Zwischenpunkte in optimaler oder fester Reihenfolge zu setzten. In diesem Projekt ist eine Verbindung beider Varianten angedacht. Wählt der User die Option “feste Reihenfolge“, so soll über die nacheinander gesetzten Punkte in der gewählten Reihenfolge geroutet werden. Bei der Variante “optimale Reihenfolge“, soll jeder Punkt in dem Sinne einmal angefahren werden, so dass letztlich die kürzeste zwischen Start- und Endpunkt über die Zwischenpunkte entsteht.
Als weitere Funktion könnten die Gebäude der Universität und der Fachhochschule mit in die POI-Suche einbezogen werden. Dadurch soll sich dem User die Chance bieten, jedes Gebäude sowohl zu finden als auch als Start- bzw. Endpunkt zu setzten, sowie eventuell wichtige

Informationen einzusehen. Besonders für angehende Studenten der Universität oder Fachhochschule Osnabrück könnte dieses Feature zur Orientierung dienen.
Nachfolgend eine kurze Auflistung dieser Funktionen:

- Routenbewertung
- Zwischenpunkte in fester und optimaler Reihenfolge
- Universitäts- und Fachhochschulsuche

Abschließend gibt es eine zusammenfassende Auflistung der Standardfunktionen, der Stärken der anderern Radroutenplaner und der eigenen Ideen, die im Rahmen des Studienprojektes realisiert werden sollen:

Standard-funktionen	**Stärken anderer Projekte**	**Weitere eigene Ideen**
- Berechnung einer Route zwischen zwei Punkten - verbale Routenbeschreibung - Druckfunktion - Export auf mobile Endgeräte	- Profilauswahl - Höhenprofil - Profileditor - Adresssuche - Themenrouten - POI-Popups - Zwischenpunkte - Mehrsprachigkeit - Bereichsvermeidung - Routenlink	- Universitäts- und Fachhochschulsuche - Routenbewertung - Zwischenpunkte in fester und optimaler Reihenfolge

Tabelle 5: *Zusammenfassung der angedachten Funktionen*
Quelle: Eigene Darstellung

3.4.3 Zusammenfassung

In diesem Abschnitt des Kapitels ist aufgezeigt worden, was für Stärken und Schwächen verschiedene Radroutenplaner aufweisen und wie man sich positive Funktionen bei der Konzipierung eines neuen interaktiven Radroutenplaners zu Nutze machen kann. Des Weiteren wurde herausgestellt, welche Funktionen als Inspirationen aus anderen Projekte dienen und welche Funktionen dieses Projekt von vergleichbaren Beispielen abheben soll. Außerdem wurde geklärt, welche Standardfunktionen generell in einem Radroutenplaner vorhanden sein sollten. Letztlich ist eine Liste entstanden, die im Zuge des Projektes möglichst vollständig umgesetzt werden sollte. Als weiteres wichtiges Merkmal wurde eine intuitive Bedienbarkeit und gute Übersichtlichkeit herausgestellt, was auch im Rahmen des im Projektes eine große Rolle spielen soll. Darauf wird im nachfolgenden Kapitel eingegangen.

3.5 Konzeption der Benutzeroberfläche der Webseite

Damit Anwender von einem Radroutenplaner überzeugt werden, genügt es nicht, nur viele Funktionen anzubieten, die von anderen Planern nicht geboten werden. Ebenso ist es notwendig, ein attraktives und moderes graphisches Erscheinungsbild zu liefern, mit dem eine einfache und intuitive Bedienung der Anwendung durch den Nutzer gewährleistet wird. Nur wenn die Interaktion zwischen Anwender und Routenplaner reibungslos und ohne unnötige Komplikationen abläuft, wird die Serviceleistung angenommen.
Einen Blickfang stellt auch ein Logo dar. Das Logo von *fahrradies.net* unterstreicht den Begriff *Fahrradies* indem es die beiden darin enthaltenen Begriffe *Fahrrad* und *Paradies* verbindend darstellt. Es stellt eine grüne Palmeninsel als Symbol des Paradieses dar, auf der ein Radfahrer fährt. Ein gelbes Feld, welches einem Ortseingangsschild nachempfunden ist, stellt den regionalen Bezug dar (s. Abb. 12). Um die Wiedererkennbarkeit nochmals zu verbessern,

Abbildung 12: *Logo des Radroutenplaners*
Quelle: Eigene Darstellung

ist es weiter sinnvoll, ein Favicon zu benutzen. Favicons sind kleine, 16 × 16 Pixel große, quadratische Grafiken, die am Anfang einer Adresszeile in einem Internetbrowser angezeigt werden. Das Favicon von *fahrradies.net* stellt eine grüne Palme dar, die sich auch im großen Logo wieder findet.
Zur Interaktion von Anwender und Recheneinheit muss eine benutzerfreundliche graphische Oberfläche implementiert, die durch den Anwender über das Internet aufgerufen wird. Benutzerfreundlichkeit beinhaltet hier die Aspekte Intuitive Bedienung, sinnvolle Farbgebung, Autovervollständigung bei Eingaben, Drag 'n Drop, umfangreiche Hilfe und moderner Look (s. Abb. 13). Die Benutzer müssen Parameter zur Routenberechnung und Optionen für die Wahl der Route eingegeben sowie andere Einstellungen getätigt werden. Außerdem muss die graphische Ausgabe der berechneten Route geschehen und auch Zwischeninformationen geliefert werden. Diese Benutzeroberfläche wurde mit PHP implementiert und mit JavaScript erweitert, um besser mit dem Anwender interagieren zu können. Mit jedem aktuellen und gängigen Internetbrowser kann daher die Benutzeroberfläche problemlos aufgerufen werden.
Die Ansicht im Browserfenster ist grob in vier Teile gegliedert, wie die a. Am linken oberen Rand befindet sich das Logo von *fahrradies.net*. Rechts schließt sich eine Kopfzeile an, die Knöpfe für verschiedene Funktionen enthält. Darunter liegt auf der linken Seite das Menüfeld, welches alle Einstellungen zur Routenplanung, Zielfindung sowie der Routenbe-

Abbildung 13: *Aspekte der Benutzerfreundlichkeit*
Quelle: Eigene Darstellung

arbeitung enthält. Die restliche Fläche wird für die Kartendarstellung genutzt. Durch diese Aufteilung mit der verhältnismäßig großen Kartenfläche wird die Wichtigkeit der Karte als Planungs- und Orientierungsmedium unterstrichen. Gleichzeitig bleibt genügend Fläche um die verschiedenen Funktionen zu platzieren.
Im Folgenden geht es um die Gestaltung und Aufteilung der Oberfläche und um die graphische Gestaltung einzelner Symbole. Die Gestaltung des Layouts der graphischen Benutzeroberfläche bedingt, dass zunächst alle Funktionen des Radroutenplaners grundlegend festgelegt sind.

Die Ansicht im Browserfenster ist grob in vier Teile gegliedert.

- Logo
- Kopfzeile mit Infobox und Buttons
- Menüleiste mit Menüauswahl und Menüfeld
- Kartenfeld

Am linken oberen Rand befindet sich das Logo von *fahrradies.net*. Rechts schließt sich eine Kopfzeile an, die Knöpfe für verschiedene Funktionen enthält. Darunter liegt auf der linken Seite das Menüfeld, dass etwa ein Drittel der Bildschirmbreite einnimmt und in dem alle Einstellungen zur Routenplanung, Zielfindung sowie der Routenbearbeitung zu finden sind. Die restliche Fläche, etwa die Hälfte der Gesamtfläche, wird für die Kartendarstellung genutzt (s. Abb. 14).

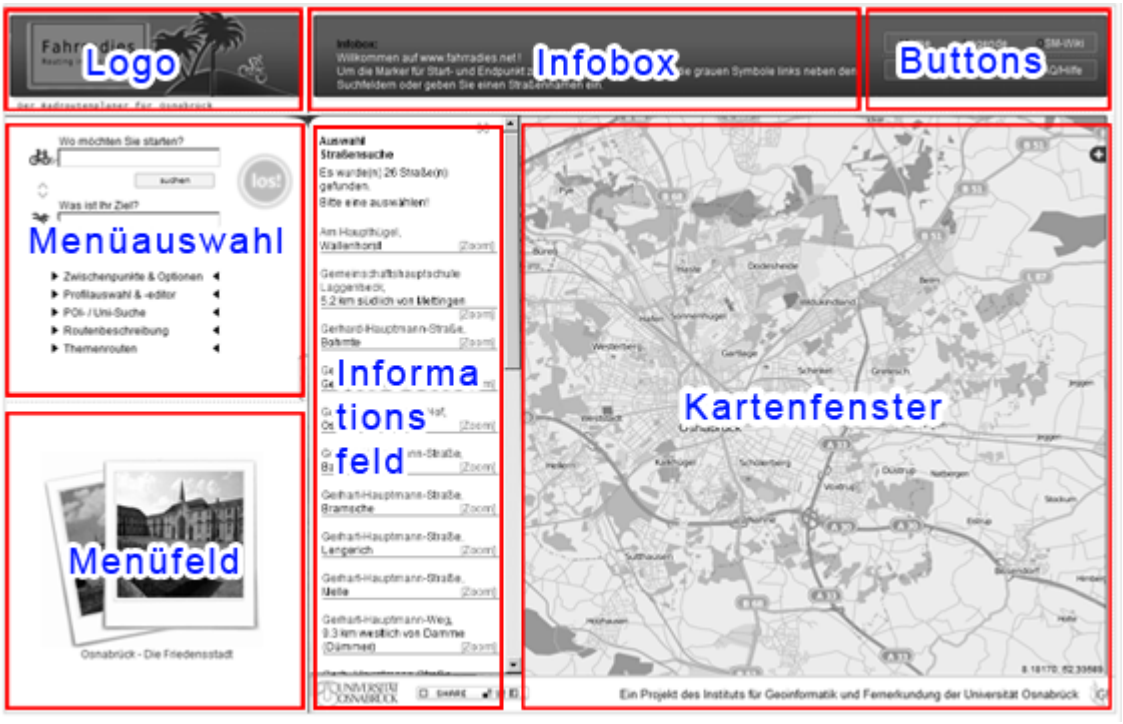

Abbildung 14: *Übersicht über das Weblayout*
Quelle: Eigene Darstellung

Das Menüfeld gliedert sich in zwei Hauptbereiche, die Start-/Zieleingabe mit Menüwahl im oberen Bereich und die Fläche darunter, in der die verschiedenen Menüs angezeigt werden. Die Eingabe von Start- und Zielort wurde bewusst ganz oben angeordnet und nicht in ein Untermenü gepackt, um dem Nutzer ein sofortiges Loslegen zu ermöglichen. Die Eingabefelder fallen sofort ins Auge und dem Anwender ist vom ersten Moment an klar, wozu diese Felder dienen. Neben den Eingabefeldern befindet sich der *los!*-Knopf. Dieser ist zunächst grau, also deaktiviert, dargestellt und ändert seine Farbe zu grün sobald in beiden Eingabefeldern gültige Werte enthalten sind, die eine Routenberechnung zulassen. Dies dient dazu, dem Nutzer gültige Eingaben sofort anzuzeigen und Fehler in der Eingabe aufzudecken. Um dem Anwender die Eingabe von Start- und Zielort zu erleichtern, wurde eine Funktion entwickelt, die parallel zum Eintippen des Textes, die Datenbank durchsucht und mögliche Ziele (Straßen und Orte) in einem darunter liegenden Vorschlagsfeld anbietet. Dabei ist es gleich, in welchem Format nach einer Straße gesucht wird. Es wird sowohl das Format *Straße, Ort*, als auch *Ort, Straße* unterstützt (s. Abb. 15).

Noch einfacher wird die Wahl von Start- und Zielpunkt, wenn der Anwender auf das Symbol vor dem jeweiligen Feld klickt. Dann hat er die Möglichkeit, einfach einen Punkt in der Karte zu selektieren und diesen als Start- bzw. Zielpunkt auszuwählen. Sobald der Marker einmal in der Karte gesetzt wurde, kann der Nutzer ihn per Drag 'n Drop versetzen. Die ebenfalls im oberen Bereich der Menüleiste dargestellten Menüpunkte befinden sich unterhalb der Ein-

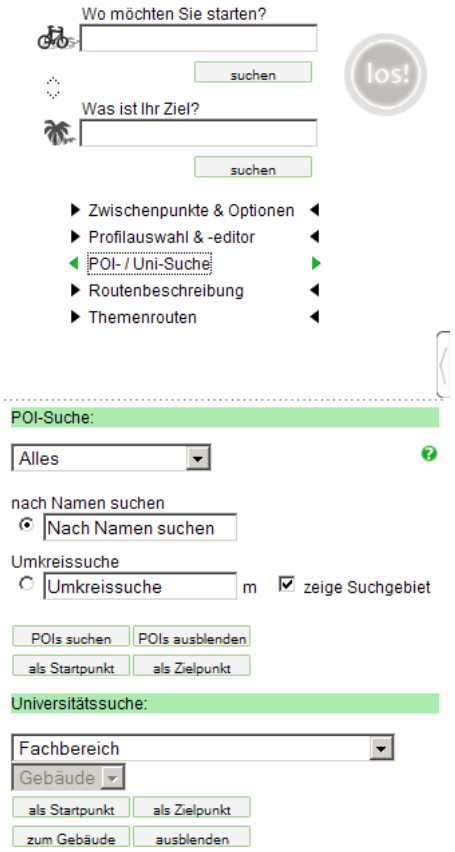

Abbildung 15: *Übersicht über die Menüleiste*

Quelle: Eigene Darstellung

gabefelder für Start- und Zielpunkt. Damit liegen sie genau im Blickfeld des Anwenders und können leicht gefunden werden. Durch einen Mausklick auf einen Menüpunkt, öffnet sich im darunter liegenden Bereich die spezielle Funktion. Aus Platzgründen war es notwendig, die einzelnen Menüs nicht dauerhaft anzuzeigen sondern sie alle nach Bedarf am selben Ort darzustellen. Die Anordnung ist dennoch intuitiv, da dem Anwender durch graphische Merkmale verdeutlicht wird, zu welchem Menü die im unteren Bereich des Menüfeldes dargestellten Funktionen gehören.
Für einige Funktionen ist es notwendig, über mehr Platz verfügen zu können. Dieser Fall tritt beispielsweise bei einer Suchanfrage, die viele Ergebnisse liefert, auf, aber auch für die Darstellung von Legende und Höhenprofil. Für diese Fälle öffnet sich rechts neben der Menüleiste ein neues Fenster in Form des *Informationsfeldes*, welches die gewünschten Informationen enthält. Dieses Fenster wird von links über das Kartenfeld geschoben. Es wird jedoch nur temporär benötigt und kann danach wieder geschlossen werden, um nicht unnötig Raum einzunehmen und den Darstellungsbereich der Karte zu verkleinern (s. Abb. 16).

Abbildung 16: *Infofenster*
Quelle: Eigene Darstellung

Über die Kopfzeile hat der Anwender Zugang zu Funktionen, die nicht nur einzelnen Menüpunkten zugeordnet sind. Dies sind z. B. die Druckfunktion, die Legende und auch die

FAQ/Hilfe. Diese haben eigene Knöpfe auf der rechten Seite der Kopfzeile und stehen dem Nutzer dort zu jedem Zeitpunkt zur Verfügung, so dass sie im Bedarfsfall nicht erst lange gesucht werden müssen. Sollte ein Anwender weitergehende Anleitung zu bestimmten Funktionen benötigen, findet sich eine Infobox ebenfalls in der Kopfzeile, jedoch auf der linken Seite. Hier werden Hinweise zu einzelnen komplexeren Funktionen geboten.
Weiterführende Hilfe kann der Anwender erhalten, indem er die Frequently Asked Questions (FAQ) über den Button *FAQ/Hilfe* drückt. Dort werden viele möglicher Fragen von Nutzern beantwortet.
Außerdem gibt es weitere Hilfestellung zu den Elementen der Menüleiste durch sog. Tooltips. Dies sind kleine Fenster, die sich öffnen, sobald die Maus längere Zeit auf einem Element stehen bleibt. Diese Tooltips geben direkte Anweisungen, wie der Anwender weiter vorgehen muss, um die Funktion zu nutzen.

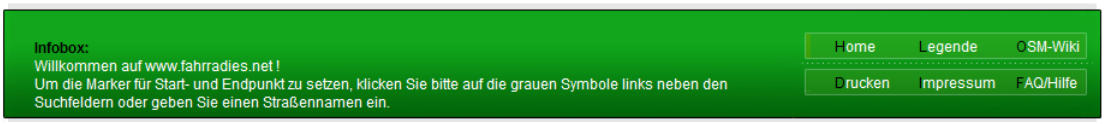

Abbildung 17: *Kopfzeile*
Quelle: Eigene Darstellung

Zur Benutzerfreundlichkeit gehört auch ein klares Farbkonzept auf einer Internetseite. Dieses sollte konsequent umgesetzt werden. Im Wesentlichen werden bei *fahrradies.net* die Farben Grün, Gelb, Schwarz und Weiß genutzt, wobei die Kopfzeile und das Logo mehrheitlich grün sind und die Menüleiste mit schwarzer Schrift auf weißem Hintergrund arbeitet. Die Tooltips werden als dunkelgrüne Schrift auf gelbem Hintergrund dargestellt.
Durch diese Farbwahl ist ein guter Kontrast von Hintergrund und Schrift sichergestellt. Gleichzeitig wurden Farben gewählt, die gut miteinander harmonieren und beim Betrachter einen positiven Eindruck erwecken.
Um dem Benutzer auch mit der Symbolik die Wiedererkennung zu erleichtern werden, statt einfacher, punktähnlicher Grafiken für die Markierung von Start- und Zielpunkten in der Karte, jeweils spezielle Grafiken entworfen. Sie sind an die in der Wortkombination Fahrradies enthaltenen Bestandteile Fahrrad und Paradies angelehnt. Dabei stellt ein blaues Fahrrad den Startpunkt, an dem bildlich gesprochen das Fahrrad steht, dar. Der Zielmarker ist eine grüne Palme, die das zu erreichende Paradies darstellt.

Alle Bestandteile der Benutzeroberfläche sind darauf ausgelegt, vom Nutzer möglichst intuitiv und ohne langen Lern- oder Probierprozess verstanden zu werden und sich zusätzlich durch Hinweise selbst zu erklären. Hierzu wird in den Menüs auf eine strikte Reihenfolge der vom Anwender zu bearbeitenden Elemente geachtet. Der Nutzer muss sich nur von oben nach unten durch die Einstellmöglichkeiten arbeiten und kann am Ende die Funktion wie

Abbildung 18: *Start- und Zielmarker*
Quelle: Eigene Darstellung

gewünscht einsetzen. Viele Funktionen und auch die Kartennavigation sind dem Anwender schon von anderen bekannten Internetdiensten bekannt und können in ähnlicher Form auch auf *fahrradies.net* genutzt werden.

4 Eingesetzte Software

Nachdem im vorherigen Kapitel u. a. die grundlegenden Funktionen des Radroutenplaners definiert wurden, enthält dieses Kapitel Informationen über die Software, die für die Implementierung verwendet wird. Zum Verständnis der Funktionen ist es ist wichtig zu wissen, wie und vor allem zu welchem Zweck die aufgeführten Softwareprodukte eingesetzt werden. Durch die Vorgabe, einen interaktiven Radroutenplaner auf Grundlage von frei verfügbarer Software zu entwickeln, handelt es sich ausschließlich um Open-Source-Software.

4.1 Betriebssystem

Der nachfolgende Abschnitt behandelt eine Gegenüberstellung der beiden populärsten Betriebssysteme, Linux und Windows, und der daraus resultierenden Nutzung der Linux-Distribution Ubuntu.

4.1.1 Linux vs. Windows

Der Streit um Linux und Windows als Serverbetriebssystem wird maßgeblich von monetären Interessen geleitet. Während Windows-Produkte einer Lizenz unterliegen, stehen Linux-Produkte kostenlos zur Verfügung. Solange der Anwender über genügend Know-How verfügt, kann ein Linux-Server günstiger betrieben werden. Erst, wenn Support für diese Systeme eingekauft werden muss, entstehen hohe Kosten. Aus diesem Grund verbreitet sich das kostenlose Linux bei Privat- und Kleinanwendern sehr schnell, hier hat es sich inzwischen einen hohen Stellenwert erreicht.
„[. . .] der Funktionsumfang [von Linux und Windows] ist schwer zu vergleichen. Längst werden keine reinen Betriebssysteme mehr angeboten. Alle Plattformen umfassen ein breites Merkmal an Funktionen, die streng genommen der Anwendungsseite zuzurechnen sind. Und viele Open-Source-Anwendungen werden zwar typischer Weise auf Linux betrieben, stehen aber ebenso für Windows zur Verfügung“ [Sch09].
Auch in Fragen der Sicherheit scheiden sich die Geister. Oftmals wird behauptet, Linux sei sicherer als Windows [vis], ähnlich oft aber auch das Gegenteil. [Jam06]
Die Entscheidung zur Wahl des Betriebssystems für den Server wurde für dieses Projekt sehr früh aus praktischen Erfahrungen heraus getroffen. Sowohl unter Linux als auch unter Windows stehen diverse MapServer zur Verfügung. Für Windows ist dies z. B. der MS4W (MapServer for Windows). Ebenso gibt es PostgreSQL und PostGIS in einer Version für Windows. Das Modul pgRouting wird aber in erster Linie für Linux entwickelt, weshalb unter Windows nur einige ältere Versionen installiert werden können. Ein weiteres Problem ist das Aufbereiten der OSM-Daten für das Routing und Schreiben dieser Daten in eine Datenbank. Dies lässt sich mit zwei verschiedenen Kommandozeilenprogrammen, OSM2pgRouting und OSM2pgSQL durchführen. OSM2pgRouting ist lediglich unter Linux nutzbar, unter Windows

dagegen nicht ausreichend getestet. OSM2pgSQL verursacht Datendurchmischungen in der erzeugten Tabelle, so werden Linien (z. B. Grenzen und Gewässer) mit Straßen vermischt, wodurch ein sinnvolles Routing unmöglich wird.
Zusätzlich ist zu beachten, dass die Unterstützung für pgRouting durch Foren, Blogs, etc. bei einem Windowssystem sehr spärlich ist und sich die meisten Supportseiten mit der Linux-Version beschäftigen.
Aus diesen Gründen ist es sehr vorteilhaft, für das Projekt ein Linuxsystem zu nutzen und nicht auf einem Windowsserver zu arbeiten. Letztlich ist die Entscheidung auf die Linux-Distribution Ubuntu gefallen.

4.1.2 Ubuntu

Ubuntu ist eine auf Debian basierende Linux-Distribution, die ein einfach zu installierendes und leicht zu bedienendes Betriebssystem mit aufeinander abgestimmter Software darstellt. Gesponsert wird das Ubuntu-Projekt von Canonical Ltd., dessen Gründer der aus Südafrika stammende Unternehmer Mark Shuttleworth ist. Der Name „Ubuntu“ stammt aus dem Zulu und bedeutet soviel wie „Menschlichkeit gegenüber anderen“. [Wip05]
Im Wesentlichen besteht Ubuntu aus den Elementen Linux und GNU. „Linux ist der eigentliche Betriebssystemkern und somit die Basis von Ubuntu. Aus dem GNU-Projekt (vgl. Kapitel 2.1) stammen viele Softwarepakete, ohne die Linux kaum nutzbar wäre und die einen essenziellen Charakter besitzen.“ [Fis06]
Jede Version von Ubuntu besitzt eine Versionsnummer, die auf dem jeweiligen Veröffentlichungsjahr und -monat basiert, und einen Codenamen. Primäres Ziel ist es, alle sechs Monate eine neue Version zu veröffentlichen, wobei die erste aus dem Jahre 2004 stammt. Bisher folgten 16 weitere Versionen.
Ubuntu ist ein oft verwendetes und frei zugängliches Betriebssystem und damit nicht nur ein Open-Source-Produkt, sondern auch ohne Einschränkung nutzbar. Mit dieser Vorraussetzung eignet es sich sehr gut für diese Projektarbeit.
Sowohl die ausgezeichnete Benutzerfreundlichkeit, als auch Internationalisierung und Barrierefreiheit stellen die wichtigsten Eigenschaften von Ubuntu dar. Neben Ubuntu selbst, welches GNOME [1] als Desktopumgebung einsetzt, existieren weitere verschiedene Abwandlungen wie Kubuntu, Xubuntu und Xfce.

4.2 Apache Weberver

Der im Projekt eingesetzte Server ist der Apache Webserver. Neben vielen anderen Betriebssystemen unterstützt er auch Linux und läuft unter Ubuntu. Er ist ein Produkt der Apache

[1] Arbeitsumgebung für Unix- und Unix-ähnliche Systeme mit einer grafischen Benutzeroberfläche und einer Sammlung von Programmen für den täglichen Gebrauch

Software Foundation und seit 1996 der meistgenutzte Webserver im Internet. Die aktuelle Version trägt die Bezeichnung 2.2.14 und stammt vom 05. Oktober 2009. [The09]
Der große Vorteil bei der Nutzung von Apache besteht vor allem in dessen Schnelligkeit und der hohen Stabilität. Um die individuellen Stärken der verschiedenen Betriebssysteme optimal ausnutzen zu können, stellt die Foundation die Apache Portable Runtime (APR) zur Verfügung, eine Bibliothek, die für die problemlose Umsetzung verschiedener Systemaufrufe sorgt.
Ergänzend dazu gibt es differenzierte Multiprocessing-Module (MPM), welche unterschiedliche Lösungen für Mehrbenutzeranfragen anbieten.
Mit dem Apache Webserver besteht für den Anwender die Möglichkeit, Webseiten dynamisch zu erstellen. Die hierfür notwendigen Skriptsprachen stellen keine Bestandteile des Webservers dar, sondern werden über das CGI, das Common Gateway Interface, eine allgemeine Schnittstelle, angesprochen oder über Module eingebunden. Die am Häufigsten verwendeten Skriptsprachen sind PHP, Perl und Ruby. [Theb]
Da Apache modular aufgebaut ist, bietet er dem Anwender nahezu unbegrenzte Nutzungsmöglichkeiten, wie z. B. die Verwendung als Proxyserver, verschlüsselter Webserver uvm. [Thea]
Für das Studienprojekt ist der Apache Webserver 2.2.14 optimal geeignet, da er neben seinen hervorragenden Eigenschaften, wie alle Produkte der Apache Foundation, ein Open-Source-Produkt ist.

4.3 UMN MapServer

„Der UMN MapServer ist eine an der University of Minnesota (UMN) in Kooperation mit der NASA und dem Minnesota Department of Natural Resources (MNDNR) entwickelte Software zur dynamischen Kartenerstellung.“ [Hac07]
Aktuell unterliegt das Projekt der Obhut der Open Source Geospatial Foundation (`http://www.osgeo.org`) und wird von einer weltweit wachsenden Zahl von Entwicklern gewartet. [Uni09b]
Beim UMN MapServer handelt es sich nicht um ein voll funktionsfähiges GIS, sondern um eine webgestützte Applikation, über die Karten visualisiert werden können. „Aufgrund des offenen Programmcodes und mit entsprechenden Programmierkenntnissen kann die Software schnell um neu entwickelte Zusatzmodule ergänzt werden.“ [Hac07]
Hilfestellungen findet man dazu auch in den zahlreichen Foren und Mailinglisten, die sich mit dem UMN MapServer beschäftigen. Somit ist es letztlich auch möglich, Funktionen einzubauen, die in einem typischen GIS vorhanden sind. So können z. B. Flächenberechnungen durchgeführt oder Abstände gemessen werden.
Die größten Vorteile des UMN MapServers liegen in der hohen Stabilität und Geschwindigkeit, welche mit einem relativ geringen Konfigurationsaufwand erreicht werden. Hinzu kommt, dass der UMN MapServer plattformunabhängig ist.

Des Weiteren werden viele Standards des Open Geospatial Consortium (vgl. Kap. 2.6) umgesetzt und eine Vielzahl von sowohl Raster- als auch Vektorformaten unterstützt [Hac07]. Über den MapServer werden außerdem Daten des PostgreSQL-Aufsatzes PostGIS eingebunden. Diese Begriffe werden im nächsten Abschnitt erläutert.
Auf das Zusammenspiel zwischen UMN Mapserver, Webserver usw. wird auch im Kapitel 5.12 eingegangen.

4.4 PostgreSQL und PostGIS

Bei PostgreSQL handelt es sich um ein freies, objektrelationales Datenbanksystem. Es wurde Mitte der 1980er Jahre als Universitätsprojekt von Michael Stonebraker unter dem Namen Postgres (Post-Ingres) gestartet und seit 1997 als Open-Source-Software von einer freien Entwickler-Community unter dem Namen PostgreSQL weiterentwickelt. [Posb]
Objektrelationale Datenbanken zeichnen sich durch die Kombination von relationalen und objektorientierten Datenbanken aus. Basis von PostgreSQL ist ein Client-Server-Modell, bei dem das Serverprogramm „postmaster“ die Datenbankdateien verwaltet, die Verbindungen mit dem Client-Programmen koordiniert und die Datenbankanfragen bearbeitet. Das Client-Server-Modell wird in der Regel mit einer verteilten Architektur umgesetzt, was bedeutet, dass Client und Server nicht notwendigerweise auf demselben Rechner installiert sein müssen. Sie können einfach per Netzwerkprotokoll, häufig TCP/IP, verbunden werden. Somit ist PostgreSQL in der Lage, viele verschiedene Verbindungen zu mehreren Clients gleichzeitig zu verwalten. [BB]
Dazu unterstützt es, anders als diverse Konkurrenzprodukte, den SQL92-, sowie den SQL99-Standard und es ist eine gewisse Plattformunabhängigkeit gegeben. Außerdem werden diverse Erweiterungen angeboten. Ein weiterer großer Vorteil von PostgreSQL-Datenbanken ist die Tatsache, dass die Größe einer Datenbank praktisch nur durch den zur Verfügung stehenden Speicher begrenzt wird und Tabellen mit Größen von bis 32 TB entstehen können. Ferner gibt es ein umfassendes Transaktionskonzept und es werden komplexe Abfragen mit Unterabfragen ermöglicht. Über Schnittstellen können verschiedene Programmiersprachen wie C, C++, Java/JDBC, PHP oder Perl eingebunden werden. [Posa]
Den Nutzern stehen mehrere Zusatzmodule zur Verfügung, darunter die Schnittstelle Generalized Search Tree (GiST), die z. B. die Anwendung von PostGIS ermöglicht, was in diesem Projekt eine große Rolle spielt.
Bei PostGIS, auch das „räumliche PostgreSQL“ genannt, handelt es sich um eine Erweiterung zur Verarbeitung räumlicher Daten. Es wird seit 2000 von Refractions Research als Open-Source-Software entwickelt und unter der GNU General Public Licence veröffentlicht. [Refb]
PostGIS erweitert PostgreSQL um komplexe geographische Objekte und Funktionen, was in diesem Projekt von enormer Wichtigkeit ist. Mit PostGIS wird die OpenGIS Simple Feature Access Spezifikation des Open Geospatial Consortium (OGC) umgesetzt. Das System unterstützt dabei folgende Geometrietypen: Well-Known Text/Binary, Extended Well-Known

Text/Binary und teilweise SQL/MM. Dadurch werden viele Funktionen wie z.B. Flächen- und Distanzberechnungen ermöglicht. [Refa]
Der Zugriff auf PostGIS erfolgt mit denselben Programmen, die auch auf PostgreSQL-Datenbanken zugreifen. Beispiele sind hier „psql“ oder „pgAdmin“. Außerdem wird PostGIS von verschiedenen Geoinformationssystemen wie z.B. Quantum GIS oder OpenJUMP unterstützt. Das Modul pgRouting dient dazu, Funktionen der Routenplanung, wie „Kürzester Pfad“ oder „Problem des Handlungsreisenden“, zu lösen, worauf zu einem späteren Zeitpunkt dann genauer eingegangen wird.

4.5 pgRouting

Das Hauptziel des pgRouting-Projektes ist es, für PostGIS- und PostgreSQL-Anwendungen Routingfunktionen bereitzustellen. [Posg]
Das Modul ermöglicht u. a. das Erstellen von Topologien sowie die Realisierung von diversen Routing-Algorithmen, wie die Lösung des Problems des Handlungsreisenden sowie den Dijkstra- und den A*-Algorithmus zum Determinieren des kürzesten Pfades. Die beiden erstgenannten Algorithmen spielen für dieses Projekt eine essentielle Rolle, da sie grundlegende, für das Routing benötigte Funktionen, beisteuern.
Bei pgRouting handelt es sich um ein Open-Source-Projekt von PostLBS, das als kostenlose Software auf der Webseite `http://pgrouting.postlbs.org` zur Verfügung steht. Es eignet sich auf Grund der oben erwähnten Routingfunktionen sehr gut für dieses Projekt und dessen ebenfalls aus Open-Source-Software bestehende PostgreSQL-/PostGIS-Datenbank. Zudem existiert mit dem Tool *osm2pgrouting* (`http://pgrouting.postlbs.org/wiki/tools/osm2pgrouting`) ein Werkzeug, welches den direkten Import von OpenStreetMap-Daten in eine pgRouting-Datenbank ermöglicht.
In den Kapiteln 5.1 und 5.2 wird die Funktionsweise dieser Routingfunktionen erläutert.

4.6 HTML und CSS

Bei der Hypertext Markup Language, kurz HTML, handelt es sich nicht um eine Programmiersprache, sondern um eine textbasierte Auszeichnungssprache, die der Beschreibung von Webseiten dient. HTML ist ein Standard des W3-Konsortiums.
HTML-Dokumente bestehen aus einfachem Text und in spitzen Klammern angegebenen Tags, die zur Strukturierung des Inhalts einer Webseite genutzt werden. Der Webbrowser liest ein HTML-Dokument und zeigt es als Webseite an. Dabei werden die Tags verwendet, um den Inhalt der Seite zu interpretieren und anschließend das HTML-Dokument im Browserfenster darzustellen. Eine HTML-Seite ist durch verschiedene Tags strukturiert, die üblicherweise paarweise benutzt werden, beginnend mit `<html>` und endend mit `</html>`. Sie ist grundsätzlich in einen head- und einen body-Teil gegliedert. [Dic01, vgl. S.78 ff]
Die Interaktion mit Webseiten wird unter Verwendung von Skriptsprachen wie JavaScript

(vgl. Kapitel 4.7) erheblich erhöht. Zwischen den Tags `<script>` und `</script>` kann der Quellcode der Skriptsprache direkt in das HTML-Dokument integriert werden, sodass sog. Dynamic HTML (DHTML) entsteht, welches Änderungen einer HTML-Seite durch den Nutzer ermöglicht, nachdem die Seite bereits geladen wurde. Darüber hinaus lassen sich sog. Event-Handler in den HTML-Code einbinden, die aufgrund eines bestimmten Ereignisses, wie z. B. das Anklicken eines Buttons, eine bestimmte Funktion der Skriptsprache aufrufen. [Dic01, vgl. S.116f]
In diesem Projekt wird als Skriptsprache JavaScript 4.7 in den Header des HTML-Dokuments integriert, da die hier verwendete Bibliothek OpenLayers 4.8 auf dieser basiert. Beispiele für den Einsatz von Event-Handlern sind Radiobuttons zur Auswahl verschiedener Funktionen, Textfelder zum Eingeben von Start- und Endpunkt sowie der Button zum Berechnen der Route.
HTML ermöglicht es zwar, den Inhalt einer Webseite zu strukturieren, für das Webdesign ist allerdings die Verwendung von Cascading Stylesheets (CSS) von Vorteil, da diese vielfältigere Gestaltungsmöglichkeiten bieten.
CSS sind ein vom W3-Konsortium normierter „Standard für die visuelle Darstellung von HTML-Dokumenten“ [Fla02, S.353]. Mit Hilfe von CSS können zentrale Formate definiert werden, um Farben, Schriftarten, Position und Sichtbarkeit von Elementen in einem Dokument festzulegen. Es existiert eine große Anzahl von Stilattributen, die auf die Elemente eines Dokuments angewendet werden können. CSS haben das Ziel den Inhalt und die Struktur von der Darstellung eines Dokumentes zu trennen. Hierzu werden Stylesheets, bestehend aus Regeln für das Aussehen der einzelnen Elemente, verwendet. Ein Stylesheet kann entweder zwischen den Tags `<style>` und `</style>` in ein HTML-Dokument eingebunden werden oder in einer eigenen Datei gespeichert werden. Dadurch entsteht ein übersichtlicherer Quellcode. Mit JavaScript ist es außerdem möglich auf die Stilarten von CSS zuzugreifen, diese dynamisch zu ändern und DHTML-Effekte zu erzeugen. [Fla02, vgl. S.353 ff]
Im Rahmen dieses Projektes werden CSS zum Design der Homepage verwendet. Die aktuelle Version ist CSS 2.1. [BCL]

4.7 JavaScript

JavaScript ist eine „interpretierte Programmiersprache mit Objektorientierung“ [Fla02, S.1] zur Optimierung von Webseiten. Sie ist kein direkter Bestandteil von HTML (vgl. Kap. 4.6) und ersetzt dieses auch nicht, sondern ist als eine Ergänzung zu verstehen.
1995 wurde JavaScript von Netscape eingeführt und lizenziert. Zu diesem Zeitpunkt handelte es sich noch um eine proprietäre Sprache, die heute allerdings zur freien Verfügung steht [Mün]. ECMA International, eine Organisation, die Standards in der Informations- und Kommunikationstechnologie sowie der Unterhaltungselektronik entwickelt, standardisierte JavaScript unter dem Namen „ECMAScript“. [ECM]
Die clientseitige Version von JavaScript, d. h. ein in einen Webbrowser integriertes JavaS-

cript, wird am häufigsten verwendet. Die Skriptingfähigkeit eines JavaScript-Interpreters wird mit dem vom Webbrowser implementierten Document Object Model (DOM) kombiniert. Auf diese Weise können ausführbare Inhalte in Webseiten eingefügt und weitergegeben werden und somit DHTML-Effekte erzielt werden, die eine hohe Interaktion mit dem Benutzer ermöglichen. [Fla02, vgl. S.4]
JavaScript-Anweisungen können entweder in eine HTML-Datei integriert oder in eine externe Datei geschrieben werden. Die Interpretation erfolgt durch den Webbrowser zur Laufzeit, wozu dieser mit spezieller Interpreter-Software ausgestattet ist.
Aus Sicherheitsgründen ist JavaScript nicht im Besitz bestimmter Fähigkeiten, wie beliebig Dateien zu lesen, zu schreiben oder zu löschen. Dadurch wird verhindert, dass Programme auf den Client-Rechner zugreifen und den dortigen Daten Schaden zufügen können, wenn eine JavaScript-unterstütze Webseite aufgerufen wird. [Fla02, vgl. S.449 f]
JavaScript ist die Basis des in diesem Projekt eingesetzten WebMapping-Clients OpenLayers (vgl. Kap. 4.8) und wird verwendet, um die OpenStreetMap-Daten als Karte in die Webseite einzubinden und außerdem verschiedene Kartenfunktionen zu implementieren.

4.8 OpenLayers

Der Webmapping-Client OpenLayers basiert auf der Skriptsprache JavaScript (vgl. Kap. 4.7 und dient der dynamischen Darstellung von Kartendaten auf einer Webseite. Es handelt sich um freie und Open-Source-Software und stellt ein Projekt der Open Source Geospatial Foundation (OSGeo) dar. OpenLayers unterstützt die Standards des Open Geospatial Consortiums (OGC) und bietet eine Schnittstelle zu Kartendiensten des UMN MapServers (vgl. Kap. 4.3) sowie zu WMS und WFS (vgl. Kap. 2.7), um Geodaten einzubinden. Es können sowohl eigene Daten als auch Geodaten von Google, MS Virtual Earth oder, wie in vorliegender Arbeit, von OpenStreetMap eingebunden werden. Darüber hinaus wird die Integration weiterer Komponenten wie z. B. einer Datenbank ermöglicht.
Im Rahmen dieses Projektes wird OpenLayers zur Anzeige der Kartengrundlage verwendet, da es gegenüber anderen Webmapping-Diensten wie z. B. Mapbender
(`http://www.mapbender.org/Main_Page`) diverse Vorteile bietet. Zum einen werden viele schon bestehende Funktionen in der Bibliothek OpenLayers.js
(`http://www.openlayers.org/api/OpenLayers.js`) bereitgestellt, zum anderen stehen zahlreiche Beispiele zur Verfügung, die die Anwendung vereinfachen. Außerdem wurden mit OpenLayers schneller Ergebnisse erzielt.
Gegenwärtig ist eine immer größer werdende Anzahl interaktiver Karten im Internet zu finden. Mit OpenLayers steht folglich ein Client zur Verfügung, mit dessen Hilfe leicht Karten und Kartenfunktionen in eine Webseite eingebunden werden können. Die aktuelle Version ist 2.8. [Oped], [GIS]

4.9 PHP

Der Begriff PHP steht als rekursives Akronym für „PHP: Hypertext Preprocessor“. Es handelt sich um eine Open-Source-Skriptsprache, die an die Syntax der Programmiersprache C angelehnt ist. [The10c] Ursprünglich 1995 von Rasmus Lerdorf als eine Art Sammlung von Perl Skripten entwickelt, unterlag PHP in der Folge sehr vielen Veränderungen. Die aktuelle Version PHP 5 wurde erstmals 2004 veröffentlicht. [The10a]
Heute wird PHP hauptsächlich zur Erstellung von dynamischen Webseiten und Webanwendungen verwendet. Zusätzlich kann mit PHP durch die Aktivierung unterschiedlicher Module auf Datenbanken verschiedener Hersteller zugegriffen werden, z. B. MySQL, MS-SQL, Oracle und auch PostgreSQL.
Genutzt wird PHP-Code in HTML-Seiten. Durch die Endung .php übermittelt man dem Webserver, dass die Daten nicht nur HTML-, sondern auch PHP-Code enthalten können. Dieser wird lediglich auf dem Server interpretiert und gibt für die Seite nur das Ergebnis des Codes zurück, nicht den ganzen Quellcode.
Durch das serverseitige Interpretieren entstehen u. a. folgende Vorteile: Der Nutzer sieht nur den zurückgegebenen Teil, jedoch keinen Quelltext. Damit benötigen die Datenbanken keine direkte Verbindung zum Client. Weiterhin treten, anders als bei z. B. JavaScript, keine Inkompatibilitäten auf, da die Clients keine Plug-Ins benötigen. [The10b]
Im Rahmen dieses Projektes wird PHP in vielen Dateien verwendet, doch besonders ist hier die Datei `routing.php` (vgl. Kap. 5.1) zu erwähnen, diese enthält alle wichtigen Funktionen und Aufrufe zur Nutzung von *fahrradies.net*.

4.10 AJAX

Ajax steht für „Asynchronous JavaScript and XML“ und dient der asynchronen Datenübertragung zwischen Server und Webbrowser. Es unterstützt die Erstellung von Web 2.0 - Anwendungen (vgl. Kap. 2.4).
Der Begriff Ajax wurde erstmals im Februar 2005 durch Jesse James Garrett, dem Präsidenten und Gründer von Adaptive Path, in seinem Essay „Ajax: A New Approach to Web Applications“ [Gar05] verwendet. Dabei ist Ajax keine neue Programmiersprache, sondern basiert auf bekannten Standards wie HTML und CSS (vgl. Kap. 4.6), dem Document Object Model (DOM), XML, XMLHttpRequest sowie JavaScript (vgl. Kap. 4.7). Beispiele für die Anwendung sind Gmail, Google Suggest und Google Maps.
XMLHttpRequest ermöglicht die asynchrone Datenübertragung zwischen Server und Webbrowser und ist eine Voraussetzung für die Realisierung von Ajax. Über ein XMLHttpRequest-Objekt ist JavaScript in der Lage, direkt mit dem Server zu kommunizieren. Dazu wird mittels JavaScript im Browser des Clienten eine Anfrage an den Server gestellt, welche von diesem ausgeführt wird. Die Antwort des Servers ist zumeist in XML codiert.
Die Datenübertragung zwischen Server und Webbrowser findet im Hintergrund statt und

erfolgt asynchron über die sog. Ajax Engine, die zwischen User und Server geschaltet ist. Diese Vorgehensweise bietet den Vorteil, dass nur die tatsächlich benötigten Daten vom Server angefordert werden. Dadurch ist es nicht erforderlich die gesamte Webseite neuzuladen, sodass der Benutzer weiterhin in der Lage ist mit der Oberfläche zu interagieren, während die Daten vom Server geladen werden. [Gar05]
In diesem Projekt ist die Interaktivität zwischen Benutzer und Karte von grundlegender Bedeutung. Eine höhere Interaktivität ist immer mit einem höheren Datenaustausch verbunden, somit ist der Einsatz von Ajax sinnvoll. Ein Beispiel für die Verwendung ist die Autovervollständigung der Benutzereingabe (vgl. Kap. 5.7.6.3). Des Weiteren existieren native OpenLayers Ajax-Funktionen, die in Kapitel 5.9.2.6 beschrieben werden.
Weitere Vorteile, die der Einsatz von Ajax bietet, sind die Erhöhung der Benutzerfreundlichkeit und der Geschwindigkeit von Internetanwendungen, da schneller auf Benutzeranfragen reagiert werden kann. Darüber hinaus sind Ajax-Anwendungen plattformunabhängig und erfordern keine Browser-Plug-Ins. Die Aktivierung von JavaScript im Browser ist allerdings notwendig. [Aja]

5 Implementation der Funktionen des Radroutenplaners

Dieses Kapitel widmet sich der Realisierung des Radroutenplaners und der Implementierung der verschiedenen Funktionen. Durch die im vorherigen Kapitel aufgeführten Software-Grundlagen soll es dem Leser nun möglich sein, nachzuvollziehen, wie die einzelnen Funktionen umgesetzt werden.
Im Folgenden werden alle wichtigen Funktionen, die zu *fahrradies.net* gehören, genauestens dargelegt. Auch wenn nicht alle Feinheiten der Umsetzung erklärt werden können, soll dieses Kapitel die interessanteste Thematik behandeln.
Durch Codefragmente wird beispielhaft gezeigt, wie und an welcher Stelle sich eine Funktion in der Umsetzung einfügt. Da außerdem der komplette Quellcode veröffentlicht wird, kann so bei der Weiterverarbeitung eine Hilfestellung geleistet werden.
Zunächst wird die wesentliche Funktion eines Routenplaners, das Routing, beschrieben. Des Weiteren wird erläutert, wie eine Route auch über Zwischenpunkte geführt werden kann. Außerdem wird die Implementierung verschiedener Profile, der Einbezug von Steigungen und die Erzeugung eines Höhenprofils sowie die Routenbeschreibung geschildert. Eine weitere wesentliche Funktion ist die Straßensuche und in diesem Zusammenhang auch die POI-Suche. Es folgen allgemeine Funktionen, die Möglichkeit des Tracklog-Exports sowie die Auswahl vorgefertigter Themenrouten. Abschließend wird die Umsetzung der Druckfunktion beschrieben und ein kurzer Überblick über die Routenbewertung gegeben.

5.1 Routing

Im Fokus dieses Kapitels steht die Integration der wichtigsten Funktion eines Routenplaners: die Berechnung einer Route zwischen zwei vom User gesetzten Punkten. Um die Bedienung dieser Funktion so möglichst einfach und intuitiv zu gestalten, wird die Option angeboten, den Start- und Endpunkt durch einen direkten Klick in die Karte selbst zu setzen. Des weiteren sollen die Punkte verschiebbar gemacht werden, um ungenau gesetzte Punkte und somit unerwünschte Routenberechnungen direkt korrigieren zu können (vgl. Kapitel 5.9.1.1).
Im Folgenden wird zunächst die clientseitige Implementation beschrieben und danach darauf eingegangen, welche Berechnungen serverseitig im Hintergrund ablaufen.

5.1.1 Clientseitige Implementation mit OpenLayers und JavaScript

Beim Laden der Hauptseite werden u.a. auch die anfangs leeren Layer diezur Darstellung der vom User gesetzten Start- und Endpunkte sowie der berechneten Route erstellt. Start- und Endpunkt sind durch unterschiedliche Symbole voneinander unterscheidbar, welches sich durch OpenLayers-Styles realisieren lässt (siehe Kapitel 5.9.2.1). OpenLayers bietet die Möglichkeit zu definieren, was beim Klick des Users in die Karte passieren soll (siehe Kapitel 5.9.1). Durch Betätigen des Schaltfeldes für das Setzen eines Markers (links neben den For-

mularen für die Straßensuche) wird eben diese Funktion aktiviert und das Symbol für den entsprechender Marker in der Karte an der gewählten Position angezeigt.
Sobald beide Punkte gesetzt sind, wird der „Los!“-Button anklickbar. Betätigt man diesen Button, wird genau wie beim Verschieben eines Punktes die Funktion `compute()` aufgerufen, die die Verantwortung für die Neuberechnung der Route trägt. Die Methode bekommt zwei Parameter übergeben: jenen der angibt, ob auf die Straße gezoomt werden soll und jenen, der bestimmt, ob die berechnete Route in sortierter Form zurückgegeben werden soll. Auf den Nutzen der Sortierung wird im späteren Verlauf noch eingegangen (s. Kapitel 5.5.1).
Sollte der Zoom aktiviert sein, wird mit geeigneten OpenLayers-Funktionen die Bounding-Box[2] ermittelt, welche beide Punkte des `markerLayer`s enthält, und dann auf diese BoundingBox gezoomt (siehe Kapitel 5.9.2.3).
Nachdem die Voraussetzungen für die Routenberechnung erfüllt sind, können die entsprechenden Routinen für Berechnung der Route in Gang gesetzt werden. Durch die Wahl von zwei Punkten innerhalb derer die kürzeste Route gefunden werden soll, sind zwei geographische Positionen in Form von jeweils x- und y- Koordinaten gegeben. Im Weiteren erfolgt die Weitergabe dieser Koordinaten mitsamt aller weiteren Übergabevariablen (Profilname und Reihenfolge der Zwischenpunkte) an die Datei `routing.php`.
Der Arbeitsweise der Routingdatei wird aufgrund ihrer Komplexität ein eigenes Kapitel gewidmet (s. Kapitel 5.1.4).
Nach der Routenberechnung geht es weiter mit der Verarbeitung des zurückgelieferten Inhaltes der PHP-Datei. Das PHP-Skript gibt seine berechnete Route als sog. `XMLHttpRequest`-Objekt zurück, auf dessen textuellen Inhalt man mit von OpenLayers bereitgestellten Funktionen zugreifen kann. Da eine XML-Struktur vorliegt, kann auch gezielt auf Inhalte einzelner Tags zugegriffen werden (siehe Kapitel 2.2.3 XML und die OpenStreetMap-XML-Struktur). Zunächst werden alle Kanten (`<edge>`-Tags) geparsed[3]. Danach wird deren Inhalt in einem Array gespeichert, aus jeder Kante das entsprechende WKT (siehe Kapitel 2.6 Open Source Geospatial Consortium und OpenGIS Standards) als String ausgelesen und dem Layer hinzugefügt, welcher sie als rote Linienzüge in der Karte darstellt.

5.1.2 Erzeugung des Routing-Graphen mit OSM2pgRouting

OpenStreetMap-Daten eignen sich in ihrer ursprünglichen Form aus zwei Gründen nicht für die Benutzung mit pgRouting: Erstens weisen sie keine Topologie auf, d. h. die Straßen haben keinen Zusammenhang. Zweitens ist „die Art der Repräsentation der Wege unbrauchbar, weil Wege oft aus mehreren Kanten bestehen können.“ [Beh09]
Abhilfe schafft das von Daniel Wendt geschriebene Tool OSM2pgRouting [Posc]. Damit lassen sich Informationen für Straßen aus der OpenStreetMap-XML-Datei (ways) direkt in die Datenbank einlesen und für die Routenberechnung aufbereiten. Jeder Straße wird eine eige-

[2] Kleinstes umgebendes Rechteck des Darstellungsbereiches
[3] parsen: Weiterverarbeitung der Eingabe in ein geeignetes Format

ne Identifikationsnummer (gid), sowie ein Startknoten (source) und ein Endknoten (target) zugeordnet. In der Abbildung 19 sind diese Zuordnungen durch das Zahlentupel (gid, source,

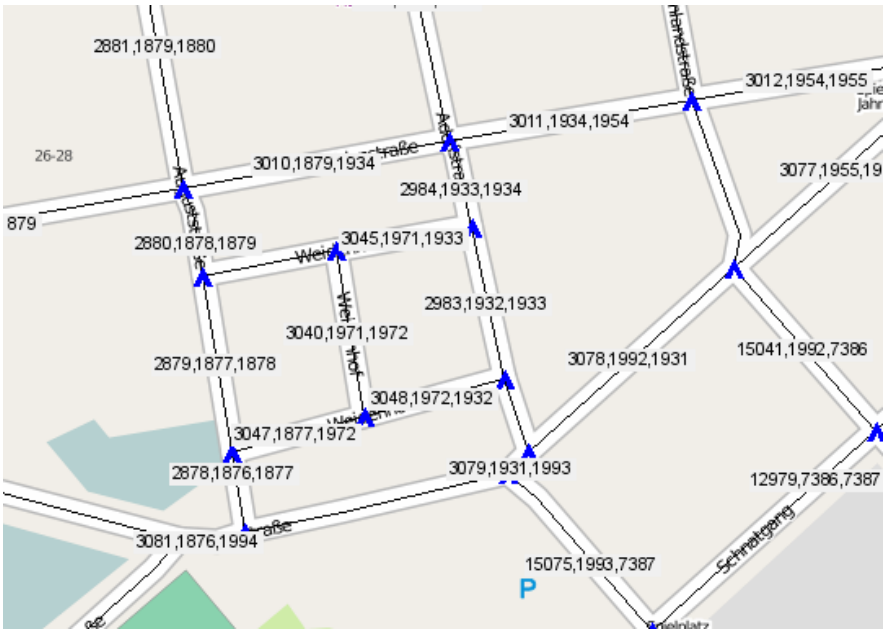

Abbildung 19: *Aus OpenStreetMap-Daten erzeugter Graph*
Quelle: Eigene Darstellung

target) über den jeweiligen Straßen dargestellt. Im Hintergrund sind die Kacheln des Mapnik-Layers zu sehen. Die schwarzen Linien und blauen Knoten stellen den durch OSM2pgRouting erzeugten Graphen dar. Die Straßen haben, definiert durch Start- und Endknoten, eine Orientierung, der Routing-Graph ist also gerichtet. Die Orientierung entspricht der Richtungsdarstellung der Straße im Editor JOSM, sie ist keineswegs willkürlich vergeben.
Weiterhin werden Straßen dort in ihre Segmente unterteilt, wo andere Segmente der gleichen Straße oder anderer Straßen anschließen. Dazu wird der Grad eines jeden Knotens gezählt und die Straße dort unterteilt, wo dieser größer als eins ist. Sobald nun die OSM-Daten vollständig für die Benutzung für pgRouting aufbereitet sind, wird mit der Funktion `assign_vertex_id()` die Topologie erzeugt (vgl. [Posd]). Zusätzlich wird vor allem für die Profile (s. Kapitel 5.3) ein Patch für OSM2pgRouting [Posh] verwendet, welcher zu jeder Kante die dazugehörige OpenStreetMap-Identifikationsnummer (OSMID) mit in die Datenbank schreibt. OSM2pgRouting bietet zwar die Möglichkeit in einer XML-Konfigurationsdatei Klassen und Typen anzugeben, um diese im Anschluss für Kostenfunktion bzw. Profile nutzen zu können, jedoch haben sich bei der Benutzung dieser Funktion einige Probleme herausgestellt:

- Man kann einer Straße lediglich einen Wert eines Tags zuordnen. Es ist nicht möglich Klassen mit mehreren Tags zu definieren. Eine Straße mit den Tags „highway“ und „cycleway“ lässt sich entweder über „highway“ oder über „cycleway“ zuordnen. Da viele unterschiedliche Eigenschaften einer Straße mit in die Routenberechnung einfließen

sollen, genügt dies den Ansprüchen des Radroutenplaners nicht.

- Straßen mit bestimmten Kombinationen von Tags wurden aufgrund von Programmfehlern überhaupt nicht in die Datenbank eingelesen.

5.1.3 Dijkstra-Algorithmus von pgRouting und damit verbundene Probleme

pgRouting stellt einen Dijkstra-Algorithmus bereit, welcher in der Lage ist, Kosten für den Hinweg über eine Kante (Spalte „to_cost“ in der Tabelle „ways“) und auch den Rückweg (Spalte „reverse_cost“) mit einzubeziehen [Posf]. Gerade Letzteres ist für einen Routenplaner von besonderer Wichtigkeit, weil nicht jede Straße in beide Richtung befahren werden darf. Ein Beispiel dafür sind Einbahnstraßen, bei denen die Kosten für den Rückweg theoretisch unendlich hoch sind.
Ein großes Problem stellt die Tatsache dar, dass lediglich zwischen Knoten geroutet werden kann. Schließlich möchte der User seine Marker in die Karte setzen und seine Route soll genau dort enden. Die gesetzten Punkte teilen die Straßen bzw. Kanten in zwei Teile, existieren jedoch in der Datenbank als zusammenhängende Kante. Der Datenbestand in der Datenbank soll auch in der Datenbank nicht im laufenden Betrieb geändert werden. Es ist also erforderlich, Straßen temporär während der Verarbeitung abzuschneiden, um eine Geometrie zu erzeugen, die später durch OpenLayers angezeigt wird. Angezeigt wird letztlich nur der am vom User gesetzten Punkt abgeschnittene Teil der Start- und Endkante. Ein Problem für die Routenberechnung ist dies auch insofern, als dass die Kosten der (abgeschnittenen) Start- und Endkante mit in die Berechnung einfließen müssen, um die optimale Route berechnen zu können. Da pgRouting dies von Haus aus nicht unterstützt, ist ein Workaround nötig.

5.1.4 Serverseitige Implementierung: Arbeitsweise der Datei routing.php

In diesem Beispiel soll zunächst die grundlegende Arbeitsweise der Datei `routing.php` aufgezeigt werden, weswegen Zwischenpunkte und die Weiterverarbeitung der Route für z. B. das Höhenprofil außer Acht gelassen werden. Hier wird die Berechnung der Route zwischen lediglich zwei Punkten beschrieben.
Die als Request übergebenen Koordinaten beider Punkte werden geparsed. Nun wird zunächst zu beiden Punkten jeweils die nächstgelegene Kante ermittelt, welche im Folgenden als Start- und Endkante bezeichnet werden (siehe Abbildung 20). Es soll immer bis zur nächstgelegenen Kante geroutet werden, weil nicht erwartet werden kann, dass der User beim Setzen der Marker in die Karte tatsächlich genau eine Kante trifft. Deshalb wird in einer BoundingBox von $200m \times 200m$ um die gesetzten Marker herum nach Kanten gesucht und die nächstgelegene ermittelt. Die Marker sind in der Abblidung grün dargestellt.
Danach wird zu beiden vom User gesetzten Punkten der nächstgelegene Punkt auf beiden zuvor ermittelten Kanten gesucht. Diese beiden Punkte stellen die Punkte dar, an welchen die Route schließlich starten und enden soll, weswegen im weiteren Verlauf die Bezeichnungen

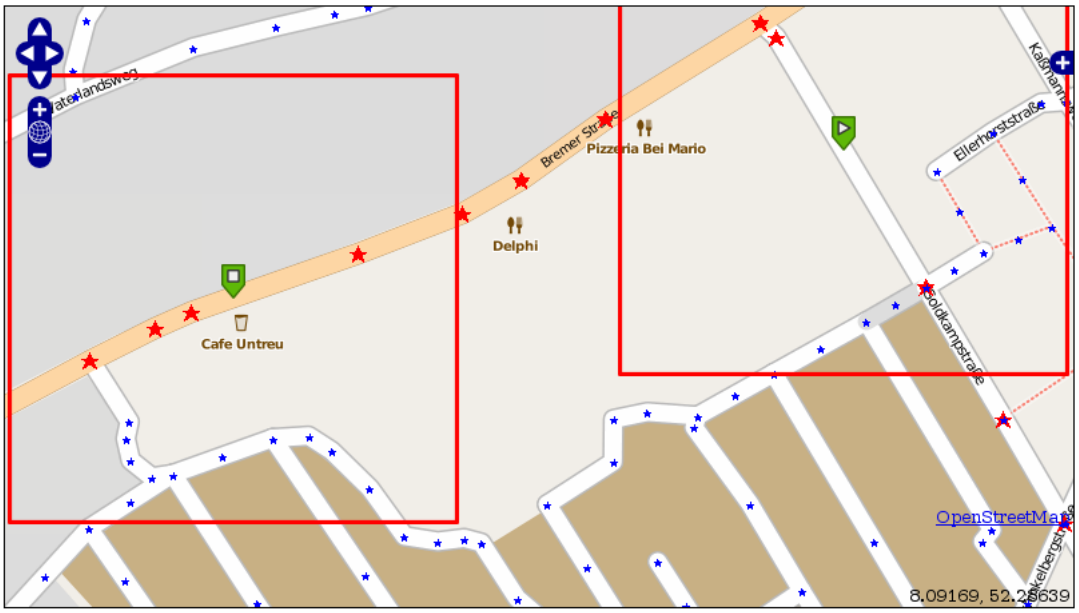

Abbildung 20: *Ermittlung der nächstgelegenen Kanten zu den vom User gesetzten Markern innerhalb einer BoundingBox von 200m*

Quelle: angepasst von `http://pgrouting.postlbs.org/wiki/LoadingtheCode3`

Start- und Endpunkt verwendet werden.
Zur Weiterverarbeitung werden mehrere Fälle unterschieden, welche sich aus der bereits erwähnten Problematik ergeben. Außerdem muss eine Sonderbehandlung vorgenommen werden, wenn Start- und/oder Endkante Einbahnstraßen sind.

Wichtiger Hinweis: Die Routenberechnung mit dem Dijkstra-Algorithmus von pgRouting liefert eine Tabelle zurück (Beispiel s. [Posf]). Die kürzeste Route kann natürlich auch über die Start- und/oder Endkante selbst zu diesen Knoten führen. Mit Hilfe einer SQL-Anweisung wird sichergestellt, dass die Start- und die Endkante *nicht* mit in der Ausgabe sein werden, weil sie nicht vollständig in der Route enthalten sind. Es kann also passieren, dass zwischen zwei Knoten geroutet wird, aber einer der beiden oder gar beide Knoten in der darauffolgenden Betrachtung keine Rolle mehr spielen, weil die Route aufgrund dieses Ausschlusses am anderen Knoten derselben Kante endet.

5.1.4.1 Reguläre Route

Im Mittelpunkt der Betrachtung steht zunächst die Behandlung einer regulären Route, d. h. Start- und Endpunkt sind nicht auf der gleichen oder auf direkt aneinander grenzenden Kanten gesetzt (s. 5.1.4.4). Es werden wiederum zwei Fälle unterschieden:

- Je kürzer die Route ist, desto vermeintlich größer ist der Anteil eines Umwegs an der Gesamtroute. Der User benutzt den Routenplaner nicht, um eine längere Radtour zu planen, sondern um sich voraussichtlich die kürzeste Route ausgeben zu lassen und er

möchte auf einer kurzen Strecke keinen Umweg in Kauf nehmen.

- Bei langen Strecken spielen kleine, von der optimalen Route abweichende Umwege eine untergeordnete Rolle und sind eher zu vernachlässigen. Der User plant voraussichtlich einen längeren Ausflug mit dem Rad. Hier ist die Geschwindigkeit der Routenberechnung, vor allem bei sehr langen Routen, entscheidend.

Um diesen Unterschied zu ermitteln, wird die euklidische Distanz zwischen Start- und Endpunkt gemessen. Wenn diese unter einem Kilometer liegt, wird das Verfahren für die optimale Routenfindung verwendet und anderenfalls ein schnelleres Verfahren. Die euklidische Distanz zwischen Start- und Endpunkt ist in einem Graphen natürlich kein optimal geeignetes Kriterium, weil damit keine genaue Aussage über die Zahl der zu traversierenden Kanten gemacht wird. Letztgenannte ist entscheidend für die Geschwindigkeit der Berechnung, setzt jedoch eben diese voraus. In ländlichen Räumen mit weniger Straßen ist die Distanz von einem Kilometer aufgrund des weniger komplexen Graphen anders zu bewerten als in urbanen Räumen. In der Praxis hat sich das Maß jedoch im Durchschnitt als tauglich erwiesen.
Die beiden aufgelisteten Routenberechnungsmethoden werden im Folgenden näher betrachtet.

5.1.4.2 Behandlung kurzer Routen

Möchte man die kürzeste Route zwischen Start- und Endpunkt ermitteln und kann nur zwischen den Knoten der Start- und Endkante routen, lässt sich keine Aussage darüber treffen, welche zwei der vier Punkte es sein müssen. Dementsprechend müssen alle vier Möglichkeiten der Kombination von Knoten, zwischen welchen die Route berechnet werden kann, in Betracht gezogen werden:

- vom source-Knoten der Startkante zum source-Knoten der Endkante
- vom source-Knoten der Startkante zum target-Knoten der Endkante
- vom target-Knoten der Startkante zum source-Knoten der Endkante
- vom target-Knoten der Startkante zum target-Knoten der Endkante

Weil auch ein Teil der Längen der Start- und Endkante für die exakte Berechnung der Gesamtlänge der Route eine Rolle spielt, müssen zu jeder der vier berechneten Routen die Start- und Endkante in Richtung des jeweils anschließenden Segmentes der Route zwischen den Knoten abgeschnitten werden.
In Abbildung 21 sieht man dies an einem Beispiel verdeutlicht. In rot dargestellt ist die Route zwischen zwei Knoten, grün der an die Route anschließende Teil der Kante und gelb der übrige Teil der Kante. Die Kosten der abgeschnittenen Kanten stellten den prozentualen Anteil entweder für Hin- oder Rückweg dar, je nachdem in welcher Richtung über die entsprechende Kante zur anschließenden Route gelaufen wurde.

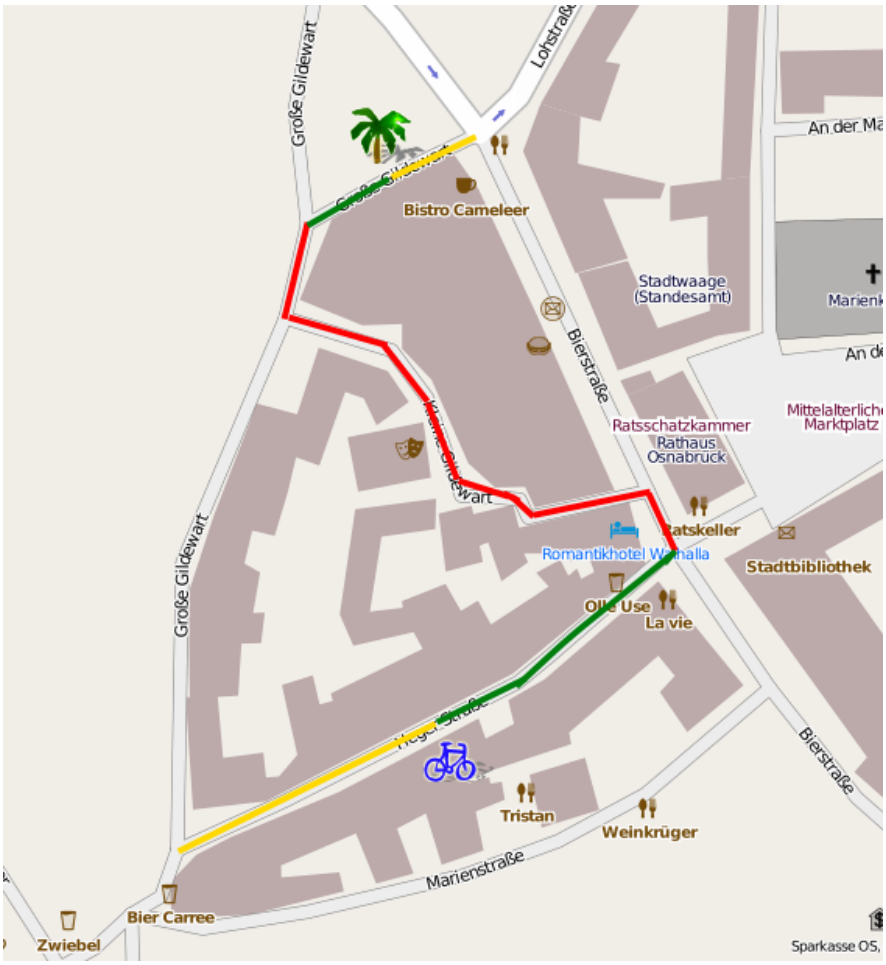

Abbildung 21: *Beispielhaftes Abschneiden der Kanten*
Quelle: Eigene Darstellung

- Für die Startkante gilt:
  - Schließt die Route am source-Knoten an, muss die Startkante vom Startpunkt (vom User gesetzt) bis zum anschließenden Rest der Route rückwärts traversiert werden. Das bedeutet wiederum, dass die Kosten für den Rückweg benutzt werden müssen.
  - Schließt die Route am target-Knoten an, werden die Kosten für den Hinweg über die Kante benutzt.

- Für die Endkante gilt entsprechend:
  - Schließt die Route am source-Knoten an, muss die Route über die Endkante in Richtung des Endpunktes (vom User gesetzt) traversiert werden. Es werden die Kosten für den Hinweg über die Kante benutzt.
  - Schließt die Route am target-Knoten an, werden die Kosten für den Rückweg über die Kante benutzt.

Die Kosten für die Route zwischen den Knoten müssen nicht gesondert ermittelt werden, da der Dijkstra-Algorithmus von pgRouting automatisch die beim Durchlauf der Kante verwendeten Kosten liefert, sei es für Hin- oder Rückweg. Abschließend werden die Kosten für alle vier Routen einzeln aufsummiert und anhand der Gesamtkosten die kürzeste Route ausgewählt.

5.1.4.3 Behandlung langer Routen

Das soeben vorgestellte Verfahren hat sich als nicht besonders performant erwiesen. Das viermalige Berechnen einer Route bedeutet besonders bei langen Routen eine Verzögerung, ist aber für die Ermittelung der optimalen Route unumgänglich. Gerade jedoch bei langen Routen machen sich kleine Umwege in der Gesamtroute kaum bemerkbar und es ist zudem wahrscheinlicher, dass der User nicht auf der Suche nach der optimalen bzw. kürzesten Route im Sinne der geringsten Kosten ist. Hier wird auf ein Verfahren gesetzt, welches in der Praxis oft gute Ergebnisse erzielt, die Route nur ein einziges Mal berechnet und damit deutlich schneller arbeitet. Folgender Algorithmus wird angewendet:

1. Überprüfe für beide Kanten, ob Einbahnstraßen vorliegen, indem nach dem Tag oneway=yes gesucht wird. Liegt zusätzlich das Tag bicycle=opposite vor, behandele die Kante, als wäre sie keine Einbahnstraße.
   a) Es liegt eine Einbahnstraße vor:
      i. Handelt es sich um die Startkante, nehme den target-Knoten als Ausgangspunkt für die Routenberechnung.
      ii. Handelt es sich um die Endkante, nehme den source-Knoten als Endpunkt für die Routenberechnung.

      Damit ist sichergestellt (obwohl bei diesem Verfahren ansonsten die Eigenschaften der Start- und Endkante außer Acht gelassen werden), dass der Routenplaner den User nicht in Rückrichtung über Einbahnstraßen schickt und ihn einer Gefahr aussetzt.
   b) Liegt *keine* Einbahnstraße vor, bestimme den Knoten der Kante, der dem vom User gesetzten Punkt am nächsten liegt und nehme diesen als Ausgangspunkt für die Routenberechnung, falls es sich um die Startkante handelt, anderenfalls als Endpunkt.
2. Berechne die Route zwischen den zwei zuvor ermittelten Knoten.
3. Schneide Start- und Endkante in Richtung der anschließenden Route ab (vgl. Abb. 21).

Es ist nicht gesichert, dass mit diesem Verfahren die optimale Route im Sinne der geringsten Kosten berechnet wird, da in einigen Fällen die kürzeste Route zum Start- und Endpunkt

über einen anderen als den gewählten Knoten führt. Je länger die berechnete Route ist, desto kleiner ist der Umweg im Verhältnis zur Gesamtroute, wodurch der Umweg zu vernachlässigen ist. Außerdem führt die Tatsache, dass der nächstgelegenste der beiden Knoten gewählt wird, zu optisch guten Ergebnissen. Es wirkt auf den Benutzer als sei die kürzeste Route berechnet worden, der Umweg ist kaum bemerkbar. Das vorgestellte Verfahren benötigt für die Berechnung einer gleich langen Route ungefähr ein Drittel der Zeit, was das ausschlaggebende Kriterium zur Benutzung dieser Methode bei längeren Routen ist.

5.1.4.4 Sonderfälle

Es treten einige Sonderfälle auf, die dadurch entstehen, dass die Start- und Endkante nicht Teil der Route sind. Zusätzlich erfordert das Auftreten von Einbahnstraßen, dass die Geometrien der Kanten anders als bisher gewohnt abgeschnitten werden. Die folgenden Beispiele werden die Problematik verdeutlichen und den Lösungsweg aufzeigen. Weil es sich hier um sehr kurze Distanzen handelt, wird angenommen, dass der User auf der Suche nach der kürzesten Route ist.

5.1.4.5 Start- und Endpunkt befinden sich auf derselben Kante

Durch einen Vergleich der `gid` beider Straßen lässt sich ermitteln, ob die Start- und Endkante identisch sind und der User somit beide Marker auf dieselbe Straße gesetzt hat. Für den weiteren Verlauf gilt es nun herauszufinden, ob mit oder entgegen der Orientierung geroutet werden soll. Dazu wird mit Hilfe der PostGIS-Funktion `distance()` die Entfernung der beiden Punkte zum source-Knoten der Kante gemessen. Befindet sich der Startpunkt näher am source-Knoten als der Endpunkt, wird in Orientierungsrichtung der Straße geroutet. Liegt jedoch der Endpunkt näher am source-Knoten, wird entgegen der Orientierung geroutet. Es muss eingestanden werden, dass diese Methode nicht immer die korrekten Ergebnisse liefert. In Abbildung 22 ist schematisch eine Straße als gerichtete Kante dargestellt, auf welcher der

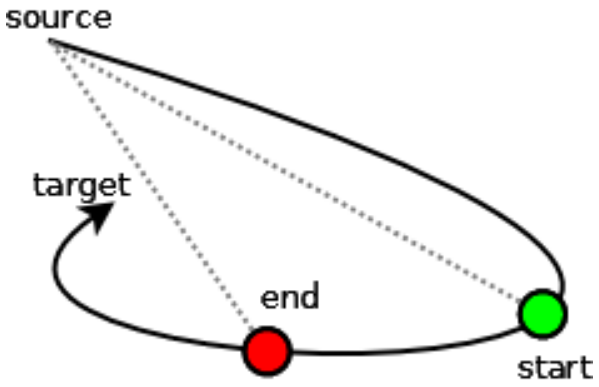

Abbildung 22: *Konstruierte Möglichkeit, bei welcher der Algorithmus nicht die korrekte Lösung liefert*

Quelle: Eigene Darstellung.

User Start- (grün) und Endpunkt (rot) gesetzt hat. In diesem Fall ist der Endpunkt näher am source-Knoten, obwohl er im Verlauf der Kante hinter ihm liegt. Dafür verantwortlich ist die kurvenartige Form der Kante. In der Praxis ist dieser Fall jedoch äußerst selten und die Methode liefert in nahezu 100 Prozent der Fälle das korrekte Ergebnis.
Nachdem nun bekannt ist, ob mit oder gegen der Orientierung der Straße geroutet werden muss, lassen sich drei Möglichkeiten für den Fall, dass der User seine Marker auf der gleichen Kante gesetzt hat, definieren:

1. Muss mit der Orientierung geroutet werden, ist es unerheblich, ob eine Einbahnstraße vorliegt oder nicht. Die Kante wird außen am Startpunkt und am Endpunkt abgeschnitten, so dass der Teil zwischen Start- und Endpunkt erhalten bleibt.

2. a) Wird die Kante entgegen der Orientierung befahren und liegt keine Einbahnstraße vor, wird die Kante wie zuvor beschrieben abgeschnitten.

   b) Wird entgegen der Orientierung über die Kante geroutet und liegt tatsächlich eine Einbahnstraße vor, wird zuerst mit dem Dijkstra-Algorithmus die Route vom target-Knoten zum source-Knoten berechnet. Dabei ist zu beachten, dass wirklich von target zu source geroutet wird, da sonst nicht die Kosten für den Rückweg, welche bei Einbahnstraßen theoretisch unendlich hoch sind, berücksichtigt werden. Danach wird die Kante zwischen source-Knoten und Endpunkt, sowie zwischen target-Knoten und Startpunkt abgeschnitten.

Im letztgenannten Fall wirkt die Gesamtroute, als ob sozusagen „drumherumgeroutet“ wurde (s. Abb. 23). Es wird deutlich, warum hier eine Sonderbehandlung erforderlich ist: Genau derjenige Teil der Kante, der in Punkt 1 und 2(a) das Endergebnis darstellt, fällt in Punkt 2(b) weg.

Anmerkung: Es wurde bei den genannten Fällen völlig außer Acht gelassen, dass die Kosten in Orientierungsrichtung hoch gesetzt sein könnten und entgegen der Orientierung niedrig. Da den Autoren dieser Arbeit kein solcher Anwendungsfall bekannt ist und in OpenStreetMap Einbahnstraßen in Orientierungsrichtung eingetragen sind, wurde dies vernachlässigt.

5.1.4.6 Start- und Endkante grenzen direkt aneinander

Wenn Start- und Endkante direkt aneinander grenzen, gibt es ebenfalls eine gesonderte Vorgehensweise:

1. Ermittle den Knoten, den beide Kanten gemeinsam haben.

2. Berechne mit dem Dijkstra-Algorithmus von pgRouting die Route zwei Mal: ausgehend von dem Knoten der Startkante, der *nicht* der gemeinsame Knoten ist, bis zu beiden Knoten der Endkante.

Abbildung 23: *Beispiel für den Fall, dass Start- und Endmarker auf einer Einbahnstraße in Rückrichtung gesetzt sind*

Quelle: Eigene Darstellung.

3. Behandele beide Routen nach folgendem Schema:

   - Wenn der Dijkstra-Algorithmus kein Ergebnis hat, bedeutet dies, dass die Route nur die zwei ursprünglichen Kanten enthält (Zur Erinnerung: Diese werden durch eine SQL-Anweisung nicht mit ausgegeben, falls sie Teil der Route sind). Schneide also beide Kanten in Richtung der jeweils anderen ab und summiere den prozentualen Anteil der Kosten auf. Dieser errechnet sich durch das Verhältnis der Länge der abgeschnittenen Kante zur Länge der ursprünglichen Kante multipliziert mit den Kosten. Wähle die Kosten für Hin- oder Rückweg je nachdem, ob die Kanten vorwärts oder rückwärts traversiert wurden.
   - Wenn das Ergebnis nicht leer ist, ist mindestens eine der beiden Kanten eine Einbahnstraße. Die Route wird wie eine reguläre Route behandelt: Schneide Start- und Endkante in Richtung der anschließenden Route ab und summiere die Gesamtkosten inkl. der anteiligen Kosten der Start- und Endkante auf. Wie bereits zuvor gilt: Wähle die Kosten, je nachdem ob die Kanten vorwärts oder rückwärts traversiert wurden.

4. Wähle die Route mit den geringeren Gesamtkosten aus.

5.1.5 Ausgabe der Route im XML-Format

Zu guter Letzt wird die gesamte Route inkl. der abgeschnittenen Geometrien in einer XML-Struktur wie folgt ausgegeben:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
<route>
  <edge id='1'>
   <id>21603</id>
   <wkt>MULTILINESTRING((895360.609152215
      6850182.48643123,895334.739633228 6850201.29092619))</wkt>
   <length>0.0195934439568297</length>
  </edge>
  <edge id='2'>
    <id>21604</id>
    <wkt>MULTILINESTRING((895360.609152215
       6850182.48643123,895405.844199718 6850157.72846767))</wkt>
    <length>0.0316014165340881</length>
  </edge>
</route>
```

Jede Kante trägt das Tag `<edge>` mit einer fortlaufenden Identifikationsnummer, welche die Reihenfolge widerspiegelt. Außerdem enthalten ist das Tag `<id>` mit der `gid` einer jeden Kante, deren Geometrie im WKT-Format in der Google-Projektion und die Länge der Kante in der Einheit Kilometer.

5.2 Zwischenpunkte

Als weiteres Feature kann der User Zwischenpunkte setzen, welche sowohl in fester, als auch in optimaler Reihenfolge angefahren werden können. Fest heißt in diesem Zusammenhang, dass die Punkte in der Reihenfolge, in welcher der User sie gesetzt hat, abgearbeitet werden sollen. Im anderen Fall wird die Reihenfolge so gewählt, dass die Gesamtroute durch alle Punkte minimal kurz bleibt.

5.2.1 Benutzeroberfläche

Die Benutzeroberfläche der Webseite bietet dem User die Wahl zwischen den beiden eingangs erwähnten Varianten (Abbildung 24). Voreingestellt ist die feste Reihenfolge, weil sie bei der ersten Benutzung die intuitivere Möglichkeit ist. Man erwartet zunächst, dass die Zwischenpunkte in der gesetzten Reihenfolge angefahren werden. Sobald der User seine Auswahl trifft, wird intern eine Variable für die Berechnungsart auf den entsprechenden Wert (fest/optimal) gesetzt. Zwischenpunkte können eingefügt werden, nachdem Start- und Endpunkt bereits gesetzt sind. Die Koordinaten der Punkte werden in der Reihenfolge, in der sie gesetzt wurden,

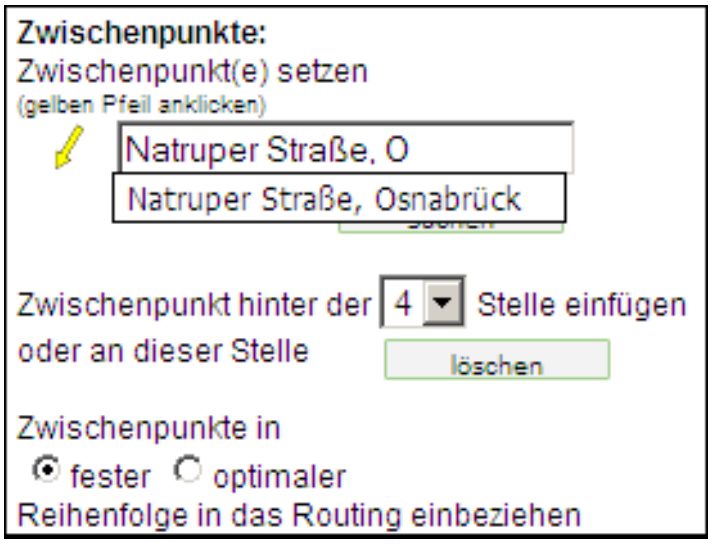

Abbildung 24: *Benutzeroberfläche Zwischenpunkte*
Quelle: Eigene Darstellung

zusammen mit der ausgewählten Berechnungsart an die Datei `routing.php` übergeben. Die feste Reihenfolge der Zwischenpunkte bietet dem Nutzer den Vorteil, seinen Routingverlauf individueller planen zu können. Da bei dieser Variante die Reihenfolge der gesetzten Zwischenpunkte einen maßgeblichen Einfluss auf die berechnete Route hat, ist es komfortabel, dem Nutzer die Möglichkeit zu geben, die Reihenfolge nicht allein davon abhängig zu machen, in welcher Abfolge die Punkte der Route hinzugefügt wurden, sondern es zu ermöglichen diese Reihenfolge auch im Nachhinein noch zu verändern. Aus diesem Grund wurde eigens eine Array-Datenstruktur implementiert, welche die Zwischenpunkte wie in einer Art Liste verwaltet und es somit ermöglicht, nicht nur an der letzten, sondern an beliebiger Stelle einen neuen Zwischenpunkt einzufügen oder einen schon existierenden Punkt zu entfernen. Die Datei `updateViaDropDown.js` wurde zu dem Zweck implementiert, die Benutzeroberfläche nach dem Löschen oder Einfügen eines Zwischenpunktes der neuen Zwischenpunktsituation anzupassen. Das gesamte Konstrukt ist so aufgebaut, dass das Einfügen oder Löschen eines Zwischenpunktes durch die Benutzeroberfläche zunächst die entsprechende Funktion der Datenstruktur aufruft. Diese aktualisiert sich, um alle sich anschließend in ihr befindlichen Punkte in entsprechender Reihenfolge durch Aufrufen der Funktion `setzeZwischenpunkt()` in der Datei `openlayers_general_functions.php` in der Karte zu zeichnen. Als Beispiel soll hier das Einfügen eines Zwischenpunktes an beliebiger Stelle dienen:
Der Nutzer wählt, nachdem er durch Anklicken des gelben Pfeils das Einfügen eines Zwischenpunktes aktiviert hat, mithilfe der Dropdown-Liste auf der Oberfläche die Position, hinter welcher der neue Zwischenpunkt eingefügt werden soll. Er klickt anschließend in die Karte, um den Ort des Punktes zu definieren. Intern wird die Methode `insertViaOnIndex()` aufgerufen, welche als Paramater die x- und y-Koordinate des Zwischenpunktes sowie seine Nummer, d. h. seinen Rang in der Liste, übergeben bekommt.

```
function insertViaOnIndex(x, y, index) {
  removeAllViasInMap();
  point = new OpenLayers.Geometry.Point(x, y);
  zwischenpunktliste.splice(index, 0, point);
  redrawAllViasInMap();
  addVia2DropDown();
}
```

In der Methode `insertViaOnIndex()` werden nun zunächst durch Aufruf der Methode

```
function removeAllViasInMap() {
  for ( var i = 0; i <= zwischenpunktliste.length - 1; i++) {
    loeschen(i + 1);
  }
}
```

alle Zwischenpunkte gelöscht, um eine komplette Neuzeichnung der Punkte vorzubereiten. Diese Herangehensweise mag inperformant wirken, hat sich jedoch als sicher erwiesen, da es mit anderen Methoden während der Testphase in einigen Fällen zu Doppelzeichnungen oder Darstellungsfehlern nach dem Neuzeichnen der Zwischenpunkte gekommen ist. Als nächstes wird ein neuer `Point` erstellt, der einer Repräsentation des später in der Karte hinzugefügten Zwischenpunktes entspricht. Die Punktrepräsentation wird nun an der gewünschten Stelle in das Zwischenpunktarray eingefügt und anschließend werden alle Punkte der Zwischenpunkt-Datenstruktur durch Aufruf von

```
function redrawAllViasInMap() {
  for ( var i = 0; i <= zwischenpunktliste.length - 1; i++) {
    setzeZwischenpunkt(zwischenpunktliste[i].x, zwischenpunktliste[i].
        y,
         i + 1);
  }
}
```

wieder der Karte hinzugefügt. Die zuletzt aufgerufene Methode

```
function addVia2DropDown() {
  element = new Option(zwischenpunktliste.length, zwischenpunktliste.
      length, false, true);
  document.zwischenpunkte.viadropdown.options[document.zwischenpunkte.
      viadropdown.options.length] = element;
}
```

sorgt dafür, dass die Benutzeroberfläche an die neue Anzahl der Zwischenpunkte angepasst wird und somit in diesem Fall in der Dropdown-Auswahlliste ein Platz mehr zur Verfügung steht. Das letzte `true` im Konstruktor des neuen Listeneintrags steht hierbei dafür, dass diese letzte Stelle selektiert wird, sodass der nächste Zwischenpunkt standardmäßig immer als letzter Punkt in die Liste eingefügt wird. Das Löschen eines Zwischenpunktes erfolgt nach

analogem Verfahren.
Ein weiteres Feature der Datenstruktur, die das JavaScript-Array mit sich bringt, ist, dass diese einfach invertiert werden kann. Das macht sich der Radroutenplaner zu Nutze, um die Zwischenpunktreihenfolge anzupassen, wenn der Benutzer die gesamte Route umkehrt. Da es sich bei den Punkten in der Liste, wie bereits erwähnt, nur um Repräsentationen der tatsächlich eingezeichneten Punkte handelt, ist es auch hier notwendig, die tatsächlichen Zwischenpunkte vorher zu löschen und anschließend neu zu zeichnen:

```
function onReverse() {
  removeAllViasInMap();
  zwischenpunktliste.reverse();
  redrawAllViasInMap();
}
```

5.2.2 Serverseitige Implementierung

5.2.2.1 Feste Reihenfolge

Zur Realisierung der festen Reihenfolge der Zwischenpunkt bildet man eine Liste der Punkte, deren erstes Element der Startpunkt und deren letztes Element der Endpunkt der Route ist. Die Zwischenpunkte werden dazwischen eingefügt. Jetzt liegt bereits die korrekte Reihenfolge der Punkte vor, weil die Zwischenpunkte in der Reihenfolge, in der sie gesetzt wurden, bereits an die Datei `routing.php` übergeben wurden. Die Routenberechnung wird nun zwischen jeweils zwei aufeinanderfolgenden Punkten der Liste ausgeführt, wobei beim Startpunkt und dem ersten Zwischenpunkt begonnen wird, bis alle Punkte abgearbeitet sind.

5.2.2.2 Optimale Reihenfolge mit der TSP-Funktion von pgRouting

Die optimale Reihenfolge der Zwischenpunkte wird mit Hilfe des Traveling Salesman Problem (TSP)-Algorithmus umgesetzt. Die Implementierung dieses Algorithmus in pgRouting „basiert darauf, die Punkte nach ihrer euklidischen Distanz zu sortieren“ [Pose]. Die Entwickler entschieden sich aufgrund einer wissenschaftlichen Arbeit dafür, welche belegte, dass die Qualität der Ergebnisse mit euklidschen Distanzen nur wenig schlechter als die in Graphen sind. Die Implementierung bildet den Spagat zwischen Performance der Berechnung und Qualität der Ergebnisse [Pose]. Die Signatur der Funktion sieht folgendermaßen aus:

```
CREATE OR REPLACE FUNCTION tsp(sql text, ids varchar, source_id
    integer)
  RETURNS SETOF path_result
```

Die Funktion erwartet die Parameter

- sql: Eine SQL-Query, deren Ergebnis eine Tabelle von Knoten zurückgibt. Sie enthält drei Spalten: Die Identifikationsnummer `id` sowie x- und y-Koordinate.

- ids: Die `ids` der Punkte, die sortiert werden sollen.
- source_id: Die `id` des Startpunktes, bei welchem die Route beginnen soll.

5.2.2.3 Umsetzung in der Datei routing.php

Die Punkte sollen mit der TSP-Funktion von pgRouting sortiert werden. Liegt nur ein Zwischenpunkt vor, muss keine Sortierung vorgenommen werden. Mit Start-, Zwischen- und Endpunkt liegt bereits die korrekte Sortierung vor. Eine Eigenheit der in pgRouting implementierten Funktion ist die Tatsache, dass die Knoten lediglich über ihren Abstand zum Startknoten sortiert werden. Der vom User gesetzte Endpunkt kann also zunächst außen vor gelassen werden, weil er ohnehin zuletzt angefahren werden soll und nicht mit sortiert werden muss.
Dabei treten jedoch zwei Probleme auf:

1. Der implementierte Algorithmus kann nur die Knoten des Routinggraphen sortieren. Die vom User gesetzten Punkte können jedoch nicht einfach zu diesem Zweck in die Datenbank eingefügt werden, weil der Datenbestand möglichst nicht geändert werden soll.

2. Ein Bug verhindert, dass bei drei Knoten (Startknoten und zwei verschiedene, zu sortierende Knoten) korrekt sortiert wird. Die Rückgabe der Funktion ist fehlerhaft, welches ein bekannter, aber bisher noch nicht behobener Fehler im Quellcode ist.

Das erste Problem wird dadurch umgangen, dass jedem Punkt sein nächstgelegener Knoten zugewiesen wird. Dies führt dazu, dass der Algorithmus beim Sortieren mit falschen Abständen rechnet, weil der Abstand zwischen Punkten und Knoten vernachlässigt wird. Jedoch sind die Unterschiede zum optimalen Ergebnis für den User kaum nachzuvollziehen, weswegen mit den Ungenauigkeiten vorlieb genommen wird.
Das zweite Problem löst sich, indem man für den Fall, dass zwei Zwischenpunkte vorliegen, mit Hilfe der PostGIS-Funktion `distance()` den Abstand beider Zwischenpunkte zum Startpunkt misst und diese „von Hand“ sortiert, ohne die TSP-Funktion zu benutzen.

Die Zwischenpunkte werden also ihren nächstgelegenen Knoten zugeordnet und in einer Liste gespeichert. Daraufhin reduziert man die Liste, so dass jeder Knoten nur einmal vorkommt, weil es passieren kann, dass mehrere dicht bei beieinander liegende Punkte zu einem Knoten werden. Jedoch muss man zur Behandlung des vorher genannten Fehlers (Problem 2) wissen, wie viele *verschiedene* Knoten genau vorliegen. Gibt es vier oder mehr verschiedene Knoten (ohne Endpunkt!), kann die TSP-Funktion ausgeführt werden. Sie gibt eine Tabelle zurück, welche die Knoten in absteigender Sortierung zurückgibt (Beispieltabelle über Link abrufbar [Pose]).
Aus der Reihenfolge der Knoten in der Rückgabe kann anschließend auf die Reihenfolge

der ursprünglichen Punkte, die den Knoten zugewiesen wurden, geschlossen werden. Wenn mehrere Punkte demselben Knoten zugewiesen wurden, landen sie in der Ordnung direkt hintereinander. Für diesen Fall wird keine weitere Sortierung vorgenommen. Die Punkte landen einfach der Reihenfolge nach in einer Liste der sortierten Punkte, so wie sie ursprünglich gesetzt wurden, d. h. in der ursprünglichen unsortierten Liste vorlagen. Die Auswirkungen auf die Länge der Gesamtroute sind hier i.d.R. sehr marginal, weil die Punkte ohnehin sehr dicht beieinander liegen, wesweswegen sie auch dem gleichen Knoten zugeordnet wurden. Das Ergebnis der Sortierung stellt eine Liste von Punkten dar, welche, in der Reihenfolge von Anfang bis Ende durchlaufen (bei allen in Kauf genommenen Ungenauigkeiten), die kürzeste Route ergeben sollten.
Abschließend wird zwischen jeweils zwei Punkten der sortierten Liste, angefangen bei den ersten beiden Elementen, die Route, wie in Kapitel 5.1 beschrieben, berechnet.

5.3 Profile

In diesem Kapitel wird die Realisierung der unterschiedlichen Profile vorgestellt. Es gibt eine Auswahl voreingestellter Profile, welche die Route nach unterschiedlichen Kriterien berechnen. So wird eine individuelle Routenplanung ermöglicht. Berechnung nach unterschiedlichen Kriterien soll bedeuten, dass unterschiedliche Eigenschaften bevorzugt werden, welche in Form von Tags in den OpenStreetMap-Daten der Straße zugeordnet sind. Die Tags aller Straßen liegen in einer über das Tool Osmosis [Opeh] eingelesenen Datenbank in der Tabelle `way_tags`. Den OSM-IDs ist hier jeweils ein `key` (Schlüssel) seinem entsprechenden `value` (Wert) zugeordnet. Eine Straße kann beliebig viele Tags, also `key`/`value`-Paare, besitzen.

5.3.1 Profile bei der Routenberechnung

Eine nähere Betrachtung der Signatur des Dijkstra-Algorithmus pgRouting's [Posf]:

```
1 CREATE OR REPLACE FUNCTION shortest_path(sql text, source_id integer,
2     target_id integer, directed boolean, has_reverse_cost boolean)
3 RETURNS SETOF path_result
```

Für `sql` wird ein String erwartet, der folgende Tabelle zurückliefert:

```
1 SELECT id, source, target, cost FROM edge_table
```

Die genaue Struktur der Tabelle:

- id: `gid` der Kante
- source: `id` des source-Knotens
- target: `id` des target-Knotens
- cost: Kosten für das Überqueren einer Kante in Orientierung

- reverse_cost: Kosten für das Überqueren einer Kante entgegen seiner Orientierung

Wie bereits beim Routing erwähnt (vgl. Kap. 5.1), bezieht der Dijkstra-Algorithmus von pgRouting bei der Routenberechnung zwei Tabellenspalten für die Kosten einer Kante ein: für das Überqueren der Kante in Orientierungsrichtung (im Anwendungsfall `to_cost`) und entgegen dieser (`reverse_cost`). Um die Profile zu verwirklichen, bietet es sich daher an, je nach Profil unterschiedliche Tabellenspalten für das Überqueren der jeweiligen Kante (= Straße) an die `shortest_path`-Funktion zu übergeben. Ein Ausschnitt aus der `routing.php`,

```
1 ...
2 SELECT * FROM shortest_path('SELECT gid AS id, source::int4, target::
     int4, to_cost_".PROFILE."::double precision AS cost, reverse_cost_
     ".PROFILE."::double precision AS reverse_cost FROM ways',
     $startVertex, $endVertex, true, true))
3 ...
```

welcher zeigt, dass für die Routenberechnung die globale Variable `PROFILE` an die Tabellennamen herangehängt wird. Sie wurde beim Aufruf der `routing.php` als Request übergeben und vervollständigt den Namen der Tabellen um das verwendete Profil. Die Spalten für das Profil `offroad` tragen beispielsweise die Namen `to_cost_offroad` und `reverse_cost_offroad`, wobei sich `offroad` hier durch den Namen eines beliebigen Profils ersetzen lässt. In den verschiedenen Profilen werden die Kanten sowohl gewichtet, je nachdem, welche Tags die dazugehörige Straße trägt und wie diese entsprechend der Präferenzen des Benutzers zu gewichten sind, als auch dann ins Verhältnis zur Länge der Straße gesetzt.

5.3.2 Konzeption der Profile

Jedem Profil ist eine zusätzliche Tabelle mit unterschiedlichen Gewichtungen für die relevanten Tags (vgl. Kap. 3.1) zugeordnet. Die Tabellen enthalten eine Auflistung der 37 möglichen `key`/`value`-Paare der für Radfahrer interessanten Eigenschaften, sowie zwei jeweils zugehörige Faktoren für die Gewichtung der Tags in der Spalte `factor_to_cost` und `factor_reverse_cost`. Zum Beispiel sind im Profil `offroad` die Faktoren für die Oberflächenbeschaffenheit (`surface`) von unbefestigten Straßen (bspw. `surface=unpaved`) im Vergleich zu denen befestigter Straßen (bspw. `surface=asphalt`) deutlich niedriger (s. Abb. 6). Zum Beispiel sind im Profil `offroad` die Faktoren für die Oberflächenbeschaffenheit (`surface`) von unbefestigten Straßen (bspw. `surface=unpaved`) im Vergleich zu denen befestigter Straßen (bspw. `surface=asphalt`) deutlich niedriger. Dadurch findet eine Bevorzugung der Straße statt, was sich in der späteren Routenberechnung niederschlägt. Dadurch findet eine Bevorzugung der Straße statt, was sich in der späteren Routenberechnung niederschlägt. Das Konzept der festen Profile (im Gegensatz zu den Profilen des Profileditors) sieht folgende grundlegende Richtlinien bei der Festlegung der Faktoren vor:

- Eine neutrale Behandlung eines Tags findet mit dem Faktor 1 statt.

key	value	factor_to_cost	factor_reverse_cost
surface	paving_stones	1	1
surface	gravel; dirt	0.25	0.25
surface	unpaved	0.25	0.25
highway	residential	0.75	0.75
bicycle	opposite	1	0.0001
...	...	...	...

Tabelle 6: *Auszug aus der Tabelle über die Kosten zum Profil offroad*
Quelle: Eigene Darstellung

- Tags von Straßen, die nicht befahren werden dürfen oder sollen, erhalten den Faktor 10000. Die verhältnismäßig hohen Kosten sorgen dafür, dass sie vollständig umfahren werden.

- Ausgewiesene Fahrradstraßen (`highway=cycleway`) und Radfahrstreifen (`cycleway/cycleway:left/cycleway:right`) werden bevorzugt. Baulich getrennte Radwege (`track`) werden etwas stärker bevorzugt als Radfahrsteifen neben der Fahrbahn (`lane`), weil sie für Radfahrer einfach sicherer sind.

- Einbahnstraßen (`oneway=yes`) dürfen im Normalfall nur in Orientierungsrichtung von Radfahrern befahren werden, in Rückrichtung nicht.

- Wenn eine Einbahnstraße im Zusammenhang mit dem Tag `bicycle=opposite` auftaucht, darf sie befahren werden. Der Faktor 10000 wird durch 0.0001 neutralisiert. Allgemein wird ein Faktor x wird immer mit $\frac{1}{x}$ neutralisiert. Dies muss bei der Wahl der Faktoren beachtet werden.

- Ist die Straße für Fahrradfahrer ausgewiesen (`bicycle=yes/designated`), kann dies ebenfalls einige Beschränkungen neutralisieren (`highway=footway/pedestrian`).

5.3.3 Profile per writecosts.php automatisch schreiben

Das PHP-Skript `writecosts.php` berechnet die Profile automatisch aus den Tabellen für die Gewichtungen. Für die Erstellung eines Profils muss das Skript folgendermaßen aufgerufen werden: `writecosts.php?profil=PROFILNAME`
PROFILNAME muss der Teil des Namens der zugehörigen Tabelle `kosten_PROFILNAME` in der Datenbank `kostenprofile` sein. Das Skript legt, wenn nicht bereits vorhanden, die Spalten `to_cost_PROFILNAME` und `reverse_cost_PROFILNAME` in der osm2pgRouting-Datenbank in der Tabelle `ways` an. Optional lässt sich das Skript über die Kommandozeile per Secure Shell (SSH) auch per
`php writecosts.php ALL` aufrufen, so dass *alle* Profile in der Datenbank `kostenprofile`

automatisch ausgelesen und geschrieben werden. Dies kann bei einer großen Anzahl von Profilen nützlich sein.
Für jedes Profil wird jede einzelne Straße (bzw. jede Kante des Graphen) auf seine vorhandenen Tags überprüft, daraufhin die Faktoren zur Gewichtung aller (vorhandenen) Tags jeweils für `to_cost` und `reverse_cost` aufmultipliziert, dann der Faktor mit der Länge der Kante multipliziert und schließlich beide Werte in die Spalten geschrieben. Wenn keine Tags vorhanden sind, wird einfach die Länge übernommen. Dies entspricht einer neutralen Behandlung.

5.3.4 Auswirkungen auf die Routenberechnung

Beispielhaft soll eine Route zwischen den gleichen Punkten, aber mit unterschiedlichen Profilen geroutet werden. Der Vergleich zeigt einen deutlichen Unterschied: Im Profil `sportlich`

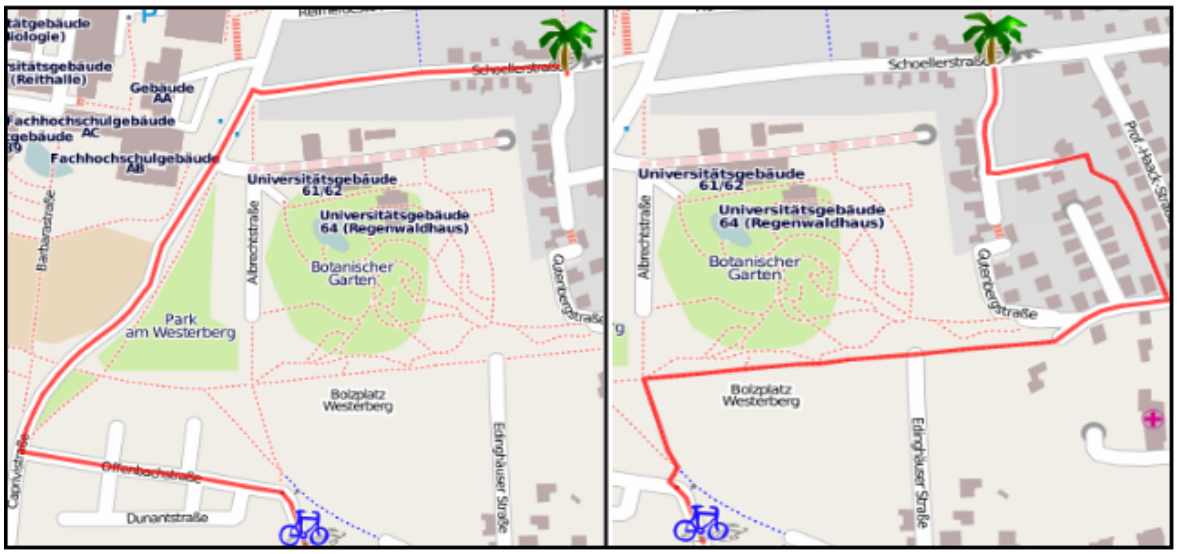

Abbildung 25: *Vergleich einer Routenberechnung der gleichen Strecke mit dem Profil sportlich (links) und offroad (rechts).*

Quelle: Eigene Darstellung

werden Straßen mit glattem Untergrund für Rennradfahrer bevorzugt, weswegen die Route in der Abbildung 25 über die asphaltierte Hauptverkehrsstraße führt. Im für Mountainbiker ausgelegten Profil `offroad` dagegen spielen Wege abseits befestigter Straßen eine größere Rolle, weswegen über die Fuß- und Radwege geleitet wird.

5.3.5 Profileditor

Im Vergleich zu den voreingestellten Profilen wird dem Nutzer durch den Einsatz des Profileditors eine noch individuellere Routenplanung ermöglicht. Der Profileditor erlaubt dem Nutzer auch einzelne Aspekte, wie den Einbezug von Fußwegen oder Treppen, zu bestimmen. Die folgenden Optionen stehen zur Auswahl (s. Abb. 26):

- Radwege bevorzugen oder nicht

- Straßenqualität nicht weiter beachten, „Offroad“ oder „befestigt“ bevorzugen
- Fußwege umfahren oder zulassen (in letzterem Fall muss das Fahrrad geschoben werden)
- Treppen umfahren oder zulassen (in letzterem Fall muss das Fahrrad getragen werden)
- Steigung ignorieren oder ab 3% bzw. 7% vermeiden
- Eingabe der Durchschnittsgeschwindigkeit

Ähnlich wie bei den voreingestellten Profilen ist jeder der 72 möglichen Kombinationen durch den Profileditor eine Tabelle in der Datenbank mit den entsprechenden Gewichtungen für die Tags zugeordnet, die insgesamt 37 mögliche `key`/`value`-Paare sowie die Spalten `to_cost` und `reverse_cost` enthält.
Auch die Richtlinien für die Festlegung der Faktoren sind bis auf einige Stellen identisch. Eine neutrale Behandlung wird durch den Faktor 1 erreicht. Durch den Faktor 10000 werden Straßen, die aufgrund einer bestimmten Eigenschaft umfahren werden sollen, ausgeschlossen. Entsprechend kann dieser Faktor durch den Wert 0.0001 neutralisiert werden. Unterschiede zeigen sich beispielsweise darin, dass bei den voreingestellten Profilen ein Routing über Fußwege, Fußgängerzonen oder Treppen grundsätzlich ausgeschlossen, im Profileditor nach expliziter Auswahl jedoch möglich ist. Des Weiteren werden Fahrradstraßen und -streifen nicht generell bevorzugt, sondern nur falls dies ausgewählt wird.

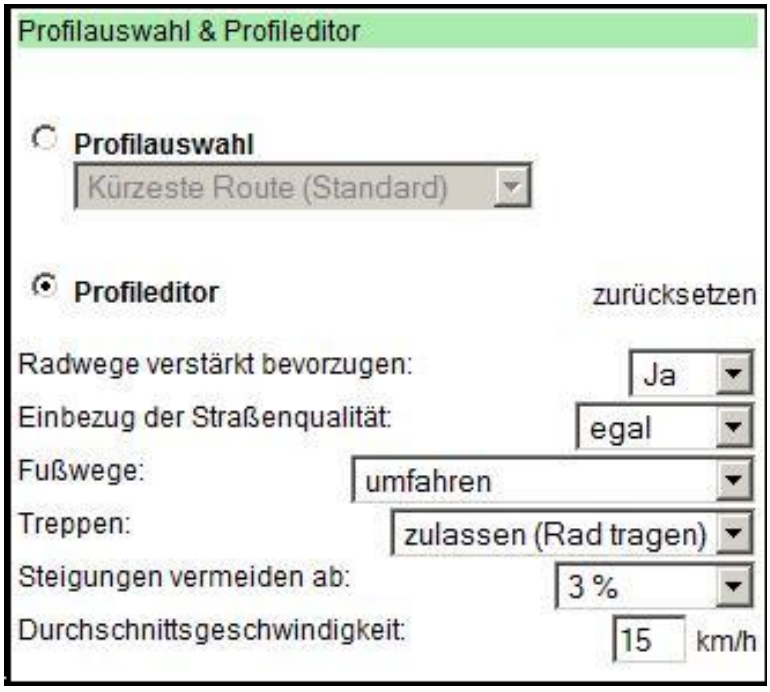

Abbildung 26: *Benutzeroberfläche des Profileditors*
Quelle: Eigene Darstellung

5.4 Steigungseinbezug

Ein weiteres wichtiges Feature stellt der Einbezug von Höhendaten dar. Dem User soll es möglich sein, Steigungen unterschiedlichen Grades nach Belieben zu vermeiden. Das Überwinden von Steigungen stellt je nach Steigungsgrad, Ausstattung des Fahrrades (Anzahl der Gänge), Gepäck oder aber auch (Gegen-)Wind einen gewissen Kraftaufwand für den Radfahrer dar. Im Gegensatz zu einem Autofahrer, für den die Berechnung der schnellsten oder kürzesten Route das ausschlaggebende Kriterium ist, ist ein Radfahrer eher bereit einen Umweg in Kauf zu nehmen, um einen starken Höhenanstieg zu vermeiden. Daher ist die Berücksichtigung von Höhen und Höhenunterschieden ein wesentlicher Bestandteil des Radroutenplaners.
Der Allgemeine Deutschen Fahrrad-Clubs e.V. (ADFC) schlägt folgende Kategorien von Steigungen vor [All]:

- bis 3 %: nicht beachtenswerte Steigung
- 3 bis 7 % bei mindestens 30 Meter Höhendifferenz: beachtenswerte Steigung
- über 7 % bei mindestens 30 Meter Höhendifferenz: starke Steigung

Ein Höhenunterschied von mindestens 30 Metern bedeutet im Umkehrschluss, dass bei 3% Steigung die Strecke, auf welcher der Höhenunterschied zu messen ist, mindestens 1 km lang sein muss. Die Längen der Kanten des durch osm2pgRouting erzeugten Graphen des aktuell eingelesenen Datensatzes (ca. zwei Drittel der Fläche des Landkreises Osnabrück) liegen jedoch im Durchschnitt deutlich darunter: Der SQL-Befehl `select avg(length) from ways` gibt einen Wert von umgerechnet etwa 172 m zurück. So erweist sich der Höhenunterschied von 30 m als nicht geeignetes Maß, im Rahmen des Projektes werden jedoch die von ADFC vorgeschlagenen Kategorien als Richtwerte beibehalten.
Um die Steigungen der Straßen zu ermitteln, werden im Rahmen dieses Projektes die freien SRTM Höhendaten verwendet 2.10.

5.4.1 Filterung der SRTM-Höhendaten

Da die SRTM-Daten kein digitales Geländemodell, sondern ein digitales Oberflächenmodell sind, stellt sich die Frage, ob man aus diesen Daten Höhenabweichungen vom Gelände, die durch Gebäude oder Bäume entstehen, filtern kann. Eine Möglichkeit besteht darin, einen Layer mit Gebäudepolygonen aus den OpenStreetMap-Daten als Maske zu verwenden. Falls der Mittelpunkt einer Höhenrasterzelle in einem Gebäudepolygon liegt, wird der Höhenwert dieser Zelle aus dem Mittelwert der umliegenden acht Rasterzellen bestimmt.

Wie man in der Abbildung 27 sieht, sind die Rasterzellen mit 3 Bogensekunden im Vergleich zu Gebäuden (hellbraun) sehr groß. Außerdem fällt auf, dass gefilterte Zellen (rot) nicht immer einige Meter höher sind als umliegende nicht gefilterte Zellen. Dies lässt sich zum einen

Abbildung 27: *Überlagerung von OpenStreetMap Gebäude- und Straßenlayern mit gefilterten SRTM-Höhendaten*

Quelle: Eigene Darstellung

mit der Extrapolierung der Höhendaten aus Daten, die eine Auflösung von 1 Bogensekunde besitzen, erklären. Der Höhenwert in der 3-Bogensekunden-Rasterzelle entsteht durch Mittelwertbildung von neun Punkten. Außerdem fließen systembedingte Ungenauigkeiten des Radars ein. Zum anderen ist die Gebäudeerfassung in OpenStreetMap noch lückenhaft, sodass sich durch Verwendung eines solchen Filters keine Verbesserung der Höhendaten ergibt. Die SRTM-Höhendaten liefern also eher großräumige Höheninformationen. Kürzere Steigungsstrecken und kleinere Höhenunterschiede können durchaus von der Realität abweichen. Bei den für Radtouren üblichen Streckenlängen von mehreren Kilometern liefern die Daten trotzdem wertvolle und hinreichend genaue Entscheidungshilfen für die Wahl der gewünschten Strecke.

5.4.2 Einlesen und Umwandeln der SRTM-Höhendaten

Der Landkreis Osnabrück wird durch die beiden unter `http://www2.jpl.nasa.gov/srtm/` herunterladbaren Dateien N52E007.hgt und N52E008.hgt vollständig abgedeckt. Die SRTM-Höhendaten liegen in einem Binärformat vor, lassen sich aber mithilfe der GDAL Tools

in ein ASCII-Textformat umwandeln. Die GDAL Tools sind eine Open-Source-Bibliothek mit Kommandozeilenwerkzeugen zur Übersetzung und Weiterverarbeitung von räumlichen Rasterdaten unterschiedlichster Formate [Opec]. Der folgende Kommandozeilenbefehl bewirkt eine Umwandlung in ein ASCII-Textfile:

```
gdal_translate -of AAIGrid  N52E007.hgt  N52E007.txt
```

Im nächsten Schritt soll die Textdatei in eine SQL-Datei umgewandelt werden, um die Höhendaten in die Datenbank einzulesen. Dies geschieht mit dem Tool SrtmImport, das im Rahmen dieses Projektes selbst in Java geschrieben wurde. Es macht sich zunutze, dass die zuvor mit GDAL translate erzeugte Textdatei einen Header mit wichtigen Informationen enthält und ansonsten ein regelmäßiges Raster von Zeilen und Spalten der Höhen der Höhenpunkte aufweist.

```
ncols         1201
nrows         1201
xllcorner     6.999583333333
yllcorner     51.999583333333
cellsize      0.000833333333
NODATA_value  -32768
```

Der Header enthält, wie dargestellt, die geographischen Koordinaten (lat/lon) des Höhenpunktes der linken unteren Ecke sowie den Abstand der Höhenpunkte zueinander. Weil das Raster,

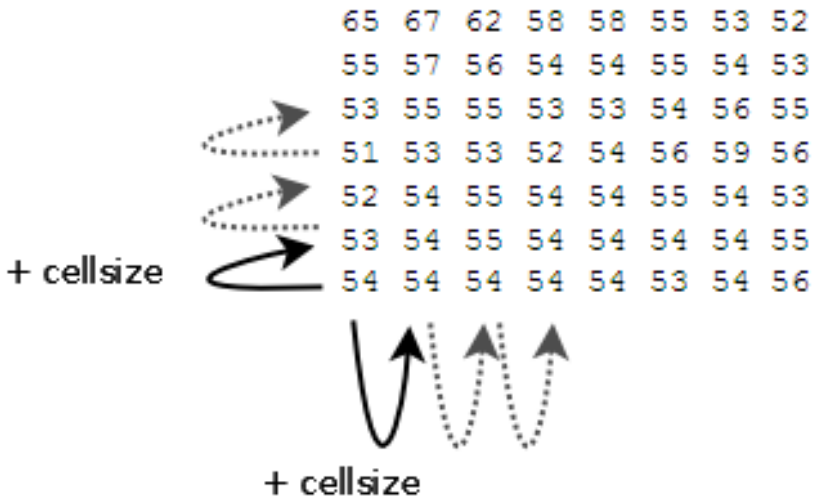

Abbildung 28: *Arbeitsweise von SrtmImport anhand eines Auszuges des umzuwandelnden Textfiles*
Quelle: Eigene Darstellung

wie in Abbildung 28 dargestellt, regelmäßig ist, lässt sich damit leicht auf die Koordinaten aller Höhenpunkte schließen: Der nächste Punkt in der gleichen Zeile hat die gleiche y-Koordinate, seine x-Koordinate hat sich um `cellsize` erhöht. Analog gilt dies auch für den nächsten Punkt in der gleichen Spalte. Die Anzahl der Zeilen und Spalten ist ebenfalls im Header enthalten. `NODATA_value` beschreibt den Wert, den solche Höhenpunkte erhal-

ten, zu denen keine Information über die Höhe vorliegen. Hierbei handelt es sich um Fehler bei der Datenaufnahme. SrtmImport errechnet zu jedem einzelnen Höhenpunkt die entsprechenden Koordinaten (lat/lon) und erzeugt daraus einen SQL-String, der den Punkt in eine Datenbank einfügt. Es wird eine Tabelle mit drei Spalten erzeugt: `lat`, `lon` und `hoehe`.

5.4.3 Aufbereitung der Höhenpunkte für die Verwendung

Um die Höhenpunkte weiterverwenden zu können, muss man aus den rohen Koordinaten noch Geometry-Objekte erzeugen. Hier hilft PostGIS:

```
1 SELECT AddGeometryColumn('hoehen','the_geom','4326','POINT',2);
2 UPDATE hoehen SET the_geom=ST_SetSRID(ST_MakePoint(lon, lat),4326);
3 ALTER TABLE hoehen RENAME COLUMN the_geom TO the_geom_4326;
4 ALTER TABLE hoehen ADD COLUMN the_geom geometry;
5 UPDATE hoehen SET the_geom=transform(the_geom_4326, 900913);
6 UPDATE geometry_columns SET srid = 900913;
7 CREATE INDEX hoehenindex ON hoehen USING GIST (the_geom);
```

Zunächst wird eine Spalte für die Geometrie in der Tabelle angelegt und anschließend aus den rohen Koordinaten ein Point-WKT erstellt, dem das Koordinatensystem WGS84 mit dem EPSG-Code 4326 zugewiesen wird. Danach nennt man die Spalte `the_geom` in `the_geom_4326` um, legt eine neue Spalte `the_geom` an und transformiert die Daten von WGS84 in die Google-Projektion mit dem EPSG-Code 900913, um mit den Routingdaten kompatibel zu sein. Abschließend muss noch die Information über das Koordinatensystem in `geometry_columns` korrigiert werden. Zudem bietet es sich an einen Index zu erstellen, damit die Berechnungen auf der Tabelle schneller laufen. Bei der enormen Anzahl von aktuell fast 500.000 Höhenpunkten erhöht dies die Geschwindigkeit der Berechnungen drastisch.

5.4.4 Berechnung der Steigungen der Straßen

Das PHP-Skript `steigungen.php` berechnet zu jeder Kante des Graphen aus der `ways`-Tabelle seine Steigung. Dazu nimmt man sich, wie in Abbildung 29 dargestellt, den Start- und Endpunkt einer jeden Kante und ermittelt zu beiden die nächstgelegenen vier Höhenpunkte sowie deren Distanz zum jeweiligen Referenzpunkt. Zusätzlich dargestellt ist das regelmäßige Raster, an dessen Eckpunkten sich die Höhenpunkte befinden. Anschließend wird der gewichtete Mittelwert aus den vier Punkten jeweils für Start- und Endpunkt interpoliert, sodass nun Höhenwerte vorliegen. Mithilfe der Länge der Straße lässt sich die Steigung nun mit folgender Formel ermitteln: **(Höhe Endpunkt − Höhe Startpunkt) / Länge der Straße = Steigung**. Die Steigung wird anschließend in die eigens dafür angelegte Spalte geschrieben. Abschließend bleibt zu sagen, dass nach einigen Tests die Interpolation der Punkte mit dem vorgestellten Verfahren deutlich bessere Ergebnisse hervorbringt als die Verwendung der Höhe des nächsten Nachbarn (Nearest Neighbour). Diskussionswürdig ist die Tatsache, dass zu jeder Kante eine eigene Steigung ermittelt wird. Wie auf Seite 82 erwähnt, liegt die Durch-

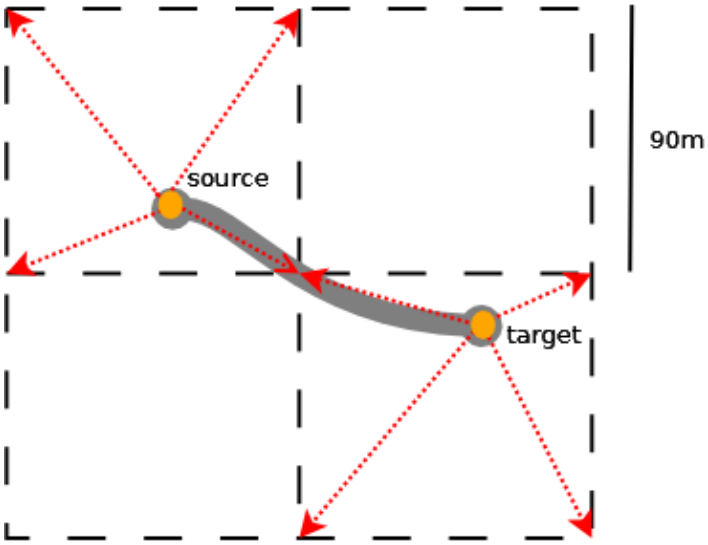

Abbildung 29: *Interpolation der Höhen der Endpunkt einer Straße*
Quelle: Eigene Darstellung

schnittslänge der Kanten bei etwa 172 m, was natürlich bei einer Auflösung von 90 m Zweifel an den Ergebnissen lässt. Nichtsdestotrotz liefert dieses Verfahren in der Praxis sehr gute Ergebnisse. Ebenfalls positiv an der Tatsache, dass für jede einzelne Kante eine Steigung definiert wird, ist, dass sich so Steigungen sehr effektiv auch auf kurzen Strecken vermeiden lassen.

5.4.5 Steigungen in den Profilen

In Kombination mit der Funktion der Profile (vgl. Kap. 5.3) können Steigungen nun bei der Routenberechnung berücksichtigt werden, indem sie entsprechend gewichtet werden. Zudem weist die Steigung eine Orientierung auf: Gefälle tritt als Steigung mit negativem Vorzeichen auf. Das bedeutet, es kann sogar beim Schreiben der Kosten durch Einbezug der Orientierung der Steigung berücksichtigt werden, wann den Radfahrer beim Überqueren der Straße eine Steigung erwartet und wann ein Gefälle. Realisiert wird die Vermeidung von Steigung dadurch, dass überprüft wird, ob die Steigung

- ein positives Vorzeichen hat: Es handelt sich um Steigung in Orientierungsrichtung bzw. Gefälle entgegen der Orientierungsrichtung, also Kosten in Spalte `to_cost` hochsetzen, oder

- ein negatives Vorzeichen hat: Es handelt sich um Gefälle in Orientierungsrichtung bzw. Steigung entgegen der Orientierungsrichtung, also Kosten in Spalte `reverse_cost` hochsetzen.

5.5 Höhenprofil

Unter einem Höhenprofil versteht man ein Diagramm, welches einen senkrechten Schnitt durch das Gelände und somit die Höhe entlang des Weges darstellt [Wikb]. Das Höhenprofil ist u. a. durch die Berichterstattung der Tour de France bekannt geworden und hat seither auch bei Radroutenplanern Einzug gehalten. Da in diesem Projekt bereits Höhendaten vorliegen, soll es auch möglich sein, das Höhenprofil zu einer berechneten Route anzuzeigen.

5.5.1 Kantengeometrien sortieren

Um das Höhenprofil berechnen zu können, müssen die Kanten der Route in der Reihenfolge vorliegen, in der sie befahren werden. Die Route muss eine fortlaufende, in sich geschlossene Struktur aufweisen, weil das Höhenprofil die sich verändernde Höhe entlang der Route widerspiegelt. Glücklicherweise gibt der Dijkstra-Algorithmus die Kanten der Route zumindest in der korrekten Reihenfolge zurück. Des Weiteren ist es erforderlich, dass auch die Stützpunkte der Geometrien aufeinanderfolgender Kanten die korrekte Reihenfolge aufweisen. Das bedeutet: Eine Kante in der Route schließt an dem Punkt an, an welchem ihr Vorgänger endet. Das Höhenprofil stellt letztlich die Höhen bestimmter Punkte der fortlaufenden Route dar. Dies ist nicht immer gegeben, da die Geometrien der Kanten als WKT und WKB in der Datenbank fest abgespeichert werden und zur Darstellung lediglich abgerufen werden. Durch OpenLayers wird die gesamte Geometrie der Kanten, die Teil der Route sind, dargestellt, so dass lediglich der Eindruck einer fortlaufenden Route entsteht. Die als Linienzug dargestellte Route ist zwar zusammenhängend, doch sind die ihr zugrunde liegenden Stützpunkte nicht fortlaufend.
Um die Geometrien der Kanten in die richtige Reihenfolge zu bringen, wird in der Datei `routing.php` eine Kantensortierung durchgeführt. Die Kanten der Route werden unmittelbar nach der Berechnung zwischen jeweils zwei aufeinanderfolgenden Punkten der Routenberechnung (Startpunkt, Zwischenpunkte, Endpunkt) durchlaufen. Die Lösung des Problems basiert auf der Tatsache, dass zwei aufeinanderfolgende Kanten einen gemeinsamen Punkt aufweisen. Die Start- und Endpunkte von jeweils zwei aufeinanderfolgenden Kanten der gesamten Route werden ermittelt und daraufhin folgende vier Fälle unterschieden:

1. Wenn der *Endpunkt* der ersten Kante der *Startpunkt* der nächsten Kante ist, liegt die korrekte Reihenfolge vor.

2. Wenn der *Endpunkt* der ersten Kante der *Endpunkt* der nächsten Kante ist, muss die Geometrie der letzteren umgekehrt werden. Dies erledigt die PostGIS-Funktion `ST_reverse()`.

3. Wenn der *Startpunkt* der ersten Kante der *Startpunkt* der nächsten Kante ist, muss die Geometrie der ersteren umgekehrt werden.

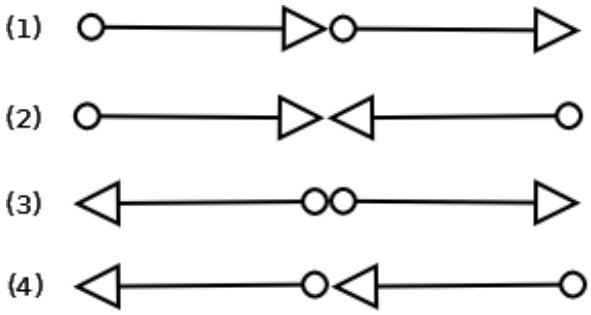

Abbildung 30: *Die vier Möglichkeiten der Reihenfolge der Stützpunkte von Kantengeometrien*
Quelle: Eigene Darstellung

4. Wenn der *Startpunkt* der ersten Kante der *Endpunkt* der nächsten Kante ist, müssen beide Geometrien umgekehrt werden.

Sobald alle Kanten der Reihenfolge nach dieser Prüfung unterzogen wurden, ist die Route für die Weiterverarbeitung bereit. Die Geometrien der Kanten und ihre Längen werden in JavaScript aus der XML-Rückgabe der Datei `routing.php` geparsed und direkt an die Datei `hoehenprofil.php` weitergegeben. Hier wird letzten Endes die Grafik erzeugt.

5.5.2 Arbeitsweise der Datei hoehenprofil.php

Die Datei `hoehenprofil.php` benutzt die PHP-Grafikbibliothek *PHPlot* zum dynamischen Generieren von Diagrammen. „PHPlot erlaubt Entwicklern Kuchen-, Linien-, Balken-, Punktdiagramm etc. zu erzeugen.“ [php] Dazu benutzt PHPlot die in der Programmiersprache C geschriebene GD-Bibliothek, welche dynamisch Grafiken erzeugen kann.
In der Datei `hoehenprofil.php` wird zunächst ein zweidimensionales Array für die x- und y-Werte des Diagramms deklariert, aus dem PHPlot später automatisch ein Diagramm erzeugt. Die x-Achse stellt die Länge der Route dar, während die y-Achse die Höhe zeigt. Ziel ist es, von bestimmten Punkten der Route die Höhe zu bestimmen und diese zusammen mit der Länge der Route beim Überqueren des Punktes in das Array zu setzen. Dazu bieten sich die Start- und Endpunkte der Kanten an: Die Länge Route beim Überqueren dieser Punkte kann einfach durch Aufaddieren der bereits bekannten Kantenlängen (per Request übergeben) ermittelt werden.
Es wird mit der Höhe des Startpunktes der Route begonnen und wie bereits beim Steigungseinbezug (vgl. Kap. 5.4) die Höhe aus den umliegenden vier Höhenpunkten ermittelt. Der zugehörige x-Wert des Startpunktes ist Null. Dann wird im weiteren Verlauf die Höhe jedes Endpunktes der nachfolgenden Kanten ermittelt (s. Abb. 31 auf Seite 89) und der zuvor addierte x-Wert, erhöht um die Länge der jeweiligen Kante, ins Array gespeichert, solange bis alle Kanten durchlaufen sind.

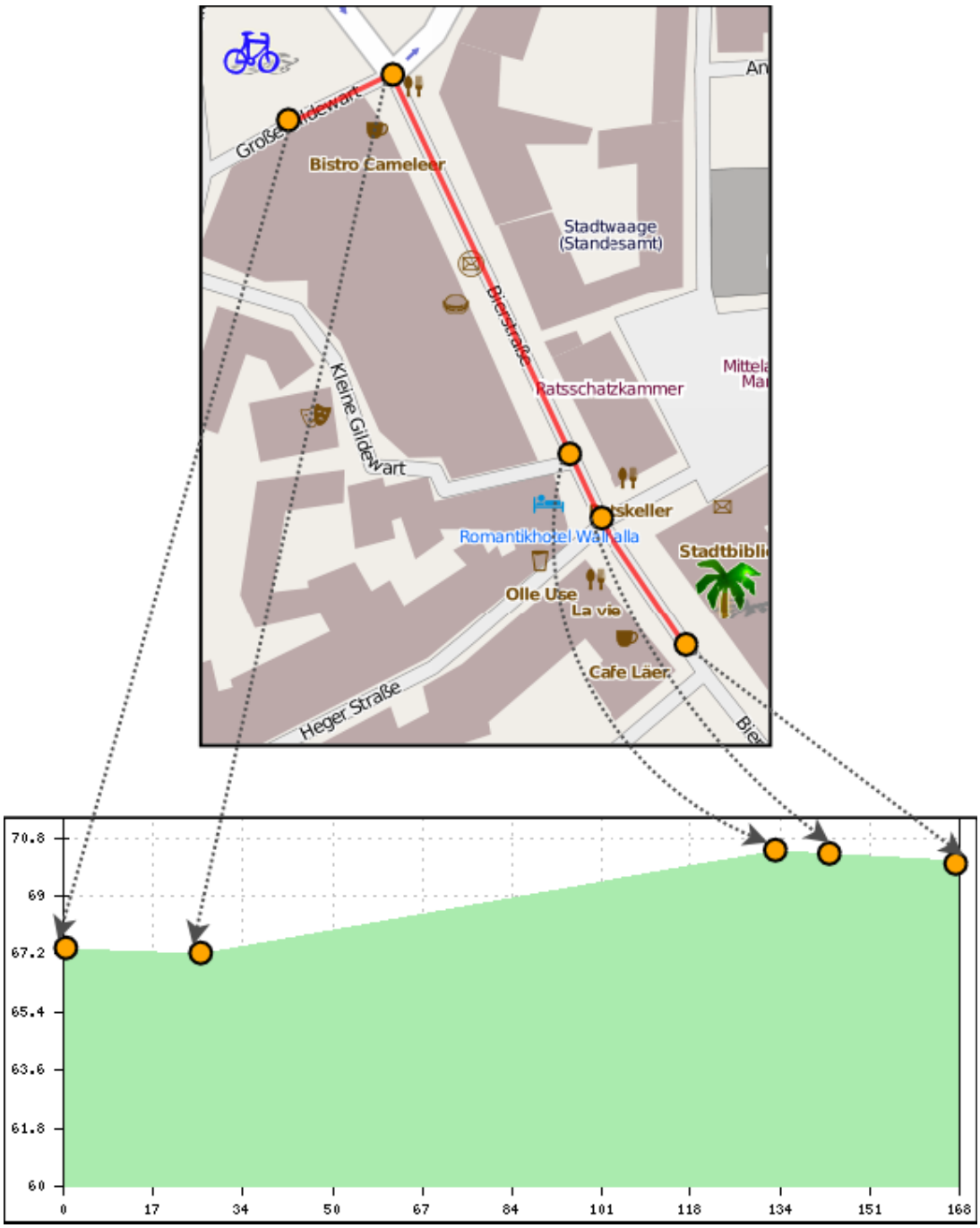

Abbildung 31: *Berechnung eines Höhenprofils aus der Höhe der Endpunkte der zugrunde liegenden Route*

Quelle: Eigene Darstellung

5.5.3 Performance

Leider ist die Sortierung der Stützpunkte der Kanten nicht sehr performant. Deshalb wird diese erst durchgeführt, sobald es nötig ist, was bei der normalen Routenberechnung nicht der Fall ist. Das wiederum bedeutet praktisch, dass die Routenberechnung ein zweites Mal aufgerufen wird, wenn der Nutzer das Höhenprofil zu seiner Route haben möchte.
Das Problem der Stützpunkte ließe sich auch in anderer Weise lösen: Beispielsweise spielt es keine Rolle, ob die Stützpunkte der Kante in der korrekten Reihenfolge vorliegen, wenn der genaue Mittelpunkt der Linie bekannt ist. Der Mittelpunkt liegt an derselben Stelle, wenn man die Kante invertiert. Dies würde allerdings eine Annäherung der Geometrie durch eine Gerade voraussetzen, was nicht immer der Fall ist. Deswegen wurde das vorgestellte Verfahren benutzt.
Eine weitere Schwachstelle ist die Tatsache, dass die Anzahl der für das Höhenprofil einbezogenen Punkte proportional zur Länge der Route ist (bei n Kanten sind es $n + 1$ Punkte). Dementsprechend länger ist also die Zeit bis zur Anzeige des Höhenprofils.

5.6 Verbale Routenbeschreibung

Nachdem ein Nutzer eine Route eingegeben und berechnet hat, möchte er in vielen Fällen auch eine genaue Anleitung bekommen, wie konkret an welchen Kreuzungen abzubiegen ist. In der Kartenansicht ist der straßengenaue Verlauf der Route nicht immer einfach erkennbar. Daher wird eine Auflistung der Abbiegevorgänge mit der Funktion `Routenbeschreibung` geliefert. Diese dient dazu, dem Nutzer eine detaillierte schriftliche Übersicht über alle Abiegevorgänge und den genauen Routenverlauf zu geben. Sie liefert Informationen darüber, in welcher Straße und in welche Richtung die Route zu Beginn verläuft. Außerdem gibt sie Auskunft über alle von der Route passierten Kreuzungspunkte. Hierzu gehören Details zur Entfernung bis zur nächsten Abbiegung und Straßenname. Darüber hinaus wird am Ende die Gesamtstrecke angezeigt und auch eine Prognose über die Fahrtdauer abgegeben.
Da der Algorithmus zur Routenbeschreibung sehr rechenaufwendig ist, wird die Routenbeschreibung nicht standardmäßig bei der Routenberechnung angezeigt. Sie muss im Menü „Routenbeschreibung“ gesondert angefordert werden. Konkret wird daraufhin die eingestellte Route erneut berechnet und im Anschluss die Routenbeschreibung erzeugt. Ausgegeben wird die Routenbeschreibung im Menüfeld.

Die Anzeige der Routenbeschreibung folgt folgendem Aufbau:

1. Angabe zum Start der Route inklusive Straßennamen und Himmelsrichtung
2. Abbiegeanweisung zu jedem Abbiegevorgang inklusive Entfernung vom letzten Abbiegen, Abbiegerichtung und Straßennamen
3. Angaben zur Zielstraße inklusive bis zum Ziel noch zu fahrender Strecke und dem

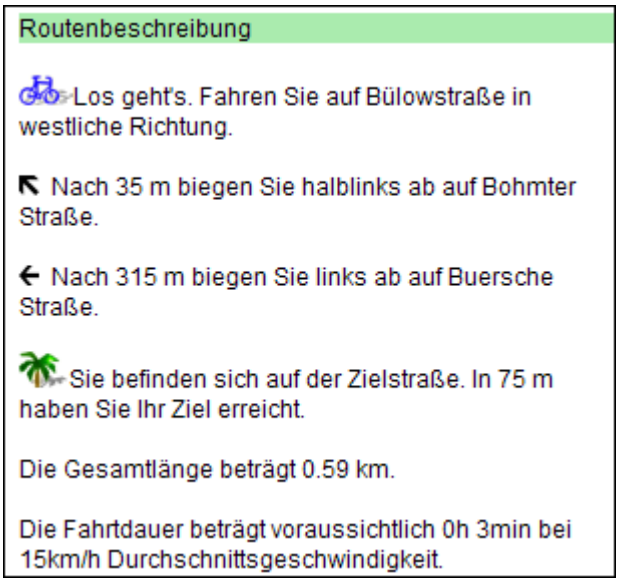

Abbildung 32: *Beispiel einer Routenbeschreibung*
Quelle: Eigene Darstellung

Straßennamen

4. Informationen zur Gesamtlänge der Strecke und zur voraussichtlichen Fahrzeit

Zu den einzelnen Abbiegeanweisungen werden jeweils auch Grafiken zur besseren Veranschaulichung geliefert. Start und Ziel werden durch die bekannten Grafiken von Fahrrad und Palme symbolisiert. Die Abbiegerichtungen werden durch Pfeile in die jeweilige Richtung verdeutlicht.
Im Folgenden wird der Algorithmus beschrieben, welcher der Routenbeschreibung zu Grunde liegt.
Zunächst wird geprüft, ob eine Route überhaupt berechnet werden kann. Andernfalls erscheint nur die Aufforderung `Bitte erst eine Route berechnen!`. Danach erscheint eine Aufforderung zum Warten und es wird die eigentliche Berechnung der Routenbeschreibung aufgerufen.

5.6.1 Berechnung der Fahrtdauer

Für die konkrete Ausgabe der Routenbeschreibung wird eine Auslagerungsdatei aufgerufen, welche die Ergebnisse anschließend wieder zurück gibt. Nachdem alle Routenhinweise ausgegeben wurden, wird die voraussichtliche Fahrtdauer und die Gesamtstrecke der Route wie folgt berechnet: Die für die Strecke benötigte Gesamtzeit wird zunächst mit der Formel

$$Dauer\_gesamt = \frac{\frac{Längeder Strecke}{Durchschnittsgeschwindigkeit} \times 3.6}{60}$$

in Minuten berechnet. Anschießend wird diese Gesamtzeit mit den Formeln

$$Dauer\_Stunden = abrunden\left(\frac{Dauer\_gesamt}{60}\right)$$

und $Dauer\_Minuten = Dauer\_gesamt \ \% \ 60$ in Stunden und Minuten umgerechnet.[4] Außerdem wird eine aktuelle Wetterinformation des Dienstleisters wetteronline.de unter der Routenbeschreibung eingefügt.

5.6.2 Kantenreihenfolge und Berechnung der Abbiegerichtung

Die Routenbeschreibung arbeitet folgendermaßen:
Nachdem die Route berechnet ist, werden die in der Route benötigten Kanten inklusive ihrer OSM-IDs aus einer Datenbank der Reihenfolge nach sortiert in eine Liste übernommen um später auf weitere Kanteneigenschaften wie Straßenname, -länge und -typ zugreifen zu können. In einigen Fällen gehört aber nicht die gesamte Länge einer Kante zu einem Routing. Dies ist unter anderem bei Start- und Zielpunkten der Fall, die nicht genau auf einem Ende sondern in der Mitte einer Kante liegen. Hier würde eine Abfrage der Kantenlänge zu einem falschen Ergebnis führen. Deshalb wird ein zweite Liste angelegt, das nur die genauen Längen der einzelnen Kanten einer Route aus der Datei, welche das Routing durchführt, in sortierter Reihenfolge enthält.
Die übergebene Reihenfolge der Kanten wird im Anschluss Stück für Stück vom Beginn bis zum Ende durchlaufen. Es wird dabei immer das Paar aus aktueller und nachfolgender Kante verglichen und die Anweisungen für die Routenbeschreibung erzeugt. Für beide werden jeweils das Azimut, der Straßenname und Länge bestimmt. Sofern der Straßenname leer ist, wird zusätzlich auch der Straßentyp abgefragt. Anhand dieser Informationen erfolgt später die Entscheidung, ob und in welche Richtung an einem Punkt eine Richtungsanweisung erfolgen soll.
Um zu überprüfen, ob eine Ausgabe einer Richtungsanweisung notwendig ist, muss jeweils die Anordnung zweier benachbarter Kanten näher untersucht werden, genauer gesagt der Schnittpunkt der beiden Kanten.

```
1 SELECT astext(pointn(w1.the_geom,2)) AS punkt
2 FROM ways w1
3 WHERE ".next." = w1.gid");
```

Eine Besondere Rolle hierbei spielt das Azimut der beiden Kanten. Zur Bestimmung des Azimuts einer Kante werden jeweils zwei Punkte benötigt. Ein Punkt davon wird durch den Schnittpunkt der benachbarten Kanten definiert. Da eine Kante aus mehreren Segmenten bestehen kann, die nicht zwangsläufig in die selbe Richtung verlaufen, muss zur Azimutberechnung der jeweils andere Stützpunkt des an den Kantenschnittpunkt angrenzenden Segmentes benutzt werden. Dadurch wird gewährleistet, dass jeweils das Azimut betimmt wird, welches auf die genaue Situation der Segemente am Kantenschnittpunkt zutrifft. Würde das Azimut jeweils mit den Kantenenden bestimmt, könnten Werte berechnet werden, die nicht

[4] Modulo-Operator: Ergibt den Rest einer Division

der Anordnung der Segmente am Kantenschnittpunkt entsprechen. In Abbildung 33 ist es also notwendig, nicht das Azimut von `Kante 1` und `Kante 2` zu berechnen sondern hierfür nur die Segmente `iv` von `Kante 1` und `i` von `Kante 2` zu nutzen, um die richtige Anordnung am Schnittpunkt zu berücksichtigen.

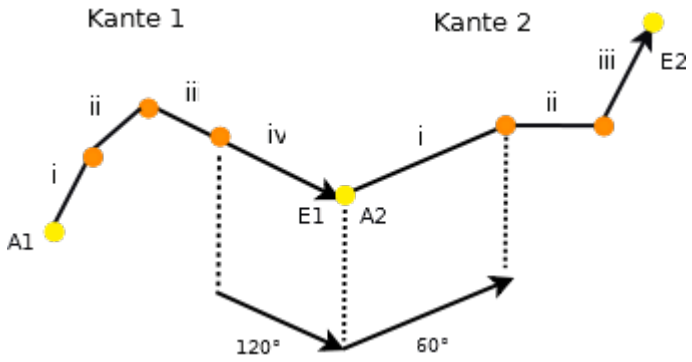

Abbildung 33: *Bestimmung des Azimuts zweier Kanten*
Quelle: Eigene Darstellung

Dieser wird mittels der PostGIS-Funktion
`ST_Azimuth(geometry pointA, geometry pointB)` ermittelt. Das Ergebnis dieser Funktion wird im Bogenmaß zurückgegeben und muss noch in Grad umgerechnet werden. Die Formel hierzu lautet: $\frac{Bogenmaß}{2\pi} * 360$.

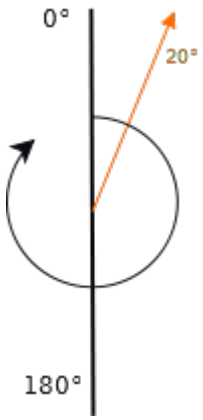

Abbildung 34: *Ermittlung des Azimuts*
Quelle: Eigene Darstellung

Weiter ist es wichtig, zu wissen, wie die beiden Kanten am Schnittpunkt gerichtet sind. Dazu wird überprüft, ob der Endpunkt der ersten Kante und der Anfangspunkt der nächsten Kante gleich dem Schnittpunkt beider Kanten sind. Für den Fall, dass dem nicht so ist, muss das Azimut der entsprechenden Kante um 180° erhöht werden um die Richtung des Azimuts umzudrehen. Dies ist unbedingt notwendig, denn andernfalls würde der Vergleich der Azimutwerte zu fehlerhaften Ergebnissen führen. Sollte dies dazu führen, dass das Azimuth

dann einen Wert größer als 360° annimmt, müssen noch 360° hiervon subtrahiert werden. Abbildung 33 zeigt zwei Kanten, welche bei einem Routing, das von links nach rechts läuft, in der richtigen Richtung verlaufen. In die Gegenrichtung, müssten die beiden Azimutwerte um 180° erhöht werden.

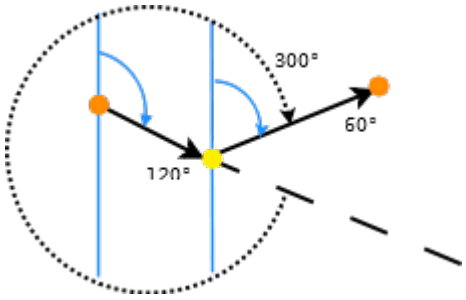

Abbildung 35: *Kantenschnittpunkt mit Azimutwerten der Kanten*
Quelle: Eigene Darstellung

Mit den beiden Azimutwerten kann nun am Kantenschnittpunkt die Richtung festgelegt werden, in die abgebogen werden muss. Hierzu wird der Azimut der zweiten Kante vom Azimut der ersten Kante subtrahiert. Dadurch ergibt sich der Abbiegewinkel am Kantenschnittpunkt. Abbildung 35 zeigt zwei Kanten mit den Azimutwerten 120° und 60°. Die Rechnung hierzu lautet also $60° - 120° = -60°$, da negative Abbiegewinkel allerdings nicht zulässig sind, müssen zu diesem Ergebnis noch 360° addiert werden um den Winkel wieder in positiver Form anzugeben. Das Endergebnis für den Abbiegewinkel des Beispiels lautet also 300°, was als `halblinks` ausgegeben wird.

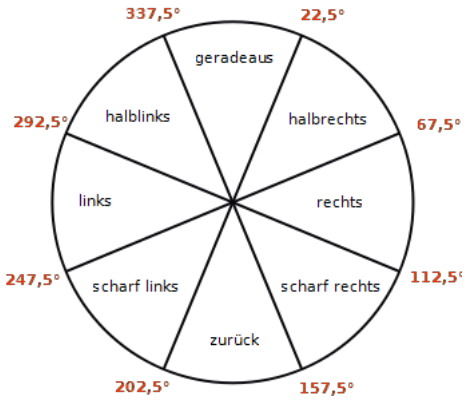

Abbildung 36: *Abbiegerichtungen*
Quelle: Eigene Darstellung

Anhand Abbildung 36 ist ersichtlich, wie die Abbiegewinkel entsprechend den Abbiegerichtungen *geradeaus*, *halbrechts*, *rechts*, *scharf rechts*, *zurück*, *scharf links*, *links* und *halblinks* zugeordnet werden. Da es nicht sinnvoll ist, in einem Routenplaner die Anweisung zum Wenden

oder zurückfahren zu geben, denn in dieser Situation läge ein Fehler in der Routenberechnung vor, außerdem würde die Aufforderung den Nutzer verwirren, wird für die Abbiegerichtung *zurück* eine Anweisung ausgegeben, die zum scharfen Rechts- bzw. Linksabbiegen auffordert. In einigen Fällen kommt es nämlich vor, dass Wege in so spitzem Winkel verbunden sind, dass die Abbiegerichtung *zurück* eigentlich passender wäre.

Für den Fall, dass an einem Schnittpunkt zweier Kanten die Abbiegerichtung nicht *geradeaus* lautet, erfolgt eine Ausgabe einer Abbiegeanweisung zu diesem Punkt. Eine Ausnahme stellt hierbei die Konstellation dar, bei welcher der Straßenname der Kante nach einer Abbiegung der selbe ist, wie derjenige der Kante vor der Abbiegung. In diesem Fall erfolgt nur dann eine Ausgabe, wenn an den Kreuzungspunkt noch weitere Straßen oder Wege anknüpfen (s. Abb. 37, Fall 1).
Sofern an einem Punkt aber geradeaus gefahren werden soll, muss überprüft werden, ob die vorherige und die folgende Kante den selben Straßennamen haben (s. Abb. 37, Fall 2). Ist dies nicht der Fall (s. Abb. 37, Fall 3), erfolgt ebensfalls die Ausgabe einer Abbiegeanweisung.

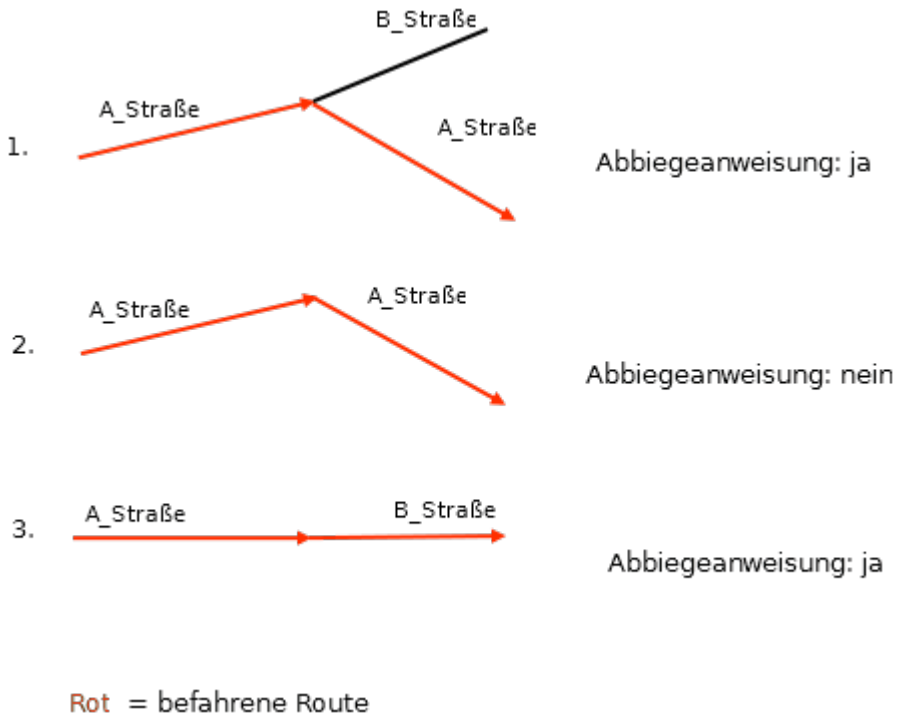

Abbildung 37: *verschiedene Abbiegeszenarien*
Quelle: Eigene Darstellung

5.6.3 Distanzberechnung

In der Abbiegeanweisung wird neben der Abbiegerichtung auch die Entfernung zur letzten Kreuzung, an der eine Abbiegeanweisung ausgegeben wurde bzw. dem Startpunkt, angegeben.

Das heißt, es müssen alle Kantenlängen addiert werden, die seit der letzten Abbiegeanweisung bzw. dem Startpunkt durchlaufen wurden. Dementsprechend werden alle Längen von Kanten, an denen keine Abbiegeanweisung erfolgte, addiert. Dieser Zähler wird jedes Mal nach der Ausgabe einer solchen Anweisung auf Null gesetzt (s. Abb. 38).

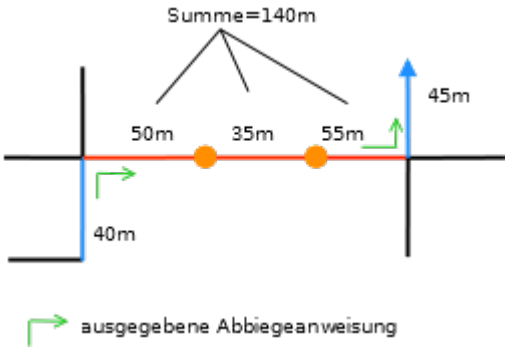

Abbildung 38: *Ausgabe kumulierter Kantenlängen*
Quelle: Eigene Darstellung

5.6.4 Straßenname und -typ

Schließlich enthält die Abbiegeanweisung außerdem den Namen der Straße oder des Weges, in die/den abgebogen werden soll. Dazu wird aus einer OSM-Daten-Datenbank der Straßenname ausgelesen. Teilweise enthalten die Kanteninformationen keine Straßennamen, weil beim Eintragen der Daten in die OSM-Datenbank diese Information nicht eingetragen wurde oder weil ein Weg einfach keinen Namen hat. Für diesen Fall existiert eine Funktion, die aus dem OSM-Tag *highway* den Straßentyp entnimmt und ins Deutsche übersetzt. Dieser Typ wird dann anstelle des fehlenden Straßennamens ausgegeben, um den Nutzer dennoch eine Orientierungshilfe zu geben.

5.6.5 Einbezug von Zwischenpunkten

Sofern sich in einer Route Zwischenpunkte befinden, ist eine gesonderte Behandlung von Nöten. Hier sind die beiden Fälle zu unterscheiden, dass nach einem Zwischenpunkt geradeaus weiter gefahren wird bzw. gewendet wird. Hierzu werden die letzte Kante vor und die erste Kante nach dem Zwischenpunkt aus der Kantenliste, in der sich lediglich die Längeninformaitonen befinden, geholt und mit dem PostGIS-Befehl
`st_equals(geometry lineA, geometry lineB)` verglichen. Sofern ihre Länge exakt gleich ist, handelt es sich bei dem Zwischenpunkt um einen Wendepunkt, der Nutzer wird aufgefordert, umzudrehen. Andernfalls ergeht eine Aufforderung, die Route in gleicher Fahrtrichtung fortzusetzen.

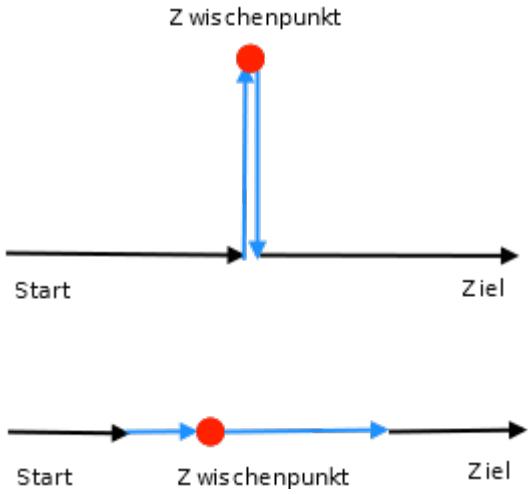

Abbildung 39: *verschiedene Zwischenpunktszenarien*
Quelle: Eigene Darstellung

5.6.6 Ausgabe der Routenbeschreibung

Nachdem alle Kanten abgearbeitet wurden, erfolgt die Ausgabe der Anweisungen als XML-Datei. Diese beinhaltet die Anweisung mit dem Straßennamen, eine Geometrie, auf die gezoomt werden kann, die Länge, damit die später zur Gesamtlänge aufaddiert werden kann sowie die Abbiegerichtung.

```
<part id='3'>
  <id>7818</id>
  <anweisung>
    Nach 40 m biegen Sie links ab auf Sedanstraße.
  </anweisung>
  <laenge>40</laenge>
  <wkt>POINT(893198.773509061 6852117.2326188)</wkt>
  <grafik>./images/links.png</grafik>
</part>
```

Da die Routenbeschreibung auch eine Funktion anbietet, die auf eine spezielle Kreuzung zentriert, wird auch ein Geometrie übergeben, die aus einem Koordinatenpaar besteht, auf welche durch OpenLayers-Funktionen gezoomt werden kann (Kapitel 5.9.2.3). Auf Clientebene wird mit Hilfe von HTML-Tags aus dem sich in der XML-Datei befindlichen Image-Link `<grafik>./images/links.png</grafik>` eine Grafik in die Beschreibung einbindet, welche dem Nutzer die Anweisung verdeutlicht.
Eine Funktion übernimmt anschließend noch die Geschwindigkeitsangabe aus dem Profile-

ditor (Kapitel 5.3.5) und erzeugt aus allen gesammelten Informationen einen String, der letztendlich auf der Webseite ausgegeben wird.

5.7 Straßensuche

Um den potentiellen Benutzern des fertigen Radroutenplaners das Auffinden von Straßen zu erleichtern, steht diesen mittels Eingabeformularen eine komfortable Suchmöglichkeit gewünschter Straßen zur Vefügung.

5.7.1 Zielvorgaben

Bei der Eingabe eines Straßennamens erscheinen bereits im Vorfeld die relevantesten Suchtreffer, aus denen der gewünschte Eintrag angeklickt werden kann - die sogenannte Autovervollständigung der Sucheingabe. Die Auswahl eines der vorgeschlagenen Straßennamen führt zur automatischen Positionierung des Start- oder Endmarkers an die zutreffende Koordinatenstelle der Karte. Um zu diesem ersten Ergebnis zu gelangen, sind einige mehr oder weniger aufwendige Schritte notwendig, die von Algorithmenentwicklungen bis hin zur Implementierung dieser in diversen Datenbank- und Skriptsprachen reichen.
Zu den grundlegenden Suchmöglichkeiten kommen diverse Funktionalitätsverfeinerungen zugunsten des Bedienkomforts hinzu, sodass sich insgesamt folgende Zielvorgaben herausstellen:

- Optimale Straßenmittelpunktfindung
- Zentrierung der Kartenebene auf den gefundenen Straßenmittelpunkt, jener der von dem Benutzer gesuchten Straße zugehörig ist.
- Nach der erfolgten Suche ist, wenn vom Anwender gewünscht, auf die Ausdehnung der gefundenen Straße zu zoomen und farblich hervorgehoben darzustellen.
- Jeder gesuchten Straße ist ein Ort zugewiesen
- Automatische Vorschläge während des Tippens
- Korrekturvorschläge bei fehlerhafter Eingabe
- Der nach der Zentrierung erfolgten Markerpositionierung auf den Mittelpunkt, der es dem Benutzer freistellt den Marker an eine beliebige Stelle der Karte zu verschieben und von dort aus das Routing zu starten oder dort enden zu lassen.

Der letzte Punkt gehört zu den allgemeinen Bedienfunktionen und wird im Kapitel 5.9.1.1 auf S.120 beschrieben.

5.7.2 Herstellen einer Datengrundlage

Zuerst muss der Datenbestandes auf **Eignung für das Geocoding** überprüft werden. Um die Eingabe des Anwenders in Koordinaten umzuwandeln, ist es von großer Bedeutung, einen geeigneten Datenbestand zu haben, der die Koordinaten zu den Straßen bereits bereithält. Dazu werden die beiden bekannten Openstreetmap-Datenimportwerkzeuge “OSM2pgrouting“ und “Osmosis“ in Betracht gezogen. Mit ihnen kann bekannterweise ein Abbild der OSM-Daten in die Datenbankebene erfolgen.
Nach unterschiedlichen Untersuchungen machen sich die beiden Datenbankschemata aufgrund ihrer weniger optimalen Datengrundlage schnell bemerkbar. Hier sind zwar nützlichche Informationen wie die Straßengeometrien oder -namen vorhanden, aber eine dort abgespeicherte Straße besteht in den meisten Fällen aus mehr als einem einzigen Linienzug. Vielmehr umfasst er mehreren Teilstücke. Jedes dieser Teilstücke ist einzeln und unter anderem mit einer eigenen ID, Geometrie und einem eigenen Namen abgespeichert. Wichtig ist dabei, dass diese Straßenelemente nicht dieselbe ID besitzen und somit nicht eindeutig einer einzigen Straße zuzuordnen sind. Die Elemente haben immer jeweils eine eigene Identifikationsnummer. Diese ID ist aber nicht fortlaufend vergeben, wodurch dem Programmierer die Option nicht gegeben ist, alle gleichnamigen Objekte mit sich anschließender bzw. ähnlicher Nummer zusammenzufügen. Möglich wäre dies, wenn jeder Straßenname im gesamten Betrachtungsgebiet nur einmalig vorkäme. Somit könnten einfach alle gleichnamigen Straßenstücke zusammenfügt werden, und dabei wäre sichergestellt, dass sie auch zusammengehören.
Letzterem werden ebenfalls die Datenbankinhalte nicht gerecht. Zwei Straßen, die die gleiche Bezeichnung tragen, können - und das nicht selten - in verschiedenen Gebieten oder Gemeinden auftauchen. Wenn man nun im gesamten Datenbestand (hier: Osnabrück und Umgebung) beispielsweise nach *Arndtstraße* sucht, werden mehrere Treffer, d.h. Teilstücke, die den gesuchten Namen enthalten, aufgelistet. Das Problem dabei ist, dass nicht ohne Zweifel gesaget werden kann, welches Teilstück zur welcher Arndtstraße gehört. In dieser Hinsicht gleichen sich die beiden mit den Importprogrammen erstellten Datengerüste. Osmosis unterteilt eine Straße nicht derart häufig wie OSM2pgrouting, da es die für das Routing unabdingbare Topologieeigenschaften nicht benötigt. Hier sind Straßen so repräsentiert, wie sie z.B. in JOSM (vgl. Kap. 3.2) zu sehen sind. Bei einem Klick auf eine Straße ist die Markierung aller dazugehörigen Teile zu sehen, sofern sie nicht bewusst unterteilt wurde, um beispielsweise einen Straßenabschnitt, der sich von dem Rest der Straße unterscheidet, mit einer andersartigen Oberflächenangabe zu versehen. Im Osmosisschema liegen die Straßensegmente als Teilelemente repräsentierende Linestrings vor.
Beispiel:

Mit einem einfachen Befehl lassen sich alle Teilstücke für die *Arndtstraße* anzeigen:

```
SELECT * FROM ways w, way_tags wt WHERE wt.way_id=w.id AND wt.v='
    Arndtstraße'
```

Jedes dieser Teilstücke hat eine andere ID und eine andere Geometrie (siehe Tabelle 43). Doch nötig ist die Zuweisung dieser Stücke zu einer einzigen, eindeutigen Straße. Ohne diese Zuordnung besteht bei der Suche einer Straße unter anderem die Möglichkeit, dass die Straßensegmente als eigenständige Wege behandelt und als mehrmalig innerhalb eines Verwaltungsgebietes auftauchend angesehen werden. Zudem gibt es in diesem Zustand auch keinen klaren Extent, auf den gezoomt werden kann.
Die nächste Aufgabe besteht also darin, die einzelnen Straßenelemente eindeutig einer Straße zuzuordnen. Zu diesem Zweck ist die Entwicklung eines Algorithmus von großem Vorteil, der eine performante Datengrundlage auf Datenbankebene schafft, damit Daten passend zur Anwendereingabe in schneller Zeit abgerufen werden können.

5.7.3 Straßensortierung

Wie bereits erwähnt, gibt es im Osnabrücker Land Straßen, deren Namen identisch sind. Beispielsweise *Neuer Graben* (in Osnabrück und Melle) oder *Parkstraße* (Osnabrück, Bad Rothenfelde, Bramsche...). Unter normalen Umständen bzw. aufgrund der weniger optimalen Form, in der die Straßen in der Datenbank gespeichert sind, ist es nahezu unmöglich diese gleichnamigen Wege voneinander zu unterscheiden. Im Nachfolgenden ist die bereits oben abgefragte Arndtstraße mit ihren Teilstücken zu sehen.

name	**way_id**	**geometry**
Arndtstraße	7763531	Linestring(Anzahl Punkte: 3)
Arndtstraße	7763592	Linestring(Anzahl Punkte: 3)
Arndtstraße	26801082	Linestring(Anzahl Punkte: 23)
Arndtstraße	26801088	Linestring(Anzahl Punkte: 2)
Arndtstraße	26801089	Linestring(Anzahl Punkte: 2)
Arndtstraße	29191353	Linestring(Anzahl Punkte: 4)
Arndtstraße	32909400	Linestring(Anzahl Punkte: 7)
Arndtstraße	39272775	Linestring(Anzahl Punkte: 5)

Tabelle 7: *Arndtstraße im Tabellenschema Osmosis*
Quelle: Eigene Darstellung

Entweder führt die Vermutung eines Betrachters dazu, dass es sich hier um Repräsentation einer einzige Straße mit mehreren Nebenwegen handelt oder er meint, es seien 8 verschiedene Straßen, die im gesamten Gebiet verteilt sind. Eine Überprüfung der Koordinaten ergibt aber, dass es sich tatsächlich um nur 4 verschiedene Wege handelt, die in unterschiedlichen Orten vorzufinden sind. Eine Zuweisung der Segmente zur jeweiligen Straße anhand der `way_id`

oder sonstigen in der Datenbank liegenden Informationen ist nicht möglich. Die zu Osnabrück gehörende *Arndtstraße* hat die IDs 26801082, 26801088, 26801089. Wie man sieht, ist sie nicht fortlaufend nummeriert, aber dennoch enthält sie Werte, die sich nur minimal voneinander unterscheiden. Man könnte daraus schließen, dass alle wertenahen Zahlen zu einer Straße zu zählen. Doch sie könnten genauso so 26801082, 2687767, 5668108 lauten und wären damit unbrauchbar. Außerdem gibt es keine weitere Spalte, die eine eindeutige ID für die gesamte Straße besitzt, wie beispielsweise:

name	**way_id**	**geometry**	**ID**
Arndtstraße	26801082	Linestring(Anzahl Punkte: 23)	2345
Arndtstraße	26801088	Linestring(Anzahl Punkte: 2)	2345
Arndtstraße	26801089	Linestring(Anzahl Punkte: 2)	2345

Tabelle 8: *Optimale Datenbankform*
Quelle: Eigene Darstellung

Alles mit 2345 gehört eindeutig zu einer bestimmten Straße. Diese Form ist aber ebenfalls nicht existent.
Warum müssen Straßen unterschieden/sortiert werden?
Wichtig bei der Suche ist die Unterscheidung der Wege. Wenn der Benutzer “Arndtstraße“ und optional den dazugehörigen Ort eingibt, soll auch eindeutig diese Arndtstraße gefunden und nicht mit einer anderen Arndtstraße vermischt werden. Außerdem gibt es hierdurch die Möglichkeit, ganz einfach an die richtige Ausdehnung dieser Straße heranzukommen, um anschließend durch einfache OpenLayers-Funktionen diesen Extent in den Fokus zu nehmen.
Um diesen Anforderungen gerecht zu werden, muss die Eindeutigkeit mit einem Algorithmus hergestellt werden.

Der Algorithmus für die Straßensortierung in Kurzform:

1. Nehme aus der Menge der gefundenen Straßenelemente das erste raus (A) und überprüfe sukzessiv, ob es sich mit den restlichen Teilen schneidet/berührt. Zur erfolgreichen Berührung zählt auch eine Distanz < 250Meter, um somit parallel verlaufende Wege, oder andere sich nicht berührende Straßenteile dennoch als zusammengehörig zu definieren.

2. Wenn Berührung vorhanden, entferne dieses Stück aus der Restmenge und

3. bringe es mit dem ersten Teilstück A zusammen (verschmelzen). Erstes Teilstück A + berührtes/geschnittenes Teilstück B aus dem Rest bilden nun gemeinsam das neue Teilstück M1. Wiederhole 1.) mit M1: Überprüfe M1 auf BerührungSchnittpunkt mit

der Restmenge.

4. Abbruchbedingung1: wenn in einem Durchgang ein Schnittpunkt mit keinem der Restelemente vorhanden war : Speichere Alle bisherigen Schnittelemente als eine zusammengehörige Straße. Wenn im Rest noch mehr als 2 Elemente sind, nehme das erste raus als A und wiederhole wie 1.). Abbruchbedingung für Ende aller Durchgänge: wenn im Rest keine Teile mehr existieren.

Beispiel Arndtstraße:

1. Nehme aus den n=8 Elementen das erste (oder irgendeins) raus und benenne es mit A.
2. Lösche A aus der Gesamtmenge. Gesamtmenge nun n-1=7
3. Überprüfe,ob A einen “touch“ oder “intersect“ mit den restlichen n Elementen hat.

```
A touches n2 - true
A touches n3 - false
A touches n4 - false
A touches n5 - false
A touches n6 - false
A touches n7 - false
A touches n8 - true
```

4. Schnittelemente: A, n1, n7. Verschmelze sie und nenne sie M1.
5. Lösche alle geschnittenen Teile aus Rest: n-n2-n8=5
6. Überprüfe,ob M1 einen “touch“ oder “intersect“ mit den restlichen n Elementen hat.

```
M1 touches n3 - false
M1 touches n4 - false
M1 touches n5 - false
M1 touches n6 - false
M1 touches n7 - false
```

7. In diesem Durchgang gab es keinen Schnittpunkt. Speichere M1 als eine zusammengehörende Straße in die Datenbank.
8. Wiederhole von Schritt 1 bis 7: Nimm aus Rest das erste raus: n3=A

```
A touches n4 - false
A touches n5 - false
A touches n6 - false
A touches n7 - false
```

In diesem Durchgang gab es keinen Schnittpunkt. Speichere M1 als eine zusammengehörige Straße in die Datenbank. (In diesem Fall besteht die Straße dann nur aus einem Element (n3)) Wiederhole von Schritt 1 bis 7: Nimm aus Rest das erste raus: n4=A

```
A touches n5 - false
A touches n6 - false
A touches n7 - false
```

In diesem Durchgang gab es keinen Schnittpunkt. Speichere M1 als eine zusammengehörende Straße in die Datenbank. (In diesem Fall besteht die Straße dann nur aus einem Element (n4)) Wiederhole von Schritt 1 bis 7: Nimm aus Rest das erste raus: n5=A

```
A touches n6 - true
A touches n7 - true
```

Schnittpunkte vorhanden. Verschmelze n5,n6 und n7 als M1 und gucke, ob Schnittpunkte mit der Restmenge.

9. Restmenge ist aber leer. Beende.

Das Ergebnis ist also:
Straße 1 = n1+n2+n8
Straße 2 = n3
Straße 3 = n4
Straße 4 = n5+n6+n7

Und somit 4 verschiedene, im Raum verteilte Straßen.

Die **Implementierung dieses Algorithmus** ist der *Streetsort*. Realisiert wird das Ganze mit PHP (vgl. Kap. 4.9), PostgreSQL und PostGIS (vgl. Kap. 4.4). Nach jeder Sortierung, d.h. Erstellen einer zusammengesetzten Straße, werden Daten der sortierten Straße in eine Streetsort Tabelle *gc_berechnet* geschrieben. Als geeignet angesehen werden Informationen über den Extent der Straße, den Ort, in dem sie sich befindet, die Geometrie, welche die komplette Straße als eine Einheit beschreibt und letztenendes ein geeigneter Referenzpunkt dieser Straße. Was dieser Referenzpunkt ist und wie er gebildet wird, ist weiter unten zu lesen. Dazu wird mit dem einfachen SQL Befehl eine Tabelle erstellt, bei der die Straßen eine eindeutig und automatisch inkrementierende ID (OID) haben. Die Geometrie der bearbeiteten Wege erfährt eine Speicherung in der WKT-Form (Linestring oder Multilinestring). PostGIS bietet anschließend Voraussetzungen zur erweiterten Verarbeitung der erstellten Geometrien,

um etwa durch die PostGIS-Funktion ST_Envelope den Extent einer Geometrie zu bekommen.

Spalte	Datentyp
oid	integer
name	text
referenz	geometry
extent	geometry
gemeinde	text
geom	geometry

Abbildung 40: *Spalten der Tabelle gc berechnet*
Quelle: Eigene Darstellung

Eine zentrale Rolle im Sortierungsskript trägt der PostGIS-Befehl ST_Dump, der die Fähigkeit besitzt, beliebige Multigeometrien in zwei Bestandteile aufzuteilen. Die von der Funktion returnierten Elemente sind zum einen Geometrien, die sich in der Collection befinden und zum anderen die Nummern dieser Geometrien in der Collection. Im Folgenden ist die Nutzungsform dieser Funktion zu sehen:

```
1 SELECT (ST_Dump(geom)).geom, (ST_Dump(geom)).path as nummer
2 FROM irgendEinMultiLineString
```

Die Ausgabe liefert die Form

geom	nummer
LINESTRING ('...')	{1}
LINESTRING ('...')	{2}

Tabelle 9: *Ausgabe ST_DUMP*
Quelle: Eigene Darstellung

Aus der Geometriecollection lassen sich nun beliebige Elemente herausnehmen und zu einer neuen Geometrie zusammenstellen, weil die Möglichkeit gegeben ist, bestimmte Elemente auszuschließen. Genau dieses Verfahren ist für die Umsetzung des Algorithmus nötig.
Der Vorteil dieser von Streetsort mit Daten gefüllten statischen Tabelle ist nicht nur die eindeutige Zuordnung der Einzelteile zu einer Straße, sondern auch die Art und Weise der Speicherung, die mit Hilfe einfacher SQL-Selects Befehlen der Art

```
1 select name from gc_berechnet where name='Neuer Graben'
```

eine schnelle, mit der Datenbankabfrage zusammenhängende Zugriffszeit bietet.
Diese Begünstigung ist für die Implementierung einer Autovervollständigung ebenfalls von

Wichtigkeit. Denn hier ist es üblich, dass mit jedem eingetippten Buchstaben eine Anfrage an die Datenbank läuft. Ohne das Vorhandensein einer vorab mit notwendigen Informationen gefüllten Datenbank käme es aufgrund komplexerer SQL-Abfragen zur großen Belastung dieser.
Eine weitere Aufgabe dieses Skriptes ist es, die Spalte "gemeinde" zu füllen. Für Straßenobjekte innerhalb der Grenzen des Landkreises Osnabrück fügt es den Namen der Samtgemeinde, in dem sich die betrachtete Straße befindet, ein. Liegt die Straße außerhalb des Landkreises, so wird jeweils die Richtungsangabe und die Entfernung zum nächstgelegenen Ort an die Spalte übergeben.

5.7.4 Zuweisung der Straße zu einem Ort

Die in OpenStreetMap vorhandenen Grenzen in Form von Relationen sind in weiten Gebieten Niedersachsens noch größtenteils unvollständig. Aus diesem Grund wird zunächst auf Grenzdaten zurückgegriffen, die diesem Projekt durch die Behörde für Geoinformation, Landentwicklung und Liegenschaften (GLL) zur Verfügung gestellt werden. Die Daten befinden sich in einer die Grenzen der Samtgemeinden des OS Landes beinhaltenden ESRI-Shape-Datei. Doch um diese Datei ohne Bedenken nutzen zu können, ist die Durchführung einiger Operationen von essentieller Notwendigkeit. Zur Fehleruntersuchung kommen im ersten Schritt die FWTools [Mit08, S.86] zum Einsatz, mit deren Hilfe herausgefunden wird, dass ein ein Defizit in der Definition des Projektionssystems herrscht:
Befehl:

```
ogrinfo -summary c:/gemeindgr.shp gemeindegr
```

Ergebnis: Layer SRS WKT <unknown>. Doch zu sehen sind hier auch die Koordinaten der Eckpunkte des Extent:
<3405296.869970, 5767385.419970> -<3467060.369970, 5842461.899970>.
Dabei ist zu erkennen, dass es sich um Gauss-Krüger-Koordinaten handelt. Mit QuantumGIS können die Polygone der Gemeindegrenzen visualisiert und die Koordinaten nochmals angesehen werden. Um dieser Shape-Datei ein geeignetes Projektionssystem zuzuweisen, werden die ogr-Tools benutzt, dessen Funktionalitäten in den FWTools integriert sind.

Befehl 2:

```
ogr2ogr -a_srs epsg:31467 c:/gemeindegr.shp  c:/gemeindegr_31467.shp
```

Zudem fehlen die Bezeichnungen der Gemeindenamen, welche nachträglich mit QGIS eingepflegt werden. Das nun brauchbare Shapefile kann jetzt mit `shp2pgsql` in die PostgreSQL-PostGIS Datenbank eingeladen werden. Die Darstellung der Datenbankinhalte mit den Gren-

zen mittels OpenJump zeigt, dass die Daten richtig und verlustfrei importiert wurden. Die Weiterverwendung dieser importierten Daten ist also möglich.
Um zu verhindern, dass eine Straße mehreren Orten zugeteilt wird, stellt sich heraus, dass die Verwendung des Referenzpunktes (vgl. Kap. **??**: "Referenzpunktbestimmung") am sinnvollsten ist, da dieser Punkt nur in einer Gemeinde liegen kann.

Behandlung von Straßen innerhalb des Osnabrücker Landes: Nachdem mit `Streetsort` der Referenzpunkt berechnet ist und die Grenzen in der Datenbank vorliegen, kann mit diesem eine räumliche Abfrage erfolgen. Dafür bietet sich die Abfrage des Enthaltenseins an. Falls der Punkt innerhalb (contains) der 35 Gemeindepolygone liegt, wird der entsprechende Gemeindename zurückgeliefert und in die `gc\_berechnet-Tabelle` eingetragen. [hier noch ein bild zur veranschaulichung]

```
1  WHERE w.id = wt.way_id  AND wt.k='name' AND wt.v='".$Strassenname."'
2  and ST_Contains(g.the_geom_31467,ST_Geomfromtext('$mittelpunkt
       ',31467))
```

Grenzennamen mit g.name ausgeben lassen, in der sich die Straße `\$Strassenname` befindet Dazu muss die Geometrie bzw. der Referenzpunkt, hier die Variable `\$mittelpunkt`, sich in diesem Grenz-Polygon befinden.

Behandlung von Straßen außerhalb des Osnabrücker Landes: Falls der Referenzpunkt der Straße in keinem der Polygone liegt, so muss er sich außerhalb befinden. Dann gilt es, die nächstgelegene Stadt zu diesem zu bestimmen. Entfernung und Richtung dazu sind ebenfalls auszurechnen. Gesucht wird hier also die minimale Distanz vom Mittelpunkt der Straße zur nächsten Stadt (Einwohner größer als 10000, definiert durch OSM-Tag "town").
Für den Fall, dass die Straße außerhalb des Osnabrücker Landkreises liegt, erfolgt die entsprechende SQL-Abfrage, um aus der mit Osmosis erstellten Punktetabelle, die sämtliche OSM-Tags beinhaltet, den nächstgelegenen Punkt mit dem "place"-tag zu lokalisieren. Die hier verwendeten und verschiedene Ortskategorien repräsentierende Attribute sind:
Zur Bestimmung der Richtungen werden die Verhältnisse der x-und-y-Abstände vom Straßenmittelpunkt und der in der Datenbank mit einer Punktgeometrie hinterlegten Stadt betrachtet.

Beispiel zur Bestimmung des Bereiches nordöstlich:
Der Bereich besitzt wie alle anderen Richtungsangaben eine Spannweite von 45°.
Bekannt ist also der Winkel zwischen der positiven y-Achse und der ersten gestrichelten, diagonalen Linie auf der rechten Seite im ersten Quadranten und die y-Differenz zwischen diesen beiden Geraden. Der Winkel beträgt 22.5°, der hier wichtige y-Wert beträgt 1 [Einheit].
Mit einfachen Winkelfunktionen lässt sich die x-Differenz berechnen. Eine anschließende Di-

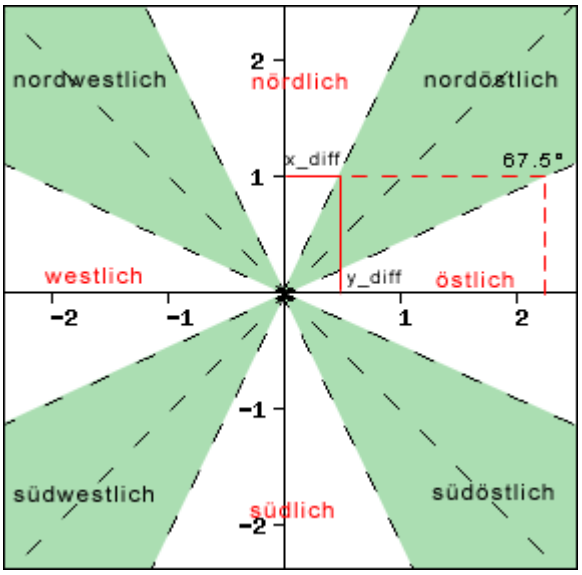

Abbildung 41: *Angabe der Himmelsrichtungen*
Quelle: Eigene Darstellung

vision dieser beiden Differenzen gibt das Verhältnis, das den Anfang des *nordöstlich*-Bereiches definiert:
$\frac{x\_diff}{y\_diff} = \frac{0.414}{1} = 0.414$
Die zweite Begrenzung dieses Bereiches ist bei 67.5°:
$\frac{x\_diff}{y\_diff} = \frac{2.414}{1} = 2.414$

Nachdem der nächstgelegene Ort ausfindig gemacht ist, wird mit Hilfe der oben beschriebenen Berechnungen der Richtungsangaben bestimmt, in welcher Himmelsrichtung sich der jeweilige Ort befindet und wie weit dessen Distanz zur Straße ist.

Beispiel: Die Suche nach “Sandstraße“ ergibt folgende Ortszuweisungen:

Straße 1: Sandstraße - Bad Iburg
Straße 2: Sandstraße - Hagen a.T.W.
Straße 3: Sandstraße - Hasbergen
Straße 4: Sandstraße - Osnabrück
Straße 5: Sandstraße - 6.5 km nordöstlich von Damme, Dümmer
Wie anfangs gefordert, werden auch diese Informationen der Ortsangaben in die Tabelle `gc_berechnet` geschrieben.

5.7.5 Referenzpunktbestimmung

Eine weitere Herausforderung stellt die Bestimmung der Mitte einer Straße dar. Da in OpenStreetMap in der Regel keine Hausnummern vorhanden sind und eine Erfassung von diesen

einen unzumutbaren Mehraufwand darstellen würde, ist es bei der Routenberechnung nicht möglich, die Hausnummer zu einer Straße anzugeben.
Aus diesem Grund muss ein geeignetes Verfahren gefunden werden, um einen Punkt auf der Straße als Referenzpunkt zu bestimmen, um dort die anfangs geforderte Markerpositionierung auf die Straße durchzuführen. Dieser Referenzpunkt sollte sinnvollerweise die Mitte der Straße repräsentieren. Doch dieser ist nicht ohne Weiteres zu bestimmen, zumal nicht jede Straße aus nur einem einfachen Linestring bzw. einer zusammenhängenden Linie besteht, sondern oftmals auch aus vielen kleinen - zum Teil sehr ausgearteten - Nebenwegen besteht. Nun gilt es, alle kleinen Nebenwege zu erkennen und sie bei der Mittelpunktbestimmung zu vernachlässigen zwecks der Erkennung des Wege-Hauptverlaufes. Der Marker wird, aufgrund der Bestimmungsschwierigkeit, an welchem Punkt sich der Anfang oder das Ende befindet, nicht genau dorthin gepackt. Ziel ist es, den Marker inmitten der Gesamtgeometrie zu platzieren. Somit gäbe es auch eine Einheitlichkeit bei der Positionierung, denn bei den Start -oder Endpunkten kann es vorkommen, dass der Ort für den Marker mal am Start/Ende des Hauptverlaufes und mal an irgendeinem Nebenzweigsende angenommen wird.
Zur exakten und bestmöglichen Bestimmung des Hauptverlaufes werden diverse Algorithmen, unter anderem auch der Dijkstra-Algorihtmus, ins Visier genommen. Dieser ist nicht mehr für das Finden des kürzesten Pfades zuständig, sondern für die Ermittlung des längsten Pfades. Von diesem längsten Pfad müsste dann nur noch der Mittelpunkt berechnet werden.
Doch aufgrund des Zeitdruckes und der prioritären Zielsetzung, wird die zuletztgenannte Methode als Lösungsansatz wieder verworfen, da sie zu viel Zeit und Aufwand für ein so relativ kleines Problem in Anspruch nehmen würde. Letztendes kommt die sog. *Schwerpunktmethode* zum Einsatz, welche einen Kompromiss zwischen schneller Implementierung und gutem Ergebnis liefert. Der Referenzpunkt ist also: Punkt auf der Straße, der dem Schwerpunkt der Straße am nächsten ist und den Mittelpunkt approximiert. Ein Beispiel der Schwerpunktmethode mit der Caprivistraße: Das kleine rote Quadrat im linken Bild stellt den Schwerpunkt der Caprivistraße dar. Ausgehend von diesem Punkt wird der ihm am nächsten gelegene Punkt auf der Caprivistraße lokalisiert, hier dargestellt durch das rote Kreuz.
Bei der konkreten Straßensuche wird ein Marker direkt auf diese Position gestellt (rechtes Bild).Dafür verantwortlich ist ein unkomplizierter pgSQL-Befehl:

```
1 multiline_locate_point(Straßengeometrie ,ST_CENTROID(Straßengeometrie)
    )
```

Im ersten Schritt wird hierdurch der Schwerpunkt, im zweiten Schritt ein auf der Straße liegender, dem Schwerpunkt nächstgelegene Punkt, bestimmt. Diese Art und Weise der Mittelpunktbestimmung hat als Nebeneffekt den Vorteil, dass der Rest der Straße gut im Blickpunkt bleibt.
Einige Sonderfälle müssen beachtet werden, denn bei diesen ist eine exaktere Mittelpunktbestimmung bzw. Mittelpunktbestimmung des Hauptverlaufes möglich:
Besteht die Straße aus nur 2 Segmenten- und sind diese Segmente am Anfangs oder End-

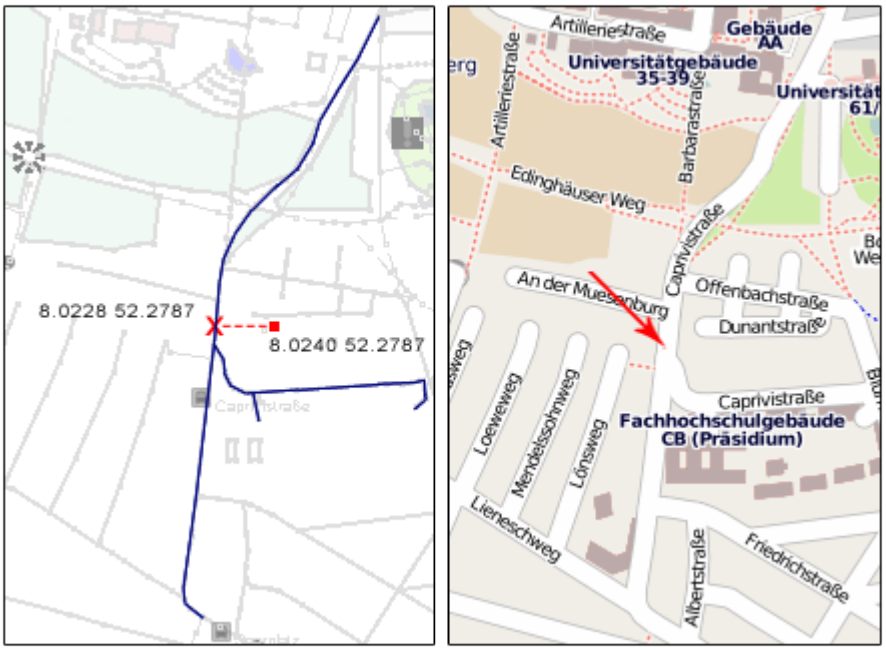

Abbildung 42: *Spalten der Tabelle gc berechnet*
Quelle: Eigene Darstellung

punkt miteinander verbunden, entsteht durch Merge-Operationen eine einzige Gometrie. Danach ist der genaue Mittelpunkt durch `ST_Line_Interpolation(Straßengeometrie, 0.5)` bestimmbar. Besteht die Straße aus nur 2 Segmenten - und bilden diese beiden Teile eine sog. T-Verbindung, erfolgt eine Mittelpunktbestimmung des längeren Elementes.

Nach einem Durchgang Streetsort mit der als 8 elementig angesehenen Arndtstraße, ist folgendes passiert:

- Die 8 Segmente wurden verteilt. Am Ende sind 4 verschiedene Straßen identifiziert.
- Jeder dieser 4 Straßen hat mit der für sie geeigneten Methode den Mittelpunkt berechnet bekommen.
- Zu jeder dieser Straßen ist eine Gemeinde/Ort zugeteilt worden.
- Zusammen mit den oben genannten Informationen kommen der Extent sowie die Geometrie in die Datenbank hinzu.

Der Extent liegt für den Fall vor, dass der Nutzer den gesamten Straßenverlauf visualisieren und den Sichtbereich der Karte auf diesen zentrieren möchte.
Die Vorteile des Skripts, dass die Referenzpunkte und ähnliches automatisiert in die Datenbank schreibt sind: Diese Tabelle enthält nachher alle für die Layersteuerung und Routingberechnung wichtigen Informationen über die vom Benutzer ausgewählte Straße. Es ermöglicht es, aufgrund des statischen Vorhandenseins von allen benötigten Informationen und effizienten

oid integer	name text	gemeinde text	referenz text	geom text	extent text
61754	Arndtstraße	Osnabrück	POINT(3434	MULTILINES	POLYGON((3
61755	Arndtstraße	Bad Laer	POINT(3438	MULTILINES	POLYGON((3
61756	Arndtstraße	Bramsche	POINT(3428	LINESTRING	POLYGON((3
61757	Arndtstraße	Bad Iburg	POINT(3435	LINESTRING	POLYGON((3

Abbildung 43: *gc_berechnet mit bearbeiteter Arndtstraße*
Quelle: Eigene Darstellung

und effektiven Datenquerys, eine viel zu lange Rechenzeit zu vermeiden und zur Datenbankentlastung beizusteuern.

5.7.6 Benutzung der Datengrundlage für die Nutzereingabe

Im folgenden wird die vorher hergestellte Datengrundlage für die anfangs erwähnte Funktion der Straßensuche benutzt wird.

5.7.6.1 Strassensuche und Markerpositionierung

Der Benutzer gibt nun also die Straßennamen der gewünschten Start- und Endposition in ein Formular ein. Diese Eingabe wird in die für das Routing notwendigen Koordinaten umgewandelt. Mit diesen Koordinaten kann in OpenLayers ein Punktfeature erzeugt und individuell dargestellt werden. Dieses Punktfeature kann zusätzlich mit einem Icon versehen werden und als Start- bzw. Stoppmarker dienen. Doch um aus der HTML-Formulareingabe an die entsprechenden Daten in der Datenbank zu kommen, sind einige zusätzliche Schritte zur Realisierung notwendig, da schlichte Formulare keine direkte Kommunikation mit der Datenbankebene erlauben. Ajax bietet hier viefältigere Funktionalitäten mit Fähigkeiten des einfachen Austausches mit einer Datenbank. Sobald eine oder mehrere Straßen zur Eingabe gefunden wurden, sollen diese in Textform ausgegeben und anklickbar gemacht werden. Hier bietet sich ebenfalls Ajax an, da es den entsprechenden Bereich der Webseite mit den Straßennamen füllen kann, ohne die komplette Seite neu laden zu müssen.
Der schematische Ablauf einer Straßensuche mit entsprechender Positionierung des Start- oder Endmarkers nach Betätigen einer Straßensuche ist in Abbildung 54 zu sehen.

Die durch die Nutzereingabe eingegebene Straße wird in einer PHP-Datei mit einer einfachen SQL-Abfrage aus der `gc_berechnet-Tabelle` herausgenommen und in XML-Struktur an eine Javascript-Funktion zurückgegeben. In dieser Datei, in der sich mitunter Geometrie, Name und Ort befinden, filtert sich eine OpenLayers-Funktion das Nötige für die Darstellung oder Positionierung einer Grafik auf den Mittelpunkt der Straße heraus. Wenn mehrere Ob-

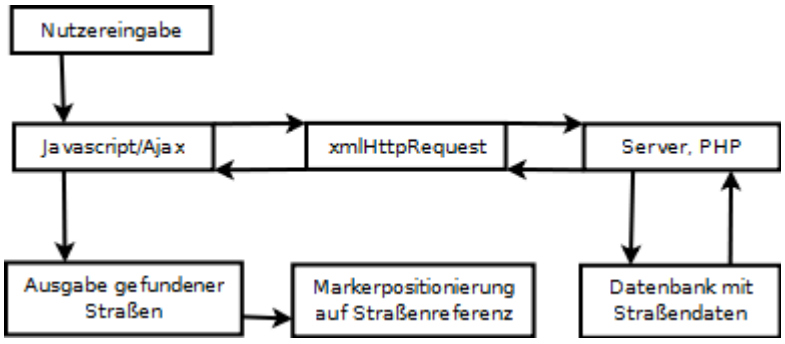

Abbildung 44: *Ablauf einer Straßensuche*
Quelle: Eigene Darstellung

jekte gefunden werden, werden diese zunächst in Form von anklickbaren Links zur Auswahl gestellt. Unterstützt werden die Formen der Eingabe:

- Straßensuche ohne Ortseingabe
- Straßensuche mit Ortseingabe bzw. Ortsangabe mit Straße
- Ortssuche
- Bis zu einem gewissen Grad fehlerhafte Eingaben

Demnach wichtig sind Fallunterscheidung bzgl. der SQL-Anweisungen zur Suche in der Datenbank:

- Auswahl eines Eintrages aus Autovervollständigunsliste: Suchen in der Datenbank nach exakt diesem Ausdruck, denn er ist in jedem Fall vorhanden.
- Eingabe nur einer Zeichenkette: Suche nach Einträgen in der Spalte `name` der `gc_berechnet-Tabelle`, die dem Eingabewert *ähnlich* sind. Die Suche nach dem Ort wird vernachlässigt.
- Falsche Eingabe z.B. durch Vertippen: Suche alle Einträge und überprüfe prozentuale Übereinstimmung der Datenbankelemente mit dem Eingabewert.
- Eingabe von Namen und Ort: Suche in der Datenbank nach Straßenname und Ort, die den Eingabewerten *ähnlich* sind.

Zum Letzten ein Codeschnipsel:

```
1 SELECT ST_AsTEXT(referenz_900913) AS referenzpunkt,name,
2       gemeinde,
3       ST_AsTEXT(extent_900913) AS extent, ST_AsTEXT(geom_900913) AS
           geom
4       FROM gc_berechnet
```

```
5       WHERE name ~* '$strassenname'
6       AND gemeinde ~* '$gemeindename'
```

Die Arbeit leistet ein spezieller Postgres Operator aus der Kategorie "POSIX reguläre Ausdrücke"[5]. Dadurch reicht es aus, wenn sich der eingegebene String irgendwo im Zielstring befindet, wobei die Groß- und Kleinschreibung ebenfalls unerheblich ist.
Abgefangen werden falsch geschriebene Straßennamen durch die php-native fuzzy-Suche similar_text[6], welcher mit der prozentualen Ähnlichkeit zweier Zeichenketten arbeitet. Wird durch die Suche in der "gc_berechnet-Tabelle" keine entsprechende Straße gefunden, werden zunächst alle Zeilen bzw. Einträge der Tabelle zur Verarbeitung herangezogen. Danach findet ein Vergleich des eingegeben Suchworts mit jedem Zeileneintrag statt. Ist die Ähnlichkeit der vorkommenden Buchstaben der Suchstrings sehr hoch bzw. übersteigt diese einen als Schwellenwert angegebenen Prozentwert, so wird der passende Eintrag aus der Datenbank herausgenommen und vorgeschlagen (siehe: Codezeilen 200-300 der Datei `routenbeschreibung.php`).
Beispiel einer korrekten Straßensuche:
Gesucht ist 'Arndtstraße, Osnabrück'. Die für die Suche verantwortliche PHP-Datei liefert nach der Anfrage an die Postgresql-Datenbank alle für die Darstellung relevanten Informationen in einer XML-Datei zurück:

```
1  <strasse>
2     <str id="1">
3    <name>Arndtstraße</name>
4    <wkt>POINT(894780.733696595 6849668.18916958)</wkt>
5    <gemeinde>Osnabrück</gemeinde>
6    <extent>
7      POLYGON((894720.02607208 6849171.76421941,..))
8    </extent>
9    <gesamtestrasse>
10     MULTILINESTRING((894739.490881831 6849172.6192812,..))
11   </gesamtestrasse>
12   <vorschlag>false</vorschlag>
13    </str>
14 </strasse>
```

Mit einer Javascript -bzw. OpenLayers-Funktion erfolgt das Setzen des entsprechenden Markers an die Referenzstelle des aus der Datenbank herausgeholten Koordinatenpaares *wkt*, nachdem aus diesen auf Client-Ebene ein Geometry-Feature erzeugt wurde, (vgl. Kap. **??** auf Seite 57).

[5] Siehe: http://www.postgresql.org/docs/7.4/interactive/functions-matching.html
[6] similar_text. Dies ist ein String-Matching-Algorithmus, (siehe: http://php.net/manual/de/function.similar-text.php)

5.7.6.2 Ortesuche

Die Ortesuche ist ähnlich einfach, da notwendige, auf dem Openstreetmap-Tag "place" beruhenden Orteinformationen in der Osmosis-Datenbank eingepflegt sind. Sie werden jedoch aufgrund der Einheitlichkeit, wie die Straßen zuvor in die *gc_berechnet* eingegetragen. Hierzu wird die Spalte *name* mit dem Ortenamen und *gemeinde* mit dem Ortetyp, den Übersetzungen nach Tabelle 10 entsprechend, gefüllt. Place ist der für Orte verantwortliche Tag in OSM.

Tag	Übersetzung
place=city	Stadt
place=town	Stadt
place=village	Stadt/-Ortsteil
place=hamlet	Ort

Tabelle 10: *Openstreetmap-Tags und Übersetzung*
Quelle: Eigene Darstellung

5.7.6.3 Autovervollständigung der Suche

Seitdem Google die Benutzung des sog. *auto-suggests*[7] eingeführt hat, erfreut sich das Verfahren, dass die Suche vereinfacht, sehr großer Beliebtheit. In einfacher Form ist die Autovervollständigung mit relativ wenig Aufwand zu realisieren, woraufhin die Entscheidung zugunsten des Einsatzes dieser Technologie fällt, zumal es die Bedienbarkeit des Radroutenplaners erhöht bzw. die Straßensuche erleichtert. Im Internet gibt es diverse Anleitungen und fertige Skripte, die auf einfache Art und Weise für eigene Zwecke benutzt werden können. Ein sehr gutes Skript, das auch in diesem Projekt Verwendung findet, ist der jQuery-Autocompleter, welcher auf folgender Seite herunterzuladen ist:
`http://bassistance.de/jquery-plugins/jquery-plugin-autocomplete/`. Hier gelangt man zu einem Autovervollständiger, der auf Ajax und jQuery[8] Basis realisiert ist. Nach nur wenigen Schritten, wie dem Datenbankverbindungsaufbau, dem SQL-Strassensuchbefehl und dem Einbau in die HTML-Seite, ist das Skript schon vollständig nutzbar.
Eine Beispielverwendung:

```
1 $(#start).autocomplete(strassensuche.php, {
2             minChars: 2,
3             matchContains: 1,
4             cacheLength: 10,
5             onItemSelect: selectItem_Start,
6 });
```

[7] `http://www.google.de|`
[8] Ein Javascript-Framework. Siehe: `http://jquery.com/|`

Bei #start handelt es sich um den durch das HTML-Tag gekennzeichneten Namen des Textfeldes. Das führt dazu, dass alle an diesem Feld vorgenommenen Interaktionen zum Autovervollständigungsskript weitergeleitet werden. In den geschweiften Klammern sind einige Beispiele zu den möglichen Optionen zu sehen. `cacheLength` speichert die vom Backend gelieferten Ergebnisse zwischen, sodass keine erneute Anfrage je Tastendruck an den Server erforderlich ist. `onItemSelect` leitet die aus der Vorschlagsliste selektierte Straße an eine Methode weiter, die wiederum das vom User benutzte Textfeld mit einem Wert füllt.
Damit sieht eine Anfrage wie folgt aus: `strassensuche.php?q=Schö`

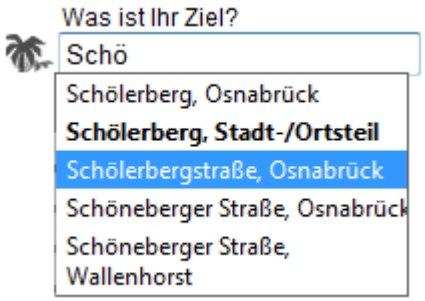

Abbildung 45: *Automatische Vorschläge für die Suche*
Quelle: Eigene Darstellung

5.8 POI- und Universitätssuche

Ein für den Komfort des Radroutenplaners sehr wichtiges Feature ist die POI-Suche. Die englische Abkürzung POI steht für „Point of Interest“. Gemeint sind damit also Orte, die für den jeweiligen Anwender von besonderem Interesse sind oder sein können, so wie beispielsweise Restaurants, Hotels, touristische Ziele, aber auch Fahrradläden und -werkstätten, sowie Stationen des öffentlichen Personenverkehrs.

5.8.1 Allgemeine Funktionsbeschreibung

Im folgenden Abschnitt werden die PI-Suche und die Universitätssuche genauer vorgestellt und ihre Funktionsweise erläutert.

5.8.1.1 POI-Suche

Da es für den Nutzer des Routenplaners sehr viele potentiell interessante Orte gibt, ist es wichtig, für eine angemessene Übersicht der Ergebnisse zu sorgen. Aus diesem Grund wurden die in der Datenbank und OSM gespeicherten Orte, die man sich mit der Suchfunktion anzeigen lassen kann, in verschiedene Kategorien eingeteilt. So kann der Anwender zum Beispiel gezielt nur nach Restaurants, Bahnhöfen, Cafés oder für das Fahrrad wichtige Orte suchen. Es wird dem Nutzer jedoch auch weiterhin noch ermöglicht, nach POIs aller Kategorien zu

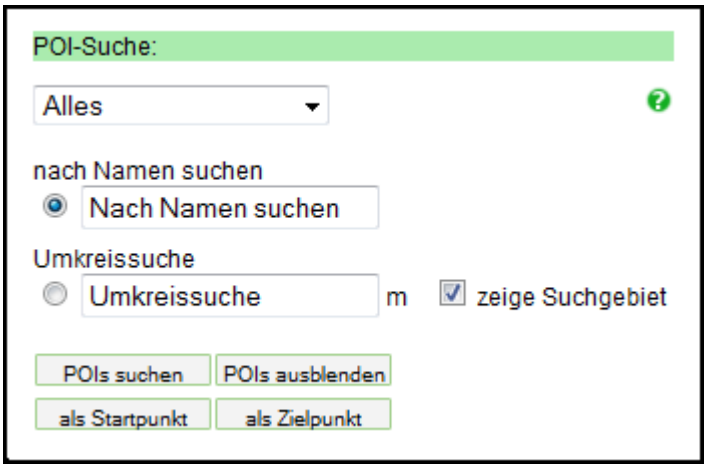

Abbildung 46: *Formular der POI-Suche*
Quelle: Eigene Darstellung

suchen, die seinen durch Namen- oder Umkreissuche spezifizierten Angaben entsprechen.

Nachdem der Anwender sich also für eine oder alle Kategorien entschieden hat, kann er zwei verschiedene Möglichkeiten auswählen, um die POI-Suche durchzuführen. Entweder er entscheidet sich für die Namenssuche, um alle entsprechenden POIs zu finden, deren Namen den eingegebenen Suchbegriff enthalten oder er wählt die Umkreissuche. Wird die Umkreissuche gewählt, so muss der Nutzer an der Stelle in die Karte klicken, die den Mittelpunkt seines Umkreises markiert und kann anschließend im POI-Formular den gewünschten zu durchsuchenden Umkreis in Metern angeben. Diese Methode ermöglicht es dem Anwender auch POIs zu einer gewünschten Kategorie zu finden, dessen Namen er noch nicht kennt, was vor allem für ortsfremde Radfahrer eine wichtige Funktion ist.
Entscheidet sich der User für die Namensuche, gibt jedoch keinen Suchbegriff im dafür vorgesehenen Fenster ein, werden ihm alle zur gewählten Kategorie passenden POIs angezeigt, die im zum Zeitpunkt der Suche sichtbaren Kartenbereich liegen.

Unabhängig von der gewählten Suchmethode werden die gefundenen POIs durch ein entsprechendes Symbol in der Karte angezeigt. Außerdem öffnet sich das Informationsfenster, in der noch einmal die Namen aller gefundenen Orte aufgelistet sind. Ein Klick auf den Namen sorgt dafür, dass der entsprechende POI in der Karte zentriert wird. Ein Klick auf ein in der Karte angezeigtes POI-Symbol öffnet ein Pop-up, in dem weitere Informationen zum entsprechenden Ort aufgeführt sind.
Um den Server nicht durch umfangreiche Anfragen unnötig zu belasten, werden immer nur maximal 100 POIs angezeigt. Überschreitet die Anzahl der gefunden POIs diese Grenze, so wird die Suche nicht gestartet und stattdessen im Informationsfenster der Hinweis „Über 100 POIs gefunden. Suche verfeinern bzw. in die Karte zoomen, oder bei Umkreissuche: Umkreis verkleinern.“ angezeigt.

Durch Klicken auf den Knopf „als Startpunkt“ oder „als Zielpunkt“ wird der gefundene bzw. ausgewählte POI direkt als Start- oder Endpunkt des Routings gesetzt.

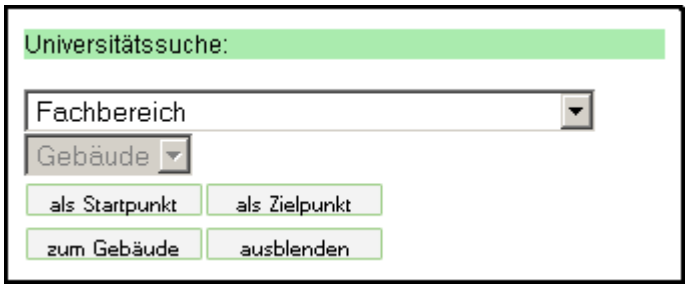

Abbildung 47: *Formular der Universitätssuche*
Quelle: Eigene Darstellung

5.8.1.2 Universitätssuche

Eine mit der POI-Suche vergleichbare Funktion des Routenplaners, die es vor allem neuen Studenten in Osnabrück erleichtern soll, ihre richtigen Universitäts- oder Fachhochschulgebäude und den optimalen Weg zu diesen zu finden, ist die sogenannte Universitätssuche. Sie befindet sich im gleichen Menü direkt unter der POI-Suche. Hier kann zunächst die Hochschule und zusätzlich bei der Universität auch der gewünschte Fachbereich selektiert werden. Anschließend kann mit der Gebäudeliste die gesuchte Gebäudenummer oder -bezeichnung ausgewählt werden. Durch einfachen Klick auf einen der darunterliegenden Knöpfe lässt sich das gewählte Gebäude wahlweise direkt als Start- oder Stopppunkt für das Routing definieren oder einfach direkt in der Karte anzeigen.

5.8.2 Realisierung der Suche

Die Realisierung der hier beschriebenen Suche funktioniert auf ähnliche Art und Weise, wie die POI-Suche.

5.8.2.1 Interne Verarbeitung der Suchparameter

Beim Klick auf die Buttons zum Anzeigen oder Setzen des Start- oder Endpunktes wird eine Funktion aufgerufen, die für eine korrekte Bearbeitung der Anfrage sorgt.
Zu diesem Zweck können der Funktion Parameter übergeben werden, die dazu dienen, die Suchoptionen zu definieren, also beispielsweise, ob es sich um eine POI-, Umkreis- oder Universitätssuche handelt. Im Falle der Umkreissuche wird zusätzlich ein Kreis als OpenLayer-Objekt erstellt, der anschließend optional in einem Layer auf der Karte dargestellt werden kann, um das Suchergebnis anschaulicher zu präsentieren.

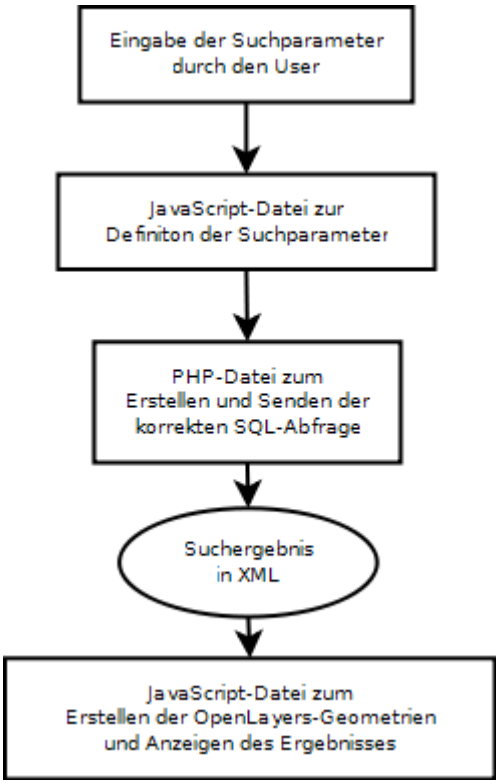

Abbildung 48: *Schema der Suchverarbeitung*
Quelle: Eigene Darstellung

5.8.2.2 Erstellen der SQL-Abfrage

Wurden alle für die Weiterverarbeitung der Anfrage benötigten Parameter definiert, so wird die Datei `createPoiRequest.php` aufgerufen, welche die Parameter, wie beispielsweise den für die POI-Suche unter Umständen wichtigen, aktuell auf dem Bildschirm dargestellten Kartenausschnitt, die vom Benutzer gewählte Suche oder, je nach Suchart, den Namen oder Umkreis, der für die Suche benutzt werden soll übergeben bekommt.
Der Zweck der PHP-Datei ist es nun, aus den übergebenen Informationen eine SQL-Anfrage zu konkatenieren, die aus der Datenbank die zur Suchanfrage passenden Ergebnisse zurückliefert und diese wiederum im XML-Format an die ursprüngliche Aufrufmethode `suchePOI()` weitergibt.

In einer früheren Version wurde anstatt der XML-Rückgabe eine Mapdatei erzeugt, welche die gefundenen Punkte aus einem WFS zurückliefert. Diese Methode erweist sich beim Testen aber als weniger geeignet, führt sie doch gleich mehrere Nachteile mit sich:

- Die Anfrage über den WFS benötigte deutlich mehr Zeit und verursachte beim Anzeigen vieler Ergebnisse eine deutliche Verzögerung der Kartensteuerungsfunktionen.

- Aufgrund von Sicherheitsrestriktionen in JavaScript war ein Proxy[9] nötig, der unumgehbare Fehler mit ausgab. Eine dieser Fehlermeldungen (*Uncaught-Exception in der XMLHttpRequest.js :: anonymous :: line 200 data: no*) kam dadurch Zustande, dass mehrere Anfragen hintereinander an den WFS geschickt wurden, ohne das die erste Anfrage vollständig bearbeitet wurde.[Ada] Zum damaligen Zeitpunkt konnte keine geeignete Lösung gefunden werden.

- Da im Gegensatz zur jetzigen Lösung, bei der aus der XML-Struktur lokal OpenLayers-Objekte erstellt werden, die GML-Objekte[10] des WFS nicht schnell genug zur Verfügung standen, konnten nicht alle Operationen von OpenLayers hierauf angewendet werden. Der Client und JavaScript waren schneller im Verarbeiten der weiterführenden Anweisungen als der Server die vom WFS benötigten Informationen liefern konnte.

Ein beispielhaftes Konstrukt der SQL-Query aus der PHP-Datei `createPoiRequest.php` lautet

```
1 SELECT DISTINCT AsText(n.geom) AS punkt, nt2.v AS name, n.id AS id
2 FROM nodes n, node_tags nt, node_tags nt2
3 WHERE nt.node_id=nt2.node_id AND nt2.k='name' AND n.id = nt.node_id
4 AND nt.k='name'
5 AND n.geom && ST_SetSRID(ST_MakeBox2D(ST_Point([X-Koordinate unten
    links],[Y-Koordinate unten links])
6    ,ST_Point([X-Koordinate oben rechts],[Y-Koordinate oben rechts]))
       ,4326)
7 ORDER BY NAME ASC;
```

In diesem Beispiel wurde die Suchanfrage nach POIs aller Kategorien ohne bestimmten Namen zu einer SQL-Abfrage verarbeitet. Die verwendete 2D-Box repräsentiert den zum Zeitpunkt der abgeschickten Suchanfrage angezeigten Bildschirmausschnitt und begrenzt den Suchraum somit auf diesen Bereich. Bei der Umkreissuche hingegen würden mit Hilfe von PostGIS-Funktionen aus dem übergebenen Radius ein Puffer um den ebenfalls übergebenen Ausgangspunkt der Suche erstellt und durch eine Contains-Abfrage nur Punkte innerhalb dieses Puffers abgefragt werden. Für die Suche nach bestimmten Kategorien wird der SQL-String um Angaben erweitert, die für die Suche nach bestimmten OSM-Schlüssel-Werte-Paaren notwendig sind. So wird zum Beispiel bei der Suche in der Kategorie „Alles ums Rad“ nur nach POIs gesucht, die sich aus den OSM-Tag-Schlüsseln „amenity“ oder „shop“ und den OSM-Tag-Werten „bicycle“, „bicycle_repair“, „bicycle_parking“ oder „bicycle_rental“ zusammensetzen. Auf diese Art und Weise können POIs mit bestimmten Tags auch mehreren

[9] proxy: Kommunikationsschnittstelle, die als Vermittler zwischen zwei Seiten auftritt

[10] GML: erlaubt die Übermittlung von Objekten mit Attributen, Relationen und Geometrien im Bereich von Geodaten

Kategorien angehören.

5.8.2.3 Verwaltung der Rückgabe

Das von der Datenbank zurückgelieferte Ergebnis wird in die XML-Struktur

```
<poi id='1'>
   <id>304320085</id>
   <name>Advena Hohenzollern</name>
   <wkt>POINT(8.058562 52.2725211)</wkt>
</poi>
```

transformiert und enthält somit u. a. sowohl die für die spätere Pop-up-Darstellung benötigte OSM-ID als auch die Geometrie eines jeden gefundenen Objektes. Dieses Ergebnis wird an die Methode `suchePOI()` zurückgeliefert und hier an die, durch einen Callback aufgerufene, Funktion `gefundenePOIs()` weitergereicht. Jene sorgt nun für die korrekte Darstellung des Ergebnisses auf dem Bildschirm des Benutzers. Zunächst wird, entsprechend der Kategorieauswahl des Nutzers, der Anzeigestil der POIs definiert, für „Cafés & Co“ also beispielsweise eine Kaffeetasse. Die Definition dieser verschiedenen Stile ist im Abschnitt 5.9.2.1 „Individuelle Darstellung von Geometrien“ beschrieben. Ist keine spezielle Kategorie, sondern „Alles“ ausgewählt, so wird ein neutrales Symbol gewählt. Für die kategoriespezifischen Symbole wurden externe Graphiken verwendet, die auf der Webseite `http://code.google.com/p/google-maps-icons/` angeboten werden. Diese stehen unter der „GNU General Public License“ bzw. der „Creative Commons Attribution-Share Alike 3.0“ und dürfen somit frei verwendet werden.
Ist der Anzeigestil definiert, wird die XML-Rückgabe analysiert und die hieraus entnommenen Punkte durch Aufruf der Funktion `ausgabe_poi()` anzeigt. Besteht das Ergebnis aus nur einem Punkt, so wird die Karte mithilfe einer OpenLayers-Funktion nach der Erstellung der Geometrie zur Bedienungserleichterung direkt auf diesen Punkt zentriert (vgl. Abschnitt 5.9.2.3). Nebenbei wird der Name jedes Punktes als Link in einem String gespeichert, der am Ende eine Liste repräsentiert, die im Informationsfenster ausgegeben wird. Diese Liste gibt einen Überblick über die gefundenen Ergebnisse und dient gleichzeitig als Auswahlmöglichkeit für den Benutzer, der nur noch auf den gewünschten Namen klicken muss, um auf den gewünschten POI zu zoomen.
Die nun in der Karte dargestellten Ergebnisse kann der Anwender anklicken, um ein Popup-Fenster mit weiteren Informationen über den POI zu öffnen.

5.8.3 Visualisierung der Pop-ups

Der nächste Schritt ist es, die Punkte, die nun in OpenLayers dargestellt werden, mit einem Informationsfenster zu verbinden, sodass, sobald der Benutzer einen Klick auf einen Punkt tätigt (s. Kapitel 5.9.2.4 „Auswahl von Features“), ein Fenster erscheint, welches Informationen zu diesem Punkt enthält. Die Integration der Informationsfenster geschieht durch die von OpenLayers bereitgestellte Klasse *OpenLayers.Popup*. Ein Listener wird an den POI-Layer gehängt, der auf Mausklicks des Benutzers reagiert und beim Selektieren eines POIs die entsprechende Methode zum Anzeigen des Informationsfensters aufruft. In dieser Methode wird hierbei die von der POI-Suche bereits mitgelieferte OSM-ID abgerufen, die dem Feature als Attribut zugewiesen wurde. Anhand der ID werden in der nun aufgerufenen PHP-Datei alle zu dem POI in der Datenbank gespeicherten Informationen abgerufen und mit HTML Inhalten inklusive CSS-Style-Elementen aufbereitet, das anschließend im Informationsfenster ausgegeben werden kann (s. Abb. 49). Die Datenbank enthält ausführliche Informationen zu bestimmten POIs aus dem Bereich Gastronomie und Fahrraddienstleistungen, darunter z. B. Adresse, Beschreibung, Kapazitäten und Preise. Sind zu der übergebenen OSM-ID keine Informationen in der speziell angelegten Datenbank vorhanden, so wird einfach der in OSM hinterlegte Namen und die Kategorie im Fenster ausgegeben.

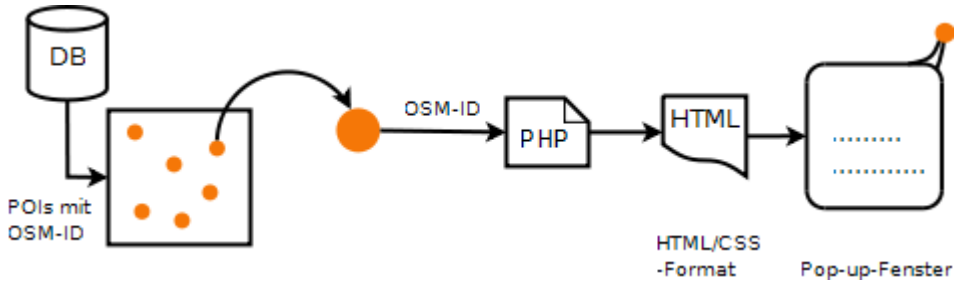

Abbildung 49: *Schema der Pop-up-Darstellung*
Quelle: Eigene Darstellung

5.9 Allgemeine OpenLayers-Methoden

Das Ziel dieses Projektes ist es, zusätzlich zur grundlegenden Routingfunktion, eine intuitiv bedienbare Seite zu erstellen. Mit nur wenig Aufwand und Einarbeitungszeit bzw. wenigen Schritten soll es den Nutzern möglich sein, Routen zu berechnen.

5.9.1 Markerbedienung

In Routenplanern ist häufig zu beobachten, dass zu viele bzw. komplizierte Schritte bis zum Definieren einer Start- oder Zielposition nötig sind. Oft ist es so, dass Aktionen wie z.B. die der Kartennavigation oder Start- bzw. Zieleingabe nicht zur selben Zeit möglich sind, da das eine das andere zunächst deaktiviert. Eine konkrete Auswahl, die das Gewünschte aktiviert oder deaktiviert, ist in solchen Fällen wichtig, aber unschön.. Darüber hinaus fehlt immer

wieder die nützliche Fähigkeit, eine zuvor gewählte Position durch Verschieben zu ändern, um so etwa eine Route ganz praktisch und unschwer zu verändern.

5.9.1.1 Zieldefinitionen

Zwecks Nachteilsreduzierung dieser genannten Punkte, d.h um eine intuitive Bedienbarkeit herzustellen, sind die nächstfolgenden Aufgaben zu erreichen.

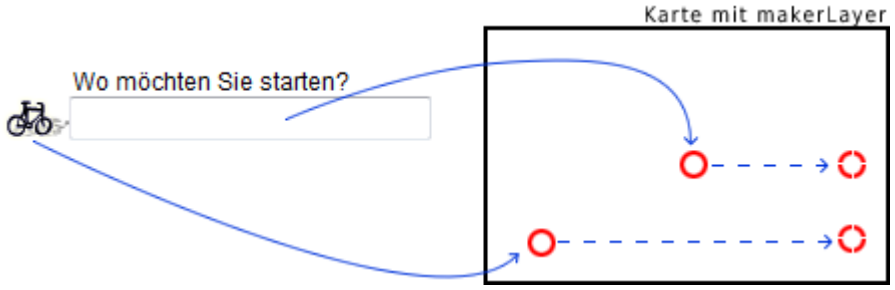

Abbildung 50: *Marker setzen durch Klick in die Karte mit anschließender Verschiebemöglichkeit*
Quelle: Eigene Darstellung

- Bei einem Klick ins leere Start- oder Zieltextfeld, erscheint das jeweilige Icon auf dem Mittelpunkt der sichtbaren Karte. Ein anschließendes Herumschieben dieses Objektes ist problemlos möglich.
- Dasselbe gilt für 2 existente Marker in der Karte. Beide sind durch einfaches Ziehen mit der Maus beliebig hin -und herfahrbar. Nachdem eines dieser beiden Positionsobjekte verschoben wurde, führt es zur automatischen Routenberechnung zwischen diesen beiden Punkten. Navigieren in der Karte ist ebenfalls ohne vorheriges Klicken einer Extra-Option möglich.
- Zusätzlich zum Hinzufügen eines Markers mittels Textfeld kann auch ein Hinzufügen durch einen Radiobutton (hier: das graue Fahrrad) angewandt werden.
- Die Möglichkeit zum Löschen eines Marker durch die rechte Maustaste

5.9.1.2 Realisierung

Möglichkeiten der Realisierung des Handlings bietet OpenLayers viele. Diverse Controller können Objekten verschiedene Fähigkeiten verleihen. So beispielsweise auch die Click-Controller-Klasse, die in die Karte gesetzte Klicks jeglicher Art verwaltet. Einmal erstellt, fängt sie alle Klicks ab und reagiert wie durch den Programmierer gewünscht. Die angeklickte Bildschirmposition kann in die für die Karte eingestellten, projizierten Koordinaten umgewandelt werden. Mit diesen Koordinaten wird weitergearbeitet. Sie werden an eine Methode weitergeleitet, die daraus ein Punktfeature mit einer Grafik erstellt und in der Karte dargestellt. Zwecks der Unterscheidung verschiedener Marker werden diesen manuelle IDs verliehen, was

vor allem beim Start- und Zielmarker wichtig ist, da sie die Start- und Zielposition festlegen und somit eine Richtung besitzen, die unter anderem für das Höhenprofil essentiell ist. Ein Doppelklick oder eine gedrückte linke Maustaste zum panen in die Karte werden hier von Click-Handler nicht behandelt, d.h. sie behalten ihre ursprüngliche Funktion.

Ein Beispiel für einen die linke Maustaste abfangende Funktion:
Der Default-Konstruktor lautet: OpenLayers.Class(Klasse1,[Klasse2],Prototyp); Klasse1 ist die Mutterklasse und der Prototyp die zu erstellende neue Klasse. Die Angabe weiterer Klassen ist optional.

```
1  OpenLayers.Control.Click = OpenLayers.Class(OpenLayers.Control, {
2     defaultHandlerOptions: {
3               'single': true,
4               'double': false,
5     },
6     initialize: function(options){
7      ...
8           this.handler = new OpenLayers.Handler.Click(this, {
9                'click': this.trigger }, this.handlerOptions);
10    },
11    trigger: function(e){
12          Hier: Koordinaten der Klickposition an Methode
13          weitergeben, die aus denen ein Geometrieobjekt mit Icon
14          erzeugt.
15          }
16 }
```

Zunächst erfolgt die Erstellung einer neuen Klasse für einen Click-Controller, der bisher nicht vorhanden war. Innerhalb der Klasse werden auch schon die gewünschten Verhaltensweisen definiert. Durch einen Handler weiß die Click-Klasse, was zur jeweiligen Aktion auszuführen ist. Wenn ein einfacher Klick in die Karte zu verzeichnen ist, wird die innerhalb der Click-Klasse vorhandene Funktion “trigger“ aufgerufen, die dafür sorgt, dass die Koordinaten der aktuell angeklickten Kartenposition in Kartenkoordinaten an die nächste Funktion weitergibt. Diese Funktion erzeugt aus den übergebenen Koordinaten ein Punktfeature und lässt es auf der Karte sichtbar werden.

Radiobutton aktiv meint, dass zuvor die Option für einen einfachen Markerklick aktiviert wird, anhand dessen der Click-Controller bestimmen kann, welche User-Interaktion derzeit im Spiel ist.
Wie in Abbildung 51 zu sehen, wird dem markerLayer ein DragControl zugewiesen. Dieser Controller ist für die Verschiebbarkeit aller Vektorgeometrien auf dem Layer zuständig. Lie-

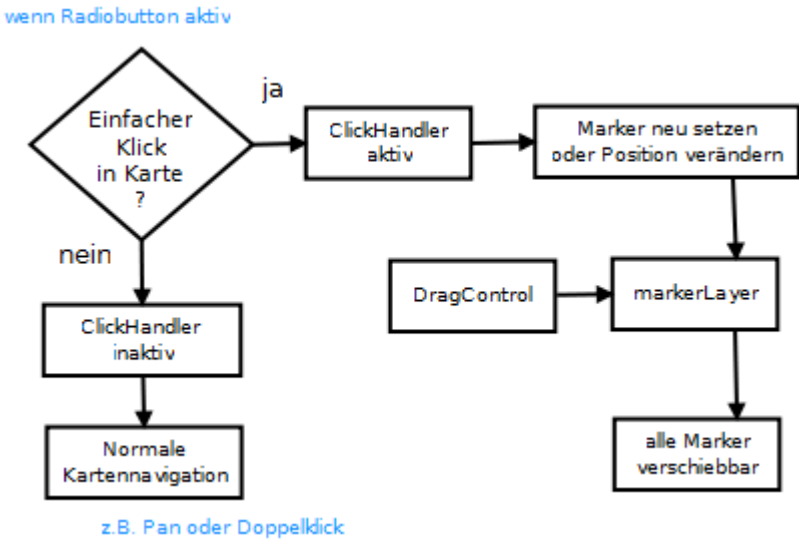

Abbildung 51: *Übersicht der mitwirkenden Komponenten bei der Markerbedienung*
Quelle: Eigene Darstellung

gen mindestens zwei Features auf dem Layer und wird einer von ihnen bewegt, führt die Aktion zur Neuberechnung und Darstellung der Route bzw. der Routenbeschreibung. Nachstehend ist ein Teil des Codes zu sehen.

```
dragFeature = new OpenLayers.Control.DragFeature(markerLayer, {
     'onComplete': function(evt){
          berechneRouteNeu();
      }
});
```

Hier ein auf dem Layer angewendetes Drag-Feature, das darüber hinaus die Eigenschaft besitzt, die Route nach dem Ende des Verschiebens zu berechnen.
Die Markerangelegenheit spielt sich insgesamt auf nur einem Layer ab: dem markerLayer. Jedes Objekt, das auf diesen Layer gelangt, ist automatisch verschiebbar. Infolge der Überprüfung, ob bspw. bereits ein Startobjekt im Feld ist, wird diesem bei jeder Erzeugung und Setzung in die Karte eine ID zugewiesen, hier z.B. “start“. Später im Code kann nach diesem Feature gefragt oder es kann, wenn nötig, gelöscht werden. Gelöscht wird, wenn der Startmarker schon im Feld ist. Es muss gelöscht und erneut ein Feature hinzugefügt werden, da die Koordinatenveränderung bzw. Veränderung der Markerposition z.B. durch eine erneute Straßensuche dazu führt, dass der in der Position geänderte Marker bei einer bestimmten Zoomstufe (beim hineinzoomen) verschwindet. Nicht nur eine voneinander verschiedene ID kann zugeordnet werden, sondern auch das Aussehen. Jedem Objekt kann vor dem Hinzufügen zum Layer ein individueller Style zugewiesen werden (Vergleiche Kapitel 5.9.2.1).
Wegen der mit einem Popup-Fenster (vgl. Kap. 5.8) ausgestatteten klickbaren Features, die der erzeugte POI-Layer enthält, ist diesem Layer ein SelectControl zwingend notwendig.

Normalerweise -als allgemein bekanntes Problem- schließt ein aktiver SelectControl ein DragControl aus, sofern sie zwei verschiedene Layer betreffen, was dazu führen würde, dass entweder die Verschiebefunktion oder die Auswahlfunktion von Features in der Karte deaktiviert würde. Nur eines ist zu einer Zeit möglich. Die Lösung für dieses bekannte Problem liegt darin, dass man den für das „dragging“ vorhergesehenen Layer mit zum SelectControl hinzufügt:

```
1 selectControl = new OpenLayers.Control.SelectFeature([markerLayer,poi
    ]);
2 map.addControl(selectControl);
3 selectControl.activate();
```

Die Verwirklichung der Benutzung der rechten Maustaste für ein Kartenobjekt beruht auf einem Tutorial, das mehrere Modifkationen der OpenLayers-Quelldateien zur Folge hat. Zu finden ist die Anleitung unter `http://www.cnblogs.com/lei3389/archive/2009/04/21/1440458.html`. Notwendig sind diese Schritte aufgrund des Nichtvorhandenseins einer solchen Funktion in OpenLayers. Ist die Fähigkeit hergestellt, muss ein “rechtes-Mausklick -Menü“ erscheinen. Eine schnelle Lösung dafür bieten die bereits in OpenLayers integrierten Pop-up-Funktionen. Es ist also ähnlich wie bei den Pop-ups in der POI-Suche realisiert, vergleiche dazu Kapitel 5.8.3.

5.9.2 Weitere OpenLayers-Funktionen

Im Zentrum dieses Kapitels stehen die Definitionen von Verhaltenweisen bzw. Reaktionen diverser Kartenkomponenten sowie ihre visuelle Darstellung.

5.9.2.1 Individuelle Darstellung von Geometrien

Damit sämtliche Objekte, die in der Karte zu sehen sind, nicht das gleiche Aussehen mit sich tragen, muss für die individuelle Gestaltung dieser gesorgt werden. Die Notwendigkeit dieses Vorhabens resultiert z.B. aus der Betrachtung der beiden wichtigen Start- oder Zielmarkierer. Ohne einen deutlichen visuellen Unterschied bei der Grafik ist es für den Anwender schwierig, das Ende oder das Ziel voneinander zu trennen. Und es ist nicht unerheblich, wo sich die jeweiligen Marker befinden, wie im Beispiel des Profils *vermeide Steigungen über 3%* oder auch bei der Höhenprofilanzeige.
Denn beide berücksichtigen die Richtung der Route bzw. setzen sie für die erfolgreiche Darstellung voraus. Des weiteren ist die Gestaltung von Punktfeatures, die bei der POI-Suche zur Geltung kommt, nicht zu vernachlässigen, um diesen ein dem Gesamtdesign ähnliches Aussehen zu geben oder zur Repräsentierung ihrer Kategorie selbst.

1.)

```
1 var start_style = OpenLayers.Util.applyDefaults( {
2   externalGraphic : "./images/start3.png",
```

Abbildung 52: *Beispielhafte Symbologie von Markern und POI-Symbolen*
Quelle: Eigene Darstellung

```
3   graphicWidth : 45,
4   graphicHeight : 28,
5   graphicYOffset : -28,
6   graphicXOffset : -19,
7   graphicOpacity : 1
8    }, OpenLayers.Feature.Vector.style['default']);
```

2.

```
1 var markerLayer = new OpenLayers.Layer.Vector("Start/Stopp Marker");
```

3.

```
1 var punkt = new OpenLayers.Geometry.Point(x, y);
2 var startpunkt = new OpenLayers.Feature.Vector(punkt,null,start_style
      );
```

4.

```
1 markerLayer.addFeatures([startpunkt]);
```

Im ersten Schritt vollzieht sich eine Aussehensdefinition mit einer externen Grafik und die Angabe der Größenangaben. Im Folgenden entsteht ein Layer, in dem sich das Objekt befinden soll. Im dritten Schritt wird eine Geometrie erstellt, die daraufhin mit dem anfangs gewählten Äußeren ausgestattet wird. Anschließend kann dieses Feature dem Layer hinzugefügt werden.

5.9.2.2 Löschen von Features

Sind ein oder mehrere Elemente im Kartenfeld, müssen diese ggf. wieder entfernt werden. Dieser Fall tritt z.B. auf, wenn eine neue Route berechnet und die alte gelöscht wird. Oder aber ein Markerklick in die Karte, denn zieht das Löschen des alten Markers nach sich, um ihn an neuer Position zu zeichnen.
Zu diesen Zwecken können entweder alle Elemente des Layer entfernt werden:

```
1 markerLayer.removeFeatures(markerLayer.features);
```

`features` ist das Array, in dem sich alle in dem Layer dargestellten Objekte befinden.
Es lassen sich aber auch einzelne Features löschen:

```
1 markerLayer.removeFeatures(markerLayer.getFeatureById("start"));
```

In diesem Fall findet das Entfernen vom Marker mit der ID `start` statt.

5.9.2.3 Zoomen und Zentrieren

Eine weiteres interessantes Merkmal von OpenLayers ist das Zoomen auf einen bestimmten Bereich. Zur Implementierung werden in diesem Kapitel drei Varianten der Realisierung vorgestellt.
Zum Zoomen auf einen bestimmten, vom User definierten Bereich bedarf es einiger, manuell gesetzter Koordinaten:

```
1 map.zoomToExtent(new OpenLayers.Bounds(892147, 6846947, 899800,
    6852679));
```

Sie stellen ein Rechteck dar, dass durch die Eckpunkte [unten, links, rechts, oben] gekennzeichnet ist.
Ohne Eingabe gewünschter Koordinaten, dafür aber die ganze Karte betreffend, kann man auf ein eingegebenes Zoomlevel gebracht werden.

```
1 map.zoomTo(17);
```

In dieser Arbeit ist eine maximale Zoomstufe von 18 eingestellt, wobei der Maßstab der Karte in Abhängigkeit einer größeren Zahl größer wird.
Die dritte Möglichkeit besteht darin, aus einem Geometrieobjekt die Ausdehnung herauszusuchen und exakt auf diesem zu zoomen.

```
1     var objekt = OpenLayers.Geometry.fromWKT(a);
```

```
1     map.zoomToExtent(objekt.getBounds());
```

Nach zunächst erstellter Geometrie, etwa die einer Straße, erlaubt OpenLayers das Herauslesen des der Geometrie minimal umgebenden Rechtecks, entsprechend dessen der Maßstab der Karte angepasst wird.

5.9.2.4 Auswahl von Features

Bedeutend ist auch die Auswahl von Objekten, bei deren Anklicken ein Prozess in Gang gesetzt wird. Hier soll ein “Zuhörer“ an ein Objekt gebunden werden, bei dessen Aktivierung, etwa einem Klick auf ein in der Karte zu sehendes Element, weitere Aktionen nach sich gezogen werden. Zu diesem Zwecke existieren dem Kartenklient zugrunde liegende SelectControls, die bei dem Aufruf der Seite oder dem Suchen von POIs (siehe Kapitel 5.8) aktiviert werden:

```
1 selectControl=new OpenLayers.Control.SelectFeature([irgendeinLayer]);
2 map.addControl(selectControl);
3 selectControl.activate();
```

Nach der Aktivierung des Controllers, bedarf es der Registrierung an einem Layer, der von nun an stets unter Beobachtung des selectControl steht:

```
irgendeinLayer.events.on({
    "featureselected": onFeatureSelect,
    "featureunselected": OnFeatureUnselect
});
```

Jeder Klick auf ein Feature dieses Layers spricht die eingegebene Methode `onFeatureSelect` an, die für weitergehende Operationen verantwortlich ist. Sie bekommt einen Parameter namens `event` mit, der durch `event.feature` die Repräsentation des angeklickten Objektes darstellt:

```
function onFeatureSelect(event) {
  var feature = event.feature;
  selectedFeature = feature;
  machwasmitdemFeature
}
```

Nun steht es dem Programmierer frei, alle Attribute oder Methoden dieses Features anzusprechen und somit z.B. die ID des Elementes herauszufinden und weiterzugeben.
Ein selektiertes Objekt kann auch wieder deselektiert werden. In diesem Fall greift `featureunselected` ein und ruft die Methode `onPopupClose` auf. W

```
function onFeatureUnselect(event) {
  selectControl.unselectAll();
}
elche Methoden bei den Pop-ups benötigt werden, wird im Folgenden
   näher erläutert.
```

5.9.2.5 Erstellen von Pop-ups

Bei dem Pop-up-Feature erscheint ein Fenster, das Informationen über den gewählten POI enthält (Näheres zur Verwendung im Kapitel 5.8). Zur Realisierungshilfe bietet OpenLayers interne Funktionen an.
Die oben genannte Variable `machwasmitdemFeature` wird durch den Aufruf

```
OpenLayers.loadURL("popup.php", "?id=" + selectedFeature.id,
   null, createPopup, null);
```

ersetzt, die eine PHP-Datei aufruft und ihr die dem POI-Objekt zuvor durch `poi.id = poiOsmID` zugewiesene OSM-ID, des ausgewählten Features übergibt und nach derem Durchlauf die Methode createPopup aufruft.

```
function createPopup(response){
   result = response.responseText;
   popup = new OpenLayers.Popup.FramedCloud("Information",
      selectedFeature.geometry.getBounds().getCenterLonLat(),
      null, result, null, true, onPopupClose);

   popup.panMapIfOutOfView = true;
   popup.contentDiv.style.overflow = "auto";

   map.addPopup(popup);
}
```

Diese Methode ist für die Darstellung des zurückgelieferten Ergebnisses der PHP-Datei zuständig. Sie erstellt ein Popup-Objekt, das die aus der Datenbank gewonnen Informationen des POIs übergeben bekommt. Sollten die Dimensionen des Informationsfenster den Kartenausschnitt überragen, so wird dafür gesorgt, dass dieser sich auf die Ausmaße des Pop-up-Fensters verschiebt. Anschließend wird das Fenster der Karte hinzugefügt.

5.9.2.6 Ajax in OpenLayers

Ein für dieses Projekt sehr wichtiges Vorgehen besteht in der Kommunikation zwischen den verschiedenen Programmier- und Skriptsprachen auf Client -und Serverebenen. Das Routing schickt eine den Positionen von mindestens zwei gesetzten Punkten in der Karte entsprechende Berechnungsanfrage an eine PHP-Datei. Nach der Abarbeitung diverser Schritte und Zugriff auf die Datenbank, erfolgt eine Rückgabe des Ergebnisses, das wiederum von dem Clienten verarbeitet werden muss. Das Zusammenspiel verschiedener Techniken wird ermöglicht durch native OpenLayers Ajax-Funktionen. Daneben gibt es eine Reihe weiterer Komponenten, welche dieses Verfahren benötigen. So auch etwa das Höhenprofil, das mit den Kantengeometrien in WKT-Form beliefert werden muss. In diesem Fall bedeutet dies, dass Übergabelänge bzw.-größe der Daten unbegrenzt sein muss, wofür sich die normalen GET- und POST-Methoden nicht eignen.

```
OpenLayers.Request.POST({
    url: : 'hoehenprofil.php',
    asynch: true,
    data: '&param1=' + zB_Geometrien + '&param2=weitereInfos',
    headers: {
    "Content-Type": "application/x-www-form-urlencoded"
    },
    callback: callbackFunction
});
```

Aus oben genannten Gründen wird eine Übergabeform gewählt, die beliebige Parameterlängen erlaubt. Zuletzt genannter Code zeigt eine in dieser Arbeit typisch verwendete Ajax-Serveranfrage. Mit `url` ist die Zieldatei dieser Anfrage angegeben, `asynch` erlaubt die asynchrone Abarbeitung, sodass der Client an dieser Stelle nicht unbedingt auf Server warten muss. `Data` in Kombination mit den in `headers` stehenden Informationen erlaubt uneingeschränkte Größe der Übergabeparameter. Letztendlich meldet sich `callback`, wenn sie eine erfolgreiche Rückgabe der Serverdatei verzeichnen kann und ruft eine diese Rückgabe weiterverarbeitende Funktion auf.

5.10 Tracklog-Export

In Zeiten großer Verbreitung von Smartphones, leistungsfähiger Handys und GPS Geräten ist es für Routenplaner ein Muss geworden, die erzeugten Routen auch für die Mitnahme ins Gelände bereitzustellen. Das von der Firma Topografix entwickelte GPS Exchange Format (GPX) ist das offene und lizenzfreie Standardformat für diesen Zweck [Fir]. Es basiert auf XML und bietet die Möglichkeit Wegpunkte, Routen und Tracklogs zu speichern, auszutauschen und auf dem mobilen Endgerät darzustellen.

Die ersten Schritte der **Arbeitsweise des Exports der GPX-Tracks** sind analog zum Höhenprofil (vgl. Kap. 5.5), da die Route zuerst aufbereitet werden muss. Wie in Kapitel 5.5.1 beschrieben, muss auch hier zunächst die korrekte Reihenfolge der Stützpunkte hergestellt werden. Der GPX-Track bildet die Route in einer bestimmten Reihenfolge in einer Datei mit der Endung *.gpx* ab. Die korrekt sortierte Route wird daraufhin an die Datei `gpx.php` weitergegeben.
Die Route wird nach dem im der Abbildung 53 dargestellten Schema weiterverarbeitet. Der gesamte Inhalt der Route wird in der XML-Rückgabe vom Tag `<trk></trk>` (für Track) umschlossen. Der Schritt, die Route in Kanten aufzuteilen, ist zum Zeitpunkt der Übergabe an die Datei `gpx.php` bereits geschehen. Eine Kante wird durch das Tag `<trkseg></trkseg>` ausgezeichnet, also einem Segment des Tracks. Die Kanten werden der Reihe nach durchlaufen und in ihre Stützpunkte eingeteilt. Jeder dieser Stützpunkte ist wiederum durch das Tag `<trkpt lat=... lon=...></trkpt>` eingeschlossen und enthält die Koordinaten des Punktes im Bezugssystem WGS84. Wichtig dabei ist, dass die korrekte Reihenfolge der Stützpunkte eingehalten wird, so wie sie in der Route angefahren werden. Die Stützpunkte stellen die Route als Track in der Reihenfolge dar, in der sie im GPX-Track vorhanden sind. Dies ist bereits dadurch sichergestellt, dass die Route mit Hilfe der Kantensortierung vorverarbeitet wurde und innerhalb der Datei `gpx.php` nicht wieder durcheinander geworfen werden darf.

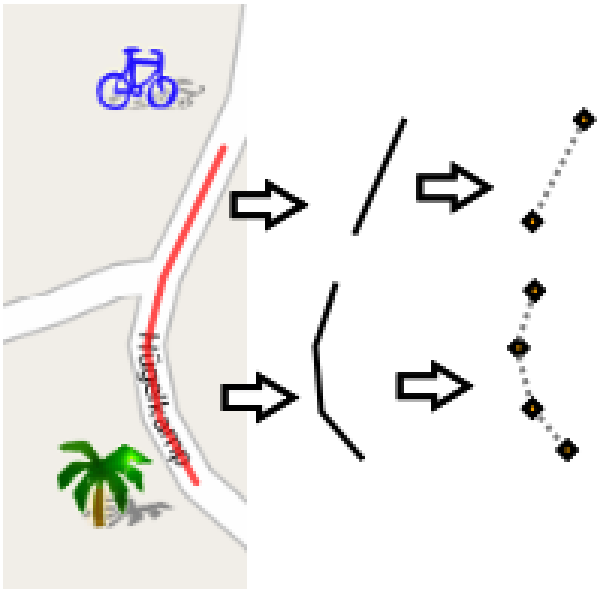

Abbildung 53: *Route wird in Kanten aufgeteilt, welche wiederum in ihre Stützpunkte aufgeteilt werden*

Quelle: Eigene Darstellung

5.11 Themenrouten

Regionale Radrouten sowie überregionale Radrouten, die durch den Landkreis Osnabrück führen, sind Anlaufpunkte für Radtouren und zudem oft beschildert. Der User soll auch durch den im Rahmen dieses Projektes entwickelten Radroutenplaner die Möglichkeit haben, sich diese Routen als vordefinierte Themenrouten anzeigen zu lassen. Der Benutzer kann sich daraufhin bei der Routenplanung an der Themenroute orientieren.

5.11.1 Einpflegen der Routen in die OpenStreetMap-Datenbank

Die Themenrouten wurden, wenn nicht bereits vorhanden, in den OpenStreetMap-Datenbestand eingepflegt. Da im Rahmen dieses Projektes nur der Landkreis Osnabrück von Bedeutung war, wurden vorher nicht vorhandene Radrouten, wie z. B. der Brückenradweg Osnabrück-Bremen Ost nur innerhalb des relevanten Bereiches aufgenommen. OpenStreetMap macht es möglich, diese Routen zu definieren, indem die zu einer Themenroute gehörigen Straßen einer gemeinsamen `relation` (vgl. Kap. 2.2.3) hinzugefügt werden. Die `relation` besitzt anschließend eine eigene OSM-ID. Die Informationen zu den Themenrouten wurden im Vorfeld im Internet und über Radwanderkarten recherchiert, jedoch waren nicht für jede existierende Themenroute ausreichend Informationen (vor allem über den Verlauf) vorhanden. Beispielsweise führt die Schlösser-Tour durch Gebiete des Landkreises, welche in OpenStreetMap bisher nur unzureichend gemapped sind. Es wurden daraufhin nur solche Routen gewählt, die zu großen Teilen durch zentrale Gebiete des Landkreises führen und nicht nur durch Randgebiete. So stehen letztlich zwölf Themenrouten zur Auswahl:

- Bahnradroute Teuto-Senne
- Brückenradweg Osnabrück-Bremen Ost
- Brückenradweg Osnabrück-Bremen West
- COLLOSAL-Tour
- Düte-Tour
- Friedensroute Osnabrück-Münster Ost
- Friedensroute Osnabrück-Münster West
- Gartentraum-Tour
- Hase-Ems-Tour
- Niedersächsiche Mühlen-Tour
- NordWestBahn-Tour
- Osnabrücker Rund-Tour

Um den Usern, die mit der Region nicht vertraut sind (beispielsweise Touristen) die Themenrouten ebenfalls nahe zu bringen, wurden noch zusätzliche, für Radfahrer interessante Informationen recherchiert:

- die Gesamtlänge der Route (über den Landkreis hinaus) in Kilometern
- der Themenschwerpunkt
- die Orte, durch welche die Themenroute verläuft
- Informationen über die Landschaftskulissen, durch welche die Themenoute führt
- der Anspruch: für wen die Route geeignet ist, vor allem im Hinblick auf Steigungen
- den GPX-Track, wenn vorhanden.

Mit Ausnahme der GPX-Tracks werden Informationen in einer Tabelle in einer lokalen Datenbank unter Zuweisung der dazugehörigen OSM-ID abgespeichert. Die GPX-Tracks liegen als .zip-Dateien vor und werden bei Vorhandensein als Link angeboten. Die Informationen sollen zusammen mit der Route angezeigt werden. Entsprechende OSM-Tags für die genannten Eigenschaften existieren nicht oder sind unbrauchbar, weswegen an dieser Stelle Informationen lokal abgespeichert werden.

5.11.2 Funktionsweise

Hinter der Auswahl einer Themenroute durch einen Klick auf den entsprechenden Namen verbirgt sich die OSM-ID der dazugehörigen Relation. Diese wird an die Datei `themenrouten.php` weitergegeben. Der folgende Pseudocode stellt das Muster dar, nach welchem diese Anfragen an die Datenbank stellt:

```
1 Hole zur Relation alle Bestandteile, aus denen sie besteht.
2 Für jeden Realtionsbestandteil:
3   Hole die jeweiligen Kanten, aus denen der Teil besteht.
4   Für jede Kante:
5     Hole zur Kante ihre Geometrie
6     und füge sie in die XML-Rückgabe ein.
7 Hole zusätzlich die weiteren Informationen zur Relation
8 und verpacke sie in die XML-Rückgabe.
9 Gebe die Informationen als XML an den Client zurück.
```

Während die Bestandteile der Relation aus der mit Osmosis eingelesenen Datenbank stammen, werden die Kanten der Relationsbestandteile aus der mit OSM2pgRouting erzeugten Datenbank abgefragt. Die Informationen zu den Themenrouten liegen in einer dritten Datenbank.
Nach der serverseitigen Verarbeitung durch die Datei `themenrouten.php` wird die XML-Rückgabe clientseitig geparsed. Die XML-Struktur ist bis auf die zusätzlichen Informationen zu den Themenrouten dieselbe wie beim Routing (vgl. Kap. 5.1.5). Die Geometrien werden anschließend durch OpenLayers in der Karte angezeigt, die Informationen zur Themenroute im Informationsfenster.
Die gewählte Implementation ist aus zeitlichen Gründen auf die bloße Darstellung der Themenrouten beschränkt.

5.12 Druckfunktion

Ein Radroutenplaner im Internet ist zwar sehr hilfreich, jedoch lässt sich der Nutzen für den Anwender stark erhöhen, wenn die Ergebnisse der Online-Planung auch auf eine Fahrradtour mitgenommen werden können. Eine Möglichkeit, diese Ergebnisse auch „live“ zur Verfügung zu haben, wäre der Aufruf der Webseite mit internetfähigen mobilen Endgeräten. Da die Webseite jedoch nicht für mobile Endgeräte ausgelegt ist, werden die meisten dieser tragbaren Geräte den größten Teil der Routingfunktionen nicht nutzen können. Mit vielen PDAs oder Handys ist nur die bloße Ansicht der Seite möglich, alle weiteren Funktionalitäten sind nur eingeschränkt oder gar nicht nutzbar. Vorteile bietet aus diesen und anderen Gründen eine analoge Form der Karte mit der Routendarstellung. Daher ist eine Druckfunktion unerlässlich!
Die praktische Umsetzung der Druckfunktion stellt den Programmierer zunächst vor ein Problem, da die Nutzung des Kartenklienten OpenLayers ein problemloses Drucken unterbindet.

Sofern man den aktuell angezeigten Kartenausschnit mit den Überlagerungen aller eingeschalteten Layer ausdrucken möchte, ist das fehlerfreie Drucken nur abhängig vom Browser und vom Druckermodell möglich. Erfolg gibt es nur mit dem Opensource-Browser Firefox, der die Darstellung der auf den Overlay gezeigten Vektoren mit der SVG-Technik [11] realisiert. Um das Drucken browserübergreifend zu ermöglichen, muss eine andere Herangehensweise gewählt werden. Eine hervorragende Art und Weise, dies zu erreichen ist Mapfish, der zahlreiche Erweiterungen von OpenLayers bietet. Hiermit ist ein gewünschter Bereich mit allen darauf zu sehenden Objekten in verschiedenen, vordefinierten beispielsweise mit einer Legende versehenen Layouts als PDF exportierbar.

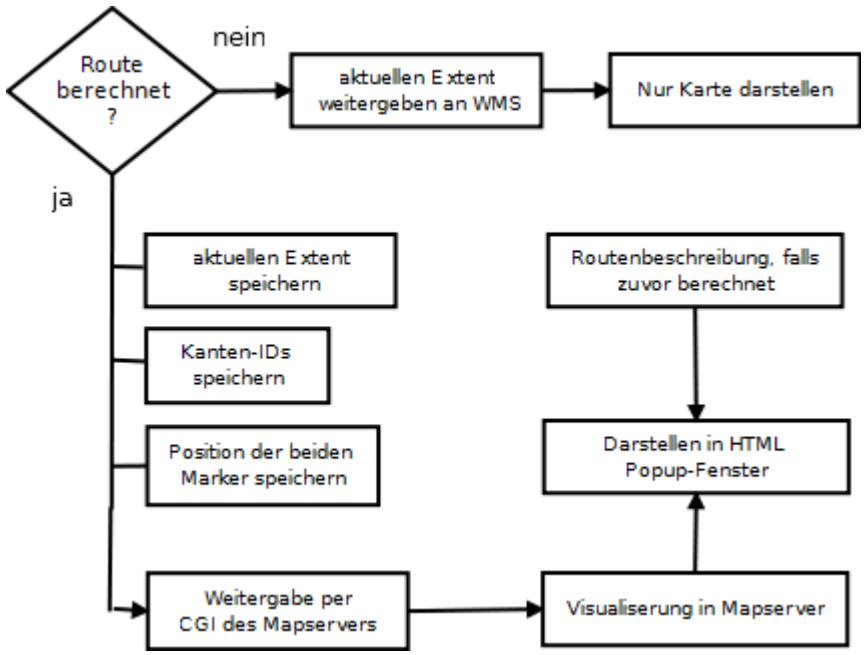

Abbildung 54: *Erzeugen einer Druckvorschau*
Quelle: Eigene Darstellung

Doch die umständliche und zeitaufwendige Einrichtung ist zum aktuellen Zeitpunkt ein Grund sich gegen den Einsatz des speziellen Kartenklienten zu entscheiden.
Die Alternative ist ein Web Mapping Service (vgl. Kap. 2.7), dessen zrückgelieferter Inhalt in jedem internetfähigen Browser dargestellt und von jedem Drucker korrekt gedruckt werden kann, weil der WMS eine entsprechende Bilddatei zur Anfrage liefert. Es ist aber gleich darauf hinzuweisen, dass hierbei aufgrund nichtvorhandener Mapnik-WMSe nur eine Annäherung an die originale Darstellung des in OpenLayers Sichtbaren entsteht.
Sobald die Route berechnet und in OpenLayers angezeigt ist, kann diese Route mit Hilfe von den Kantengeometrien referenzierenden Kanten-IDs, die zuvor gespeichert werden, in Form eines Line-Layer im Mapserver dargestellt werden. Die Hintergrundkarte besteht aus einem externen Openstreetmap-WMS von Omniscale, damit der Style des in OpenLayers benutzten

[11] SVG: Scalable Vector Graphics, Spezifikation zur Darstellung von zweidimensionalen Vektrografiken [W3Cc]

Mapnik-Renderers am besten approximiert wird.
Der für die Streckendarstellung zuständige Layer im Mapserver setzt sich wie folgt zusammen:

```
LAYER
   SYMBOLSCALE 50000

   NAME          'zwischenstrecke'
   TYPE          LINE
   STATUS        DEFAULT
   OPACITY 60
   SYMBOLSCALEDENOM 500000

   CONNECTIONTYPE postgis
   CONNECTION "user=studadm password=P4Knv8 dbname=routing_landkreis
               host=localhost port=5432"
   DATA "geom from (
          select st_setsrid(w1.the_geom,900913) as geom, gid
          from ways w1 where gid in (%anfrage%)
         ) as a using unique gid using srid=900913"

   CLASSITEM     'gid'

   DUMP TRUE
   CLASS
       STYLE
           WIDTH 1.2
           MINSIZE 1.2
           MAXWIDTH 3
           COLOR  0 0 255
           ANTIALIAS TRUE
       END
    END
 END\\
```

Nachdem die Verbindung zur PostgreSQL-Datenbank hergestellt ist, erfolgt eine Anfrage, welche alle Geometrien heraussucht, die zu den übergebenen KantenIDs zugehörig sind. Der Platzhalter für die per URL mitgeteilten Identifikationsnummern ist `%anfrage%`. Anschließend werden alle Geometrien durch Antialiasing[12] so weit es geht ohne „Treppeneffekte“ im Layer dargestellt. Das gleiche Verfahren gilt auch für die anderen Layer, wie z.B.für die Marker verantwortlichen. Im Allgemeinen ist die vom Standard abweichende SQL-Statement-Angabe[13] zu beachten, die mit *geom from* anfängt und mit *as a using unique gid using srid=900913*

[12]Methode zur Kantenglättung
[13]Postgis-Verbindung im Mapserver: `http://mapserver.org/input/vector/postgis.html`

endet, wobei eine id, hier `gid`, innerhalb der nach `geom from` folgenden Klammer auftauchen muss.

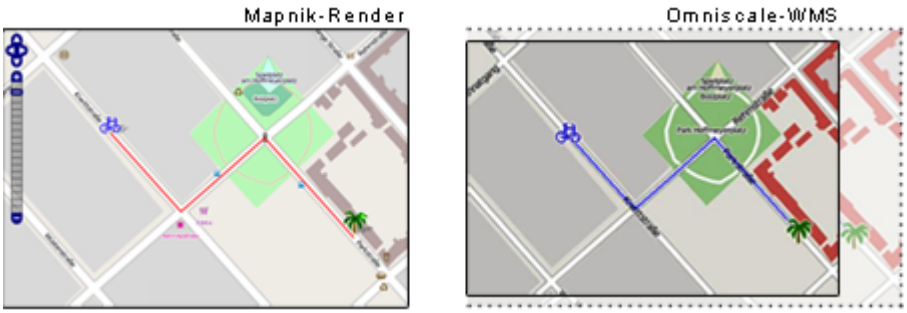

Abbildung 55: *Vergleich von Originaldarstellung mit der auf eine bestimmte Größe gebrachte WMS-Darstellung*

Quelle: Eigene Darstellung

Um eine bestmögliche bzw. scharf aufgelöste Darstellung des angeforderten Rasterbildes der Hintergrundkarte zu garantieren, muss das Verhältnis von Breite bzw. Höhe des Bildes zu den Bounding-Box-Koordinaten(BBOX, Rechteckfenster, in der die Karte dargestellt ist) stimmen. Dies ist jenes Verhältnis, das im aktuell angezeigten OpenLayers-Fenster zu sehen ist. Da aber die Webseite so konzipiert ist, dass sie sich beim ersten Aufruf an der Bildschirmbreite bzw. Höhe anpasst, kann es infolge einer vom Standard abweichenden Bildschirmauflösung sein, dass z.B. die Breite ein ungeeignetes Format für die Druckausgabe vorlegt und das Druck-Bild somit über die Grenzen des Blattes hinausragt. Die meisten Betriebssysteme haben als Druckstandard das Hochformat eingestellt, an welchem sich die Breite des Kartenbildes orientieren soll. Dafür werden 600Pixel gewählt, zu dem die Berechnung der Höhe durch einfache mathematische Länge-Breite-Beziehungen aus dem ermittelten, urünglichen OpenLayers-Extent erfolgt. Ein weiteres Problem ergibt sich: Zu der manuell gesetzten, d.h. festen Breite und der dazu automatisch berechneten Höhe, passen die Ursprungskoordinaten - die BBOX, das ebenfalls ein spezifisches Verhältnis von Länge zu Breite besitzt - nicht mehr. Es kann vorkommen, dass die anhand der BBOX angeforderte Karte im Pixelfenster einer “Quetschung“ oder aber auch einer “Dehnung“ unterliegt. Ersteres ist in Abbildung 55 zu sehen, denn hier findet eine Skalierung der urpsprünglichen Größe (transparentes Bild) in ein kleineres Rechteck statt. In der Praxis kommen diese Fälle aufgrund der häufig benutzten, eher quadratischen Auflösungen von 1280x1024 oder auch 1024x768px nicht vor, sodass gute Ergebnisse zu verzeichnen sind und damit auch komplexere Verhältnisberechnungen (z.B. Pixel-Koordinatenumrechnung und Weglassen bestimmter BBOX-Bereiche) vermieden werden.
Im Folgenden ist ein vollständiger Aufruf zu sehen, dessen Antwortbild in einem HTML-Pop-up integriert wird: `http://igf-project.igf.uos.de/cgi-bin/mapserv?map=/home/studienprojekt/public_html/druckwms/print.map` mit den Parametern:

```
&VERSION=1.1.1
&REQUEST = GetMap
&SRS = EPSG:900913
&BBOX = aktuellerOpenLayersExtent
&layers = osm
&WIDTH = Breite
&HEIGHT = Höhe
&FORMAT = image / jpeg
&SERVICE = wms
&anfrage = KantenIDs
&start_x / start_y = Startpunkt
&ende_x / ende_y = Endpunkt
&erstes / letztes = erste und letzte Kante in WKT-Form
```

Die erste und letzte Kante müssen mit übergeben werden, weil sie spezielle Geometrien sind, die mit einer normalen ID nicht aus der Datenbank geholt werden können. Es sind vielmehr Teile von Kanten, die in der Routingdatei zur jeweiligen Markerposition abgeschnitten wurden, um nicht immer die komplette Linie anzeigen zu müssen.
Der Wert GetMap des Parameters REQUEST sorgt dafür, dass eine Rasterdatei als Antwort zurückgeliefert wird, deren Visualisierung das HTML-Tag ¡img src=“Bild“¿ übernimmt. Zur Kartendarstellung inklusive der Routingkomponenten wird ggf. auch die Routenbeschreibung beigefügt.
`document.getElementById('routenbeschreibung').innerHTML` ist eine simple Javascript-Funktion, die sich die Realisierung des Letztgenannten zur Aufgabe nimmt. Damit lässt sich der aktuelle Inhalt des aktuellen DIV-Containers - in diesem Fall die auf der Webseite angezeigte Routenbeschreibung - erfragen und in dem Popup-Fenster inkludieren.

5.13 Routenbewertung

Mit der Funktion „Route bewerten“ ist es dem Nutzer des Radroutenplaners möglich, eine Bewertung für eine bestimmte Route abzugeben. Wie in Abbildung 56 zu sehen ist, kann die Route hinsichtlich der Kriterien Sicherheit, Attraktivität und Familienfreundlichkeit entsprechend dem Schulnotensystem von „sehr gut“ bis „ungenügend“ bewertet werden. Auf diese Weise können mehrere Nutzer miteinander interagieren, indem sie durch die Bewertung ihrer Route diese weiterempfehlen bzw. davon abraten. Nachdem der Nutzer seine Route hinsichtlich aller Kriterien bewertet hat, wird diese, nach einem Klick auf den Button „Route bewerten“, als Link angezeigt, der die ID der Route enthält. Dieser Link kann an andere Personen weitergegeben werden.
Darüber hinaus besteht für den Nutzer die Möglichkeit sich die am besten bewerteten Routen anzeigen zu lassen. Beim Klick auf den Button „Beste Routen“ werden für zehn Routen die

Abbildung 56: *Benutzeroberfläche Routenbewertung*
Quelle: Eigene Darstellung

jeweilige ID der Route, ihre Bewertung, dargestellt als Sterne, und die Anzahl der Bewertungen angezeigt. Die Sterne entsprechen den in der Bewertung vergebenen Noten. Null Sterne bedeuten „ungenügend“ und die maximale Zahl von fünf Sternen steht für „sehr gut“. Der Nutzer kann sich die bewertete Route durch einen Klick auf die ID in einem neuen Fenster anzeigen lassen.
Zur Zeit befindet sich die Routenbewertung noch in der Entwicklung. Es sind Optimierungen notwendig, da beispielsweise jede Route in der Liste der besten Routen erscheint, sofern sie einmal bewertet wurde und zu den zehn besten gehört.
Die Routen werden in eine Tabelle der Datenbank geschrieben und jeweils zugehörigen Bewertungen in eine zweite Tabelle. Bewertet ein weiterer Nutzer dieselbe Route, soll diese nicht erneut mit einer neuen ID gespeichert werden, sondern lediglich eine neue Bewertung hinzugefügt werden. Alle für eine Route abgegebenen Bewertungen sollen anschließend gemittelt werden. Geplant ist, dass eine Route erst dann in der Liste angezeigt wird, wenn sie mindestens fünfmal bewertet wurde.
Da die Routenbewertung eine wichtige Funktion im Hinblick auf Interaktivität und Web 2.0-Features darstellt, soll sie so schnell wie möglich fertiggestellt werden.

6 Öffentlichkeitsarbeit

Nachdem im vorherigen Kapitel die Implementierung der einzelnen Funktionen beschrieben worden ist, die in der Gesamtheit einen lauffähigen Radroutenplaner auf der Webseite `http://www.fahrradies.net` ergeben, ist es nun wichtig Öffentlichkeitsarbeit zu betreiben. Ziel ist es, den Radroutenplaner bei der Allgemeinheit bekannt zu machen und für diesen zu werben.
Werbung bedeutet, den potenziellen Nutzern des Fahrradroutenplaners, diesen so attraktiv wie möglich zu präsentieren und dabei Lust auf "mehr" zu machen. Da der Planer unter `http://www.fahrradies.net` nach Abschluss der Projektphase nicht ungenutzt bleiben soll, müssen verschiedene Maßnahmen ergriffen werden, um möglichst viele Nutzer zu gewinnen.

Nicht nur örtliche Zeitungen, sondern auch Radiosender und stadtbekannte Zeitschriften werden deshalb mehrere Male in Kenntnis gesetzt. Eine große Rolle spielen dabei schon zu Anfang die zweimal wöchentlich erscheinenden Osnabrücker Nachrichten [ON] und auch das vierteljährlich herauskommende Kettenblatt des Allgemeinen Deutschen Fahrradclubs e.V. [ADF]. Bei beiden Institutionen wird die Idee des Radroutenplaners mit großem Interesse aufgenommen, sodass schon in der Mittwochsausgabe der ON vom 09.11.2009 ein großer Artikel mit Foto erscheint. Ebenso gibt es in der Dezemberausgabe des Kettenblatts (s. Anhang) einen zweiseitigen Informationstext über *fahrradies.net*. In diesem Artikel geht es hauptsächlich darum, darüber zu informieren, dass der Radroutenplaner im Frühjahr 2010 zu nutzen sein soll.
Im Februar und März 2010 kann schon stärker darauf aufmerksam gemacht werden, dass der Planer bald aktiv wird. Da durch das ab Frühlingsanfang stetig besser werdene Wetter wieder verstärkt auf das Fahrrad als Fortbewegungsmittel zurückgegriffen wird, ist es besonders wichtig, dass am Anfang der neuen "Saison" ein großer Wert auf Werbung gelegt wird.

Besonders hilfreich ist dabei ein erneuter Artikel mit Foto in der ON und ein weiterer Artikel mit einem Screenshot der Webseite in der Aprilausgabe des Kettenblatts (s. Anhang), in welchen nochmals über verschiedene Dinge informiert wird. Dazu gehören beispielsweise, dass ab Beginn der FOSSGIS-Konferenz (s. u.) die Beta-Phase der Seite freigegeben wird und dass es einige neue Entwicklungen auf *fahrradies.net* gibt, die im Oktober/November noch nicht abzusehen waren.
Diese Artikel sind insofern wichtig, als dass zum Zeitpunkt des Erscheinens der ersten Artikel zwar klar strukturierte Ziele vorlagen, aber die Projektphase noch sechs Monate andauerte. In diesem Zeitraum sind sowohl neue Funktionen erstellt als auch geplant, aber auch alte verworfen worden.

Auch ein Radiointerview beim örtlichen Sender "osradio" hat der Öffentlichkeitsarbeit einen

weiteren Vorstoß gegeben. Abzurufen ist das Gespräch unter [Ros]. Auch hier geht es darum, die Funktionen des Radroutenplaners sowie den Nutzen und die Vorteile für die Bürger und Touristen Osnabrücks vorzustellen.
Dadurch, dass das Interview mit den Studenten geführt wird, kann sich der eine oder andere "Fahrradies-Nutzer" vielleicht sogar besser mit diesem Projekt identifizieren. Die Entwickler sind nicht in weiter Ferne, sondern Einwohner von Osnabrück und Umgebung.

Die große Aufmerksamkeit, welche das Projekt bis zum Februar 2010 schon erhalten hat, wird dadurch belegt, dass das Stadtblatt Osnabrück von sich aus um ein Interview mit einigen Vertretern des Radroutenplaners für ihre Aprilausgabe bittet. In dem Artikel, der aus dem Interview entsteht, geht es um Bürger Osnabrücks, die sich um die Nachhaltigkeit ihrer Stadt kümmern. Unter diesem Aspekt betrachtet, liegt *fahrradies.net* in einem immer stärker werdenen Trend.

Neben der Bekanntmachung durch gedruckte und digitale Medien wird auch in Fachkreisen auf den Planer aufmerksam gemacht.
Auf der diesjährigen FOSSGIS [FOS], einer Anwenderkonferenz für freie und Open-Source-Software für Geoinformationssysteme, die vom 2.-5. März in Osnabrück stattgefunden hat, konnte *fahrradies.net* durch einen halbstündigen Vortrag vielen Zuhörern näher gebracht werden. Da auf der FOSSGIS sowohl am 4. als auch am 5. März schwerpunktmäßig über freie Geodaten und OpenStreetMap referiert wird, passt die Präsentation des Projektes gut zu den anderen Beiträgen und erlangt bei vielen Zuhörern große Aufmerksamkeit.
Auch auf der Geoinformatik 2010 (17.-19. März in Kiel) [GiN] halten zwei der insgesamt 16 am Projekt beteiligten Studenten einen interessanten Vortrag, mit dem weiter auf den Planer aufmerksam gemacht werden kann. Die diesjährige Geoinformatik-Konferenz, die unter dem Thema „Die Welt im Netz" steht, zeichnet sich vor allem dadurch aus, dass viele Vertreter aus Wirtschaft, Verwaltungen und auch der Wissenschaft zugegen sind. Zudem zeigen die Reaktionen der Zuhörenden in Kiel, dass der Radroutenplaner für das Osnabrücker Land auch von überregionalem Interesse ist.
Schon durch die Zulassung des Vortrags über *fahrradies.net*, sowohl auf der FOSSGIS-Konferenz, als auch auf der Geoinformatik 2010, wird deutlich, dass dieses Projekt interessant und anspruchsvoll genug ist, um einem fachkundigen Publikum präsentiert zu werden. Um auch längerfristig in den Köpfen der Zuhörer zu bleiben, liegen sowohl auf der FOSSGIS als auch auf der Geoinformatik und im Gebäude des IGF in Osnabrück Flyer aus, die von der Universität Osnabrück genehmigt sind und die wichtigsten Informationen über *fahrradies.net* enthalten.

Um besonders den Studenten die nützliche Funktion der Universitäts- und FH-Gebäudesuche nahe zu bringen, werden außerdem verschiedene Flyer gestaltet, die mehrere Male in der Men-

sa und der Cafeteria am Westerberg und in der Stadt ausliegen.
Auch bei Fahrradläden und Reparaturwerkstätten dürfen die Flyer ausgelegt werden, sodass dort ebenfalls eine möglichst großer Personenkreis angesprochen werden kann und weitere Anhänger gewonnen werden können.
Seitdem der Routenplaner online verfügbar ist, melden sich häufig interessierte Anwender per E-Mail beim Projektteam, um eigene Vorschläge zu unterbreiten oder kleine Fehler zu melden. Dies zeigt, dass der Radroutenplaner auf ein großes Interesse stößt und in der Bevölkerung akzeptiert wird.
Nicht nur durch die eigenständige Bekanntmachung, sondern auch durch den wachsenden Bekanntheitsgrad des Projektes, wird die Öffentlichkeit aufmerksam gemacht. So liest man auch auf den OS-Screens, die mittlerweile in vielen Bussen der Stadtwerke Osnabrück installiert sind, im Lokalteil der "Kreiszeitung" für den Landkreis Diepholz oder im Regionalteil der Webseite der Bild-Zeitung [Bil] über den neuen Fahrradroutenplaner für Osnabrück. Weiterhin haben auch das IGF und die Universität Osnabrück [Uni09a] einen Artikel über *fahrradies.net* auf ihre jeweilige Internetseite gestellt.
Aufgrund dieser Entwicklung bleibt zu hoffen, dass das Projekt auch in Zukunft weiter geführt wird und den Bewohnern von Osnabrück und Umgebung zur Verfügung steht.

7 Fazit und Ausblick

Die Dokumentation hat gezeigt, dass ein Fahrradroutenplaner vielen Anforderungen genügen muss - Benutzerfreundlichkeit, Aktualität oder korrekte Routenführung sind da nur einige Beispiele einer langen Liste. *fahrradies.net* ist unter vielen Aspekten gesehen kein "normaler" Radroutenplaner. Vielmehr ist er eine Zusammenstellung der wichtigsten und interessantesten Funktionen, die in einem heutigen webbasierten Planer enthalten sind oder sein sollten. Dafür sprechen nicht nur die Nutzerzahlen der Seite, sondern auch die positive Resonanz bei den Vorträgen.
Den Besuchern soll natürlich auch langfristig ein aktueller Radroutenplaner vorliegen. Dies wird aber nur funktionieren, wenn in Zukunft auch auf die Mithilfe der Benutzer gebaut werden kann. Aus diesem Grund wurde auf der Homepage die Sektionen *FAQ/Hilfe* und *OSM-Wiki* eingefügt. Sie sollen den Nutzern helfen und dazu anregen, am Projekt mitzuwirken. Nach dem Prinzip "Mitmach"-Routenplaner sollen die Benutzer selbst die Initiative ergreifen und mit Hilfe des „Tagging-Standards", der in den Hilfsanweisungen beschrieben wird, zur Verbesserung der Kartengrundlage beitragen. Eine extra angelegte Seite im OpenStreetMap-Wiki (s. [Opeg]) bietet Informationen an, wie man sich an dem Projekt beteiligen kann. So soll langfristig die Aktualität der Daten gesichert und jedem die Möglichkeit gegeben werden, das Projekt zu erweitern und zu bereichern. Durch die aktive Beteiligung profitieren sowohl die Nutzer, als auch die Entwickler der Seite.

Trotz der vielen verschiedenen Funktionen, die der Planer aufweist und die in den vorherigen Kapiteln beschrieben wurden, konnten nicht alle Ideen verwirklicht werden. Zum einen lag dies an der Dauer des Projektes. Die angedachten Funktionen Bereichsvermeidung und Mehrsprachigkeit waren in zwei Semestern nicht zu realisieren. Zum anderen sind auch durch die beteiligte Zahl von nur 16 Studenten Grenzen gesetzt. Auch die Idee der weiterführenden Handynutzung ist zwar theoretisch sehr gut, war jedoch praktisch mit den genannten Voraussetzungen kaum umzusetzen.
Da der Radroutenplaner jedoch ein fortlaufendes und kein abgeschlossenes Projekt darstellt, kann man für die Zukunft von *fahrradies.net* schon einige Verbesserungsvorschläge geben. Zunächst wäre dies die Option, die Benutzung der Webseite für die Benutzer mobiler Endgeräte einzubauen. Auch der Implementierung von Wanderwegen, generellen Fußwegen oder auch Reitwegen wäre eine Möglichkeit, um *fahrradies.net* zu verbessern und an mehr Nutzer heranzuführen.
Trotzdem muss man bei Projekten dieser Art auch beachten, dass es niemals wirklich "abgeschlossen" werden kann. Nicht nur weitere Funktionen des Planers, sondern auch eine verbesserte Bedienbarkeit oder ein moderneres Design - neue Ideen werden laufend vorgetragen. Die generelle Zukunft für Radroutenplaner sieht vielversprechend aus. Nicht nur die Nutzung von *fahrradies.net*, sondern auch die Zahl der Radroutenplaner, die alleine in den letzten

Jahren entwickelt worden sind, zeigen dies. Durch dieses Projekt wird neuen Entwicklern der Aufbau neuer Radroutenplaner (auf OpenStreetMap-Basis) erleichtert, da der Quellcode des gesamten Projektes gemäß dem Open-Source-Gedanken zur Verfügung gestellt wird.
Es wäre außerdem wünschenswert, wenn man durch diese Arbeit einige Bürger vom Radfahren überzeugen konnte oder auch Nutzer finden würde, die sich aktiv am Gestaltungsprozess der freien Geodaten beteiligen würde, denn von beidem könnte *fahrradies.net* nur profitieren. Ob das Projekt etwas zur weiteren Entwicklung der Radroutenplaner im Internet beigetragen hat, wird sich erst in der Zukunft zeigen.

Literatur

[Ada] Adams, Till: *MapFish-Users-Mailingliste - Uncaught exception XMLHttpRequest.js.* http://lists.mapfish.org/pipermail/users/2009-June/001323.html 5.8.2.2

[ADF] ADFC Osnabrück: *Homepage.* http://www.adfc-osnabrueck.de/, Abruf: 28.04.2010 6

[Ait09] Aitchison, Alastair: *Beginning Spatial with SQL Server 2008.* New York : Apress, 2009 2.11

[Aja] AjaxPatterns.org: *Whats Ajax?* http://ajaxpatterns.org/Whats_Ajax 4.10

[All] Allgemeinen Deutschen Fahrradclubs e.V.: *Inhalte von Radkarten.* http://www.adfc.de/6827_1, Abruf: 28.04.2010 5.4

[BB] Boenigk, Cornelia ; Burger, Ralf: *Was ist PostgreSQL?* http://www.postgres.de/index.whtml#pg, Abruf: 28.04.2010 4.4

[BCL] Bos, Bert ; Celik, Tantek ; Lie, Ian Hickson Hakon W.: *Cascading Style Sheets Level 2 Revision 1 (CSS 2.1) Specification.* http://www.w3.org/TR/CSS2/, Abruf: 28.04.2010 4.6

[Beh09] Behncke, Kai: *Tutorial OpenStreetMap-Daten und OpenLayers.* Version: 2009. http://pgrouting.postlbs.org/wiki/Getyourownroutingdata, Abruf: 28.04.2010 5.1.2

[Bil] Bild.de: *Ein Fahrradies für Osnabrücks Straßen.* http://www.bild.de/BILD/regional/hannover/dpa/2009/12/18/ein-fahrradies-fuer-osnabruecks-strassen.html, Abruf: 28.04.2010 6

[BL04] Bieber, Christoph ; Leggewie, Claus: *Interaktivität - Ein transdisziplinärer Schlüsselbegriff.* 1. Aufl. Frankfurt/Main : Campus, 2004 http://books.google.de/books?id=aEPlch-cp4cC&printsec=frontcover&dq=interaktivite&ei=tGHPS6unCtP2Oe3a7csP&sa=X&oi=book_result&ct=result&resnum=3&ved=0CA4Q6AEwAg 2.3

[Bun07a] Bundeszentrale für politische Bildung: *Into the Great Wide Open.* Version: 2007. http://www.bpb.de/themen/FC8K5M,1,0,Into_the_Great_Wide_Open.html, Abruf: 28.04.2010 2.1

[Bun07b] Bundeszentrale für politische Bildung: *Open Source.* Version: 2007. http://www.bpb.de/themen/32K5CW,0,0,Open_Source.html, Abruf: 28.04.2010 2.1

[Deu] Deutsches Zentrum für Luft- u. Raumfahrt: *Digitales Höhenmodell*. http://www.dlr.de/srtm/level1/products_ge.htm, Abruf: 28.04.2010 2.10

[Dic01] Dickmann, Frank: *Web-Mapping und Web-GIS*. 1. Aufl. Braunschweig : Westermann Schulbuchverlag GmbH, 2001 4.6

[DOR] DORIS - Digitales Oberösterreichisches Raum-Informations-System: *Digitales Höhenmodell (DHM)*. http://doris.ooe.gv.at/geoinformation/metadata/pdf/dhm.pdf, Abruf: 28.04.2010 2.9, 2.10

[ECM] ECMA International: *What is Ecma International*. http://www.ecma-international.org/memento/index.html, Abruf: 28.04.2010 4.7

[Fir] Firma Topografix: *The GPS Exchange Format*. http://www.topografix.com/gpx.asp, Abruf: 28.04.2010 5.10

[Fis06] Fischer, Marcus: *Ubuntu GNU/Linux*. 4. Aufl. Bonn : Galileo Computing, 2006 4.1.2

[Fla02] Flanagan, David: *JavaScript - Das umfassende Referenzwerk*. 2. Aufl. Köln : O'Reilly, 2002 4.6, 4.7

[FOS] FOSSGIS: *Homepage*. http://www.fossgis.de/konferenz/wiki/Main_Page 6

[Fra] Fraunhofer-Institut für Angewandte Informationstechnik FIT: *Web Competence & Responsibility*. http://www.competence-site.de/e-business/EInterview-Prof-Wolfgang-Prinz%2Dzum-Virtual-Roundtable%2DWeb-Competence-and-Responsibility%2DTeil1%2DWeb-2-0-Bedeutung-Chancen-Risiken, Abruf: 28.04.2010 2.4

[Gara] Gary68: *osmdiff.pl*. http://svn.openstreetmap.org/applications/utils/gary68/osmdiff.pl, Abruf: 28.04.2010 3.3

[Garb] Gary68: *useractivity.pl*. http://svn.openstreetmap.org/applications/utils/gary68/useractivity.pl, Abruf: 28.04.2010 3.3

[Gar05] Garrett, Jesse J.: *Ajax: A New Approach to Web Applications*. Version: 2005. http://www.adaptivepath.com/ideas/essays/archives/000385.php, Abruf: 28.04.2010 4.10

[Geo] Geofabrik GmbH: *Open Data - Freie Geodaten und freie Software*. http://www.geofabrik.de/de/geofabrik/free.html, Abruf: 28.04.2010 2.2.1

[GiN] GiN e.V.: *Geoinformatik 2010*. http://www.geoinformatik2010.de/, Abruf: 28.04.2010 6

[GIS] GISWiki: *OpenLayers*. http://www.giswiki.org/wiki/OpenLayers, Abruf: 28.04.2010 4.8

[Gra04] Grassmuck, Volker: *Freie Software - Zwischen Privat- und Gemeineigentum*. 2. Aufl. Bonn : Bundeszentrale für politische Bildung, 2004 http://freie-software.bpb.de 2.1

[GT98] Goodrich, Michael T. ; Tamassia, Roberto: *Data Structures and Algorithms in Java*. 1. Aufl. New York : John Wiley & Sons, 1998 2.12

[Hac07] Hachmann, Roland: Leitfaden 6 - Der UMN Mapserver als Informations- und Partizipationsplattform. In: *Leitfäden zur interaktiven Landschaftsplanung, B. Oppermann* 1 (2007) 4.3

[Hen07] Henkel, Joachim: *Offene Innovationsprozesse - Die kommerzielle Entwicklung von Open-Source-Software*. 1. Aufl. Wiesbaden : Deutscher Universitäts-Verlag, 2007 http://www.springerlink.com/content/p05352/ 2.1

[HKS08] Hurt, Hans-Wilhelm ; Kleinwächter, Doris ; Spier, Heinfried ; Niedersächsisches Ministerium für Inneres, Sport und Integration, Referat 34 - Vermessungs- und Katasterwesen (Hrsg.): *Produkte und Dienstleistungen der VKV Niedersachsen 2008/2009*. Hannover : Landesvermessung und Geobasisinformation Niedersachsen, 2008 http://cdl.niedersachsen.de/blob/images/C21612430_L20.pdf 2.2.1

[Hol] Holz, Patrick: *Designvorlagen und Geschäftsmodelle für eine neue Software-Generation*. http://www.pytheway.de/index.php/web-20, Abruf: 28.04.2010 2.4

[Hol08] Holz, Patrick: *Das Web als Plattform*. Version: 2008. http://www.pytheway.de/index.php/web-20/64-das-web-als-plattform, Abruf: 28.04.2010 2.4

[Int07] Intevation GmbH: *Über Frida*. Version: 2007. http://frida.intevation.de/ueber-frida.html, Abruf: 28.04.2010 2.2.1

[IT-a] IT-Wissen - Das grosse Online-Lexikon für Informationstechnologie: *Artikel zum Thema Log-Datei*. http://www.itwissen.info/definition/lexikon/Port-port.html, Abruf: 28.04.2010 2.5.1

[IT-b] IT-Wissen - Das grosse Online-Lexikon für Informationstechnologie: *Artikel zum Thema Port*. http://www.itwissen.info/definition/lexikon/Port-port.html, Abruf: 28.04.2010 2.5.1

[ITO] ITO: *ITO World*. http://www.itoworld.com, Abruf: 28.04.2010 3.3

[Jam06] James, Justin: *Is Apache inherently more secure than IIS?* Version: 2006. http://blogs.techrepublic.com.com/programming-and-development/?p=32, Abruf: 28.04.2010 4.1.1

[KHL02] Koch, Andreas ; Heipke, Christian ; Lohmann, Peter: Bewertung von SRTM Digitalen Geländemodellen - Methodik und Ergebnisse. In: *PFG* 6 (2002), S. 389–398 2.10

[Mar] Martin-Luther-Universität Halle-Wittenberg: *Traveling-Salesman Problem.* http://www.informatik.uni-halle.de/ti/forschung/toleranzen/#anchor1297624, Abruf: 28.04.2010 2.13

[Mit08] Mitchell, Tyler: *Web Mapping mit Open Source-GIS-Tools.* 1. Aufl. Köln : O'Reilly, 2008 2.1, 2.2.1, 2.7, 2.8, 2.8, 2.9, 5.7.4

[Mül07] Müllegger, Christian: *Grundlagen der Datenqualität (ISO 19113).* Version: 2007. http://homepage.univie.ac.at/wolfgang.kainz/Lehrveranstaltungen/Seminar/2006%20WS/Muellegger_Text.pdf, Abruf: 28.04.2010 2.2.1

[Mün] Münz, Stefan: *SELFHTML - Einführung in JavaScript und DOM.* http://de.selfhtml.org/javascript/intro.htm, Abruf: 28.04.2010 4.7

[Moo01] Moody, Glyn: *Rebel Code - Inside Linux and the Open Source Revolution.* 1. Aufl. Cambridge : MA: Perseus Publishing, 2001 2.1

[Näh] Näher, Prof. Dr. S.: *Das Travelling Salesman Problem.* http://www-i1.informatik.rwth-aachen.de/~algorithmus/algo40.php, Abruf: 28.04.2010 2.13

[Nie] Nielsen, Henrik F.: *W3C - CERN httpd.* http://www.w3.org/Daemon/Status.html, Abruf: 28.04.2010 2.5.1

[OGPa] OGP Surveying & Positioning Committee: *Homepage.* http://www.epsg.org/main.html, Abruf: 28.04.2010 2.9

[OGPb] OGP Surveying and Positioning Committee: *EPSG Geodetic Parameter Dataset.* http://www.epsg.org/Geodetic.html, Abruf: 28.04.2010 2.9

[ON] ON: *Osnabrücker Nachrichten.* http://www.on-live.de/, Abruf: 28.04.2010 6

[Opea] Open Geospatial Consortium, Inc.: *About OGC.* http://www.opengeospatial.org/ogc, Abruf: 28.04.2010 2.6

[Opeb] Open Geospatial Consortium, Inc.: *OGC Members.* http://www.opengeospatial.org/ogc/members, Abruf: 28.04.2010 2.6

[Opec] Open Source Geospatial Foundation: *Geospatial Data Abstraction Library*. http://www.gdal.org/, Abruf: 28.04.2010 5.4.2

[Oped] OpenLayers: *Free Maps for the Web*. http://openlayers.org/, Abruf: 28.04.2010 4.8

[Opee] OpenLayers: *Spherical Mercator*. http://docs.openlayers.org/library/spherical_mercator.html, Abruf: 28.04.2010 2.8, 2.8, 2.9

[Opef] OpenStreetMap Wiki: *DE:Map Features*. http://wiki.openstreetmap.org/wiki/DE:Map_Features, Abruf: 28.04.2010 2.2.3

[Opeg] OpenStreetMap Wiki: *Fahrradies.net*. http://wiki.openstreetmap.org/wiki/Fahrradies.net, Abruf: 28.04.2010 7

[Opeh] OpenStreetMap Wiki: *Osmosis Dokumentation im OpenStreetMap-Wiki*. http://wiki.openstreetmap.org/wiki/Osmosis, Abruf: 28.04.2010 5.3

[php] phplot.sourceforge.net: *Offizielle Webseite von PHPlot*. http://phplot.sourceforge.net/, Abruf: 28.04.2010 5.5.2

[Posa] PostgreSQL Global Development Group: *PostgreSQL - About*. http://www.postgresql.org/about/, Abruf: 28.04.2010 4.4

[Posb] PostgreSQL Global Development Group: *PostgreSQL - History*. http://www.postgresql.org/about/history, Abruf: 28.04.2010 4.4

[Posc] PostLBS: *Offizielle Downloadseite von OSM2pgRouting*. http://pgrouting.postlbs.org/wiki/tools/osm2pgrouting, Abruf: 28.04.2010 5.1.2

[Posd] PostLBS: *pgRouting Documentation: Creating Data for Routing Aplli-cations*. http://pgrouting.postlbs.org/wiki/pgRoutingDocs/preparation/topology, Abruf: 28.04.2010 5.1.2

[Pose] PostLBS: *pgRouting: Dokumentation der Traveling Sales Person Implementation*. http://pgrouting.postlbs.org/wiki/TravellingSalesPerson, Abruf: 28.04.2010 5.2.2.2, 5.2.2.3

[Posf] PostLBS: *pgRouting Dokumentation des implementierten Dijkstra-Algorithmus*. http://pgrouting.postlbs.org/wiki/Dijkstra, Abruf: 28.04.2010 5.1.3, 5.1.4, 5.3.1

[Posg] PostLBS: *pgRouting Project*. http://pgrouting.postlbs.org, Abruf: 28.04.2010 4.5

[Posh] PostLBS: *Userbeitrag im pgRouting-Forum*. http://pgrouting.postlbs.org/discussion/topic/279, Abruf: 28.04.2010 5.1.2

[Pro03] Professur für Geodäsie und Geoinformatik (GG) AUF Universität Rostock: *Geoinformatik-Service - Digitales Höhenmodell (DHM).* Version: 2003. http://www.geoinformatik.uni-rostock.de/einzel.asp?ID=529, Abruf: 28.04.2010 2.10

[Ram08] Ramm, Jochen Frederik und T. Frederik und Topf: *OpenStreetMap - Die freie Weltkarte nutzen und mitgestalten.* 1. Aufl. Berlin : Lehmanns Media, 2008 2.2.2, 3.2

[Refa] Refractions Research: *PostGIS History.* http://www.refractions.net/products/postgis/history/, Abruf: 28.04.2010 4.4

[Refb] Refractions Research: *What is PostGIS?* http://postgis.refractions.net/, Abruf: 28.04.2010 4.4

[Ros] Rossmann, Marcel: *OS-Radio.* http://osradio-podcast.de/2010/03/02/fahrradies-net-der-radroutenplaner-fur-osnabruck/, Abruf: 28.04.2010 6

[Sch09] Schulze, Jan: *Linux oder Windows Welches Betriebssystem gehört auf den Server?* Version: 2009. http://www.computerwoche.de/mittelstand/1891466/index2.html, Abruf: 28.04.2010 4.1.1

[SHPer] SHP Ingenieure: *Masterplan Mobilität der Stadt Osnabrück.* Hannover 1

[SS04] Saake, Gunter ; Sattler, Kai-Uwe: *Algorithmen und Datenstrukturen.* 2. Aufl. Heidelberg : dpunkt.verlag, 2004 2.12

[Sta09] Statistisches Bundesamt Deutschland: *70 Millionen werden in Deutschland gesattelt.* Version: 2009. http://www.destatis.de/jetspeed/portal/cms/Sites/destatis/Internet/DE/Presse/pm/zdw/2009/PD09__022__p002,templateId=renderPrint.psml, Abruf: 28.04.2010 1

[Str07] Stroemer, Katrin: *Nutzerbasierte Adaption des Fahrradroutenplanungsprozesses im Internet auf Basis einer empirischen Untersuchung.* Osnabrück : repOSitorium, 2007 http://repositorium.uni-osnabrueck.de/handle/urn:nbn:de:gbv:700-2007030212 2.3

[Thea] The Apache Software Foundation: *Apache HTTP Server Project.* http://httpd.apache.org/, Abruf: 28.04.2010 4.2

[Theb] The Apache Software Foundation: *Apache HTTP Server Project - Multi-Processing-Module (MPMs).* http://httpd.apache.org/docs/2.0/mpm.html, Abruf: 28.04.2010 4.2

[The09] The Apache Software Foundation: *What IS the Apache HTTP Server Project?* Version: 2009. http://httpd.apache.org/ABOUT_APACHE.html, Abruf: 28.04.2010 4.2

[The10a] The PHP Group: *Die Geschichte von PHP.* Version: 2010. http://de2.php.net/manual/de/history.php.php, Abruf: 28.04.2010 4.9

[The10b] The PHP Group: *PHP - Sicherheit.* Version: 2010. http://de2.php.net/manual/de/security.php, Abruf: 28.04.2010 4.9

[The10c] The PHP Group: *PHP - Was ist PHP?* Version: 2010. http://www.php.net/manual/de/intro-whatis.php, Abruf: 28.04.2010 4.9

[Tra04] Trakas, Athina (Hrsg.): *Praxishandbuch WebGIS mit Freier Software.* Bonn : CCGIS und terrestris, 2004 http://www.mygeo.info/skripte/Praxishandbuch_WebGIS_Freie_Software.pdf 2.5.2

[uni] uni-protokolle.de: *P/NP-Problem.* http://www.uni-protokolle.de/Lexikon/P/NP-Problem.html, Abruf: 28.04.2010 2.13

[UNI08] UNIGIS Salzburg: *Oberflächenmodelle.* Version: 2008. http://www.unigis.ac.at/fernstudien/UNIGIS_professional/schnupper/schnuppermodul/html/lektion5/index.htm, Abruf: 28.04.2010 2.10

[Uni09a] Universität Osnabrück: *Pressemitteilung - Freizeitplanung durch Geoinformatik - Studierende erstellen einen Radroutenplaner.* Version: 2009. http://www2.uni-osnabrueck.de/pressestelle/mitteilungen/Detail.cfm?schluessel_nummer=427&schluessel_jahr=2009&RequestTimeout=50 6

[Uni09b] University of Minnesota: *MapServer - About.* Version: 2009. http://mapserver.org/about.html#about, Abruf: 28.04.2010 4.3

[USG] USGS: *Earth Resources Observation and Science (EROS) Center.* http://eros.usgs.gov/#/Find_Data/Products_and_Data_Available/Elevation_Products, Abruf: 28.04.2010 2.10

[vis] visual complexity: *Windows vs Linux Server.* http://www.visualcomplexity.com/vc/project.cfm?id=392, Abruf: 28.04.2010 4.1.1

[W3Ca] W3C: *Extensible Markup Language (XML) 1.1, 2.1 Wohlgeformte XML-Dokumente.* http://www.edition-w3c.de/TR/2004/REC-xml11-20040204/#sec-well-formed, Abruf: 28.04.2010 2.2.3

[W3Cb] W3C: *The Extensible Stylesheet Language Family (XSL).* http://www.w3.org/Style/XSL/#news, Abruf: 28.04.2010 2.2.3

[W3Cc] W3C: *Scalable Vector Graphics (SVG).* http://www.w3.org/Graphics/SVG/, Abruf: 28.04.2010 11

[W3Cd] W3C: *XML Schema.* http://www.w3.org/XML/Schema, Abruf: 28.04.2010 2.2.3

[W3C04] W3C: *Extensible Markup Language (XML) 1.1, Deutsche Übersetzung - Einführung.* Version: 2004. http://www.edition-w3c.de/TR/2004/REC-xml11-20040204/#sec-intro, Abruf: 28.04.2010 2.2.3

[Wika] WIKIPEDIA: *Artikel Webserver.* http://de.wikipedia.org/wiki/Webserver, Abruf: 28.04.2010 2.5.1

[Wikb] WIKIPEDIA: *Höhenprofil.* http://de.wikipedia.org/wiki/Gel%C3%A4ndeprofil, Abruf: 28.04.2010 5.5

[Wip05] WIPPLINGER, Andrea: *Ubuntu - ein ungewöhnlicher Name für ein verblüffendes Konzept.* Version: 2005. http://www.vhs-community.de/mod/forum/discuss.php?d=15, Abruf: 28.04.2010 4.1.2

Anhang

Auf der CD anbei sind folgende Dokumente zu finden:

- Im Rahmen der Arbeit erstellte Software mit Quellcode
- Alle erschienenen Pressemitteilungen
- Erstellte Flyer, Handzettel und Plakate
- Podcast zum Radiointerview bei OS-Radio
- Auszug aus dem Endbericht Masterplan Mobilität Osnabrück