Manuel Alfred Eder, BSc

Semi-automatisierte Wegenetzerstellung aus Rasterkarten sowie Einbindung dieser in eine passive Indoor-Navigationsanwendung

MASTERARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur

Masterstudium Geomatics Science

eingereicht an der

Technischen Universität Graz

Betreuer

Dr. Konrad Rautz

Institut für Geodäsie

Graz, Jänner 2016

EIDESSTATTLICHE ERKLÄRUNG

Ich erkläre an Eides statt, dass ich die vorliegende Arbeit selbstständig verfasst, andere als die angegebenen Quellen/Hilfsmittel nicht benutzt, und die den benutzten Quellen wörtlich und inhaltlich entnommenen Stellen als solche kenntlich gemacht habe. Das in TUGRAZonline hochgeladene Textdokument ist mit der vorliegenden Masterarbeit identisch.

_____ _____

Datum Unterschrift

Hinweise

In der Arbeit befinden sich einige Logos von Firmen, Softwareprodukten, Programmiersprachen und diversen Standards. Hiermit wird darauf hingewiesen, dass auf die Quellen dieser Logos kein Bezug genommen wird, um der Übersichtlichkeit nicht zu schaden.

Um die Lesbarkeit zu erleichtern, wird in der Arbeit die Sprachform des generischen Maskulinums verwendet. Ich weise darauf hin, dass die Verwendung dieser Form geschlechtsneutral zu verstehen ist.

Es wird darauf geachtet, dass zu schützende Daten nicht in der im Rahmen der Arbeit angelegten Datenbank eingetragen werden. Die für Studierende nicht relevanten Informationen werden vernachlässigt.

Danksagung

An dieser Stelle möchte ich mich bei all jenen bedanken, die durch ihre fachliche und/oder persönliche Unterstützung am Gelingen der Diplomarbeit beigetragen haben.

An erster Stelle möchte ich Herrn Dr. Konrad Rautz erwähnen, der bei der Diplomarbeit als mein Betreuer gewirkt hat.

Mein besonderer Dank gebührt auch meinen Eltern, Alfred und Auguste, die mir das Studium ermöglichten und mir bei meinen Entscheidungen eine wertvolle Stütze waren. Auch für die Korrektur der Arbeit möchte ich mich ganz herzlich bei ihnen bedanken.

Vielen Dank auch an alle Personen, von denen ich konstruktive Ideen für diese Arbeit erhalten habe. Außerdem möchte ich mich auch noch bei meinen Studienkollegen bedanken, die mich während meines Studiums begleitet und moralisch unterstützt haben.

Kurzfassung

Die Diplomarbeit beschäftigt sich mit der Generierung von Indoor-Wegenetzen und mit der Berechnung und Visualisierung des schnellsten Weges zwischen zwei gewählten Punkten. Dazu wurden mit einem WebGIS und einer WebApp zwei Plattformen entwickelt, die über einen Web- und Datenbankserver laufen. Um die Plattformen praktisch zu testen, wurden Gebäudepläne der Technischen Universität Graz verwendet.

Mit dem WebGIS ist es möglich, Wegenetze für den Indoor-Bereich zu generieren und zu verwalten.

Die WebApp ist eine für mobile Geräte optimierte Webseite, die standortbezogene Dienste anbietet und eine visuelle Zielführung zwischen Standort und Zielpunkt anhand von Karten ermöglicht. Die Berechnung und Visualisierung des optimalen Weges wurde neben dem Indoor-Bereich auch für den Outdoor-Bereich realisiert, um zwischen den Gebäuden ebenfalls routen zu können. Hierfür kommen zusätzlich OpenStreetMap-Daten zur Anwendung. Indoor- und Outdoor-Daten werden entsprechend miteinander verknüpft.

Für die Ortung des mobilen Gerätes wurden, neben der Positionierung mittels QR-Code, noch weitere Methoden implementiert. Die Positionsbestimmung erfolgt nur durch das bewusste Eingreifen des Nutzers. Bei der Angabe des Zielpunktes stehen mehrere Methoden zur Auswahl. Die umfangreichen Anwendungsmöglichkeiten werden durch den Einsatz von Symbolen und einer einfachen und übersichtlich gehaltenen Menüführung unterstützt.

Abstract

The master thesis deals with the generation of indoor path networks as well as with the calculation and visualization of the shortest path between two chosen points. For this purpose two platforms, a WebGIS and a WebApp were developed, which operate on a web- and database server. In order to test the platforms, building plans of Graz University of Technology were used.

The WebGIS enables the generation of indoor path networks and the data management.

The WebApp is a website, optimized for mobile devices, which provides a location-based service (LBS). With the help of maps, a visual route guidance between the starting point and the destination is possible. In addition, it is possible to calculate and visualize the shortest path for the outdoor area. The routing between two buildings is realized with the help of OpenStreetMap data. Special links are used to connect the indoor and outdoor area.

In order to enable positioning of the mobile device, QR-Codes and a number of further methods are implemented. The positioning occurs only after intervention of the user. To indicate the destination point a range of selection methods can be used. The comprehensive application possibilities are enabled by symbols and clearly arranged menu navigation.

Inhalt

1. Glossar

Begriff	Bedeutung	Beschreibung
API	**A**pplication **P**rogramming **I**nterface	Schnittstelle zum Anwendungsprogramm
App	**App**lication	Kurzform für mobile Applikation
CAD	**C**omputer **A**ided **D**esign	Rechnergestütztes Konstruieren
CSS	**C**ascading **S**tyle **S**heet	Siehe Abschnitt 4.13.2
DOM	**D**ocument **O**bject **M**odel	Spezifikation einer Schnittstelle für den Zugriff auf HTML- oder XML-Dokumente
EPSG	**E**uropean **P**etroleum **S**urvey **G**roup	Die aus dieser Arbeitsgruppe entstandenen EPSG-Codes werden weltweit verwendet, um Koordinatenreferenzsystem und weitere geodätische Datensätze zu unterscheiden.
GIS	**G**eo**I**nformation **S**ystem	Siehe Abschnitt 4.5
GNSS	**G**lobal **N**avigation **S**atellite **S**ystem	Siehe Abschnitt 4.1.1
GPS	**G**lobal **P**ositioning **S**ystem	Die Beschreibung bezieht sich prinzipiell auf alle GNSS-Systeme (GPS, Galileo, GLONASS, u.a.). Da sich die Bezeichnung GPS durchgesetzt hat, wird dieser Begriff in der Arbeit stellvertretend für alle Systeme verwendet.
GUI	**G**raphical **U**ser **I**nterface	Grafische Benutzeroberfläche
HTML	**H**yper **T**ext **M**arkup **L**anguage	Siehe Abschnitt 4.13.1
JDBC	**J**ava **D**ata**b**ase **C**onnectivity	Dabei handelt es sich um eine Datenbankschnittstelle der Java-Plattform.
JPEG	**J**oint **P**hotographic **E**xperts **G**roup	Gängiges, meist verlustbehaftetes Grafikformat
jQuery	-	Ist eine freie JavaScript-Bibliothek, die es einfach ermöglicht HTML-Dokumente zu manipulieren und grafisch aufzuwerten.
LBS	**L**ocation-**B**ased **S**ervice	LBS sind standortbezogene Dienste bzw. Applikationen, die in Abhängigkeit vom Nutzerstandort darauf abgestimmte Informationen liefern. (itwissen.info - LBS 2015)
MEO	**M**edium **E**arth **O**rbit	Die mittlere Erdumlaufbahn (MEO) befindet sich in einer Höhe zwischen 2.000 und 36.000 km. Diese wird vor allem für Nachrichten-, Navigations- und Forschungssatelliten genutzt.
NFC	**N**ear **F**ield **C**ommunication	Internationaler Übertragungsstandard zum kontaktlosen Austausch von Daten per Funktechnik.
ODBC	**O**pen **D**ata**b**ase **C**onnectivity	standardisierte SQL-Datenbankschnittstelle

OGC	**O**pen **G**eospatial **C**onsortium	OGC ist eine internationale Vereinigung zur Entwicklung und Standardisierung geografischer Daten, LBS und GIS (itwissen.info - OGC 2015). Beispiele für OGC Standards/Spezifikationen sind: KML, GML, WMS, WFS, WCS usw.
OS	**O**perating **S**ystem	Betriebssystem
OSGeo	**O**pen **S**ource **Geo**spatial Foundation	OSGeo ist eine gemeinnützige Organisation zur Förderung und Entwicklung von Open-Source Software mit Bezug zu Geoinformationssystemen sowie Förderung von freien Geodaten (osgeo.org 2015).
OSM	**O**pen **S**treet **M**ap	Frei nutzbare Geodaten in einer wiki-ähnlichen Datenbank
PHP	**P**HP: **H**ypertext **P**reprocessor	Siehe Abschnitt 4.13.4
PNG	**P**ortable **N**etwork **G**rafics	Grafikformat mit verlustfreier Kompression
POI	**P**oint **o**f **I**nterest	Dabei handelt es sich um Punkte, die für den Nutzer relevant sind.
Popup	-	Ein eigenes Browser-Fenster, das sich nach einem bestimmten Ereignis öffnet. Bei den in der Arbeit erwähnten Popups handelt es sich um einfache Info-Boxen, die an einer bestimmten Stelle im Browserfenster erstellt werden.
QGIS	**Q**uantum **GIS**	Siehe Abschnitt 4.17.8
RFID	**R**adio-**F**requency **ID**entification	Technologie, um berührungslos über elektromagnetische Wellen Informationen zwischen Lesegerät und Transponder zu übertragen.
RSSI	**R**eceived **S**ignal **S**trength **I**ndicator	Indikator für die Empfangsfeldstärke kabelloser Kommunikationsanwendungen
SBAS	**S**atellite **B**ased **A**ugmentation **S**ystem	Satellitenbasiertes Ergänzungssystem (geostationär), um durch zusätzliche Informationen die Genauigkeit von GNSS zu verbessern.
SQL	**S**tructured **Q**uery **L**anguage	Dabei handelt es sich um eine Datenbanksprache. Für genauere Informationen siehe Abschnitt 4.14.1
TUGonline	-	Das TUGonline ist das Informations- und Verwaltungssystem der TU Graz. Es bietet Zugriff auf sehr viele Ressourcen, insbesondere auf alle Daten zu Studium und Lehre (portal.tugraz.at - tugonline 2015).
UTM	**U**niversal **T**ransversal **M**ercator	Siehe Abschnitt 4.12.3

W3C	**W**orld **W**ide **W**eb **C**onsortium	W3C ist eine Mitgliedsorganisation, die für die Standardisierung der Techniken des World Wide Web zuständig ist. Beispiele für standardisierte Technologien sind: HTML, XHTML, XML, CSS, SVG usw.
WebGL	**Web** **G**raphics **L**ibrary	In Browsern integrierte Web-Grafikbibliothek
WLAN	**W**ireless **L**ocal **A**rea **N**etwork	Drahtloses lokales Netzwerk
WMS	**W**eb **M**ap **S**ervices	Unter einem WMS versteht man die webbasierte Erstellung von Karten innerhalb eines GIS. (giswiki.org - WMS 2015)
XML	E**X**tensible **M**arkup **L**anguage	Auszeichnungssprache zur Darstellung hierarchisch strukturierter Daten in Form von Textdateien

2. Einleitung

2.1 Motivation

Die Idee zur Diplomarbeit entstand, als im Rahmen einer Lehrveranstaltung an der Technischen Universität Graz ein Raum nur mit Mühe gefunden werden konnte. Nach damaligem Stand war die Raumsuche anhand von Gebäudeplänen nur mit dem TUGonline möglich. Der Aufruf dieser Webseite mit einem Smartphone war jedoch mühsam. Mittlerweile gibt es die App „TU Graz Raumsuche“, mit der man sich den gesuchten Raum anhand eines Stockwerkplans anzeigen lassen kann (play.google.com - TU Graz Raumsuche 2015). Ein Routing zu diesem Raum im Gebäude bietet die App jedoch nicht an. Daher entstand die Idee, im Rahmen der Diplomarbeit eine mobile Navigationsanwendung zu erstellen, die eine Zielführung im Indoor-Bereich ermöglicht.

Eine Herausforderung stellt die Positionsbestimmung innerhalb von Gebäuden dar, für die es noch kein einheitliches System gibt, im Gegensatz zu GPS, mit dem eine Ortung im Outdoor-Bereich problemlos möglich ist. Die für die Positionsbestimmung notwendige Infrastruktur sollte so einfach wie möglich gehalten werden, um den Installationsaufwand der geplanten Anwendung zu reduzieren.

2.2 Zielsetzung

Es soll eine Anwendung erstellt werden, die es ermöglicht, Räume in Gebäuden anhand einer visuellen Zielführung zu finden. Dazu dienen Gebäudepläne der Technischen Universität Graz. Die vorhandenen Rasterkarten der dafür ausgewählten Gebäude müssen zuerst routingfähig gemacht werden. Dieser Schritt zur Generierung der Wegenetze soll möglichst automatisiert erfolgen. Mittels QR-Codes, die primär an den Eingängen anzubringen sind, soll die Position im Gebäude bestimmt werden. Danach soll der Anwender von dieser Position aus den schnellsten Weg zum gesuchten Raum am Smartphone angezeigt bekommen. Die Darstellung des berechneten Weges soll über mehrere Stockwerke möglich sein.

Außerhalb der Gebäude soll die Zielführung anhand von Geodiensten wie der OSM erfolgen. Auch die Positionsbestimmung mittels GPS soll möglich sein. Die Bestimmung des Standortes erfolgt nur durch das Eingreifen des Anwenders und soll nicht kontinuierlich bestimmt werden, wie z.B. bei einem Fahrzeugnavigationssystem. Auf eine kontinuierliche Positionsbestimmung wird verzichtet, weil in den meisten Gebäuden die dafür notwendige Infrastruktur fehlt. Die finale Version der Anwendung soll jedoch in jedem Gebäude und ohne hohen Aufwand umsetzbar sein. Die Applikation soll auf Smartphones oder Tablets lauffähig und leicht zu bedienen sein.

3. State-of-the-Art-Analyse

Bei einer Recherche nach bereits vorhandenen Systemen im Bereich der Indoor-Navigation hat sich herausgestellt, dass bereits zahlreiche Lösungen angeboten werden. Fast alle Systeme arbeiten mit eigens aufbereiteten Karten und nutzen unterschiedliche Positionierungs-Methoden. In diesem Abschnitt sind Systeme beschrieben, die etwa ein ähnliches Einsatzgebiet haben, wie das System in der vorliegenden Arbeit.

3.1 vanillaNAV

Die vanillaNAV-Positionierungstechnologie nutzt optische Navigationsmarker zur Positionsfindung. Diese kleinen QR-Codes können am Boden oder an den Wänden angebracht werden. Dadurch ist es möglich, das System schnell, einfach und kostengünstig in jedem beliebigen Gebäude zu installieren. Nach gestarteter Zielführung erkennt die vanillaNAV-App die Navigationsmarker und führt den Nutzer über Augmented Reality und mit Hilfe eingeblendeter Navigationspfeile ans Ziel (siehe Abbildung 1). Dies funktioniert für den Innen- und Außenbereich gleichermaßen. Über den vanillaNAV-Manager können Ziele und Routen definiert bzw. Zusatzinformationen gespeichert werden.

Schritte:

- Plan des Gebäudes laden
- Navigationsziele und Zusatzinformationen definieren
- Positionen der Navigationsmarker angeben
- Navigationsmarker ausdrucken und montieren

Kosten:
Jedes Ziel sowie jede Zusatzinformation zu einem Ziel wird ab 99 Cent im Monat angeboten. In den auf der Webseite angegebenen Preisbeispielen wird eine Ausstellungshalle mit 20 Austellern mit ca. 80 Dollar pro Monat und ein Museum mit 50 Zielen mit Kosten von 2.300 Dollar im Jahr angegeben. Nähere Informationen sind unter (vanillanav.com 2015) zu finden.

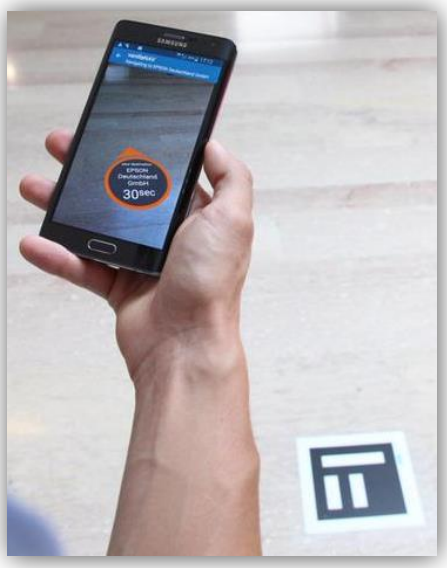

Abbildung 1 – vanillaNAV-App
(pbs.twimg.com - vanillaNAV 2015)

3.2 Deep Map

Das international agierende IT-Unternehmen Heidelberg Mobil bietet ein Service zur Erstellung von 2D/3D Indoor-Karten an. Aus CAD-Gebäudedaten werden 2D/3D Indoor-Karten erstellt und mit zusätzlichen GIS-Informationen ausgestattet. Auf Wunsch kann auch Indoor-Routing integriert werden.

Funktionen:

- 2D/3D Karten-Ansicht (siehe Abbildung 2)
- Integration aller gängigen Positionierungslösungen wie GPS, WLAN und Bluetooth-Beacons
- Platzierung von POIs
- Interaktive Karte mit Informationen zu verschiedenen Standorten
- Darstellung mehrerer Etagen sowie Realisierung eines Multifloor-Indoor-Routings (siehe Abbildung 3)
- Stufenloses Zoomen und Rotieren
- Individuelle Anpassung des Designs
- Integration von Google-Maps oder OSM

Abbildung 2 – 2D-Map/3D-Map (deep-map.com - Funktionen 2015)

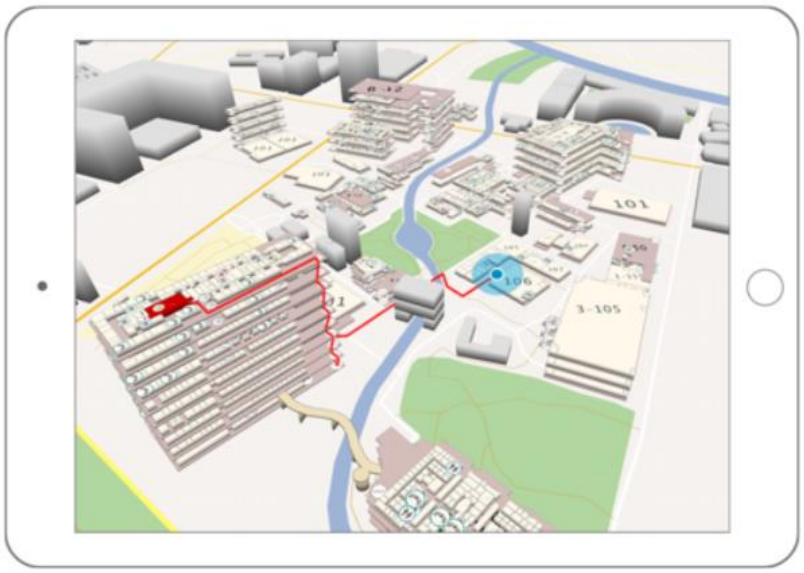

Abbildung 3 – Multifloor-Indoor-Routing (deep-map.com - Funktionen 2015)

Anwendungsbereiche:

- Events
- Messen
- Shopping Center
- Firmengelände
- Flughäfen
- Bahnhöfe
- Krankenhäuser

Die Informationen basieren auf (deep-map.com - Deep-Map 2015).

3.3 indrz

Mit der indrz-App der Firma gomogi ist es möglich, dass Studierende, Besucher und Mitarbeiter einer Universität anhand einer Karte den Weg zum gewünschten Zielort problemlos finden. Der berechnete Weg wird auf der Karte visualisiert, wobei dies auch über mehrere Stockwerke und auch für den gesamten Campus-Bereich möglich ist (siehe Abbildung 4). Karten können entweder in 2D oder 3D dargestellt werden. Durch die Verknüpfung mit den Verzeichnissen der Universität ist es möglich, bestimmte Informationen in die App einzubinden. Die Auswahl des Standortes und des Ziels erfolgt manuell oder über ein Positionierungsverfahren. Die Informationen beziehen sich auf (panoramatec.com - indrz-campus 2015).

Abbildung 4 – indrz-campus (panoramatec.com - indrz-campus 2015)

3.4 Navvis

Dabei handelt es sich um eine Applikation, mit der man in Gebäuden anhand visueller Informationen navigieren kann. Um nach dem richtigen Weg zu suchen, nimmt man mit der App eines Smartphones ein Foto der Umgebung auf (siehe Abbildung 5). Dieses Bild wird anschließend mit den gespeicherten Aufnahmen einer Datenbank verglichen. Durch entsprechende Algorithmen kann der Aufnahmestandpunkt bestimmt werden und der Weg wird, wie bei Navigationssystemen üblich, mit Pfeilen eingeblendet.

Abbildung 5 – NAVVIS-App (navvis.lmt.ei.tum.de - about 2015)

Wie bei allen andern Methoden fällt auch hier die Arbeit des Kartierens nicht weg. In diesem Fall müssen die Räume auch mit einem Laser gescannt sowie Fotos aufgenommen werden.

Der Artikel basiert auf (navvis.lmt.ei.tum.de - NAVVIS 2015).

Ein ähnliches Konzept wurde von der technischen Universität Wien und von einem Start-up-Unternehmen für den Flughafen Wien realisiert (futurezone.at - TU-Wien Navigations-App 2015).

3.5 ROOMAPS

ROOMAPS ist eine Informationslösung für mobile Endgeräte, mit der sich die Nutzer über Smartphone-Apps in einem Gebäude zurecht finden. Dabei werden bestehende digitale Gebäudepläne in ein vektorbasiertes Format übergeführt. Mit ROOMAPS ist es auch möglich, Besucher- bzw. Kundenströme innerhalb eines Gebäudes auf Heat-Maps zu visualisieren und auszuwerten.

Abbildung 6 – iBeacons (roomaps.com - Technology 2015)

Die Positionsbestimmung erfolgt durch eine Signalstärkenauswertung der iBeacons (siehe Abbildung 6). iBeacons sind kleine autarke Bluetooth Low-Energy-Sender. Durch die Nutzung der Sensoren des Smartphones kann mittels Partikelfilterverfahren und „Map-Matching“ die Positionsbestimmung verfeinert werden. Dabei soll eine Genauigkeit von ein bis zwei Metern erreicht werden.

Das verwendete Verfahren streut virtuelle Partikel auf der gesamten Nutzfläche (siehe Abbildung 7). Anhand der Schritterkennung und der begehbaren Flächen werden unwahrscheinliche Partikel schrittweise entfernt. Nach einigen Metern konzentrieren sich die virtuellen Partikel. Nähere Informationen zu diesem Thema sind auf (roomaps.com - ROOMAPS 2015) zu finden.

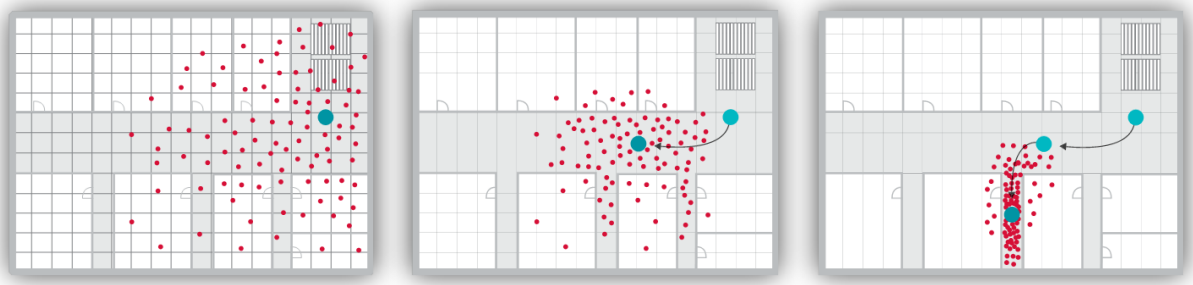

Abbildung 7 – Partikelfilterverfahren (roomaps.com - Technology 2015)

3.6 awiloc

Das Frauenhofer-Institut in Nürnberg bietet in den Bereichen Lokalisierung und Navigation innovative Technologien und Systemlösungen an. Das Angebot reicht dabei von einzelnen Modulen und Komponenten bis hin zu fertigen Lösungen.

Dabei werden über zehn verschiedene Lokalisierungstechnologien zu maßgeschneiderten Prototypen und Systemen angepasst, weiterentwickelt und kombiniert.

Einsatzfelder:

- Verkehr und Automotive
- Sicherheit
- Logistik
- Information und Unterhaltung
- Robotik

Mit awiloc wurde eine Lösung zur Positionsbestimmung entwickelt, die die charakteristische Feldstärkenverteilung vorhandener Funknetzwerke nutzt. Dabei werden drahtlose Funknetzwerke auf Basis des WLAN-Standards als Basis verwendet. Die Lokalisierung erreicht dabei eine Genauigkeit von wenigen Metern. Nähere Informationen zu awiloc sind unter (iis.fraunhofer.de - Adaptive Systemsoftware 2015) und (iis.fraunhofer.de - rssi/awiloc® 2015) zu finden.

3.7 Google Indoor-Maps

Seit zwei Jahren nutzt Google für seine Indoor-Maps WLAN-Signale, um mittels Trilateration den Standort in Gebäuden zu bestimmen. Diese Technologie kann mit mobilen Apps genutzt werden (siehe Abbildung 8).

Nach derzeitigem Stand ist die Verwendung dieser App jedoch auf sehr wenige Gebäude beschränkt. Außerdem ist die Ortung sehr ungenau, was praktische Tests gezeigt haben. Auch bei der Anzeige des richtigen Stockwerkes kommt es zu Fehlern. Meist liegt es an der mangelhaften oder schlecht optimierten Sender-Infrastruktur des Gebäudes, welche aus WLAN-Routern besteht.

Die Informationen beziehen sich auf (google.com - indoormaps 2015) und den Zeitschriftenartikel (Stelzel-Morawietz 2015).

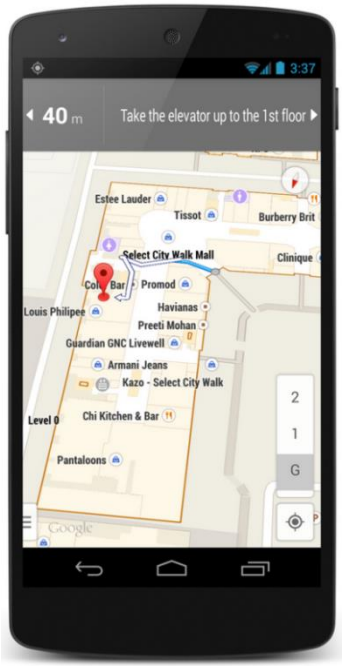

Abbildung 8 – Google Maps (India-Indoor 2015)

4. Theorie

Dieses Kapitel enthält Information über den theoretischen Hintergrund der Arbeit.

4.1 Positionsbestimmung

Um die Position eines Endgerätes zu bestimmen gibt es unterschiedliche Methoden. Dabei unterscheidet man grundsätzlich zwischen aktiver und passiver Ortung.

4.1.1 Aktive Ortung

- **IP-Adresse:** Jedes Gerät mit Internetzugang hat eine öffentliche IP-Adresse, welche von einem Internetprovider bereitgestellt wird. Damit kann die Position des Gerätes näherungsweise bestimmt werden. Die Genauigkeit der Annäherung kann jedoch unterschiedlich stark sein. Nähere Informationen sind unter (browsercheck.pcwelt.de - Geolokalisierung 2015) zu finden.

- **Zellortung:** Das drahtlose Mobilfunknetz besteht aus unzähligen Mobilfunkmasten, dessen Positionen genau bekannt sind. Jeder Sender deckt einen in etwa kreisförmigen Bereich um sich herum ab, in dem das Funksignal noch fehlerfrei decodiert werden kann. Die genaue Abdeckung ist jedoch abhängig von Größe und Typ der Antenne, der Sendeleistung, den meteorologischen Einflüssen und vor allem von der geografischen Oberfläche, da z.B. Gebäude und Berge dieses Signal abschirmen können. Auf Grund der Überlagerung der Abdeckungsbereiche entstehen sogenannte Ortungszellen (siehe Abbildung 9), die meist eine Wabenanordnung haben. Jede dieser Zellen besitzt eine eindeutige „Cell-ID“. Sobald sich ein Smartphone bei einem Mobilfunkmasten einbucht, wird das vom jeweiligen Netzbetreiber registriert. Dieser weiß damit auch, in welcher Zelle sich das Mobiltelefon gerade befindet und gibt diese Daten bei Bedarf an das Gerät weiter. Die Genauigkeit dieser Positionierungsmethode ist abhängig von der Größe der Ortungszelle, welche wiederum abhängig von der Dichte der Mobilfunkmasten ist. Da in weniger dicht besiedelten Gebieten weniger Mobilfunkmasten benötigt werden, ist diese Methode dort sehr ungenau.

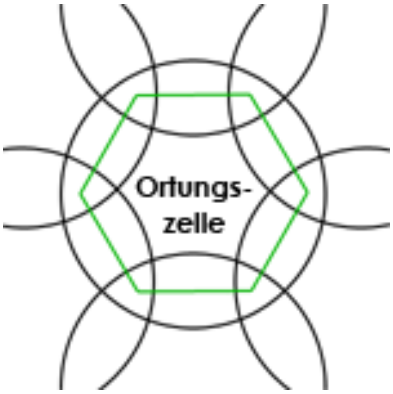

Abbildung 9 – Ortungszelle (handyortung-und-schutz.de - GSM-Ortung 2015)

Die Informationen über die Zellortung basieren auf (handyortung-und-schutz.de - GSM-Ortung 2015) und (de.wikipedia - Funkzelle 2015).

- **GPS:** Dabei werden die Signale von GPS-Satelliten verwendet, um die Position durch Laufzeitmessungen zu bestimmen. Dazu benötigt man einen im mobilen Gerät integrierten GPS-Empfänger. Um die Position berechnen zu können, ist das Signal von mindestens vier Satelliten erforderlich, da vier Unbekannte zu bestimmen sind (die Koordinaten x, y, z sowie der Uhrenfehler). Die Satelliten umkreisen die Erde in Umlaufbahnen (MEO) und decken die gesamte Erdoberfläche ab (siehe Abbildung 10). In Gebäude können die Signale jedoch nur vereinzelt eindringen und es ist daher nicht möglich, die Position im Indoor-Bereich zu bestimmen. Der Vorteil dieser Methode liegt in der weltweiten Abdeckung des Outdoor-Bereiches, bei einer Genauigkeit von ca. 8 - 12 Metern. Diese kann noch durch weitere Methoden wie z.B. SBAS zusätzlich verbessert werden. Außerdem können durch die kontinuierliche Bestimmung der Position die Geschwindigkeit und die Bewegungsrichtung bestimmt werden. Moderne Smartphones nutzen sogenanntes A-GPS (Assisted-GPS), um über das Mobilfunknetz die Almanach-Daten (Bahnparameter und Korrekturdaten) zu aktualisieren. Die Positionsbestimmung kann so erheblich beschleunigt werden, da die Übermittlung der Almanach-Daten über das GPS-Signal im Extremfall bis zu 12,5 Minuten dauern kann.

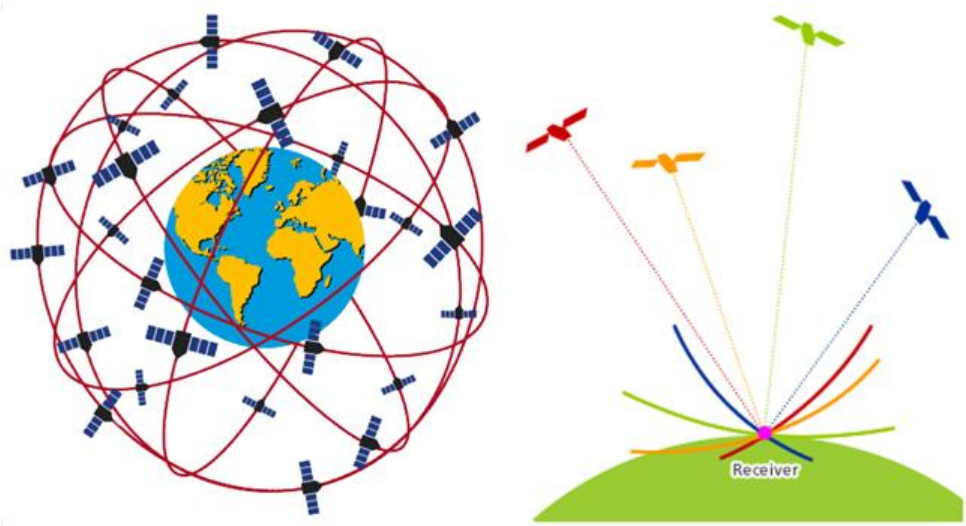

Abbildung 10 – GPS-Positionierung (seos-project.eu - Satellitenbahnen 2015) und (jaxa.jp - GPS 2015)

Die Informationen über GPS beziehen sich auf (kowoma - GPS 2015) und (quantenwelt.de - Almanach 2015).

- **WLAN:** Dabei wird die Position über die Auswertung der WLAN-Signale der näheren Umgebung berechnet. Ein Einloggen in eines der Netzwerke ist nicht notwendig. Es werden lediglich die Informationen über Signalstärke und die Kennung des Funknetzes benötigt, welche frei verfügbar sind. Um die Position zu bestimmen, ist ebenfalls die Kenntnis der Standorte der WLAN-Router erforderlich. Je mehr Signale empfangen werden, desto genauer kann die Position bestimmt werden. Im Gegensatz zu GPS funktioniert diese Methode auch innerhalb von Gebäuden. Der Nachteil ist, dass die notwendige Infrastruktur in Form von zahlreichen verorteten WLAN-Routern vorhanden sein muss. Unterstützend können auch Verfahren wie Szenenanalysen eingesetzt werden. Vorteil gegenüber anderen Methoden ist, dass das Kundenverhalten aufgezeichnet werden kann.

  Das Hintergrundwissen über WLAN basiert auf (wikipedia - WLAN-basierte Ortung 2015) und dem Zeitenschriftenartikel (MEYER 2008).

- **Bluetooth:** Ähnlich wie bei WLAN ist es seit der Version 1.1 möglich, die RSSI zu ermitteln, die ein Indikator für die Signalstärke ist. Damit kann die Position von Endgeräten bestimmt werden, wobei eine Genauigkeit von bis zu zwei Metern erreicht werden kann. Die Signalstärke ist jedoch stark von der Umgebung und den Materialien abhängig, welche sich in unmittelbarer Umgebung befinden. Um eine hohe Genauigkeit zu erreichen, ist wie bei der Positionsbestimmung mittels WLAN eine entsprechende Infrastruktur notwendig, indem man eine hohe Dichte an sogenannten Bluetooth-Beacons garantiert. Nähere Information sind unter (indoor-ortung.de - Ortung mit Bluetooth 2015) zu finden.

- **Inertialsensoren:** Mit Hilfe der in Smartphones oder Tablets eingebauten Inertialsensoren (Beschleunigungssensor/Accelerometer und Drehratensensor/Gyroskope) kann die relative Position bestimmt werden. Um daraus die absolute Position zu bekommen, ist auch die Position des Gerätes zum Zeitpunkt des Messbeginns erforderlich. Daher kann dieses Verfahren, welches auch mit „Dead Reckoning“ bezeichnet wird, nur in Kombination mit anderen Verfahren eingesetzt werden. Mit dem Faktor Zeit bzw. mit der Entfernung summieren sich die systematischen Fehler auf, so dass die Positionierungsgenauigkeit darunter leidet. Einen großen Nachteil dieses Verfahrens stellen die schlechten Sensoren dar, welche in den mobilen Endgeräten eingebaut sind. Aus diesem Grund sollen solche Systeme in Zukunft zusätzlich mit Gebäudemodellen (Map-Matching) und/oder Bewegungsmodellen (Schrittdetektion bei Fußgängernavigation) ausgestattet werden, um die Genauigkeit zu verbessern. Detaillierte Information zu Inertialsensoren sind unter (ree-wifi-service.de - Passive Indoor-Navigation 2015), (LOULIER 2011) und (Leitfaden – Hardware, Software, IT-Sicherheit V2.0 2015) zu finden.

- **Weitere Methoden:** Nutzung von Licht, Ultraschall oder Magnetfeldern zur Ortung.

4.1.2 Passive Ortung

- **QR-Codes:** Bei der Positionsbestimmung mittels QR-Code ist das Einscannen des Codes am Standort erforderlich, welches mit nahezu jedem Smartphone möglich ist. Über eine Anwendung kann dieser Code entschlüsselt werden und die darin enthaltenen Informationen können zur Bestimmung der Position verwendet werden. Diese Information kann eine Referenz zur Ortsinformation sein oder kann auch die Position selbst enthalten. Folgende Vor- und Nachteile bietet diese Lösung gegenüber den Methoden der aktiven Ortung:

  Vorteile:
  - im Indoor-Bereich einsetzbar
  - für eine grobe Positionsbestimmung ausreichend
  - leicht umsetzbar
  - kostengünstig
  - zusätzliche Informationen und Angebote sind im Code integrierbar
  - wird auch schon von einigen Webbrowsern unterstützt

  Nachteile:
  - eine Internetverbindung ist erforderlich
  - QR-Codes können leicht manipuliert/zerstört werden
  - Der Zugriff auf die Kamera-API ist derzeit nur mit wenigen Browsern möglich, was die Entwicklung von WebApps erschwert.
  - Die Messung von Kundenströmen oder der Frequentierung ist nicht möglich.

Weitere Informationen über die Ortung mittels QR-Code sind unter (ree-wifi-service.de - Passive Indoor-Navigation 2015) zu finden.

- **Bildverarbeitende Methoden:** Neben QR-Codes gibt es noch weitere bildverarbeitende Möglichkeiten, die Position anhand von Bildern bzw. Symbolen zu bestimmen.

- **RFID/NFC:** Bei diesem Verfahren wird die Position auf einem RFID-Transponder gespeichert (siehe Abbildung 11). Bekannt ist diese Technik auch durch den internationalen Übertragungsstandard NFC. Mit einem Lesegerät kann die gespeicherte Information berührungslos über eine kurze Distanz übermittelt werden. Jedoch haben heute nur die modernsten Smartphones ein NFC-Lesegerät eingebaut. Vorteile sind hingegen der geringe Preis und der niedrige Stromverbrauch sowie die geringe Größe. Nähere Informationen sind unter (RFID Radiofrequenz-Identifikation 2010) und (praxistipps.chip.de - NFC - Was ist das? 2015) zu finden.

Abbildung 11 – NFC-Chip (electronics360.globalspec.com - NFC-Tag 2015)

4.2 Indoor-Navigation

Unter dem Begriff Indoor-Navigation versteht man die Positionsbestimmung und die Berechnung des optimalen Weges zum Ziel innerhalb eines Gebäudes. Auch die Zielführung (guidance) ist genaugenommen ein Teilbereich davon.

Während für den Outdoor-Bereich GPS als Standard akzeptiert wird, gibt es für den Indoor- bzw. Gebäudebereich noch kein einheitliches System, welches wirklich überzeugen kann. Jedes in Abschnitt 4.1 beschriebene System hat seine Vor- und Nachteile, wobei meist eine geringe Fehlerquote mit hohen Kosten verbunden ist oder umgekehrt. Bei der Ortung mittels QR-Code ist es außerdem nicht möglich, die Position automatisiert bzw. kontinuierlich zu bestimmen, sondern nur dann, wenn der Benutzer aktiv eingreift.

Die erreichbare Genauigkeit spielt bei der Indoor-Positionierung ebenfalls eine wichtige Rolle. Während man bei GPS schon mit einer Genauigkeit von zehn Metern zufrieden ist, wird innerhalb von Gebäuden eine präzisere Ortsangabe benötigt. Die Detektion des richtigen Stockwerks ist für eine Indoor-Navigationsanwendung ebenfalls entscheidend.

Der Einsatz einer Navigationsanwendung innerhalb von Gebäuden ist auch davon abhängig, ob der Gebäudebesitzer die entsprechenden Karteninformationen und Infrastruktur bereitstellt.

Anwendungsgebiete:

- Privater Komfort/Unterhaltung
- Marketing
- Tourismus
- Einsatzkräfte/Militär
- Industrie

Detailliertere Information sind unter (CZYCHOLL 2015) zu finden.

4.3 Routing

Unter Routing versteht man in der Navigation die Berechnung von möglichen Wegen in einem Wegenetz. Damit werden Fragen behandelt wie „Wohin geht es?“ und „Wie komme ich dorthin?“ (HOFMANN-WELLENHOF, LEGAT, and WIESER 2003). Die Führung entlang einer berechneten Route anhand von akustischen oder visuellen Informationen wird als Zielführung (engl. guidance) bezeichnet. Unter einer Route wird eine Folge von Wegpunkten verstanden, die miteinander verbunden sind.

4.4 Location-Based Services

Damit werden standortbezogene Dienste bezeichnet, die anhand der Positionsdaten eines mobilen Endgerätes bestimmte Informationen und Dienste bereitstellen. Die Art und Genauigkeit der Positionierung ist geräteabhängig. Neben der Verwendung in Navigationsanwendungen (Freizeit, Flottenmanagement u.w.) werden diese Dienste auch vermehrt in der Unterhaltung oder zur Optimierung von Werbekampagnen eingesetzt. Die Nutzung ortsbezogener Dienste ist in der Europäischen Union durch Richtlinien und Gesetze geregelt. So werden z.B. standortbezogene Informationen als schützenswert betrachtet. Daher wird in vielen Fällen vom Nutzer die Bestätigung des Standortes angefordert. Die Informationen beziehen sich auf (itwissen.info - LBS 2015).

Ein LBS besteht im Wesentlichen aus den drei Komponenten:

- Geoinformation (Aufbereitung, Analyse und Visualisierung von Geodaten)
- Navigation (Bestimmung des Nutzerstandortes, Berechnung des optimalen Weges sowie Zielführung)
- Kommunikation (Datenübertragung, meist über Internet)

4.5 GIS – Geografische Informationssysteme

Geografische Informationssysteme sind rechnergestützte Systeme, bestehend aus Hardware, Software und Daten. Diese finden im Bereich der raumbezogenen Daten Verwendung, wobei in einem GIS folgende Aufgaben vereint werden:

- Erfassung
- Speicherung
- Verwaltung
- Aufbereitung
- Analyse
- Visualisierung

Ein solches System verbindet grafische Geometriedaten (Kartenmaterial) mit numerischen Attributen. Nähere Informationen über GIS sind unter (artalis.de - GIS 2015) und (cartogis.de - Was ist GIS? 2015) zu finden.

4.6 Graphentheorie

4.6.1 Definition

Die Graphentheorie ist ein Teilgebiet der Mathematik, welches sich mit Graphen und ihren Beziehungen zueinander beschäftigt. Die ersten Ansätze dieses Teilgebietes entstanden 1736 (Königsberger Brückenproblem) und um 1850 (Vierfarbenproblem).

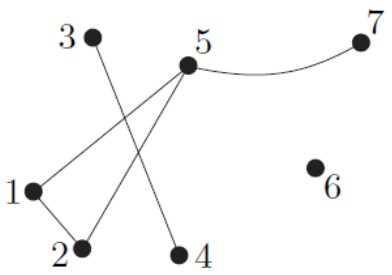

Abbildung 12 – Beispiel für einen Graphen (BINDER 2006)

Ein Graph besteht aus einer Menge von Punkten (Abbildung 12), die man als Knoten oder seltener auch als Ecken bezeichnet.

Zwei der Knoten können durch sogenannte Kanten bzw. Bögen miteinander verbunden sein. Ob diese Kanten gerade oder geschwungen sind, disjunktiv sind oder sich überkreuzen, hängt von der grafischen Darstellung ab und ist von der formalen Definition des Graphen unabhängig. Graph $V = \{1, \dots, 7\}$ mit der Kantenmenge $E = \{\{1,2\},\{1,5\},\{2,5\},\{3,4\},\{5,7\}\}$

Die Graphentheorie wird unter anderem bei der Berechnung von kürzesten Wegen benötigt. Ein Weg ist eine natürliche Folge seiner Knoten.

Einen Graphen kann man nach folgenden Kriterien unterscheiden:

- gerichteter Graph: Ein Graph, der mindestens eine gerichtete Kante enthält. Eine gerichtete Kante wird anstatt durch Linien mit einem Pfeil verbunden, wobei die Verbindung nur in eine Richtung besteht. Die Pfeilrichtung gibt an, in welche Richtung die Verbindung möglich ist.
- ungerichteter Graph: Graph, der keine gerichteten Kanten enthält. Die Kanten werden als Linien dargestellt und eine Verbindung besteht in beide Richtungen.
- planarer Graph: Ein Graph heißt planar, wenn er in der Ebene so gezeichnet werden kann, dass sich seine Verbindungslinien (Kanten) nicht kreuzen. Dabei ist die Anzahl der Kanten linear abhängig von der Anzahl der Knoten. Diese erzeugen auch weniger Rechenaufwand.
- finit Graph: Ein Graph mit endlicher Anzahl von Knoten.
- vollständiger Graph: Ein Graph, in dem jeder Knoten durch eine Kante mit einem anderen Knoten verbunden ist.

Weitere Begriffe:

- adjazent: Ein Knoten ist adjazent, wenn er durch mindestens eine Kante verbunden ist.
- Baum: Ein Graph, der keinen Kreis enthält. Ein Baum mit genau n Knoten hat genau n-1 Kanten. Bäume kommen jedoch in Wegenetzen nur untergeordnet vor. Bei Bäumen gibt es immer nur einen kürzesten Weg. Baumstrukturen finden Verwendung bei der Optimierung von Speicherplatz, so z.B. auch bei der Verwendung von Karten (siehe Tiles, Abschnitt 4.10).

Der Abschnitt basiert auf (DIESTEL 2006), (HOFMANN-WELLENHOF, LEGAT, and WIESER 2003), (BINDER 2006) und (mathe.tu-freiberg - Graphentheorie 2015).

4.6.2 Speicherung von Graphen

Um einen Graphen zu speichern, gibt es mehrere Möglichkeiten. Die einfachste und logischste Variante ist die Verwendung einer Adjazenzliste, in der von jeder Kante Anfangs- und Endknoten sowie dessen Attribute gespeichert werden. Eine weitere Möglichkeit ist, eine Adjazenzmatrix zu verwenden, in der jede Zeile einem Anfangsknoten entspricht und jede Spalte einem Endknoten. Sollte eine Verbindung zwischen den Knoten bestehen, wird das Feld „1“ eingetragen oder auch die entsprechenden Kosten c. Ansonsten wird „0“ eingetragen. Nachteilig erweist sich bei solchen Matrizen die hohe Dimension ($n \times n$) und dass nur ein Kantenattribut verwendet werden kann. Der Vorteil liegt aber im leichten Zugriff auf solche Tabellen. Eine Alternative ist die gekettete (indizierte) Adjazenzliste. Dabei werden eine Knoten- und eine Bogentabelle angelegt. In der Knotentabelle steht der Bogenindex. Aus der Differenz zum nächsten Bogenindex erhält man die Anzahl der Bögen, die von diesem Knoten ausgehen. Die jeweiligen Kanten der Bogentabelle erhält man wiederum über den Bogenindex. Dabei müssen die Bögen unbedingt aufsteigend und lückenlos nummeriert sein. Durch die Aufteilung in zwei Tabellen wird weniger Speicherplatz benötigt, egal wie viele zusätzliche Attribute die Bogentabelle enthält. Ein wichtiges Kantenattribut sind die Kosten, mit der jede Kante bewertet wird.

Die Kosten c entsprechen einem Gewicht, das von der Länge abhängig ist. Aber auch eine zusätzliche Abhängigkeit von weiteren Kantenattributen ist möglich, wie z.B. erlaubte Geschwindigkeit.

In Abbildung 13, Tabelle 1, Tabelle 2 und Tabelle 3 ist ein Beispiel einer geketteten Adjazenzliste dargestellt.

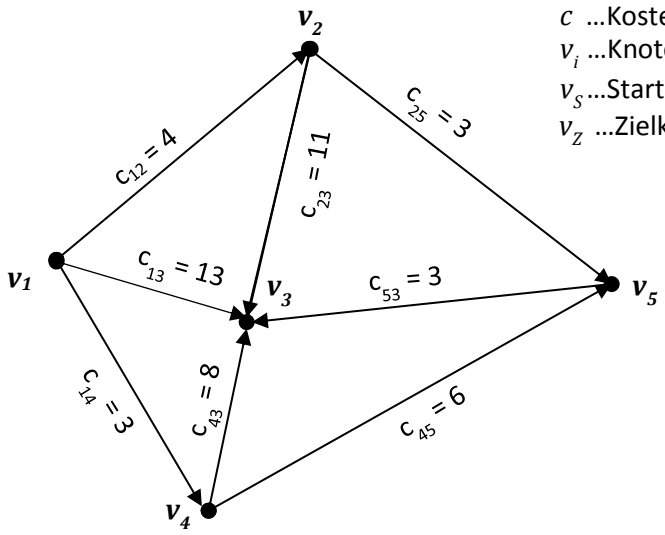

Abbildung 13 – Gerichteter Graph, eigene Darstellung nach (HOFMANN-WELLENHOF, LEGAT, and WIESER 2003)

Tabelle 1 – Adjazenzliste

ID	v_S	v_Z	c
1	1	2	4
2	1	3	13
3	1	4	3
4	2	3	11
5	2	5	3
6	4	3	8
7	4	5	6
8	5	3	3

Gekettete Adjazenzliste:

Tabelle 2 – Knotenliste

v_S	1	2	3	4	5	6
Bogenindex	1	4	6	6	8	9

6 - 4 = 2 → 2 Bögen

Tabelle 3 – Bogenliste

Bogenindex	1	2	3	4	5	6	7	8
v_Z	2	3	4	3	5	3	5	3
c	4	13	3	11	3	8	6	3

Speicherplatzbedarf:

Formel 1 – Vergleich zwischen den Speichervarianten

Adjazenzmatrix:

$d_{AM} = 8 \cdot 8 = 64\ Zellen$

Adjazenzliste:

$d_{AL} = 3 \cdot 8 = 24\ Zellen$

gekettete Adjazenzliste:

Vergleich:

$d_{gekAL} = 6 + 2 \cdot 8 = 22\ Zellen$

$d_{AM} \gg d_{AL} > d_{gekAL}$

Der Vergleich der Speichervarianten (siehe Formel 1) zeigt, dass die Speicherung eines Graphen, der aus 8 Bögen besteht, in einer Matrix am meisten Zellen benötigt und die gekettete Adjazenzliste am wenigsten Speicherplatz benötigt. Dieser Abschnitt basiert auf (HOFMANN-WELLENHOF, LEGAT, and WIESER 2003) und (WIESER 2013).

4.7 Algorithmen zur Berechnung der kürzesten Wege

In diesem Abschnitt werden Graphen-Algorithmen behandelt, die die zentrale Frage beantworten, wie man am schnellsten von Punkt A nach Punkt B kommt. Diese Algorithmen basieren darauf, dass die Kanten des Graphen durch Kosten bewertet sind.

4.7.1 Dijkstra-Algorithmus

Beschreibung:

Der Algorithmus ist nach seinem holländischen Erfinder Edsger W. Dijkstra benannt und löst das Problem der kürzesten Pfade. Der Dijkstra-Algorithmus wird verwendet, um anhand der Kosten der einzelnen Kanten den günstigsten Weg vom Startknoten zu allen anderen Knoten des Graphen zu berechnen. Ausgehend von einem Startknoten werden die Nachbarknoten verarbeitet und von diesen wiederum die Nachbarknoten, bis schließlich alle Punkte besetzt wurden. Die Kosten des Pfades werden dabei aufsummiert und für jeden besetzten Punkt gespeichert.

Am Anfang werden die Kosten zu allen Knoten auf unendlich gesetzt, da zu diesen zunächst noch kein Weg bekannt ist. Die Kosten des Startknotens werden auf null gesetzt. Der Knoten, der die geringsten Kosten aufweist, ist der nächste Knoten, der besetzt wird und von dem die weiteren Berechnungen ausgehen. Im ersten Schritt ist das der Startknoten. Jeder Knoten kann nur einmal besetzt werden. Ausgehend vom besetzten Knoten werden die Kosten zu den Nachbarknoten laufend verbessert, sollte folgende Bedingung zutreffen: „Wurde der (Nachbar-)knoten noch nicht besetzt und sind die Kosten über die neue Kante geringer als die bisherigen Kosten?“. Die Kosten der Knoten ergeben sich immer aus der Summe der Kosten zu seinem Vorgängerknoten und den Kosten der aktuellen Kante. Die einmal berechneten Kosten zwischen Startpunkt und einem bereits besuchten Knoten werden nicht mehr geändert. Allerdings garantiert der Algorithmus, dass beim Erreichen eines Knotens kein kürzerer Pfad zu ihm existieren kann, als der zuvor berechnete. Die Kosten zu den noch nicht besetzten Knoten können sich jedoch laufend verringern.

Bei jeder Aktualisierung der Kosten wird der aktuell besetzte Knoten als Vorgängerknoten für jenen (Nachbar-)knoten gespeichert. Damit kann am Ende der Berechnung der schnellste Pfad zu jedem Knoten konstruiert werden. Da immer der Vorgänger gespeichert wird, erfolgt die Konstruktion in umgekehrter Reihenfolge, also vom Zielknoten zum Startknoten. Letzterer hat keinen Vorgänger.

Der Algorithmus wird in der Regel so lange durchgeführt, bis alle Knoten besetzt sind. Um die Performance zu verbessern, kann auch eine Abbruchbedingung implementiert werden, z.B. wenn ein oder mehrere Punkte erreicht bzw. besetzt wurden.

Es wird zwischen zwei Varianten unterschieden:

- single-source shortest path:
  Die Kosten zwischen Startpunkt und allen Knoten werden berechnet.
- single-pair shortest path
  Die Berechnung wird solange durchgeführt, bis ein definierter Zielknoten erreicht wird.

Negative Kantenkosten sind mit diesem Algorithmus nicht möglich.

Quellen: (m9.ma.tum.de - Der Dijkstra-Algorithmus 2014) und (HOFMANN-WELLENHOF, LEGAT, and WIESER 2003).

Laufzeit des Dijkstra-Algorithmus:

Es wird angenommen, dass der Algorithmus auf einem Graphen mit n Knoten und m Kanten ausgeführt wird. Wenn jeder Knoten des Graphen besetzt wird, gibt es mindestens n Schritte. Da aus allen n Knoten derjenige mit der geringsten Distanz auszuwählen ist, braucht der Algorithmus also $n \times n$ Einzelschritte.

Bei jeder Knotenbesetzung werden die Nachbarknoten der ausgehenden Kanten betrachtet und anhand der Kosten entschieden, ob einer dieser Knoten als nächstes besetzt wird. Dadurch werden alle Kanten des Graphen betrachtet, was wiederum bedeutet, dass der Dijkstra-Algorithmus weitere m Einzelschritte benötigt. Die Gesamtlaufzeit des Algorithmus liegt also in der Größenordnung $m + n^2$. (m9.ma.tum.de - Der Dijkstra-Algorithmus 2014)

Pseudocodes:

```
FUNCTION Dijkstra(graph, source)

# Initialisierung
FOR EACH vertex i IN graph
    d(i) → ∞
    p(i) → none (NaN)
END

d(source) = 0

# Verarbeitung der Knoten
WHILE sum(T ≠ none) > 0
        v_i = v_j in T with smallest d
        T(v_i) = none (NaN)

         FOR EACH v_j OF N(v_i)
                costs = d_i + c_ij

                IF costs < d_j
                        d_j = costs
                        p_j = v_i
                END
        END
END
```

In Anlehnung an: (gitta.info - Dijkstra Algorithm: Short terms and Pseudocode 2015) und (m9.ma.tum.de - Der Dijkstra-Algorithmus 2014)

```
# Dijkstra Algorithmus
# Eingabe: Graph, Startknoten (source), (Zielknoten bei single-pair shortest path)

# Initialisierung - Vorbereitung des Graphen:
für alle knoten des Graphen:
    setze Kosten (d) auf unendlich
    setze Vorgänger (p) auf none

setze Kosten (d) von Startknoten auf 0
```

```
# Verarbeitung der Knoten:
SOLANGE noch Knoten in Liste der zu verarbeitenden Knoten (T):
    bestimme den Knoten mit den geringsten Kosten (aus d) → minKnoten
    entferne minKnoten aus Liste der zu verarbeitenden Knoten (T)

    für alle Nachbarknoten von minKnoten:
        neue Kosten = Kosten von minKnoten + Kosten der Kante von minKnoten
        zu Nachbarknoten (c)

        WENN neue Kosten < Kosten von Nachbarknoten (d):
                setze Kosten von Nachbarknoten (d) auf neue Kosten
                setze Vorgänger von Nachbarknoten (p)  auf minKnoten

# Ausgabe:
Kostenliste (d) für jeden Knoten,
kürzester Pfad (anhand einer Liste mit Vorgänger p)

Die Gesamtkosten vom Startpunkt zu den jeweiligen Punkten ergeben sich aus
Kostenliste (d).
```

In Anlehnung an: (inf-schule.de - Der Algorithmus von Dijkstra 2014)

In Tabelle 4 sind die Variablen der Pseudocodes beschrieben.

Tabelle 4 – Beschreibung der Variablen

Kürzel	Beschreibung	Dimension
$\boldsymbol{d}$	Liste mit Kosten zu allen anderen Knoten, Startknoten hat Kosten 0, nicht besetzte Knoten am Beginn sind ∞	$1 \times n$
$\boldsymbol{p}$	Liste mit jeweiligen Vorgängerknoten, Startknoten hat keinen Vorgänger	$1 \times n$
$\boldsymbol{T}$	Liste aller Knoten des Graphen, besetzte Knoten werden auf none gesetzt	$1 \times n$
$\boldsymbol{N}$	Liste der Nachbarknoten	$1 \times \ldots$
$\boldsymbol{c_{ij}}$	Kosten der Kanten zu den Nachbarknoten	$1 \times \ldots$
$\boldsymbol{v}$	Knoten ID	-
$\boldsymbol{n}$	Anzahl der Knoten	

Beispiel in Anlehnung an das Buch (HOFMANN-WELLENHOF, LEGAT, and WIESER 2003) und an (WIESER 2013):

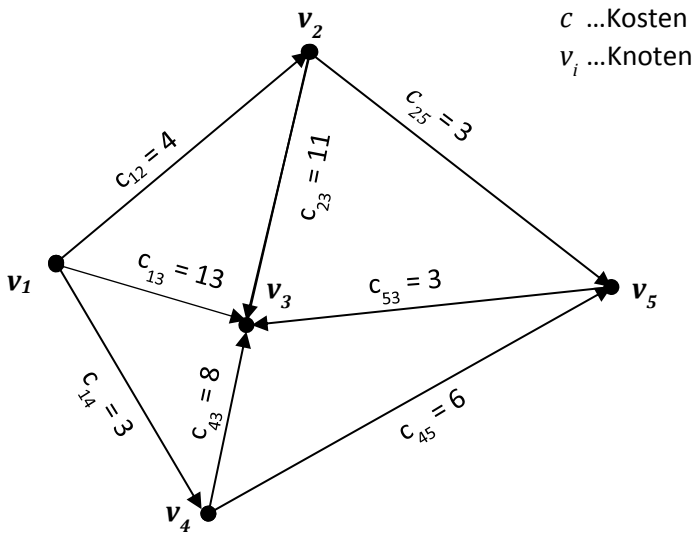

Abbildung 14 - Gerichteter Graph, eigene Darstellung nach (HOFMANN-WELLENHOF, LEGAT, and WIESER 2003)

Tabelle 5 – Liste der Kosten

d						
#	***v***	v_1	v_2	v_3	v_4	v_5
1	**-**	0	∞	∞	∞	∞
2	v_1	*0*	4	13	3	∞
3	v_4	*0*	4	11	*3*	9
4	v_2	*0*	*4*	11	*3*	7
5	v_5	*0*	*4*	10	*3*	*7*

Tabelle 6 – Liste der Vorgänger

p						
#	***v***	v_1	v_2	v_3	v_4	v_5
1	**-**	-	-	-	-	-
2	v_1	-	v_1	v_1	v_1	-
3	v_4	-	v_1	v_4	v_1	v_4
4	v_2	-	v_1	v_4	v_1	v_2
5	v_5	-	v_1	v_5	v_1	v_2

Im Beispiel (siehe Abbildung 14, Tabelle 5 und Tabelle 6) wird der Knoten v_1 als Startpunkt gewählt.

- Im 1. Schritt, der sogenannten Initialisierung, werden die Kosten der Liste d für jeden Knoten auf unendlich gesetzt. Die Kosten des Knoten v_1 bekommen den Wert 0.
- Aufgrund dessen, dass alle anderen Werte in d höher sind, wird v_1 als erstes besetzt (2. Schritt). Als nächstes werden die Kosten zu jedem Nachbarknoten berechnet. Für den Startpunkt gilt, dass in Liste d die Gesamtkosten zu den Nachbarknoten gleich groß sind, wie die Kosten der jeweiligen Kante (c_{ij}). Für alle Nachbarknoten kann in der Liste der Vorgängerknoten (p) der Knoten v_1 eingetragen werden.
- Da v_4 der Knoten mit den geringsten Kosten in Liste d ist, wird dieser als nächstes besetzt. Anzumerken ist hier, dass alle Knoten, die bereits besetzt wurden, von der Knotenliste gestrichen werden und nicht mehr als Kandidaten in Betracht gezogen werden können. Bei der Aktualisierung der Liste d müssen ab dem 3. Schritt die Kosten des aktuell besetzten Knotens zu den Kosten der jeweiligen Kante (c_{ij}) addiert werden (Beispiel - Für den Nachbarknoten v_3 gilt: $d[v_3] = d[v_4] + c_{43}$). Sollten die neu berechneten Gesamtkosten geringer ausfallen, als zuvor berechnet, werden diese in der Liste d aktualisiert. Sollten sich die Kosten ändern, kommt es gleichzeitig auch zu einer Aktualisierung des Vorgängerknotens in Liste p.
- Die Schritte 4 und 5 erfolgen analog zu Schritt 3, wobei die Berechnung nach dem 5. Schritt beendet wird, da alle Knoten abgearbeitet wurden. Anhand der Listeneinträge in d des 5. und letzten Schrittes kann man die jeweiligen Gesamtkosten vom Startknoten v_1 zu den jeweiligen Knoten im Beispiel entnehmen. Die Reihenfolge der kürzesten Route kann mit Hilfe der Liste der Vorgängerknoten (p) des letzten Schrittes nachvollzogen werden. Für den Knoten v_3 wäre das $v_3 \rightarrow v_5 \rightarrow v_2 \rightarrow v_1$, wobei anzumerken ist, dass der Beginn immer beim Zielknoten erfolgt.

4.7.2 A*-Algorithmus

Allgemeines:

Der A*-Algorithmus ist ein Algorithmus zur Bestimmung des kürzesten Weges zwischen zwei Knoten. Bei diesem Algorithmus spricht man auch von einem heuristischen Algorithmus bzw. einem informierten Suchverfahren. Der A*-Algorithmus verwendet im Gegensatz zum Dijkstra-Algorithmus eine Schätzfunktion, um zielgerichtet suchen zu können. Durch das Miteinbeziehen von zusätzlichen Informationen können jene Knoten ausgewählt werden, die wahrscheinlich schneller zum Ziel führen. Damit kann die Laufzeit für die Berechnung des schnellsten Weges zwischen zwei Knoten verringert werden.

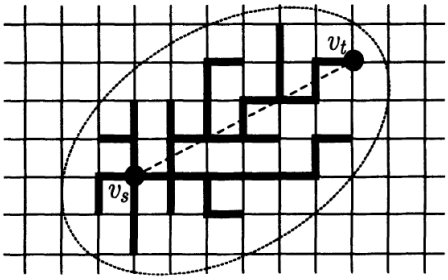

Abbildung 15 – Heuristisches Suchverfahren (HOFMANN-WELLENHOF, LEGAT, and WIESER 2003)

Charakteristisch für diesen Algorithmus ist, dass das Suchfeld sich nicht in alle Richtungen gleich ausbreitet und stattdessen zum Zielpunkt orientiert ist (siehe Abbildung 15). Die Schätzfunktion ist ein Gewicht, das für jeden Knoten die direkten Kosten, die sich aus der Luftlinie zum Zielknoten berechnen, angibt. Damit kann jener Knoten als nächster ausgewählt werden, der sich am nächsten zum Zielknoten befindet. Die berechnete Strecke zum Zielpunkt ist im Normalfall nie kürzer als die direkte Verbindung.

(m9.ma.tum.de - Der A*-Algorithmus 2014)

Berechnung:

Jedem Knoten *j* werden Kosten *f(j)* zugeordnet:

Formel 2 – Geschätzte Kosten (HOFMANN-WELLENHOF, LEGAT, and WIESER 2003)

$$f(j) = g(j) + h(j)$$

- *g(j)*: Kosten vom Startknoten bis Knoten *j*
- *h(j)*: Geschätzte Kosten vom Knoten *j* bis zum Zielknoten

Der Algorithmus baut auf zwei Listen auf, der OPEN-Liste, in welcher die besetzten Knoten enthalten sind, deren Nachfolger noch nicht untersucht wurden und der CLOSED-Liste, welche die Knoten enthält, deren Nachfolger bereits ermittelt wurden. Beide sind am Beginn der Berechnung leer. Gleich wie beim Dijkstra-Algorithmus wird mit dem Startknoten gestartet. Dabei wird im ersten Schritt dieser Knoten zur OPEN-Liste hinzugefügt und dessen Kosten auf 0 gesetzt.

Anschließend werden alle Knoten so lange besetzt, bis der Zielknoten erreicht wird. Der nächste Knoten ist immer jener mit den geringsten Gesamtkosten d. Darauf wird jener Knoten aus der OPEN-Liste entfernt und dessen Nachbarknoten werden untersucht. Neben der Berechnung der Kosten zum Nachbarknoten über den kostengünstigsten Weg spielt auch der heuristische Kostenanteil h eine wichtige Rolle, der sich aus der direkten Distanz (Luftlinie) zum Zielknoten ergibt. Sollte sich der Nachbarknoten bereits in einer der Listen befinden und die neu berechneten Kosten f (Formel 2) geringer sein, als deren aktuelle Kosten *d(j)*, wird der Nachbarknoten *j* der OPEN-Liste hinzugefügt. Ansonsten wird der Knoten verworfen. Am Ende der Untersuchung der Nachbarn wird der aktuell besetzte Knoten der CLOSED-Liste hinzugefügt. Weitere Informationen zur Berechnung sind unter (web.mit.edu - Pathfinding using A* 2002) und (HOFMANN-WELLENHOF, LEGAT, and WIESER 2003) nachzulesen.

Laufzeit:

Die Laufzeit des Algorithmus ist abhängig von dessen verwendeter Heuristik-Funktion. In jedem Fall besitzt der A*-Algorithmus eine bessere Laufzeit als der Dijkstra-Algorithmus. Die Komplexität liegt bei O(n^3). Die Stärke des Algorithmus liegt jedoch am zielgerichteten Suchen, da dadurch nur ein Bruchteil der Knoten des Graphen betrachtet werden muss (m9.ma.tum.de - Der A*-Algorithmus 2014).

Im Gegensatz zum Dijkstra-Algorithmus muss dieser Algorithmus für die Berechnung der kürzesten Wege zwischen Startknoten und allen anderen Knoten des Graphen entsprechend oft ausgeführt werden.

Pseudocodes:

```
FUNCTION A_star(graph, source, target)

# Initialisierung
init OPEN_list
init CLOSED_list

OPEN_list.add(node = source, d = 0)

# Verarbeitung der Knoten
WHILE length(OPEN_list) > 0
        d_min = smallest d in OPEN_list
        v_i = node in OPEN_list where d = d_min

        OPEN_list.delete(node = v_i)

         FOR EACH v_j OF N(v_i)
                IF v_j == target
                        "Ziel erreicht"
                        break;
                ELSE
                        g = c_ij
                        h = dist(v_i, source)
                        f  = g + h

                        IF v_i exist in OPEN_list AND d <= f
                                nothing
                        IF v_i exist in CLOSED_list AND d <= f
                                nothing
                        ELSE
                                OPEN_list.add(node = v_i, d = f)
                        END
                END
        END

        CLOSED_list.add(node = v_i, d = d_min)
END
```

Der Code basiert auf (web.mit.edu - Pathfinding using A* 2002)

```
# A* - Algorithmus

# Eingabe:
Graph, Startknoten, Zielknoten

# Initialisierung - Vorbereitung des Graphen:
initialisiere OPEN-Liste      → noch abzuarbeitende Knoten
initialisiere CLOSED-Liste    → bereits abgearbeitete Knoten

füge den Startknoten der open-Liste hinzu
setze die Kosten (d) des Startknotens auf 0

# Verarbeitung der Knoten:
SOLANGE der Zielknoten nicht erreicht ist:
        finde die geringsten Kosten (d) in der OPEN-Liste
        finde den Knoten in der OPEN-Liste, wo Kosten am geringsten → v
        entferne u aus der OPEN-Liste

        für alle Nachbarknoten:
                g = Summe der Kosten vom Startknoten über den
                kostengünstigsten Pfad
                h = direkte vom aktuellen Knoten (v) zum Zielknoten
                f = g + h

                WENN sich der gleiche Knoten bereits in der OPEN-Liste
                befindet und seine Kosten d geringer oder gleich hoch sind
                als f:
                        verwerfe den neuen Knoten
                WENN sich der gleiche Knoten bereits in der CLOSED-Liste
                befindet und seine Kosten d geringer oder gleich hoch sind
                als f:
                        verwerfe den neuen Knoten.
                SONST
                        füge aktuellen Knoten der OPEN-Liste hinzu (node)
                        aktuelle Kosten in OPEN-Liste speichern (d)

        füge u der CLOSED-Liste hinzu (node)
        aktuelle Kosten (d_min) des Knotens in CLOSED-Liste speichern (d)

# Ausgabe:
Kosten (Zielknoten),
kürzester Pfad (aus CLOSED-Liste)
```

Der Code basiert auf (web.mit.edu - Pathfinding using A* 2002)

In Tabelle 7 und Tabelle 8 sind die Listen und Variablen der Pseudocodes beschrieben.

Tabelle 7 – Beschreibung der Listen

Kürzel	Beschreibung
OPEN_list	Liste der besetzten Knoten, deren Nachfolger noch nicht untersucht wurden, Attribute: node, d (Kosten)
CLOSED_list	Liste der Knoten deren Nachfolger bereits ermittelt wurden, Attribute: node, d (Kosten)

Tabelle 8 – Beschreibung der Variablen

Kürzel	Beschreibung
N	Liste der Nachbarknoten
f	Gesamtkosten
g	Kosten vom Startknoten bis Knoten j
h	Geschätzte (heuristische) Kosten vom Knoten j bis zum Zielknoten
c	Kosten der Kanten zu den Nachbarknoten
d	Liste mit Kosten zu allen anderen Knoten
$node$	Knotenliste
v	Knoten ID

4.7.3 Weitere Graphen-Algorithmen

- **Bellman-Ford-Algorithmus:**

  Im Gegensatz zu den beiden vorherigen Algorithmen berechnet der Bellman-Ford-Algorithmus auch kürzeste Wege, wenn negative Kantengewichte gegeben sind (m9.ma.tum.de - Graphalgorithmen 2014).

- **Floyd-Warshall-Algorithmus:**

  Der Algorithmus von Floyd-Warshall berechnet den kürzesten Weg zwischen allen Paaren von Punkten. Er funktioniert ebenfalls bei negativen Gewichten (m9.ma.tum.de - Graphalgorithmen 2014).

- **Bidirektionale Suche:**

  Bei diesem Suchverfahren (siehe Abbildung 16) werden zwei Berechnungen simultan durchgeführt (z.B. mittels Dijkstra-Algorithmus). Dabei werden Start- und Zielknoten als Ausgangsknoten verwendet. Die Abbruchbedingung des Verfahrens ist erreicht, wenn zwei idente Knoten besetzt wurden und damit eine durchgehende Verbindung zwischen beiden Ausgangspunkten möglich ist. Bei diesem Verfahren kann man im Idealfall eine Zeitersparnis von 50% erreichen (Beweis siehe Formel 3). Dabei ist es jedoch unbedingt erforderlich, dass die Algorithmen parallel arbeiten. Die Informationen basieren auf (HOFMANN-WELLENHOF, LEGAT, and WIESER 2003).

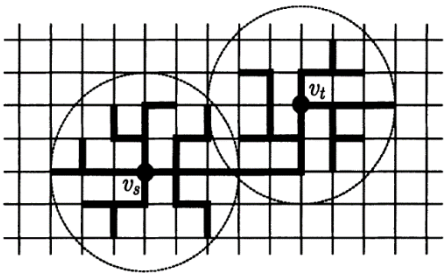

Abbildung 16 – Bidirektionales Suchverfahren (HOFMANN-WELLENHOF, LEGAT, and WIESER 2003)

Formel 3 – Kreisfläche (WIESER 2013)

$$F_{1\ Kreis} = r^2 \cdot \pi \qquad F_{2\ Kreise\ (Bi)} = 2 \cdot \left(\frac{r}{2}\right)^2 \cdot \pi = \frac{r^2}{2} \cdot \pi$$

$$F_{2\ Kreise\ (Bi)} < F_{1\ Kreis}$$

4.8 Datenbanken

Eine Datenbank ist eine selbstständige und für den flexiblen und sicheren Gebrauch ausgelegte Datenorganisation zur systematischen Ansammlung von Daten. Diese wird dazu verwendet, um große Datenmengen strukturiert zu speichern und zu verwalten (wirtschaftslexikon.gabler.de - Datenbanken 2015). Ein Datenbanksystem (DBS) besteht aus zwei wesentlichen Teilen, basierend auf (reeg.junetz.de - Datenbanksystem 2015):

- Datenbankverwaltungssystem (DBMS): Diese hochkomplexe Verwaltungssoftware besteht aus einer Vielzahl von Werkzeugen, um intern die strukturierte Speicherung zu organisieren und um Datenzugriffe zu kontrollieren. Es können auch mehrere Datenbanken zugleich verwaltet werden.
- Datenbank: Diese besteht aus einem zusammengehörigen Datenbestand und wird vom DBMS (siehe Abbildung 17) verwaltet.

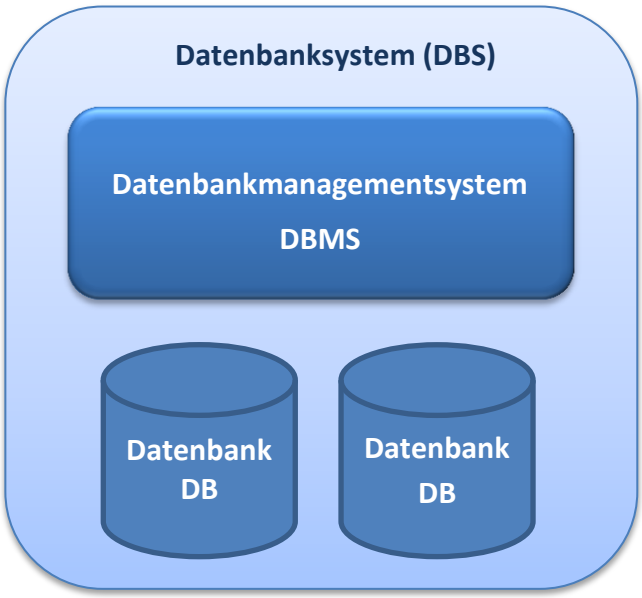

Abbildung 17 – DBS (Eigene Darstellung)

Es gibt verschiedene Arten von Datenbanksystemen, welche durch das Datenbankmodell festgelegt werden. Um die Verwaltung und den Zugriff auf die Daten zu realisieren, gibt es für jedes Datenbanksystem (DBS) eine Datenbanksprache.

Grundlegende Anforderungen an Datenbank-Systeme (sql-und-xml.de - Grundbegriffe und Konzepte von Datenbank-Systemen 2015):

- Vermeiden von Redundanzen:
  Informationen sollten nicht mehrfach abgelegt werden.
- Sicherstellung einer maximalen Integrität der Daten:
  Alle abzulegenden Informationen zerfallen in kleinste Einheiten, die vielfachen Einschränkungen unterliegen können.
- Datensicherheit:
  Der Schutz gegen Datenverlust sowie Schutz gegen unerlaubten Zugriff wird gewährleistet.

- Transaktionen:
  Regelung des parallelen Zugriffs mehrerer Benutzer.

4.8.1 Datenbankmodell

Die Grundlage für die Strukturierung der Daten und deren Beziehungen untereinander wird durch das Datenbankmodell festgelegt. Dabei gibt es folgende Modelle (datenbanken-verstehen.de - Datenbankmodel 2015):

- Relationales Datenbankmodell
- Objektorientiertes Modell
- Objektrelationales Modell
- Hierarchisches Modell
- Netzwerk-Modell
- Dokumentorientiertes Modell

4.8.2 Relationales Datenbankmodell

Dieses tabellenbasierte Modell speichert die Daten zeilenweise und ist das am weitesten verbreitetste Modell.

Eine solche Datenbank besteht aus einer Sammlung von Tabellen (Relationen) und Beziehungen, die über Schlüssel miteinander verknüpft sein können (siehe Abbildung 18). Jede Zeile (Tupel) einer Tabelle entspricht einem Datensatz, wobei jedes Tupel aus mehreren Spalten (Attributen) besteht. Das Relationsschema einer Tabelle legt die Anzahl und den Typ der Attribute fest.

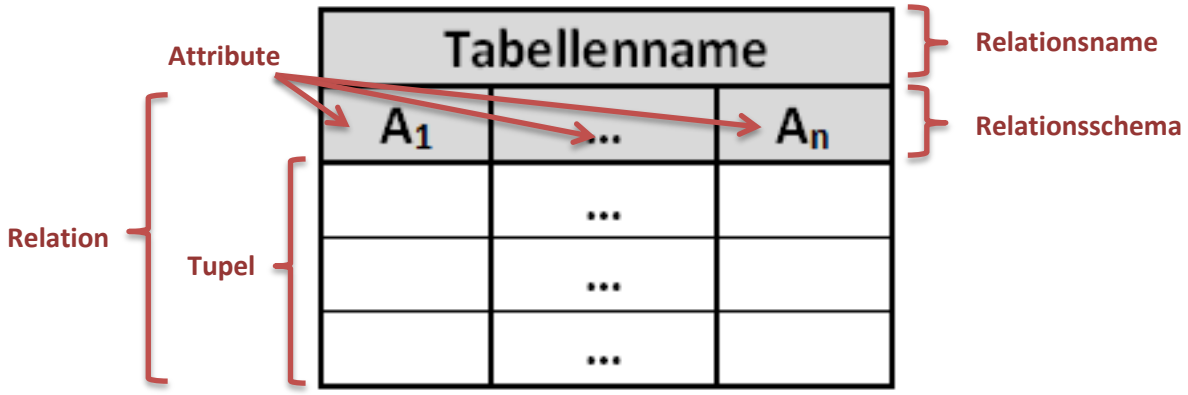

Abbildung 18 – Bestandteile einer DB (Eigene Darstellung)

Die Operationen auf diese Relationen werden durch die relationale Algebra bestimmt, wobei diese auch Grundlage für die Datenbanksprache SQL ist.

Relationale Datenbanken bauen auf atomaren Attributen auf. Dadurch wird das Arbeiten mit raumbezogenen Daten, die zwei oder drei Dimensionen haben, erschwert. Als Alternative werden spezielle Geodatenbanken bzw. Erweiterungen angeboten, die Geometriedatentypen unterstützen. Die Informationen beziehen sich auf (datenbanken-verstehen.de - Relationales Datenbankmodell 2015).

Normalisierung

Darunter versteht man die Aufteilung von Attributen auf mehrere Relationen (Tabellen). Dies wird unter Berücksichtigung der Normalformen (NF) realisiert, um Redundanzen zu vermeiden. Außerdem sollen dadurch funktionelle und transitive Abhängigkeiten vermieden werden. Das Ziel ist ein klar strukturiertes Datenbankmodell.

1. Normalform:
   Eine Relation (Tabelle) befindet sich in der 1. NF, wenn die Wertebereiche aller Attribute der Tabelle atomar sind. Ein Wert ist atomar, wenn dieser nicht mehr weiter in seine Bestandteile zerlegbar ist. Beispiel: Trennung des Namens in Vor- und Nachname.

2. Normalform:

   Eine Relation (Tabelle) befindet sich genau dann in der 2. NF, wenn diese sich in der 1. NF befindet und jedes Nichtschlüsselattribut von jedem Schlüsselkandidaten voll funktional abhängig ist.

3. Normalform:
   Eine Relation (Tabelle) befindet sich genau dann in der 3. NF, wenn diese sich in der 2. NF befindet und kein Nichtschlüsselattribut transitiv von einem Kandidatenschlüssel abhängig ist.

Die 3. NF ist das Ziel einer erfolgreichen Normalisierung, da sie einerseits Anomalien und Redundanzen verhindert, aber auch ausreichend Performance für SQL-Abfragen und noch genügend Flexibilität für den Entwickler bietet. Nähere Informationen zum Thema Normalisierung sind unter (datenbanken-verstehen.de - Normaliserung 2015) zu finden.

Schlüssel

- Primärschlüssel (Primary Key):

  Dieser wird zur eindeutigen Identifizierung eines Datensatzes verwendet. In einer normalisierten Datenbank besitzen alle Tabellen einen eindeutigen Primärschlüssel. Sollte ein Datensatz anhand eines Attributes nicht eindeutig identifizierbar sein, ist es auch möglich, den Primärschlüssel aus mehreren Attributen zusammen zu setzen (z.B.: Vor- und Nachname). Dabei muss sichergestellt werden, dass jede dieser Kombinationen nur einmalig auftritt. Ansonsten wird ein künstlicher Schlüssel angelegt, der aus einer fortlaufenden Nummer besteht. Im Falle, dass mehrere Attribute der Relation eindeutig sind, werden diese Attribute als Schlüsselkandidaten bezeichnet (datenbanken-verstehen.de - Primärschlüssel 2015).

- Fremdschlüssel (Foreign Key):

  Dabei handelt es sich um eine Spalte, die auf einen Primärschlüssel einer anderen Tabelle verweist (datenbanken-verstehen.de - Fremdschlüssel 2015).

4.8.3 Objektorientiertes Datenbankmodell

In einem objektorientierten Datenbankmodell (OODB) werden Daten zusammen mit ihren Funktionen in einem Objekt gespeichert. Dieses Datenmodell basiert auf den Gesetzmäßigkeiten der objektorientierten Programmierung und vermeidet damit das wiederholte Zusammensuchen zusammengehörender Daten. Die Datenabfrage erfolgt über Funktionen des Objektes oder über eine SQL-ähnliche Sprache (OQL, Object Query Language).

Da die Verbreitung objektorientierter Programmierung die Integration der Daten in relationale Datenbanken zunehmend erschwert, gewinnt dieses Modell zunehmend an Bedeutung. Der Einsatz einer Objektdatenbank ist wesentlich effizienter, weil die aufwendige Verknüpfung von Relationen wegfällt. Außerdem können multimediale Inhalte und komplexere Datentypen integriert werden.

Auf Grund der geringen Verbreitung von OODB gibt es dafür jedoch nur wenige kompatible Schnittstellen. Die Informationen basieren auf (datenbanken-verstehen.de - Objektorientiertes Datenbankmodell 2015).

4.8.4 Hierarchisches Datenbankmodell

Dabei handelt es sich um das älteste Datenbankmodell, in dem die Daten in Form einer Baumstruktur dargestellt werden. Die Verknüpfungen werden über Eltern-Kind-Beziehungen realisiert. Dieses Modell ist jedoch sehr starr und bietet dem Entwickler wenig Freiheit. Im Bereich der XML-Entwicklung kann dieses Modell dagegen sehr effektiv genutzt werden (datenbanken-verstehen.de - Hierarchisches Datenbankmodell 2015).

4.8.5 Netzwerkdatenbankmodell

Im Vergleich zum relationalen Datenbankmodell besitzt das Netzwerkdatenbankmodell keine strenge Hierarchie. Ein Datenfeld besteht aus einem Namen und einem Wert. Der Vorteil dieses Modells ist, dass es unterschiedliche Lösungswege angibt. Das kann auch problematisch sein, wenn der Entwickler nach einem eindeutigen Lösungsweg sucht. Wenn das Modell ständig weiter wächst, verringert sich auch schnell die Übersichtlichkeit (datenbanken-verstehen.de - Netzwerkdatenbankmodell 2015).

4.8.6 Objektrelationale Datenbank

Eine objektrelationale Datenbank (ORDB) integriert die Konzepte einer klassischen relationalen Datenbank mit Paradigmen der objektorientierten Programmierung. Das Ziel dieser Synthese ist, die ausgereifte Technologie der relationalen Datenbanken auf die Organisation beliebiger Daten auszuweiten. Im Vergleich zu relationalen Datenbanken unterstützt dieses Datenbankmodell Datenobjekte und Operationen höherer Komplexität. Beispiele für Objekte wären: Bild- und Audiodateien oder Karten. Da die Verknüpfungen über Objekte erfolgen, können ORDB wesentlich effizienter sein. In der Geoinformation geht der Trend immer mehr in Richtung objektorientierter Ideen.

Von verschiedenen Herstellern wurde das relationale Modell um objektorientierte Konzepte ergänzt, ohne vorhandene Entwicklungen zu stören. So konnten für das ursprüngliche Datenbanksystem zusätzliche Erweiterungen bereitgestellt werden.

Nähere Informationen sind unter (itwissen.info - Objektrelationale Datenbank 2015) und (BARTELME 2005) zu finden.

4.9 PostgreSQL

PostgreSQL ist ein frei verfügbares objektrelationales Datenbankmanagementsystem (ORDBMS), das ohne Lizenzierung genutzt werden darf. Dieses Open-Source-Datenbanksystem ermöglicht die Speicherung nicht atomarer Daten, Vererbung und Objektidentitäten. Es erlaubt den Benutzern auch, das System durch selbstdefinierte Datentypen, Operatoren und Funktionen zu erweitern.

PostgreSQL basiert auf dem Client-Server-Modell, in dem ein Server die Dateien der Datenbank sowie die Verbindungen zum Client verwaltet. Außerdem werden die Anfragen der Client-Programme bearbeitet. Kommuniziert wird meist über eine TCP/IP-Verbindung, wobei ein PostgreSQL-Server Verbindungen zu mehreren Clients parallel verwalten kann.

Eine Tabelle einer Datenbank kann bis zu 32 Terabyte groß werden und kann eine unbegrenzte Anzahl an Datensätzen enthalten (Zeilen). Die Anzahl der Attribute (Spalten) ist jedoch je nach Datentyp auf 250 bis 1.600 begrenzt.

PostgreSQL bietet mächtige Werkzeuge an und kann über Module auch prozedurale Sprachen in den Server einbinden, wie z.B. die PostgreSQL-eigene Sprache PL/pgSQL. Zusätzlich gibt es auch noch C/C++, Python- und PHP-Bibliotheken sowie JDBC- und ODBC-Schnittstellen. Der Abschnitt basiert auf (postgresql.de - das freie objektrelationale Open Source Datenbanksystem 2015) und (c't zu Postgresql v7.0.2 2000).

4.10 Tiles

Unter Tiles (englisch für Kacheln) versteht man die mosaikartig zusammengesetzten Einzelteile, die gemeinsam ein vielfaches größeres Gesamtbild ergeben. Die Trennung in einzelne Kacheln ermöglicht eine einfachere Berechnung und einen geringeren Arbeitsspeicherverbrauch. Die Idee ist, hochauflösende Karten in Tiles zu unterteilen und einzelne Tiles client-seitig zwischenzuspeichern, da meist nur ein kleiner Kartenausschnitt benötigt wird. Die Rechen- und die Übertragungszeit des Bildes werden beim Kartenaufbau ausbalanciert. In Tabelle 9 sind die Vor- und Nachteile der Nutzung von tiled und untiled Quellen aufgelistet.

Tabelle 9 – Vor- und Nachteile von Tiled Layern

Layer	Vorteile	Nachteile
tiled	• Hohe Geschwindigkeit: Es müssen nur die Bilder übertragen werden • Client-seitiges Zwischenspeichern • Unterstützung bei großen Datenmengen: Jede Kachel besteht nur aus einem kleinen Bild, das den Browser nicht überladet. • Stabilität	• Hohe Server-Beanspruchung um die Tiles zu erzeugen • Hoher Speicherplatzverbrauch wenn Tiles regeneriert werden • Sichtbares Nachladen (Pausen) bei stufenlosem Zoom • Für jede Projektion muss ein eigenes Tileset erzeugt werden.
untiled	• Verminderter Speicherplatzverbrauch • Unterstützung von Mehrfachprojektionen • Rendering der Bildqualität für jede Maßstabsstufe und nicht nur auf definierten Stufen	• Kein Zwischenspeichern des Bildes - kurze Zeitverzögerung, weil Bild zuerst vom Server geladen werden muss • Hohe Datenlast auf der Serverseite bei einer großen Anzahl von Nutzern.

Zusammenfassung der Vor- und Nachteile aus Tabelle 9: Wenn eine hohe Performance und wenig Flexibilität gefordert sind, sollten Tiled-Raster verwendet werden. Sollten laufend aktuelle Daten benötigt werden und die Layer nur aus einer oder zwei Datenquellen bestehen, sind Untiled-Raster die bessere Wahl. Es ist auch möglich, beide Varianten zu kombinieren, indem man für bestimmte Zoomstufen Tiles verwendet und dazwischen Untiled-Inhalte.

Auflösung bei Tiled-Layern

Die Auflösung bei Tiled-Layern gibt die Anzahl der Projektionseinheiten pro Pixel an. Wenn der Benutzer in eine Karte hinein oder heraus zoomt, wechselt er in eine andere Zoomstufe. Das Zoomen ändert die Anzahl der Kacheln, die verwendet werden, um die Karte darzustellen (siehe dazu Abbildung 19). Jede Zoomstufe verdoppelt die Anzahl der Kacheln, in die die Karte aufgeteilt wird und halbiert die Auflösung. Zoom-Level 0 hat ein Tile (1x1) und Zoom-Level 1 hat vier Tiles (2x2). Zoom-Level 2 hat 16 Tiles (4x4) und so weiter. Für die Berechnung der Auflösung jeder Zoomstufe wird Formel 4 verwendet, für die man die Projektionsgrenzen (W) und die Zoomstufe (z) benötigt.

Formel 4

$$Auflösung = \frac{W}{2^z}$$

Rechenbeispiel:

Bildgröße:

$320 \times 360\ [km]$

Berechnung für Level 3:

$W_x = 320, W_y = 360$

x: $\frac{W_x}{2^z} = \frac{320}{2^3} = 40\ [km]$

y: $\frac{W_y}{2^z} = \frac{360}{2^3} = 45\ [km]$

Kachelauflösung:

$40 \times 45\ [km]$

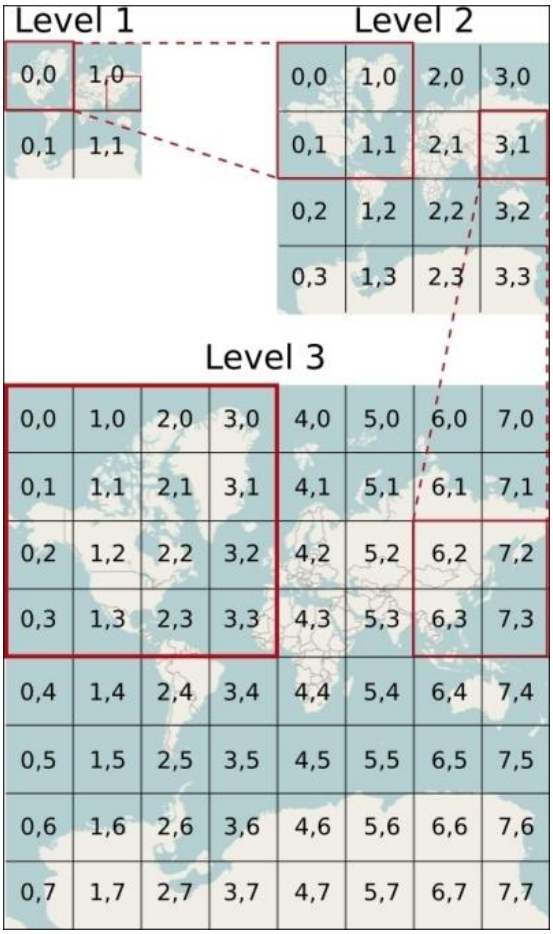

Abbildung 19 – Tiles (GRATIER, SPENCER, and HAZARD 2015)

Der Inhalt des Abschnitts basiert auf (GRATIER, SPENCER, and HAZARD 2015).

4.11 Koordinatensysteme

Koordinatensysteme sind Hilfsmittel, um die Position eines Punktes durch Koordinaten zu definieren. Durch Koordinaten lässt sich die Lage eines Punktes durch Zahlenwerte beschreiben. Jedes Koordinatensystem ist fix mit einem Objekt verbunden und wird durch Datum und Maßstab definiert. Dabei gibt das Datum an, wie das System im Raum verankert ist und der Maßstab, wie die Koordinatenachsen (meist zwei oder drei) skaliert sind. Ein digitales Bild wird durch zweidimensionale Bildkoordinaten beschrieben. Jene Koordinaten, die mit der Erde in Bezug stehen, werden als Weltkoordinaten bezeichnet.

Systeme mit Weltkoordinaten (siehe Abbildung 20) lassen sich unterteilen in geodätische (φ, λ) und kartesische (x, y, z) Koordinatensysteme.

Dreidimensionale kartesische (rechtwinklige) Koordinatensysteme kommen vor allem im Zusammenhang mit der Satellitenpositionierung vor. Um die Oberfläche der Erde mit zweidimensionalen kartesischen Koordinaten zu beschreiben, werden Projektionen verwendet (siehe 4.12).

Das geodätische (auch geografische) Koordinatensystem wird durch die geografische Länge λ und die geografische Breite φ definiert. Aufgrund der in Näherung kreisförmigen Form der Erde wird ein Kugelkoordinatensystem verwendet und die Zahlenwerte in der Einheit Grad angegeben. Dabei werden die Breitengrade vom Äquator aus gezählt und die Längengrade vom Nullmeridian, der durch Greenwich und die Erdachse definiert ist. Als dritte Dimension wird die Höhe verwendet, welche von unterschiedlichen Bezugsflächen aus definiert sein kann. Längen- und Breitenkreis stehen in jedem Punkt orthogonal aufeinander, wobei jeder Längenkreis (Meridian) gleich lang ist und der Umfang der Breitenkreise in Richtung der Pole abnimmt.

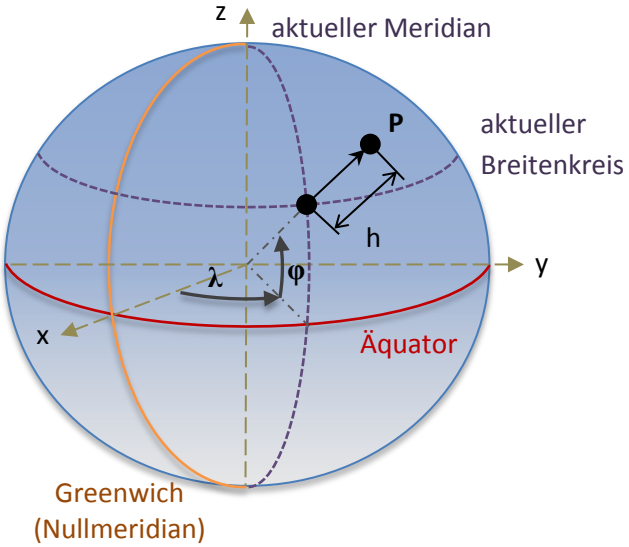

Abbildung 20 – Weltkoordinatensystem, eigene Darstellung nach (BARTELME 2005)

Da die Erde keine Kugel ist, sondern ein Geoid, haben die Koordinaten je nach Region unterschiedliche Referenzsysteme. Diese richten sich nach deren Referenzellipsoid, das wiederum einem regionalen Schwerefeld angepasst wird. International wird heute meist das World Geodetic System 1984 (WGS84) als Referenzsystem benutzt. Durch die Definition eines Bezugsellipsoids wird die Realisierung eines geodätischen Datums möglich.

Ein Bildkoordinatensystem wird durch zweidimensionale Koordinaten (X, Y) definiert und hat meist links oben/unten seinen Ursprung (0,0). Die Einheit wird in Pixel angegeben. Durch die Detektion von Passpunkten mit Weltkoordinaten können diese Bildkoordinaten georeferenziert werden.

Detaillierte Informationen über Koordinatensysteme sind unter (kartenkunde-leichtgemacht.de - Koordinatensysteme 2015) zu finden.

4.12 Projektionen

4.12.1 Allgemeines

Die Projektion legt fest, wie die reale Welt, welche in Näherung ein Ellipsoid ist, im zweidimensionalen Raum abgebildet wird. Aufgrund der Erdkrümmung ist die Abbildung nur mit Einschränkungen möglich. Durch den Abbildungsprozess kommt es immer zur Verfälschung der Konformität, Entfernung, Richtung und/oder Größe (kowoma - Kartenprojektionen 2007). Je größer das abzubildende Gebiet ist, desto gravierender wirken sich diese Fehler aus. Bei der Abbildung der Erde in die Ebene wird versucht, eine oder mehrere der folgenden Eigenschaften anzustreben – basierend auf (arcgis.com - Kartenprojektionen 2015) und (kowoma - Kartenprojektionen 2007):

- **Winkeltreue (Konformität):** Eine Abbildung ist winkeltreu bzw. konform, wenn die Richtungswinkel (Azimute) eines Punktes in alle Richtungen richtig abgebildet werden. Dabei entsteht ein krummliniges Koordinatennetz (siehe Abbildung 21). In konformen Karten können Winkel korrekt abgelesen werden, was für die Luft- und Seenavigation von großer Bedeutung ist.

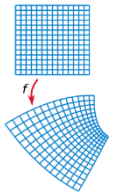

Abbildung 21 – Konforme Abbildung (wikipedia - Konforme Abbildungen 2015)

- **Längentreue:** Längentreue Abbildungen bewahren die Entfernung zwischen bestimmten Punkten. Der Längenmaßstab kann jedoch nicht überall auf der Karte korrekt beibehalten werden. In den Abbildungen gibt es meist ein oder mehrere Meridiane, in denen die Längentreue ordnungsgemäß beibehalten wird. Eine generell längentreue Abbildung der Kugel in die Ebene ist praktisch nicht umsetzbar.

- **Flächentreue:** Eine Abbildung ist flächentreu, wenn alle Flächen auf der Karte das gleiche Verhältnis zueinander haben wie auf der Erde. Meist wird diese Eigenschaft vernachlässigt.

Es gibt zahlreiche Projektionen für unterschiedliche Anwendungen. Drei der häufigsten Projektionen (siehe Abbildung 22) sind Zylinder- und Kegel-Projektion sowie die Azimutale Projektion, die Abbildung in die Ebene.

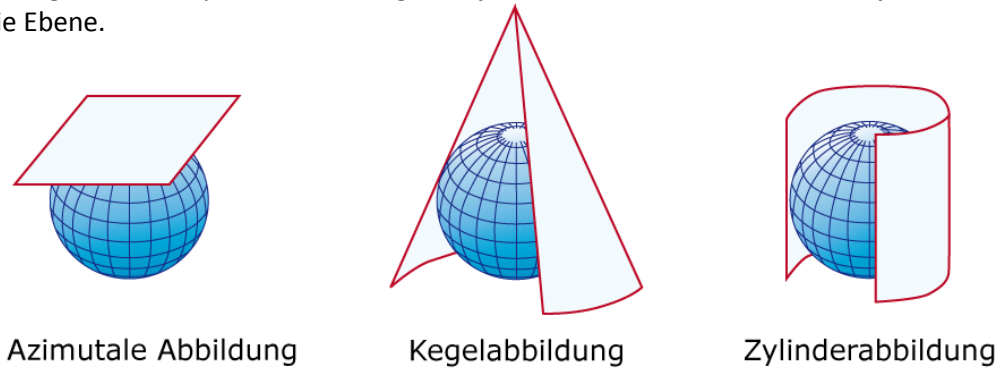

Abbildung 22 – Projektionen (artalis.de - Projektionen 2015)

Es wird nach der Lage der Transformationsfläche (siehe Abbildung 23) unterschieden, wobei diese normal, transversal oder schiefachsig sein kann.

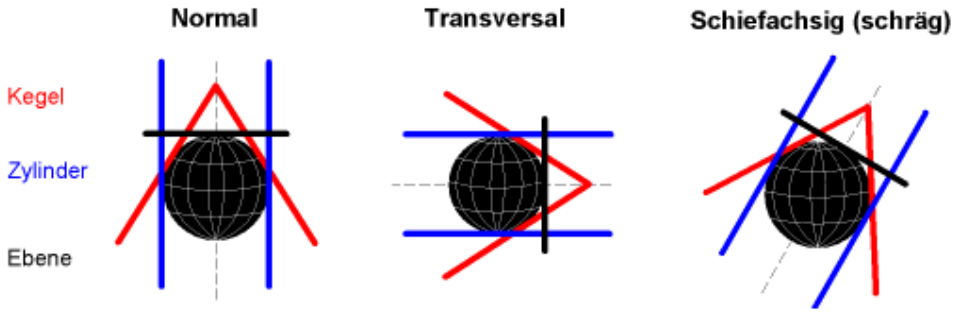

Abbildung 23 – Lage der Transformationsfläche (kartenkunde-leichtgemacht.de - Kartennetzentwurf 2015)

Die Abbildungsfläche muss die Erdkugel nicht zwangsläufig berühren, sondern kann diese auch schneiden, man spricht dann von einer orthogonalen Schnittprojektion (siehe Abbildung 24).

Abbildung 24 – Abbildungsfläche (kartenkunde-leichtgemacht.de - Kartennetzentwurf 2015)

4.12.2 Normale Mercatorprojektion

Diese entspricht einer konformen Zylinderprojektion, in der der Zylinder die Erdoberfläche am Äquator berührt. Die Erdoberfläche ist in erster Näherung eine Kugeloberfläche, die nicht verzerrungsfrei auf eine ebene Karte abgebildet werden kann. Aufgrund ihrer Beschaffenheit hat diese Projektion starke Verzerrungen an den Polen, welche Richtung Äquator geringer werden (siehe dazu Zellgrößen Abbildung 25). Der Äquator wird längentreu abgebildet. Der Vorteil dieser Projektion ist, dass Kurslinien über längere Distanzen als Gerade abgebildet werden können. Die Mercator-Projektion kommt für eine Abbildung der Polregionen nicht in Betracht.

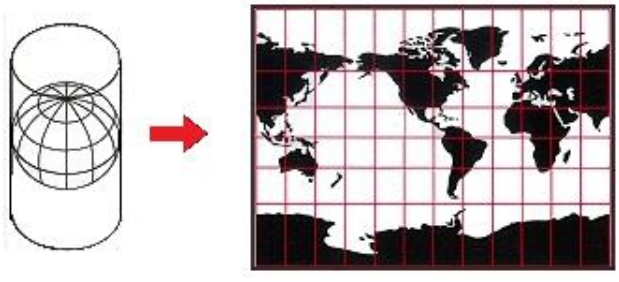

Abbildung 25 – Normale Mercatorprojektion (GRATIER, SPENCER, and HAZARD 2015)

Nähere Information sind unter (olanis.de - Kartenprojektionen 2015), (artalis.de - Mathematische Grundlagen 2015), (kowoma - Kartenprojektionen 2007) und (kartenkunde-leichtgemacht.de - Kartennetzentwurf 2015) zu finden.

4.12.3 Transversale Mercatorprojektion

Diese ist die bedeutendste Projektion und ist auch als Gauß-Krüger-Projektion bekannt (siehe Abbildung 26). Dabei werden kleine Abschnitte der Erde mit einem festgelegten Meridianstreifensystem auf einen querachsigen (transversalen) Zylinder übertragen. Einer der zwei Berührungslinien wird als Haupt- oder Zentralmeridian bezeichnet. Da bei der Zylinderprojektion die Verzerrung dort am geringsten ist, wo der Zylinder die Erde berührt, ist dieser Meridian längentreu. Auf beiden Seiten des Bezugsmeridians wird durch die Meridianstreifen eine Fläche definiert und auf dem Zylinder abgebildet. Je schmaler der abzubildende Meridianstreifen gewählt wird, desto geringer fallen die Verzerrungen aus. Mit wachsendem Abstand zum Bezugsmeridian nehmen diese Verzerrungen zu und daher wird diese Art der Projektion nur für sehr schmale Streifen verwendet (3°).

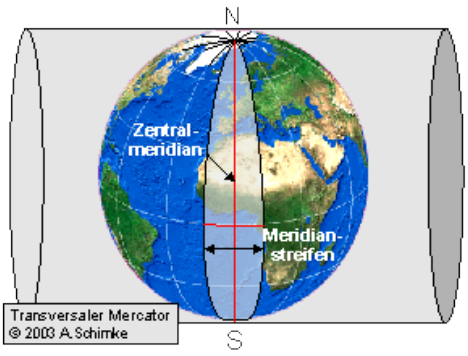

Abbildung 26 – Transversale Mercatorprojektion (olanis.de - Kartenprojektionen 2015)

Bei der UTM-Projektion wird die Erde durch einen Zylinder rechts und links des Zentralmeridians geschnitten (siehe Abbildung 27), so dass diese zwei Schnittlinien in Nord-Süd-Richtung längentreu sind. Für den Bezugsmeridian wird ein Skalierungsfaktor (0.9996) verwendet. Im Vergleich zur Gauß-Krüger-Projektion sind bei UTM die Streifen mit einer Breite von 6° doppelt so groß. Da UTM das GRS80-Ellipsoid verwendet, auf welches sich auch das geodätische Datum zur einheitlichen Landesvermessung bezieht, findet diese Projektion immer mehr Anwender. Im Vergleich dazu bezieht sich die Gauß-Krüger-Projektion in Österreich auf das Bessel-Ellipsoid.

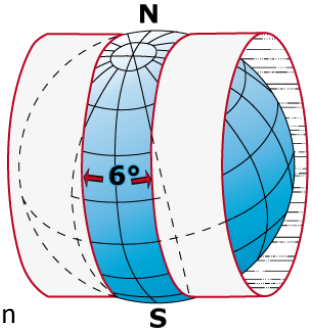

Abbildung 27 – UTM-Projektion (artalis.de - Mathematische Grundlagen 2015)

Nähere Information sind unter (olanis.de - Kartenprojektionen 2015), (artalis.de - Mathematische Grundlagen 2015), (kowoma - Kartenprojektionen 2007) und (kartenkunde-leichtgemacht.de - Kartennetzentwurf 2015) zu finden.

4.12.4 Kegelprojektion

Bei einer Kegelprojektion wird die abzubildende Erdoberfläche auf einen umhüllenden Kegelmantel projiziert. Die Kegelprojektion ist für die Abbildung von verhältnismäßig kleinen Regionen in mittleren Breiten geeignet. Dabei gibt es drei Möglichkeiten, wie die Projektion durchgeführt werden kann, basierend auf (kartenkunde-leichtgemacht.de - Kartennetzentwurf 2015):

- Bei der ***einfachen Kegelprojektion*** (siehe Abbildung 28) wird die abzubildende Erdoberfläche durch den Projektionskegel entlang eines Breitengrades berührt. Entlang dieses Bezugsbreitengrades ist die Projektion verzerrungsfrei. Die Verzerrung nimmt jedoch proportional zur Entfernung zum Breitengrad zu.

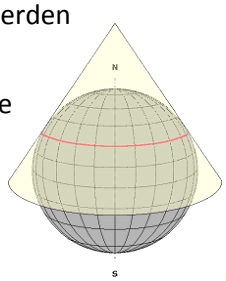

Abbildung 28 (kartenkunde-leichtgemacht.de - Kartennetzentwurf 2015)

- Die ***konforme Kegelprojektion*** (siehe Abbildung 29) schneidet die abzubildende Erdoberfläche durch den Projektionskegel in zwei Breitengrade, welche verzerrungsfrei abgebildet werden. Je schmaler der Streifen zwischen den zwei Breitenstreifen ist, desto geringer fallen die Verzerrungen aus. Sie fallen jedoch deutlich geringer aus als bei der einfachen Kegelprojektion.

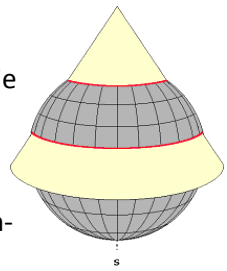

Abbildung 29 (kartenkunde-leichtgemacht.de - Kartennetzentwurf 2015)

- Bei der ***polykonischen Kegelprojektion*** (siehe Abbildung 30) werden mehrere Kegelprojektionen von mehreren Breitenstreifen durchgeführt. Jeder Kegel berührt die abzubildende Erdoberfläche an einem anderen Breitengrad. Um die Verzerrungen zu minimieren, wird der abzubildende Breitenstreifen bewusst schmal gehalten. Durch das Zusammenführen mehrerer Kegelprojektionen können größere Gebiete mit geringer Verzerrung dargestellt werden.

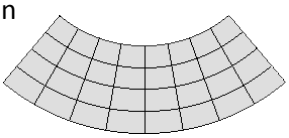

Abbildung 30 (kartenkunde-leichtgemacht.de - Kartennetzentwurf 2015)

4.13 Skript- und Auszeichnungssprachen

4.13.1 HTML

HTML (**H**yper **T**ext **M**arkup **L**anguage) ist die Kernsprache von Webinhalten. So gut wie alle Inhalte, die im Webbrowser angezeigt werden, sind mittels HTML strukturiert. Genauer gesagt beschreibt diese die Struktur und Semantik der Inhalte einer Webseite. HTML ist ein internationaler Standard, der sich technologisch laufend weiterentwickelt und durch das W3C definiert wird. Die derzeit aktuellste Spezifikation ist HTML5. (mozilla.org - HTML 2015)

HTML5 ist die Sprache des Web 2.0 und seit 1999 die letzte akzeptierte HTML-Weiterentwicklung. Sie bietet neue Funktionen an, um z.B. grafische Elemente dynamisch zu erstellen und problemlos einzubinden, was zuvor nur mit Hilfe zusätzlicher Plug-Ins realisierbar war. Weitere Änderungen sind semantische Divs wie z.B. Header und Footer sowie einige weitere Formular-Elemente.

Das Canvas-Element, welches mit seinen vielzähligen und mächtigen Methoden erlaubt, dynamisch Grafiken in ein HTML-Dokument zu zeichnen, ist dabei die spektakulärste Neuerung.

Ein Video-Tag bietet nun die Möglichkeit, sehr einfach verschiedenste Videoformate in einer Webseite einzubinden. (selfhtml5.org - HTML5 Features 2015)

4.13.2 CSS

Bei CSS (**C**ascading **S**tyle **S**heets) handelt es sich um eine Formatierungssprache für HTML-Dokumente (aber auch für XML). Neben einigen Formatierungen, die bereits ein Bestandteil von HTML sind, kann mithilfe von CSS die Darstellung von diesen Dokumenten auf vielfältige Art verändert werden. Diese Formatierungssprache klinkt sich nahtlos in HTML ein und erlaubt das beliebige Formatieren einzelner HTML-Elemente. Der Abschnitt basiert auf (wiki.selfhtml - CSS 2015).

4.13.3 JavaScript

JavaScript wurde geschaffen, um HTML mit einem Werkzeug auszustatten, mit dem es möglich ist Webseiten zu optimieren. Es handelt sich dabei jedoch um keinen direkten Bestandteil von HTML, sondern um eine eigenständige Programmiersprache mit zahlreichen Bibliotheken. Da die JavaScript-Dokumente am Webbrowser ausgeführt werden, spricht man dabei auch von einer client-seitigen Sprache. Die Skripte können entweder direkt in der HTML-Datei oder in einer separaten Datei implementiert werden. (molily.de - JavaScript 2015)

Die wichtigste Aufgabe besteht darin, auf die Benutzereingaben der Webseite zu reagieren und Änderungen im HTML-Dokument vorzunehmen. Diese Änderungen finden nur client-seitig statt, sodass das Dokument am Webserver unangetastet bleibt. Die Gestaltung des Dokumentes erfolgt dadurch interaktiv und dynamisch. (molily.de - JavaScript 2015)

Den Zugriff auf das Dokument, mit dem dieses gelesen und verändert wird, regelt das DOM-Modell. Dieses Modell ist gleichzeitig die Schnittstelle zwischen HTML und JavaScript. (wiki.selfhtml - DOM 2015)

Heutzutage nimmt JavaScript neben HTML und CSS im Web eine wichtige Rolle ein. Beinahe jede Webseite verwendet ein oder mehrere dieser Skripte. Die Weiterentwicklung von JavaScript besteht im Wesentlichen darin, dass neue Bibliotheken angelegt und neue APIs erstellt werden, die der Webbrowser unterstützt. (wiki.selfhtml - JavaScript 2015)

4.13.4 PHP

PHP (**P**HP: **H**ypertext **P**reprocessor) ist eine OpenSource-Skriptsprache, welche in der Webprogrammierung eingesetzt wird und in HTML eingebettet werden kann. Es handelt sich um eine serverseitige Skriptsprache, mit der dynamische Webseiten und Inhalte erstellt werden. Die Stärken von PHP sind die Unterstützung für eine breite Masse von Datenbanken (z.B. PostgreSQL, MySQL, ODBC, usw.), der Datenaustausch mit anderen Programmiersprachen, die Internetprotokoll-Einbindung, das Lesen und Generieren von Dateien sowie die Integrationsmöglichkeit zahlreicher Bibliotheken.

Typische Anwendungsbeispiele von PHP sind:

- Verarbeitung von Formularen: Auswertung von Eingaben und Speichern der Daten in einer Datenbank bzw. Versenden per E-Mail an den Seitenbetreiber
- Login-Möglichkeit: Ermöglichung von Benutzer-Registrierung, Verschlüsselung von Daten
- Zusammensetzung verschiedener Dateien: Einfügen der Daten in ein Template

Nähere Informationen sind auf (php.net - PHP-Handbuch 2015), (1php.de - PHP-Einführung und Grundlagen 2015) sowie auf (wiki.selfhtml - PHP 2015) zu finden.

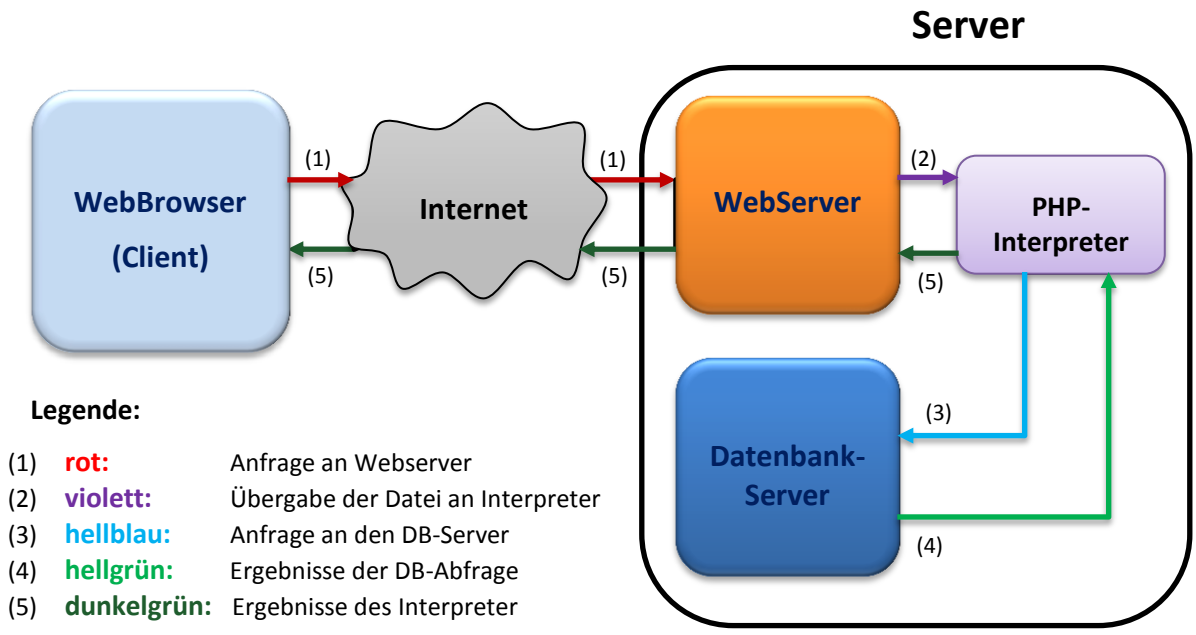

Abbildung 31 – PHP (Eigene Darstellung)

Beschreibung der Abbildung 31: Wenn der Besucher eine dynamische PHP-Webseite aufruft, sendet der Webbrowser eine Anfrage an den Webserver (1). Dieser lädt die angeforderte Datei (2) und schickt diese an den PHP-Interpreter des Webservers. Der PHP-Interpreter ist dafür zuständig, aus der PHP-Datei eine HTML-Seite zu generieren und sich mit einer Datenbank zu verbinden, um eine Abfrage (3) an den Datenbankserver zu stellen. Der Interpreter sendet die Ergebnisse (4, 5) anschließend an den Webserver zurück und dieser wiederum gibt die Daten an den Webbrowser weiter.

Im Unterschied zu client-seitigen Programmiersprachen wie JavaScript, wird der Code am Server ausgeführt und HTML-Ausgaben werden generiert, die anschließend an den Client gesendet werden. Client-seitig ist es damit nicht möglich, herauszufinden, wie der PHP-Code lautet, da der Client nur das Ergebnis der Skriptausführung erhält.

4.14 Datenbanksprachen

4.14.1 SQL

Die Abfragesprache SQL (**S**tructured **Q**uery **L**anguage) ist eine etablierte Datenbanksprache für die Erstellung von Datenbankstrukturen in relationalen Datenbanken, sowie zum Bearbeiten und Abfragen der darauf basierenden Datenbestände. Es existieren unterschiedliche Standards, und jeder Hersteller von Datenbanksystemen hat seine eigenen Erweiterungen und Besonderheiten. Man spricht von leicht voneinander abweichenden SQL-Dialekten, wobei es sich bei PL/pgSQL um einen dieser Dialekte handelt. Der Abschnitt bezieht sich auf (datenbanken-verstehen.de - SQL Einführung – Was ist SQL ? 2015) und (wikibooks - Einführung in SQL 2015).

4.14.2 PL/pgSQL

PL/pgSQL (**P**rocedural **L**anguage/**P**ost**G**reSQL **S**tructured **Q**uery **L**anguage) ist eine prozedurale Programmiersprache für das objektrelationale Datenbanksystem PostgreSQL. Diese Sprache wurde zur Erweiterung des SQL-Funktionsumfangs eingeführt und kann dabei auch als Code in der Datenbank gespeichert werden.

Die Einführung dieser Sprache ermöglicht das Schreiben von Funktionen und Triggerprozessen. Außerdem können Kontrollstrukturen implementiert werden und es stehen benutzerdefinierte Typen, Funktionen und Operatoren zur Verfügung. Damit können komplexe Berechnungen durchgeführt werden. Durch das direkte Ausführen der PL/pgSQL-Programme in der Datenbank, kommt es zu einer deutlichen Steigerung der Performance. Diese ergibt sich dadurch, dass die Client-Server-Kommunikation wegfällt und die Kommunikation zwischen den Prozessen vermieden wird.

Nähere Informationen sind unter (postgresql.org - PL/pgSQL: SQL prozedurale Sprache 2013) zu finden.

4.15 Objektorientierte Programmierung

4.15.1 Allgemeines

Die objektorientierte Programmierung öffnet neue Wege durch die Verwaltung der Codes mithilfe von Objekten, im Gegensatz zur prozeduralen Programmierung, bei der der Code der Lesereihenfolge folgt. Ein Grundkonzept der objektorientierten Programmierung besteht darin, Daten und deren Methoden in einem Objekt zusammenzufassen.

Methoden sind Funktionen, die auf diese Daten angewendet werden können. Nach außen sind die Objekte gekapselt, sodass die Daten nur über eine klar definierte Schnittstelle des Objekts gelesen und verändert werden können.

Die Definition der Objekte erfolgt über eine Klasse, welche eine formale Beschreibung darstellt, wie ein Objekt beschaffen ist (Attribute, Methoden). Man spricht auch von Bauplänen, nach denen Objekte erzeugt werden. Statt einem Objekt spricht man auch von einer Instanz einer Klasse.

Man kann Objekte auch als leicht wiederverwertbare Bausteine beschreiben, die die Wiederverwendbarkeit unterstützen.

Objektorientiertes Denken erlaubt die Verwaltung von komplexen Gebilden (Objekten) auf unterschiedlichen Niveaus. Darüber hinaus stellt ein Objekt ein unverwechselbares Ganzes dar, das durch seine Eigenschaften und durch die darauf anwendbaren Methoden geprägt ist. Eine objektorientierte Betrachtungsweise hat auch konkrete Vorteile, wenn es um Interoperabilität von Systemen geht, wo mehrere autonome Komponenten zusammenwirken.

Der Abschnitt basiert auf (openbook.rheinwerk-verlag - Objektorientierte Programmierung 2015), (python-kurs.eu - OOP-Klassen 2015) und (BARTELME 2005).

4.15.2 Vererbung

Die Vererbung bietet die Möglichkeit, eine neue Klasse zu erzeugen, indem eine existierende Klasse erweitert wird. Durch die Deklarierung einer Elternklasse können dessen Methoden und Eigenschaften verwendet werden. In folgendem Diagramm (siehe Abbildung 32) wird dies anhand der Objekte *Cat* und *Dog* erklärt, die ihre Eigenschaften und Methoden vom Objekt *Animal* vererbt bekommen haben und zusätzliche Methoden besitzen. Nähere Informationen sind unter (it-infothek.de - Objektorientierte Programmierung 2003) zu finden.

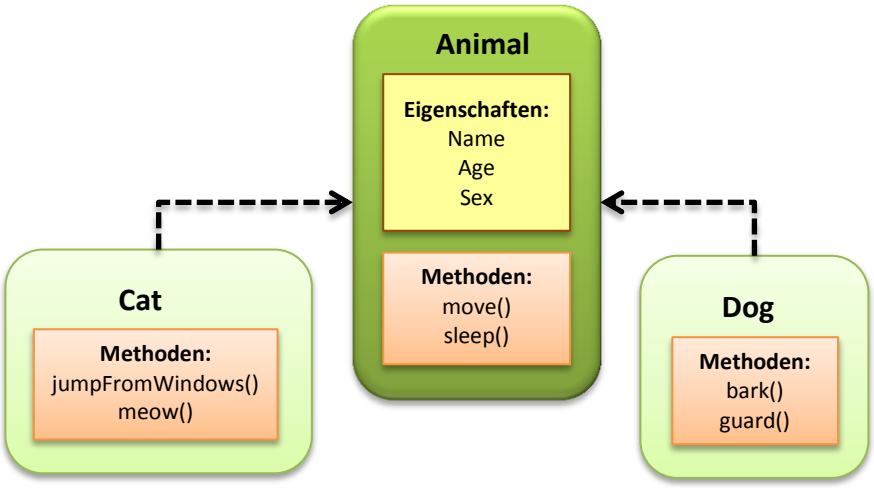

Abbildung 32 – Beispiel für OOP, eigene Darstellung nach (GRATIER, SPENCER, and HAZARD 2015)

4.16 OpenLayers

Allgemeines:

OpenLayers ist eine Open-Source JavaScript-Bibliothek für die Erstellung von interaktiven Web-Maps. Da es sich um eine client-seitige Bibliothek handelt, wird keine spezielle server-seitige Software benötigt. OpenLayers ermöglicht es, komplette Karten-Anwendungen von Grund auf zu erstellen, mit der Möglichkeit, jeden Aspekt der Karte (Layer, Controls, Events etc.) jederzeit anzupassen. Die 2.x Version der OpenLayers-Bibliothek hat sich inzwischen zu einem reifen und beliebten Framework mit vielen leidenschaftlichen Programmierern entwickelt.

Vorteile von OpenLayers:

- Open-Source
- einfache Anwendung – man muss kein Programmierer sein
- Starke Community
- Unterstützung von Smartphones/Tablets
- nicht auf eine proprietäre Technologie oder ein Unternehmen gebunden
- die Bibliothek ist auch von der OSGeo anerkannt

Hauptunterschiede zwischen OpenLayers 2 und OpenLayers 3:

- OL2 ist acht Jahre älter und hat Konstruktionsfehler. Dadurch wurde es schwer, neue Entwicklungen zu integrieren.
- OL3 bietet die Möglichkeit Animationen leicht zu integrieren
- Der standardmäßige Canvas-Renderer ist effizienter als der DOM-Renderer in OL2 und die WebGL-Unterstützung in der Roadmap ermöglicht eine bessere Darstellung.
- Gesteigerte Performance

In den finalen Versionen der Web-App bzw. des WebGIS wird ausschließlich OpenLayers 3 verwendet. Im Anhang befinden sich detailliertere Informationen zu OpenLayers 3.

Der Abschnitt basiert auf (GRATIER, SPENCER, and HAZARD 2015) und (JANSEN and ADAMS 2010).

4.17 Sonstige Technologien und Softwarepakete

4.17.1 QR-Codes

QR-Codes („Quick Response“) sind zweidimensionale Strichcodes, die 1994 von der japanischen Firma Denso Wave entwickelt wurden. Ziel war es, mehr Informationen auf einer kleinen Fläche unterzubringen und geringe Anforderungen an die Lesegeräte zu stellen.

QR-Codes sind wie kleine Datenspeicher, die bis zu 40.000 Zeichen Text verschlüsseln können. Im Vergleich dazu erfasst ein Strichcode gerade einmal 13 Zeichen (HENGSTBACH 2011). Ein großer Vorteil dieser Codes ist, dass bei einer Verschmutzung oder Zerstörung von bis zu 60 % der Fläche die Dekodierung noch fehlerfrei möglich ist (in den meisten Anwendungen jedoch nur 7 - 30 %). Außerdem ist die Verwendung des QR-Codes lizenz- und kostenfrei (wikipedia - QR-Code 2015). Neben der industriellen Anwendung werden QR-Codes vor allem für mobile Anwendungen eingesetzt. Ein typisches Bespiel ist das Codieren von Webadressen, die einem potentiellen Kunden zu Werbe- oder Informationszwecken näher gebracht werden sollen. Mit praktisch jedem Smartphone ist es möglich, diese Codes mit einer entsprechenden Software einzuscannen, zu entschlüsseln und die Webseite aufzurufen. Der Anwender muss dadurch nicht mehr die meist langen Webadressen eintippen (KRAMER 2013).

Um die entsprechenden QR-Codes zu generieren, gibt es einige online QR-Code-Generatoren sowie fertige Skripte, die man in die eigene Webseite einbauen kann. Dabei wird eine Grafik erstellt, die den codierten Text enthält. Die Größe der QR-Codes liegt je nach Variante zwischen 9x9 und 422x422 Modulen, wobei ein Modul einem Rasterfeld entspricht. Die Anzahl der Zeichen, die verschlüsselt werden können, sind abhängig von der Anzahl der Module (wikipedia - QR-Code 2015).

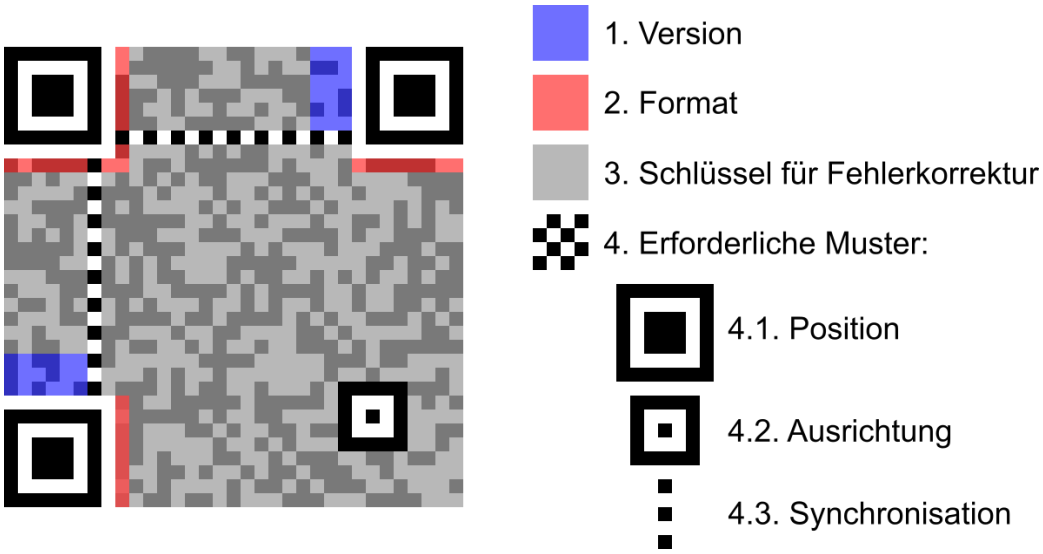

Abbildung 33 – Aufbau eines QR-Codes (wikipedia - QR-Code, svg 2015)

Um die QR-Codes (siehe Abbildung 33) korrekt lesen zu können, müssen noch weitere Informationen im Grafikcode enthalten sein (wikipedia - QR-Code 2015):

- Positionsmarkierungen in den drei Ecken, um die Ausrichtung des QR-Codes mittels Lesegerät zu erkennen.
- Synchronisationselemente, um das Raster der Matrix zu bestimmen.
- Ausrichtung, um die perspektivische Verzerrung der Aufnahme zu bestimmen.
- Datenformat
- Version

QR-Codes können auch in Farbe dargestellt werden, wobei jedoch unbedingt ein hoher Kontrast zwischen den hellen und dunklen Pixeln erforderlich ist. Für die Generierung und zum Einlesen der QR-Codes werden im Internet einige Open-Source und kostenpflichtige APIs und Plug-Ins angeboten. (HENGSTBACH 2011)

Varianten:

Neben dem quadratischen QR-Code, der in Abbildung 33 dargestellt ist und der auch am häufigsten verwendet wird, gibt es noch weitere Varianten, die zu erwähnen sind (wikipedia - QR-Code 2015):

- Design-QR-Code: Prinzipiell ist eine individuelle Gestaltung durch die Variation der Module und der Farbe möglich, sowie durch das Einbetten von Symbolen. Letzteres ist aufgrund der hohen Fehlertoleranz (meistens 30 %) möglich.

- Mini-QR-Code: Dabei handelt es sich um eine Variante, die auf eine geringe Abmessung optimiert ist.

- iQR-Code: Dieser QR-Code ist eine Weiterentwicklung und übertrifft die Ergebnisse aller anderen Varianten. Außerdem muss dieser QR-Code auch nicht zwingend quadratisch sein.

- Frame-QR: Bei diesem QR-Codes ist es möglich, Symbole im Code zu integrieren, ohne dabei Teile des Codes zu überlagen. Dadurch geht keine Information verloren. Als Grundformen sind verschiedene Varianten möglich, z.B.: Quadrat, Kreis, und Dreieck.

- Secure-QR-Code: Ist ein QR-Code mit einer zusätzlichen Funktion, um Daten zu verschlüsseln. Dabei muss der Schlüssel dem Decoder des Lesegerätes bekannt sein.

- QArt-Code: Bei diesem QR-Code ist neben der zu kodierenden URL auch eine Bildvorlage anzugeben. Die Datenpixel des Codes werden so angeordnet, dass sie die Helligkeitsunterschiede der Bildvorlage wiedergeben und das Bild im Code als Schwarzweißbildausgabe erkennbar ist.

Da QR-Codes leicht zu erstellen sind, können diese aber auch leicht manipuliert werden, z.B. durch Überkleben (KRAMER 2013).

4.17.2 AJAX

Mit AJAX bezeichnet man ein Konzept, dass die asynchrone Datenübertragung zwischen Webbrowser und dem Webserver ermöglicht. Die Idee dahinter ist, dass der Browser statt ganzen HTML-Seiten nur Daten lädt, die gebraucht werden. Diese Daten werden dann in der aktuellen Seite verwendet.

AJAX basiert auf verschiedensten Web-Technologien (HTML, CSS, XML, DOM), welche das interaktive Arbeiten im Web ermöglichen.

Die Daten werden mittels JavaScript mit dem Objekt XMLHttpRequest geladen. Dabei kann der XMLHttpRequest entweder synchron oder asynchron aufgerufen werden.

Bei einem synchronen Aufruf wartet das JavaScript-Programm (und damit meistens der ganze Browser), bis die Daten geladen werden. Dies hat den Nachteil, dass die Benutzeraktivität unterbrochen ist, solange der Client auf die Antwort vom Server wartet. Außerdem werden bei dieser Art der Datenübertragung eine Menge nicht relevanter Daten übermittelt.

Bei einem asynchronen Aufruf ist es nicht mehr notwendig, bei einer erforderlichen Änderung die komplette Seite neu aufzubauen. Es werden nun gezielte Anfragen an den Server gestellt, die im Hintergrund ablaufen, ohne dass es auf der Webseite zu Einschränkungen kommt. Durch den geringeren Datenaustausch zwischen Client und Server können Anfragen auch schneller abgearbeitet werden. Da es während der Verwendung einer Anwendung, die auf AJAX basiert, kaum zu Unterbrechungen kommt, macht dies den Eindruck, als würde die Anwendung rein client-seitig ablaufen. Der Unterschied zwischen einer synchronen und asynchronen Webapplikation ist in Abbildung 34 grafisch dargestellt.

Die Informationen beziehen sich auf (openbook.rheinwerk-verlag - Objektorientierte Programmierung 2015), (BARTHEL et al. 2005) und (ADOLF, FLIEGE, and LAUN 2006).

Vorteile:

- Seiteninhalte können durch neue Daten vom Server geändert werden, ohne dass die Seite komplett neu geladen werden muss.
- die Serverlast wird verringert
- es können Oberflächen geschaffen werden, die Desktop-Anwendungen ähneln (Stichwort WebApps)

Nachteile:

- JavaScript muss aktiviert sein
- Beim Internet Explorer müssen ActiveX-Objekte aktiviert sein
- Deutlich mehr Aufwand in der Programmierung

- Manche Methoden funktionieren nicht in jedem Browser bzw. unterscheiden sich in den Ergebnissen. Dadurch sind mehr Tests notwendig, um jeden Browser zu überprüfen.

Eine genauere Beschreibung der Vor- und Nachteile ist in (BAUER 2009) zu finden.

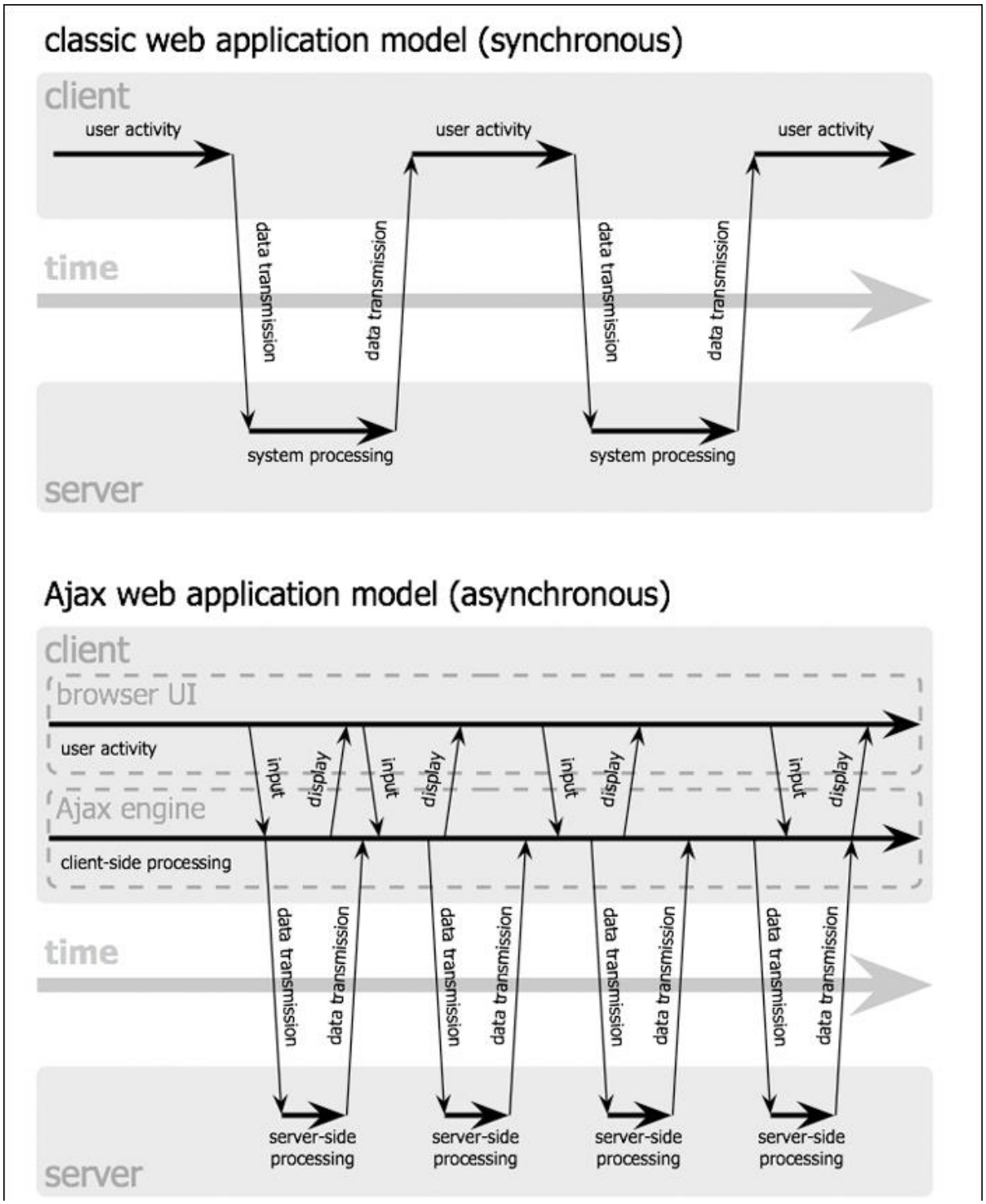

Abbildung 34 – AJAX (GRATIER, SPENCER, and HAZARD 2015)

4.17.3 Geolocation API

Über die von W3C zur Verfügung gestellte HTML 5 Geolocation-API ist es möglich, Informationen über den Standort des mobilen Geräts zu bekommen. Die Geolocation-API wird von den gängigen Webbrowsern (IE 9+, Firefox 3.5+, Google Chrome und weitere) unterstützt. Der Standort kann durch folgende Methoden bestimmt werden, die mit aufsteigender Genauigkeit gereiht sind:

1. IP-Adresse
2. Zellortung
3. WLAN
4. GPS (nur bei „enableHighAccuracy“)

Diese Ortungsmethoden sind in Abschnitt 4.1 genauer beschrieben. Die WLAN- und GPS-Ortung ist jedoch nur möglich, wenn diese Module am mobilen Gerät aktiviert sind. Erst wenn keine genauen Daten vorliegen, greift der Browser auf eine der anderen Methoden zurück, wobei die Ortung mittels IP-Adresse am ungenauesten ist. Die Informationen beziehen sich auf (selfhtml5.org - geolocation 2015) und (wiki.selfhtml.org - Geolocation 2015).

Der Zugriff auf die Ortsinformation erfolgt über das Navigator-Objekt:

```
navigator.geolocation.getCurrentPosition(function(position){
     successFunction(position),
     errorFunction(),
     {
            timeout: 0,
            enableHighAccuracy: true,
            maximumAge: Infinity
     }
});
```

Programmcode nach (dev.w3.org - Geolocation API Specification 2015)

Man erhält daraus das Objekt „position“, das folgende Eigenschaften hat (siehe

Tabelle 10):

Tabelle 10 – „position“-Objekt (w3schools.com - geolocation 2015)

Eigenschaft	Beschreibung	Typ	Einheit
coords.latitude	Breite	double	Grad
coords.longitude	Länge	double	Grad
coords.altitude	Höhe	double/null	Meter
coords.accuracy	Genauigkeit	double	Meter
coords.altitudeAccuracy	Genauigkeit	double/null	Meter
coords.heading	Kurs	double/null	Grad (im Uhrzeigersinn von Norden)
coords.speed	Geschwindigkeit	Double/null	Meter/Sek.

Tabelle 11 – Genauigkeiten der Gerätetypen (selfhtml5.org - geolocation 2015)

Gerätetyp	Genauigkeit [m]
PC	2.000 bis 140.000
Smartphone ohne GPS	20 bis 80
Smartphone mit GPS	5 bis 20

In Tabelle 11 ist zu erkennen, dass mit GPS die Position am genauesten bestimmt werden kann. Die Verwendung des GPS-Moduls kann durch „enableHighAccuaracy“ aktiviert werden. Diese Methode hat jedoch folgende Nachteile (selfhtml5.org - geolocation 2015):

- Die Positionsdaten sind nach dem Aktivieren aufgrund der Initialisierungsphase (meist 10 bis 30 Sekunden) nicht sofort verfügbar.
- Meist ist GPS jedoch deaktiviert, da sich sonst die Akkulaufzeit erheblich verringert.
- Die Zustimmung des Nutzers wird benötigt. Daher erscheint bei jeder Ortung eine Warnmeldung per Popup. Dies wird von W3C so vorgeschrieben.

4.17.4 Geocoding API

Mit der Google Maps Geocoding API ist es möglich, aus Adressangaben geografische Koordinaten (φ und λ) zu bekommen. Eine Konvertierung in umgekehrter Richtung ist ebenfalls möglich und wird „Reverse Geocoding“ bezeichnet.

Der Datenaustausch erfolgt im JSON- oder XML-Format. Folgender Aufruf, in dem die Leerzeichen durch „+“ ersetzt worden sind, erzeugt eine Antwort im JSON-Format:

```
https://maps.googleapis.com/maps/api/geocode/json?address=30+N+Steyrergasse,
+Graz,+Austria&key=YOUR_API_KEY
```

Um die API nutzen zu können, wird ein Schlüssel benötigt, der eine Registrierung erfordert. Die Informationen basieren auf (developers.google.com - geocoding 2015).

4.17.5 Cookies

Cookies werden verwendet, um Nutzerinformationen von Webseiten zu speichern. Diese werden beim erneuten Aufruf der Seite geladen. Das hat folgende Vorteile:

- Die Seite kann dementsprechend aufgebaut werden.
- Bestimmte Einstellungen, die der Nutzer auf dieser Seite getätigt hat, müssen so nicht erneut ausgeführt werden.

Die Informationen werden in kleinen Textfiles am Computer des Nutzers über einen bestimmten Zeitraum gespeichert. Der Benutzer muss vom Speichern seiner Informationen nicht unbedingt etwas mitbekommen. Meist werden Cookies zu Werbezwecken verwendet und um das Surfverhalten der Nutzer zu analysieren. Diese Daten können durch eine entsprechende Software entfernt werden. Bei WebApps können Cookies verwendet werden, um nutzerspezifische Einstellungen zu speichern. Nähere Informationen sind unter (w3schools - JavaScript Cookies 2015) zu finden.

4.17.6 MATLAB

MATLAB (**MAT**rix **LAB**oratory) ist eine kommerzielle Software des Unternehmens MathWorks, in der in einer proprietären Programmiersprache gearbeitet wird (etools.fernuni.ch - MATLAB 2015). Diese bietet eine interaktive Umgebung für numerische Berechnungen, Visualisierung von Daten und Erstellung von grafischen Benutzeroberflächen (GUI). Mit diesem Werkzeug ist es möglich, Algorithmen und spezielle Ansätze zu entwickeln sowie Daten zu analysieren. Dabei gelangt man wesentlich schneller zu einer Lösung als mit herkömmlichen Programmiersprachen, wie C/C++ oder Java (mathworks - Documentation 2015). Unterstützung erhält man durch einen großen Umfang an mathematischer Funktionen und einer beachtlichen Internet-Community.

MATLAB findet vorwiegend Anwendung in der Verarbeitung, Auswertung und Darstellung numerischer Daten (siehe Diagramm in Abbildung 35). Ein besonderer Vorteil von MATLAB gegenüber den meisten anderen Programmiersprachen ist, dass es auch als Interpreter bedient werden kann. Um das Programm auszuführen, muss es vorher nicht kompiliert werden und arbeitet trotzdem fast so schnell wie mit anderen Programmiersprachen (etools.fernuni.ch - MATLAB 2015).

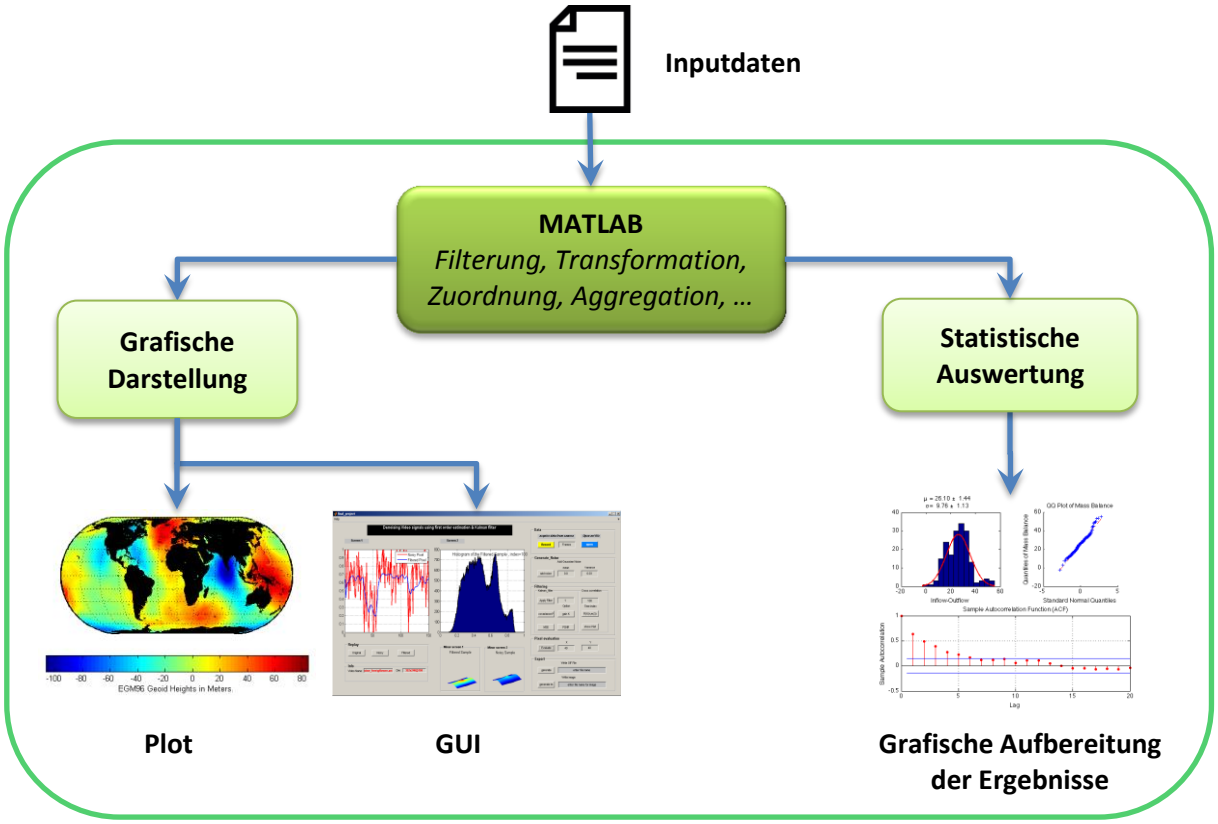

Abbildung 35 – MATLAB, eigene Darstellung nach (etools.fernuni.ch - MATLAB 2015)

Quellen der Grafiken in Abbildung 35: (clipartbest.com - Weltkarte 2015), (nd.edu - Statistik 2015) und (mathworks.com - GUI 2015).

4.17.7 osm2po

Dabei handelt es sich um eine Open-Source-Anwendung, mit der es möglich ist, OSM-Karten in ein PostGIS-taugliches Format zu bringen. Dieses Format ist auch routingfähig. Unterstützte Eingabe-

formate für das Kartenmaterial sind *.bz2, *.pbf, *.o5m und *.o5m.gz. Die Anwendung wird mittels Eingabeaufforderung ausgeführt und die Ergebnisse als SQL-Skript unter „localhost“ gespeichert. Um die Karte letztendlich in die Datenbank zu bekommen, kann dieses Skript beispielsweise mit dem pgAdmin ausgeführt werden. Das Ergebnis ist eine Tabelle, die mit den Funktionen von pgRouting kompatibel ist. Große Datensätze, wie z.B. die europe.osm, können mit diesem Werkzeug leicht in eine routingfähige Form gebracht werden. Die Informationen beziehen sich auf (osm2po.de 2015) und (anitagraser.com - osm2po-Quickstart 2011).

4.17.8 QGIS

QGIS (**Q**uantum **GIS**) ist eine freie, leicht zu bedienende, Open-Source GIS-Anwendung, die dem Benutzer das Visualisieren, Verwalten, Editieren und Analysieren von Geodaten ermöglicht. Außerdem können aus den Daten aussagekräftige Karten zusammengestellt werden (wiki.openstreetmap.org - QGIS 2015). Es wird eine Vielzahl von Vektor-, Raster- und Datenbankformaten und -funktionen unterstützt. Mithilfe der Plug-In-Architektur können weitere Funktionalitäten einfach ergänzt werden (docs.qgis.org - Vorwort 2015). Erweiterungen können über Schnittstellen sowohl in C++ als auch in Python entwickelt werden. QGIS ist ein offizielles Mitglied der OSGeo, welche das Ziel hat, der Allgemeinheit den Zugang zur Verarbeitung räumlicher Daten zu ermöglichen. Die Informationen basieren auf (qgis-anwendertreffen.de - Über QGIS/Ziele 2015).

4.17.9 PgAdmin III

Das Programm PgAdmin III ist ein freies Entwicklungs- und Verwaltungssystem für PostgreSQL-Datenbanken. Es enthält eine grafische Verwaltungsoberfläche und Werkzeuge, mit denen die Entwicklungs- und Verwaltungsaufgaben auf einfache Weise erledigt werden können. Damit wird man vom Schreiben einfacher SQL-Abfragen bis hin zur Entwicklung komplexer Funktionen sowie der Erstellung von Datenbanken unterstützt. Nähere Informationen sind unter (teialehrbuch.de - PostgreSQL - Werkzeuge 2015) zu finden.

4.17.10 pgRouting

pgRouting ist eine Erweiterung der PostGIS/PostgreSQL-Geodatenbank, um die Funktionen für “Kürzeste-Wege”-Berechnung (Routing) bereit zu stellen.

Vorteile dieser Variante des Datenbank-Routing:

- Daten und Attribute können von mehreren Clients, wie z.B. QGIS über JDBC, ODBC oder direkt über PL/pgSQL verändert werden.
- Datenänderungen können durch das Routing-System sofort reflektiert werden. Es besteht keine Notwendigkeit, vorher weitere Berechnungen durchzuführen.
- Der Kostenparameter kann über SQL dynamisch berechnet werden.

pgRouting unterstützt folgende Graphen-Algorithmen:

- Dijkstra (Shortest Path)
- A* (Shortest Path)
- Floyd-Warshall Algorithm (All Pairs Shortest Path)
  Johnson’s Algorithm (All Pairs Shortest Path)

- Bi-directional Dijkstra (Shortest Path)
- Bi-directional A* (Shortest Path)
- Driving Distance
- K-Shortest Path, Multiple Alternative Paths
- K-Dijkstra, One to Many Shortest Path
- Traveling Sales Person
- Turn Restriction Shortest Path
- Shortest Path Shooting Star

Bevor einer der Routing-Algorithmen angewendet werden kann, müssen die Datensätze in einer entsprechenden Topologie vorliegen.

Der wichtigste Befehl, um eine solche Topologie zu realisieren, lautet: pgr_createTopology. Dabei wird eine Knotentabelle generiert.

Befehl: `SELECT pgr_createTopology('edge_table', 0.00005, 'geom_way');`

Name der mit dem Befehl generierten Knotentabelle: `<edge_table>_vertices_pgr`

Gleichzeitig werden für die Kantentabelle Indizes erstellt, um bei der Berechnung der Routen einen schnelleren Zugriff zu ermöglichen. Die Grundlagen des Abschnitts basieren auf der offiziellen Projektseite (pgrouting.org - Project 2015) und dem Manual (pgRouting Manual 2013).

4.18 Formate

4.18.1 WKT

Das WKT-Format (**W**ell **K**nown **T**ext) ist ein Format um Vektor-Geometrien zu beschreiben. Es ist aus der „Simple Features“-Spezifikation des OGC hervorgegangen (giswiki - WKT 2015). Beispiele für Geometrie-Typen sind:

Tabelle 12 – WKT-Format (giswiki - WKT 2015)

Typ	Definition mit Beispiel
Punkt	`POINT(10 20)`
Linie	`LINESTRING(10 10, 20 20, 30 30)`
Multilinie	`MULTILINESTRING((10 10, 20 20),(30 30, 40 40, 50 50))`
Polygon	`POLYGON((0 0, 0 10, 10 10, 10 0, 0 0))`
Multipolygon	`MULTIPOLYGON(((0 0, 10 0, 0 10, 0 0)),((20 20, 20 30, 30 20, 20 20)))`
GeometryCollection	`GEOMETRYCOLLECTION( … )`

4.18.2 WKB

Das **W**ell-**K**nown **B**inary Format ist eine Darstellung der Geometrien als eine kontinuierliche Byte-Datenkette. Als Datentypen verwendet WKB Integer ohne Vorzeichen. Dieses Format wird z.B. für Geometriedaten in einer Datenbank mit Spatial-Erweiterung eingesetzt (giswiki - WKB 2015).

4.18.3 JSON

JSON steht für **J**ava**S**cript **O**bject **N**otation. Es handelt sich dabei um ein leicht lesbares Datenformat, das für den Datenaustausch zwischen verschiedenen Web-Anwendungen und mobilen Anwendungen konzipiert wurde. JSON basiert auf der Notation, in der JavaScript-Objekte und die Werte ihrer Eigenschaften initialisiert werden. Genauer gesagt handelt es sich um eine vereinfachte Version der Initialisierungsform der Objekte in JavaScript. Die Grammatik von JSON ist sehr einfach und entsprechend leicht lassen sich die Parser in so gut wie jeder Programmiersprache entwickeln. (openbook.rheinwerk-verlag - Objektorientierte Programmierung 2015)

JSON ist viel einfacher zu verwenden als XML, vor allem in den JavaScript-basierten Web-Apps. Für die meisten JavaScript-Bibliotheken mit GIS-Bezug wie OpenLayers und Leaflet ist JSON bereits Standard. Jedoch sind die OGC-Standards fast alle XML-basiert und durch XML-Schemata spezifiziert. (fossgis.de - OGC Web Services in JSON 2015)

Beispiel:

```
{
  "Name": "Franz",
  "Alter": 46,
  "Schulabschluss": "Hauptschule"
  "Verheiratet": true,
  "Beruf": null,
  "Kinder": [
    {
      "Name": "Felix",
      "Alter": 19,
      "Schulabschluss": "Gymnasium"
    },
    {
      "Name": "Barbara",
      "Alter": 12,
      "Schulabschluss": null
    }
  ]
}
```

JSON-Variablen können verschiedene Typen haben (wiki.selfhtml - JSON 2015)**:**

- Null: Damit kann angezeigt werden, dass sich kein Wert in der Variable befindet.
- Boolean: Zuweisung eines booleschen Wertes
- Zahl: Ziffern zwischen 0 und 9 mit positiven oder negativen Vorzeichen und Dezimalpunkt
- Zeichenkette: Eine Zeichenkette wird mit doppelten Anführungszeichen deklariert und wird auch String genannt.
- Array: Die Arrays werden durch eckige Klammern (`[]`) deklariert und die Werte innerhalb der Klammern werden durch Komma getrennt.
- Objekt: Ein Objekt wird durch geschweifte Klammern (`{}`) deklariert. Im Unterschied zum Array wird jeder Wert durch einen eindeutigen Schlüssel definiert, der von doppelten Anführungszeichen umgeben ist. Schlüssel und Wert werden durch einen Doppelpunkt voneinander getrennt und mehrere Schlüssel werden durch ein Komma getrennt.

4.19 WebApp versus Native App

- Native Apps sind mobile Anwendungen, die über das Internet von einem AppStore heruntergeladen und am Smartphone oder Tablet installiert werden.
- WebApps sind Webseiten, die für mobile Geräte optimiert sind und über einen Webbrowser laufen.

WebApps haben gegenüber gewöhnlichen Native Apps (wie z.B. runtastic oder WhatsApp) eine Reihe von Vorteilen und natürlich auch Nachteile. Die Unterschiede sind in Tabelle 13 und Tabelle 14 angeführt.

Tabelle 13 – Vor- und Nachteile von Native Apps

	Native Apps
+	optimale Ausnutzung der Smartphone Hardware (Kamera, Sensoren, Sprache, GPS etc.)
+	konsistente Darstellung der Oberfläche
+	hochwertige Wahrnehmung durch Nutzer
+	Marketing und Verkauf der Apps wird durch AppStores ermöglicht
+	schnellere Ladezeiten
+	kann auch im Offline-Modus verwendet werden
+	Personalisierungsmöglichkeit
-	keine einheitlichen Programmierrichtlinien zwischen den Anbietern der unterschiedlichen Betriebssysteme (Android, IOS, Windows)
-	mehr Programmieraufwand
-	teilweise kostenpflichtig

Tabelle 14 – Vor- und Nachteile von WebApps

	WebApps
+	mit allen Betriebssystemen weitestgehend kompatibel (egal ob Android, IOS oder Windows)
+	plattformunabhängig (PC, Smartphone, Tablet)
+	kein Download notwendig
+	einfache Wartung und Aktualisierung durch den Anbieter
+	einfache Erstellung der WebApps durch HTML5, JavaScript und deren Bibliotheken
+	die Nutzung der Gerätehardware wurde durch HTML5 ermöglicht
+	kann auch am PC genutzt werden
+/-	Personalisierungsmöglichkeit (nur) über Cookies möglich
-	dauerhafte Internetverbindung erforderlich
-	Ladezeiten von Internetverbindung abhängig
-	browserabhängige Darstellung (abweichende Form der Icons, Schrift usw.)
-	Unterstützung von Hardware-Features teilweise noch unbefriedigend (vom Browser abhängig)
-	kein Marketing über AppStore möglich
-	keine Push-Nachrichten

Quellen der Vor- und Nachteile von Native- und Web-Apps sind: (html-seminar.de - Web App versus Native App 2015), (plazz-entertainment.com - Native App vs. Web App vs. Mobile Seite 2014), (signalinc.com - Choosing the Right Technology for Your Mobile App Strategy 2013) und (GASSMANN 2014).

Fazit:

Die WebApp konnte sich durch den HTML5-Standard technologisch stark weiterentwickeln. Durch die Möglichkeit, die Gerätehardware nutzen zu können, ist der größte Nachteil gegenüber Native Apps Geschichte. Frühestens ab 2020 wird erwartet, dass HTML5-Webseiten auf unterschiedlichen Browsern identisch aussehen und funktionieren werden. Momentan werden die APIs, die z.B. den Zugriff auf die Hardware ermöglich, nicht von jedem Browser unterstützt.

Bei Offline-Anwendungen sowie bei Apps, die nahtlos auf spezielle Hardware zugreifen möchten, oder bei Anwendungen, bei denen komplexe Funktionen und Aufgaben erforderlich sind, kommt man jedoch nicht umhin, eine Native App zu erstellen.

Der Entwicklungsaufwand von Native Apps ist jedoch wesentlich höher als bei WebApps, wobei sich die Entwicklungskosten für Unternehmen oft rentieren oder sie machen sich durch den Verkauf in App-Stores bezahlt.

5. Umsetzung

Dieses Kapitel beschreibt die während der Diplomarbeit durchgeführten Tätigkeiten. Neben der Ausarbeitung eines Konzeptes enthält dieses Kapitel eine genaue Beschreibung über die Datenbeschaffung, den Aufbau der Datenbank, die entwickelten Lösungen und die Beschreibung der Ergebnisse.

5.1 Konzepterstellung

Bei der Suche nach einer geeigneten mobilen Plattform für die Navigationsanwendung fiel die Wahl auf eine WebApp. Die leichtere Programmierbarkeit und die Unabhängigkeit vom Betriebssystem waren bei der Entscheidungsfindung ausschlaggebend (Gegenüberstellung siehe Abschnitt 4.19). Potentielle Nutzer der Navigationsanwendung sollen in erster Linie Studenten sein. Bei der Berechnung soll auch berücksichtigt werden, ob der Nutzer Fußgänger, Radfahrer oder körperlich beeinträchtigt ist.

Neben der Positionsbestimmung mit QR-Codes, deren Verfügbarkeit nicht überall gegeben ist, sind für die Navigationsanwendung noch weitere Ortungsmethoden hinzugezogen worden. Diese sind:

- Manuelle Auswahl (Dropdown) oder Eingabe des Raumes (textbasiert)
- Positionsbestimmung mittels IP-Adresse, Zellortung, GPS oder WLAN (Geolocation-API)
- Positionsbestimmung durch Eingabe der Adresse (Geocoding-API)

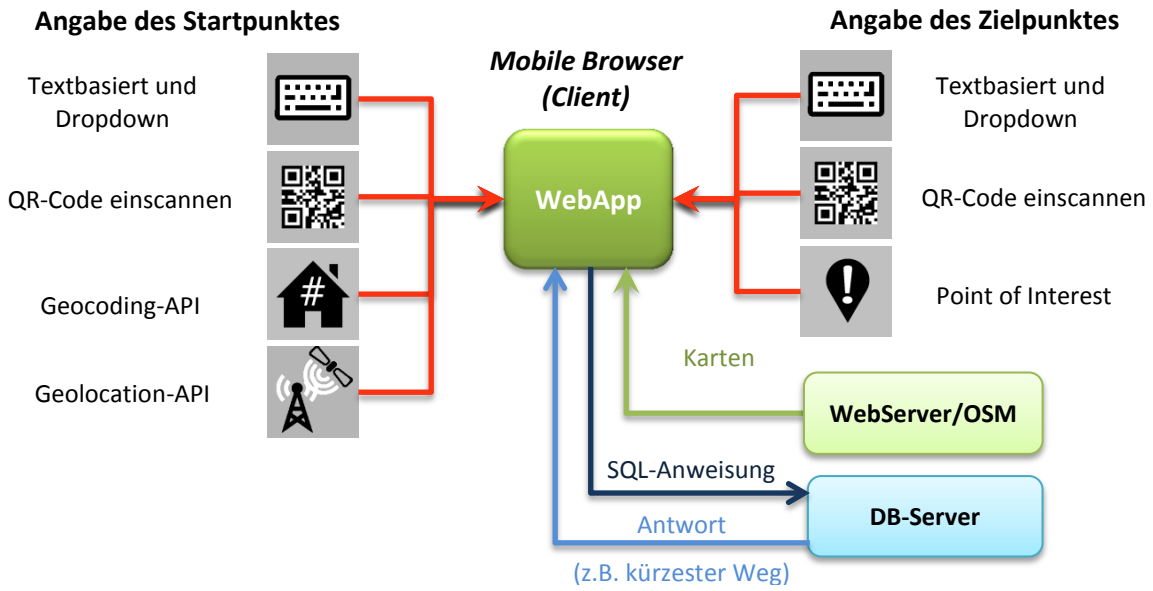

Abbildung 36 – Konzept der WebApp (Eigene Darstellung)

In Abbildung 36 ist das konzeptionelle Schema der WebApp grafisch dargestellt. Zwischen Datenbank-Server und WebApp werden während einer Anwendung laufend Informationen ausgetauscht. Der kürzeste Weg entspricht dem Endergebnis mehrerer Anfragen, da in der Realität laufend Daten zwischen Datenbank und Client ausgetauscht werden.

Schon bei der Datenbeschaffung wurde klar, dass für die Generierung und Verwaltung der Daten eine zusätzliche Plattform geschaffen werden muss. Wegen der technologischen Gemeinsamkeiten mit der mobilen Plattform wurde eine webbrowserbasierte Variante ausgewählt, die als WebGIS bezeichnet wird. In beiden Plattformen kommen dieselben Technologien zum Einsatz, nämlich HTML, PHP, JavaScript und Openlayers 3. Die Alternative wäre die Entwicklung eines QGIS-Plug-Ins gewesen und das hätte durch die Verwendung einer weiteren Programmiersprache (Python bzw. QPy) einen beträchtlichen Mehraufwand bedeutet.

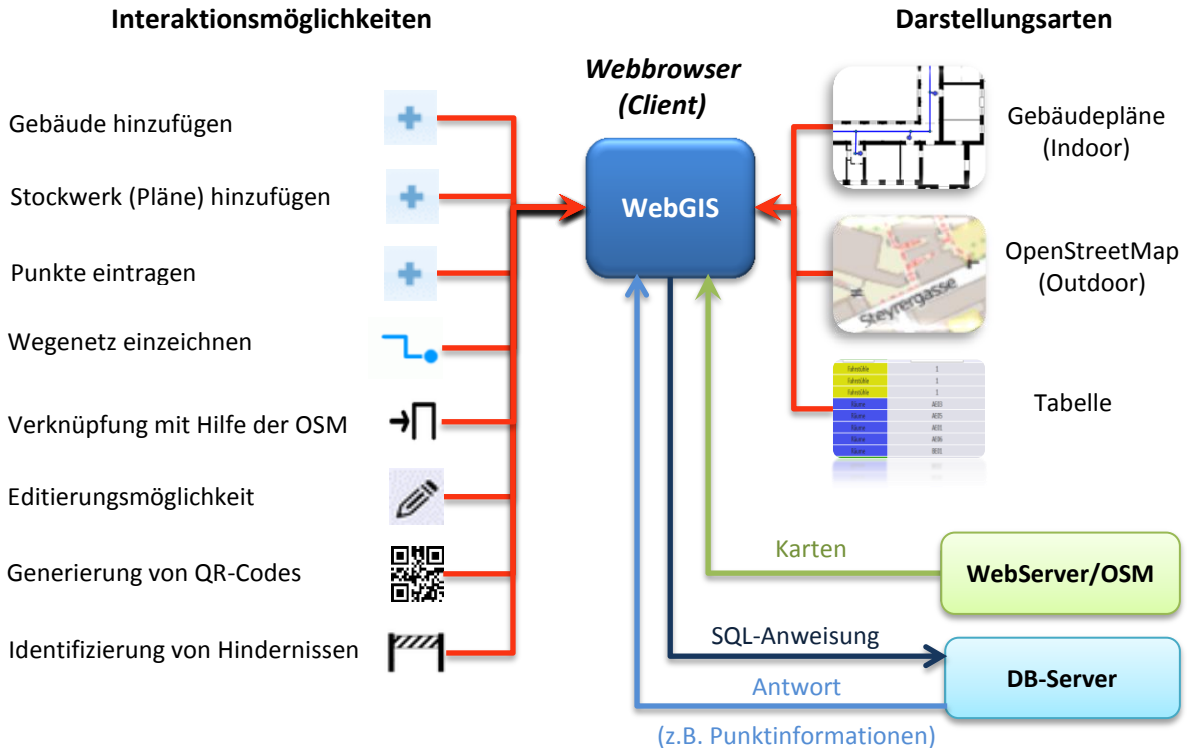

Abbildung 37 – Konzept des WebGIS (Eigene Darstellung)

Das konzeptionelle Schema des WebGIS wird in Abbildung 37 grafisch dargestellt. Die zahlreichen Interaktionsmöglichkeiten können in einer der Karten (Indoor oder Outdoor) sowie in einer Tabelle bedient werden. Diese Werkzeuge dienen der Verwaltung des Datenbankservers, den auch die WebApp als Datengrundlage nutzt.

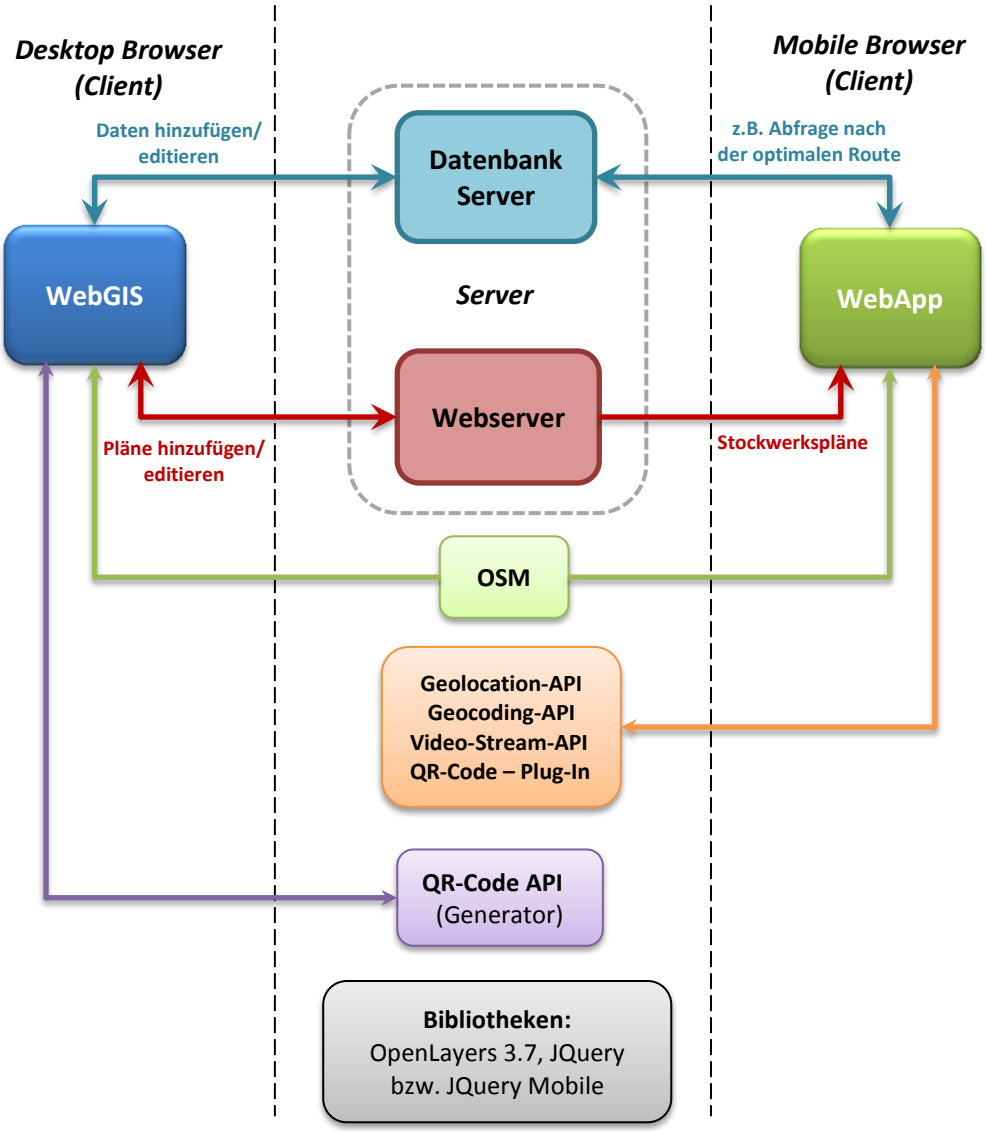

Abbildung 38 – Konzept der zwei Plattformen (Eigene Darstellung)

In Abbildung 38 ist das Konzept der zwei Plattformen dargestellt. Es zeigt, dass beide Plattformen auf gemeinsame Technologien zugreifen. Um den mobilen Ansprüchen gerecht zu werden, wird für die WebApp die JavaScript-Bibliothek JQuery-Mobile verwendet. Vor allem bei der Kartenerstellung mit OpenLayers und in der Datenbank-Kommunikation mittels PHP und JavaScript konnten die Gemeinsamkeiten vorteilhaft genutzt werden.

5.2 Datenbeschaffung

Für die Diplomarbeit musste eine Reihe von Daten gesammelt, erzeugt und aufbereitet werden. Die Schwerpunkte der Datensuche kann man unterteilen in Indoor- und Outdoor-Bereich.

5.2.1 Indoor-Wegenetz

Als Grundlage für die Indoor-Navigation wurden die Rasterkarten aus dem TUGonline gewählt, da diese für die Stockwerke aller Gebäude der Universität vorhanden sind. Im selben System sind gleichzeitig auch nützliche Informationen zu den Räumen abrufbar. Außerdem sind diese Karten leicht zu beschaffen und in einer akzeptablen Qualität verfügbar. Wegen der geringeren Verfügbarkeit von CAD-Files (Vektordaten) und der aus rechtlichen Gründen schwierigeren Beschaffung wurde bewusst auf deren Verwendung verzichtet.

Um die Räume, Eingänge, Stiegen, Fahrstühle und sonstigen Punkte des Gebäudes zu identifizieren, wurde ein webbasiertes GIS (WebGIS) erstellt. Dabei werden die Bildkoordinaten und Attribute der Punkte in einer Datenbank gespeichert. Die Datenbank kann auch mit zusätzlichen Informationen aus dem TUGonline befüllt werden (wie z.B. Raumnummer oder Nutzungsart). Für die Generierung des Wegenetzes und der dazwischen liegenden Wegpunkte wurden zwei unterschiedliche Varianten entwickelt.

- **Automatisierte Wegenetzerstellung:**
  Die Identifizierung der Punkte erfolgt im WebGIS. Anschließend werden die Punkte und deren Attribute am Webserver anhand von Textfiles gespeichert. Die Generierung des Wegenetzes erfolgt mittels MATLAB, wobei als Input, neben den Textfiles des Gebäudes, die Karten der Stockwerke verwendet werden. Diese Variante ist in Abschnitt 5.3 genauer beschrieben.

- **Semi-automatisierte Wegenetzerstellung:**
  In dieser Variante erfolgt die Wegenetzerstellung, wie auch die Identifizierung der Punkte, im WebGIS. In dieser Variante zeichnet der Anwender die Wege jedoch selbst in der Karte ein. Das wird durch die visuelle Interpretation des Anwenders realisiert. Diese Variante wird im Abschnitt 5.6 genauer beschrieben.

Speicherformat: Die im Gebäude eingetragen Punkte werden in einer Tabelle der PostgreSQL-Datenbank gespeichert. Das eingezeichnete bzw. generierte Indoor-Wegenetz wird in einer geketteten Adjazenzliste gespeichert (siehe 4.6.2). Für jedes Gebäude wird dafür eine eigene Tabelle angelegt.

Bestandteile eines Gebäudes: Ein Gebäude besteht aus mehreren Stockwerken, welche über Verknüpfungspunkte miteinander verbunden sind. Diese Verknüpfungspunkte können entweder Stiegen oder Fahrstühle sein. Die Anbindung des Gebäudes an den Outdoor-Bereich bzw. zu einem angrenzenden Nachbargebäude erfolgt über die Punktart „Eingang“. Um einen Gebäudeeingang mit dem Außenbereich zu verknüpfen, ist außerdem noch eine Identifizierung des Eingangs in der OSM notwendig. Weitere Bestandteile sind Zugänge zu relevanten Räumen sowie Points of Interest (z.B. Kopierer).

Pixelbasierte Kosten: Die Kosten einer Kante werden zunächst anhand des Pixelabstandes zwischen Anfangs- und Endknoten der Kante bestimmt. Da der Maßstab der Rasterkarten nicht bekannt ist, lässt sich aus den Pixeln keine metrische Größe ableiten, und damit ist auch keine Zeitangabe möglich.

Geschätzte metrische Kosten: Durch die Identifizierung von mindestens zwei Eingängen in der OSM-Karte wird es möglich, einen Kartenmaßstab für die Gebäudepläne zu schätzen. Bei mehr als zwei Eingängen wird der Maßstab durch eine Mittelwertbildung ermittelt. Dieser Maßstab ist natürlich nicht sehr präzise und es wird auch vernachlässigt, dass der Kartenmaßstab für jedes Stockwerk unterschiedlich sein kann. Da die Webapplikation nur die benötigte Zeit in Minuten angibt, welche man anhand des angenommenen Schritt- bzw. Fahrtempos berechnet, kann diese Problematik vernachlässigt werden. Die Informationen zu Schritt- und Fahrtempo beruhen auf (sejox.de - Geschwindigkeit eines Fußgängers 2015) und (trafficmaxx.de - Geschwindigkeit eines Radfahrers 2015).

Raumregistrierung: Da die Navigationsanwendung primär auf Studenten zugschnitten ist, werden auch nur die relevanten Räume registriert. Auf die Eintragung von Serverräumen und Büros wurde verzichtet, um die Sicherheit und Privatsphäre nicht einzuschränken.

5.2.2 Outdoor-Wegenetz

Vor der Beschaffung des Kartenmaterials zur Routenberechnung waren folgende Punkte (Tabelle 15) zu überlegen:

Tabelle 15 – Fragestellungen zu den Outdoor-Daten

#	Fragestellung	Antwort
1	Wie soll die Routenberechnung erfolgen?	pgRouting (per Datenbank)
2	Welcher Inhalt soll in der Karte dargestellt werden?	Alle Verkehrswege und zusätzlich alle Fuß- und Radwege
3	Welcher Bereich soll abgedeckt werden?	Graz
4	Welche Hintergrundkarte soll verwendet werden?	OpenStreetMap
5	Sollen die Daten frei verfügbar sein?	ja

Erhebung von routingfähigem Kartenmaterial:

Um einen der Routing-Algorithmen (siehe Abschnitt 4.7) der postgreSQL-Datenbank anwenden zu können, muss die Kantentabelle in einer entsprechenden Topologie vorliegen.

Für die Konvertierung der Daten des OSM-Verkehrswegenetzes in die entsprechende Form kann die Anwendung osm2po verwendet werden. Jedoch müssen auch die Rohdaten in einem der folgenden Dateiformate vorliegen: bz2, pbf, o5m oder o5m.gzm (osm2po.de 2015). Durch das Ausführen von folgendem Befehl in der Eingabeaufforderung wird eine SQL-Datei erzeugt:

```
java -Xmx1408m -jar osm2po-core-5.0.0-signed.jar prefix=at tileSize=x, c
http://download.geofabrik.de/europe/austria-latest.osm.pbf
```

Die SQL-Datei, die sehr groß sein kann, wird unter `http://localhost:8888/Osm2poService` gespeichert und mittels der Datenbank-Verwaltungssoftware pgAdmin ausgeführt. Für den Datensatz wird in der Datenbank eine Kantentabelle in pgRouting-tauglicher Form angelegt. Nähere Informa-

tionen zu pgrouting sind unter (docs.pgrouting.org - pgr-create-topology 2015), (docs.pgrouting.org - pgRoutingDocumentation.pdf 2015) und (live.osgeo.org - pgrouting_quickstart 2015) zu finden.

Die Rohdatensätze des Verkehrswegenetzes sind auf den Seiten wie Planet-OSM oder Geofabrik im pbf-Format frei verfügbar (geofabrik.de - Download 2015). Diese Datensätze sind jedoch für Straßenfahrzeuge optimiert und enthalten daher auch nur Verkehrswege, die von Fahrzeugen benutzt werden dürfen. Um die Problematik zu verdeutlichen, wurde im QGIS eine Karte erstellt (siehe Abbildung 39). Dabei handelt es sich um einen Ausschnitt der Innenstadt von Graz rund um die Herrengasse.

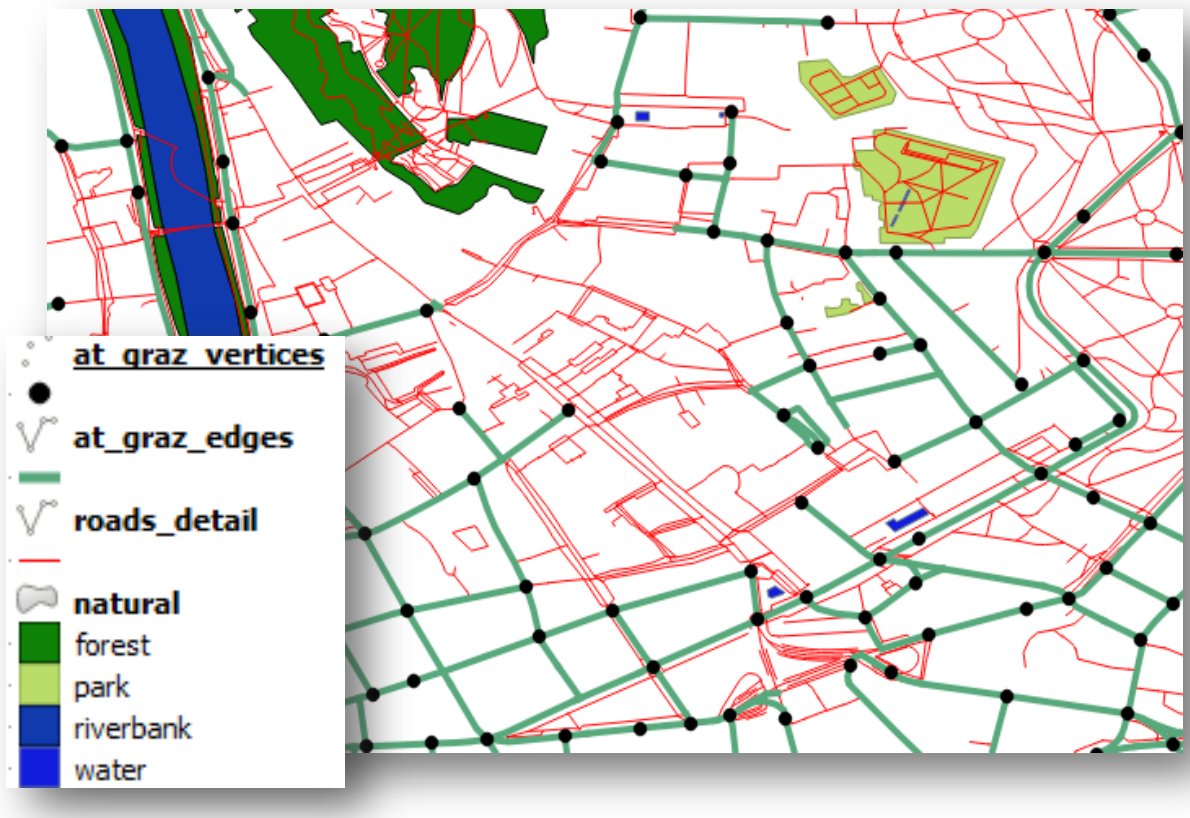

Abbildung 39 – Graz (QGIS-Karte, eigene Darstellung)

Die grünen Linien stellen das Straßennetz für Fahrzeuge dar, welches mit osm2po in ein routingfähiges Datenbankformat konvertiert wurde (`at_graz_edges`). Die schwarzen Punkte sind die mit dem Befehl `pgr_createTopology` berechneten Knoten (`at_graz_vertices`). Der rote Layer liegt im Shape-Format vor (`roads_detail`). Dieser besteht aus Fuß- und Radwegen sowie dem kompletten Straßennetz (ebenfalls auf Geofabrik frei verfügbar). Dabei sind auch jene Straßen mit Fahrverbot enthalten wie z.B. die Fußgängerzonen oder der Jakominiplatz. Das Shape-Format ist jedoch mit osm2po oder pgRouting nicht kompatibel und daher auch nicht routingfähig.

Der Vergleich zeigt, dass der routingfähige Datensatz nur auf das normale Straßenverkehrsnetz beschränkt ist. Benötigt man für Routingzwecke jedoch auch Fuß- und Radwege, muss man nach einer anderen Lösung suchen, um das detaillierte Wegenetz, welches im Shape-Format vorliegt, ebenfalls routingfähig zu machen.

Generierung eines routingfähigen Outdoor-Wegenetzes:

Das routingfähige Datenformat hat wie folgt auszusehen:

Tabelle 16 – routingfähiges Datenformat

id	osm_id	osm_name	clazz	source	target	cost	reverse_cost	kmh	X1	Y1	X1	Y1
INT	BIGINT	VARCHAR	INT	INT		DOUBLE	DOUBLE	INT	DOUBLE			

Die rot markierten Attribute sind unbedingt erforderlich. Das Attribut „id“ dient der eindeutigen Identifikation der Kante. „source“ und „target“ entsprechen der ID des Start- bzw. Zielpunktes des Pfades und „cost“ der Distanz in Kilometer. Die Attribute „X1“, „Y1“, „X2“, „Y2“ sind die Koordinaten des Anfangs- bzw. Endknotens des Pfades. Die weiteren Attribute „clazz“, „reverse_cost“ und „kmh“ sind für die Fahrzeugnavigation wichtig, da aus diesen die spezifischen Kosten berechnet werden können.

Der zu verwendende Vektordatensatz (*.shp) der Geofabrik musste zuerst in dieses Format gebracht werden. Dieser enthielt auch noch zusätzliche Attribute, wie „bridge“, „oneway“ oder „tunnel“, die jedoch nicht berücksichtigt wurden. Da der Datensatz sehr groß ist (ca. 300MB), weil er alle Straßen und Wege von Österreich enthält, wurde dieser im ersten Schritt auf das Gebiet der Stadt Graz beschränkt. Dazu wurde im QGIS mit Hilfe eines Auswahlpolygons und der Clip-Funktion der relevante Teil herausgeschnitten.

Mit QGIS wurde der Shape-Datensatz auch noch in eine für die Datenbank kompatible Form gebracht (Linestring statt Multilinestring) und in die postgreSQL-Datenbank geladen. Die übertragenen Daten enthalten auch die Geometrie des vollständigen Pfades mit all seinen Wegpunkten. Diese Geometrie ist für die Visualisierung des Wegenetzes in der Karte wichtig. Folgende Änderungen wurden in der Datenbank an der Tabelle `shape_roads_graz_raw` vorgenommen:

- Umbenennung der Attribute
- Hinzufügen des Attributs „geom_text“, das die Geometrie des Pfades im WKT-Format enthält.
- Alle Straßennamen („osm_name“), die `NULL` enthalten, werden auf ‚footpath‘ geändert.

Mit der Software FLOW-HEATER[1] V3.x wurde die Tabelle schließlich als Textfile (semicolon-separated-Format) aus der Datenbank importiert. Dieses File wurde in MATLAB eingelesen, wo eine Konvertierung der Daten in ein routingfähiges Format erfolgte. Dabei passierte Folgendes:

1. Eliminierung des Geschwindigkeitsanteils aus den Kosten, da dieser für die Fußgänger nicht benötigt wird.
2. Für die Straßentypen *‚motorway‘* und *‚motorway_link‘* wurden die Kosten („km“) mit einem hohen Wert versehen, da diese nicht benutzt werden sollen.
3. Mit einem separaten Kosten-Attribut („cost_disabled”) werden Stufen („type“ = *‚steps‘)* ebenfalls mit einem hohen Wert versehen. Dieser Schritt wurde durchgeführt, um die Kanten für Radfahrer oder körperlich beeinträchtigte Menschen als Hindernis zu kennzeichnen.
4. Generierung einer Liste mit unterschiedlichen Straßentypen.
5. Jeder Kante für den Straßentyp („clazz“) den entsprechenden Integer-Wert zuweisen.
6. Detektion des Start- und Zielknotens aus dem vollständigen Pfad, der im WKT-Format (LINESTRING) vorliegt.

[1] Freeware Version auf 10.000 Datensätze beschränkt. https://flowheater.net/de

7. Zusammenfassen aller Start- und Zielpunkte zu einer Knotenliste, in der jeder Knoten nur einmal vorkommt.
8. Jedem Pfad die ID des Start- und Zielknotens zuweisen, basierend auf der zuvor erstellten Liste. Dabei entspricht der Startknoten dem „source“- und der Zielknoten dem „target“-Attribut.
9. Speichern der Ergebnisse in einem entsprechenden Format.

Die Implementierung dieses Algorithmus in einer eigenen pl/SQL-Funktion hätte den Vorteil gehabt, dass die Berechnung direkt in der Datenbank erfolgen würde. Da jedoch in pl/SQL keine Matrizen-Berechnung unterstützt wird und die Datenaufbereitung des Grazer Wegenetzes nur einmal notwendig ist, wurde stattdessen MATLAB verwendet.

Das mit MATLAB erzeugte Textfile wurde anschließend mit der FLOW HEATER Software[2] als neue Tabelle (`roads_graz_pedestrian`) in die Datenbank exportiert. Auch dieses Mal mussten wieder Änderungen in der Tabelle vorgenommen werden:

- Hinzufügen des Attribut „geom_way“ (Format: `GEOM`), da dieses durch den Konvertierungsvorgang nicht übermittelt wird.
- Hinzufügen des Attributes „cost“. Dieses bekommt den Wert des Attributes km zugewiesen, da die fußgängeroptimierten Kosten nur von der Distanz abhängig sind. Dabei ist auch noch zu erwähnen, dass für eine spätere Änderung zu einem für Fahrzeuge optimierten Routing die entsprechenden Attribute angepasst bzw. erhalten geblieben sind.
- Entfernung aller Punkte, wo der „source“- oder „target“-Punkt fehlt und kein Eintrag in der Geometriespalte enthalten ist.
- Generierung einer Punkttabelle mit dem `pgr_createTopology`-Befehl.
- Hinzufügen des Attributes „disabled”. Diese wird dazu verwendet, um bestimmte Kanten für körperlich beeinträchtigte Menschen entsprechend zu kennzeichnen.

Durch eine Reihe von Transformationen und SQL-Befehlen ist es möglich geworden, das Wegenetz vom Shape-File in eine routingfähige PostgreSQL-Tabelle zu transformieren.

Alternativ wurde auch noch getestet, aus der Kanten- und Knotenliste eine verkettete Adjazenzliste (siehe Abschnitt 4.6.2) zu erstellen und mit einem eigenen Dijkstra-Algorithmus den schnellsten Weg zu berechnen. Die Berechnungszeit ist jedoch aufgrund der hohen Punktanzahl sehr hoch und die Variante wurde nicht weiter verfolgt.

5.2.3 Universitätsgebäude

Von der Geofabrik wurde auch ein Shapefile herunter geladen, indem nahezu alle Gebäude Österreichs enthalten sind. Aus diesem Datensatz wurden mit Hilfe von QGIS alle Universitätsgebäude der Stadt Graz herausgefiltert.

Befehl:

```
CREATE TABLE buildings_university AS SELECT * from buildings WHERE type LIKE
'university' AND ST_Within(the_geom, ST_Transform((SELECT polygon.the_geom
FROM polygon_graz AS polygon), 4326));
```

[2] Freeware Version auf 10.000 Datensätze beschränkt. https://flowheater.net/de

5.3 Automatisierte Wegenetzerstellung

In der Arbeit wurde unter anderem untersucht, ob es Sinn macht, das Wegenetz eines Gebäudes automatisch zu generieren. Für die Entwicklung der Algorithmen wurde MATLAB verwendet. Die Berechnung benötigt folgende Eingangsdaten:

- Rasterkarten der Stockwerke im png-Format
- Informationsdatei über die Anzahl und Bezeichnung der Stockwerke
- Liste der registrierten Zielpunkte, welche im WebGIS als Raum, Eingang, Fahrstuhl, Stiege und sonstiger Punkt eingetragen werden.

Der erste funktionierende Algorithmus, der entwickelt wurde, basiert auf der Generierung von Rechtecken zwischen den Zielpunkten. Dazu wird der Gebäudeplan, welcher als Rastermatrix eingelesen wird, am Beginn in eine binäre Matrix konvertiert. Bei der Konvertierung bekommen alle dunklen Pixel (Wände/Säulen/Türen) den Wert 0 und alle hellen Pixel den Wert 1. Mithilfe dieser Matrix kann für jedes generierte Rechteck überprüft werden, ob dieses die Wand schneidet oder nicht. Alle Rechtecke, die keine Wandpixel enthalten, bleiben übrig und werden so weit in alle vier Richtungen ausgedehnt, bis auch diese die Wand berühren. Rechtecke, die sich zur Gänze innerhalb eines anderen Rechteckes befinden, werden ebenfalls aussortiert. Im nächsten Schritt werden aus den Rechtecken Weglinien erzeugt und aus dem Schnitt mehrerer Linien Wegpunkte erstellt. Das Ergebnis ist ein Wegenetz. Es kann jedoch möglich sein, dass aus einem oder mehreren registrierten Zielpunkten kein gültiges Rechteck generiert werden kann. In diesem Fall kann der Anschluss des Punktes an das Wegenetz nicht hergestellt werden. Daher wird der Prozess der Rechteck-Generierung iterativ durchgeführt, indem nach jedem Iterationsschritt die berechneten Wegpunkte zu den Zielpunkten hinzugefügt werden. Sollten die Gänge aber so verzweigt sein, dass dennoch keine Verbindung hergestellt werden kann, ist es erforderlich, im WebGIS manuell Wegpunkte einzutragen.

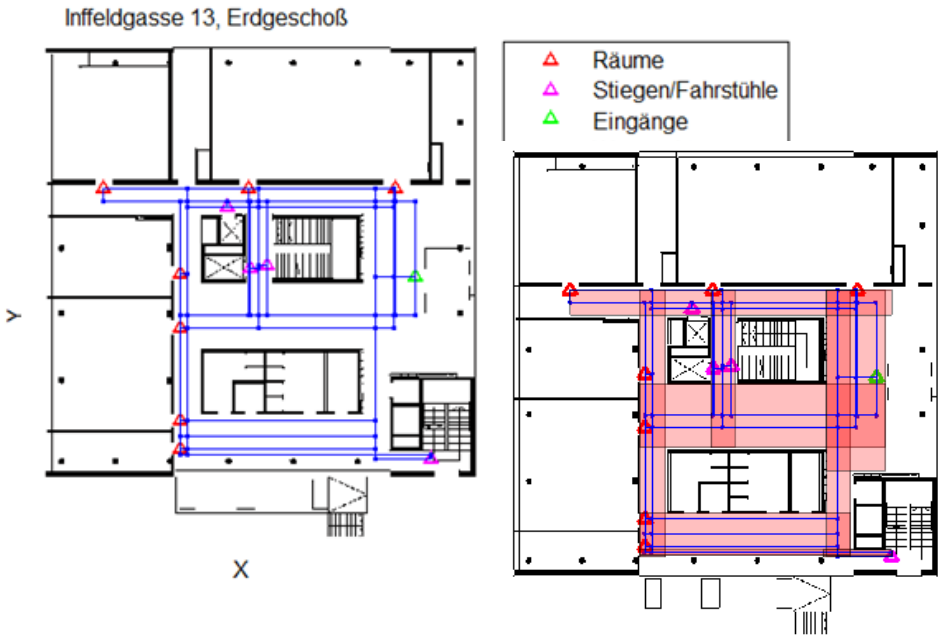

Abbildung 40 – Orthogonales Wegenetz, Inffeldgasse 13 (Eigene Darstellung)

Die Abbildung 40 zeigt das berechnete Wegenetz eines Gebäudes, in dem die Wände orthogonal angeordnet sind. In diesem Fall war das Ergebnis zufriedenstellend.

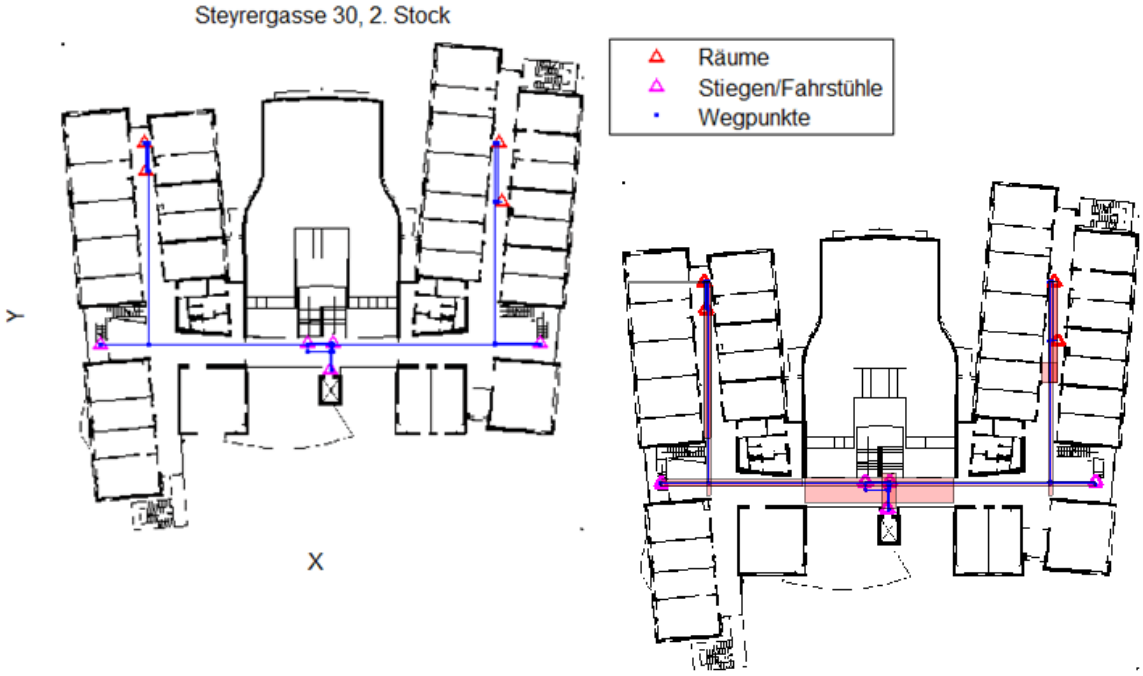

Abbildung 41 – Orthogonales Wegenetz, Steyrergasse 30 (Eigene Darstellung)

In Abbildung 41 ist das Gebäude Steyrergasse 30 (Mathematik und Geodäsie) als Beispiel abgebildet. Gerade dieses Gebäude eignet sich sehr gut für die Entwicklung des Algorithmus, da die Gänge hier teilweise nicht orthogonal angeordnet sind. Der Nachteil der zuvor beschriebenen Rechteck-Methode ist in diesem Beispiel gut zu erkennen. Es können keine schrägen Verbindungen erzeugt werden und die berechneten Wege haben eine unrealistische Form.

Um diese Anforderung zu lösen, wurde eine weitere Methode entwickelt, die eine Abtastung entlang der Verbindungslinie zwischen zwei Zielpunkten durchführt, siehe Abbildung 42 (1). Diese Abtastung wird dabei entweder in x- oder y-Richtung durchgeführt. Dies geschieht in negativer und positiver Richtung, bis ein Wandpixel detektiert wird. In negativer und positiver Richtung wird also jeweils ein Pixel bestimmt und anhand der Pixelkoordinaten die Differenz zwischen diesen berechnet. Entlang der Verbindungslinie ergeben sich meist unterschiedliche Differenzen, die im Anschluss klassifiziert werden. Daraus kann schließlich der dominanteste Wert bestimmt werden. Dieser Wert entspricht der mittleren Breite des Korridors. Da auch die Koordinaten der Wandpixel gespeichert werden, können zwei neue Wegpunkte berechnet werden. Diese sind dort, wo der dominante Wert das erste bzw. das letzte Mal aufgetreten ist. Die neu berechnete Verbindungslinie zwischen den zwei berechneten Wegpunkten ist im Normallfall parallel zum detektierten Korridor. Im letzten Schritt wird noch zwischen den jeweiligen Weg- und Zielpunkten eine Verbindung hergestellt, siehe Abbildung 42 (2).

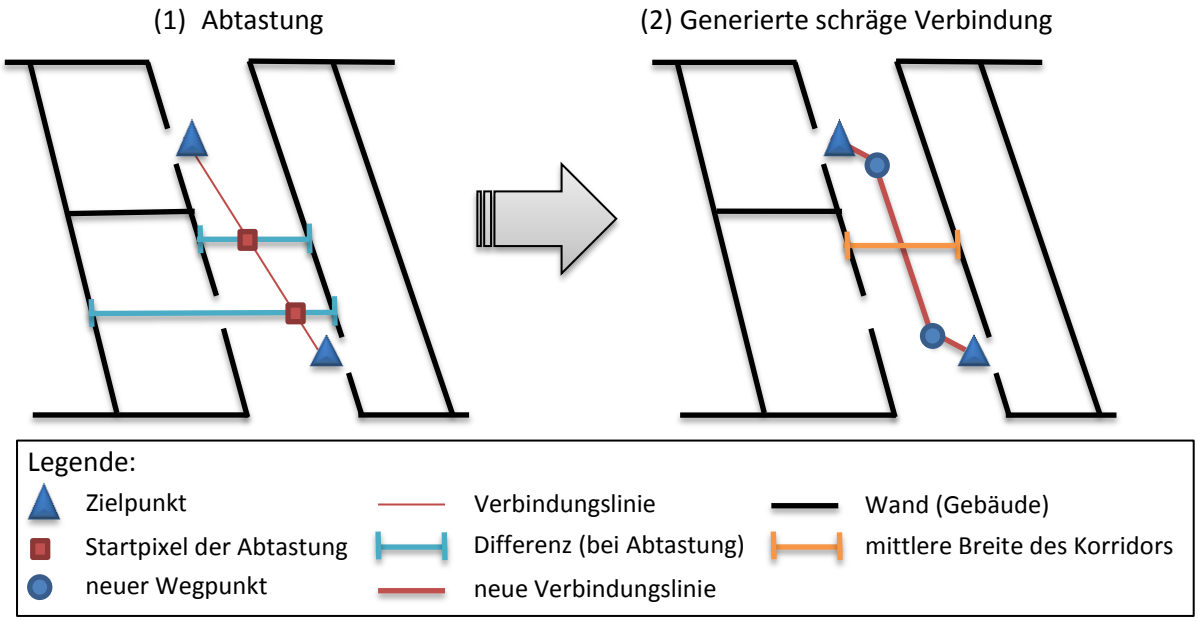

Abbildung 42 – Detektion des Korridors, Grundriss (Eigene Darstellung)

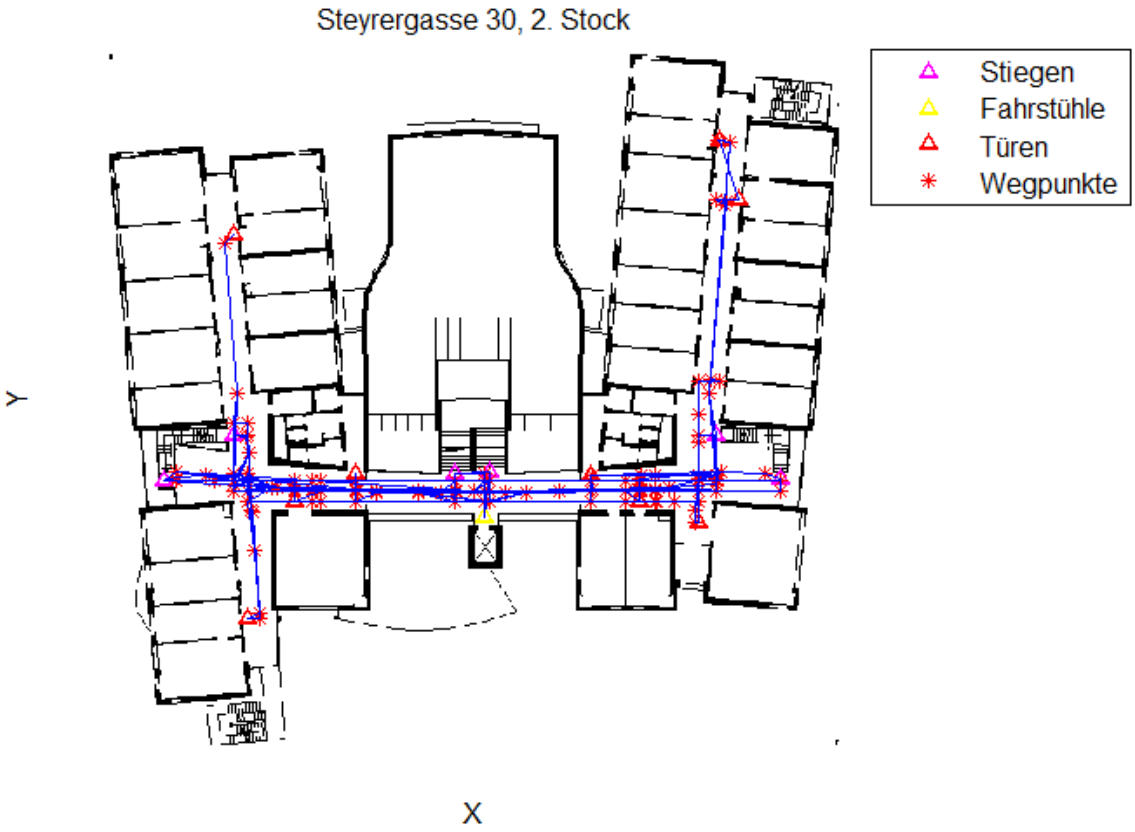

Abbildung 43 – Optimierte Methode, Steyrergasse 30 (Eigene Darstellung)

Abbildung 43 zeigt das Ergebnis der zuvor beschriebenen Methode, welche auch auf schräge Verbindungen optimiert ist. Man erkennt, dass in den Korridoren das Wegenetz nicht mehr orthogonal ist und dass Wege in der Mitte des Korridors platziert sind. Die Methode hat jedoch den Nachteil,

dass die Anzahl der Wegpunkte und Weglinien stark zunimmt. Das einwandfreie Reduzieren und Zusammenfassen der Punkte stellte die nächste Herausforderung dar, die nur ansatzweise gelöst werden konnte.

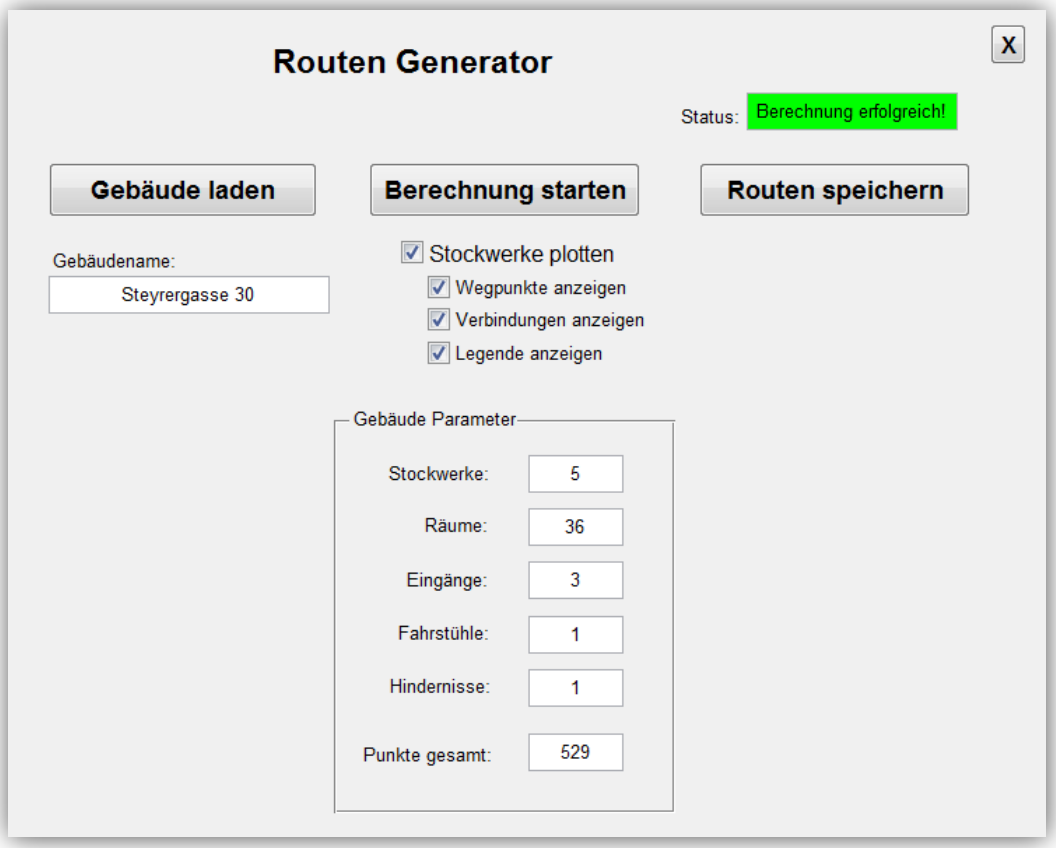

Abbildung 44 – GUI (Eigene Darstellung)

Zur Wegenetz-Generierung wurde auch ein GUI erstellt (siehe Abbildung 44). Durch die Auswahl einer Informationsdatei wird der komplette Input-Datensatz des entsprechenden Gebäudes geladen („Gebäude laden“). Um die Ergebnisse der Berechnung zu überprüfen, ist es möglich, das Wegenetz stockwerksweise zu plotten und bestimmte Gebäudeparameter im GUI anzuzeigen. Zur Weiterverarbeitung lässt sich das Wegenetz bei Bedarf in einem entsprechenden Datenformat speichern („Routen speichern“).

Die Kanten der generierten Wegenetze (Routen) werden in MATLAB mittels einer Adjazenzmatrix gespeichert. Diese Matrix enthält die Kanten aller Stockwerke, die über dieselbe Matrix miteinander verknüpft sind. Das Endergebnis der automatischen Wegenetzerstellung ist eine Knoten- und Kantenliste (gekettet Adjazenzliste, siehe 4.6.2), die in zwei Textdateien gespeichert wird.

Die Nachteile einer automatisierten Berechnung sind, dass:

- Ein hoher Zeitaufwand betrieben werden muss, um bestimmte Regeln im Algorithmus zu implementieren.
- Diese Regeln nur für einen bestimmten Kartenstandard gültig sind, z.B.: bezüglich der Darstellung der Türen (¼-Kreis oder komplett offen).
- Eine rasterbasierte Berechnung nur in x- und y-Richtung optimiert ist und die Realisierung von schrägen Verbindungen eine zusätzliche Herausforderung bedeutet.

- Die berechneten Routen oft den visuellen Ansprüchen nicht gerecht werden.
- Fehler meist nicht erkannt werden.

5.4 Aufbau der Datenbank

Für die Realisierung der beiden WebGIS- und WebApp-Plattformen werden zwei Server verwendet, auf denen folgende Daten gespeichert sind:

- Webserver:
  - WebGIS
  - WebApp
  - Gebäudepläne (je Stockwerk)
  - generierte QR-Codes
  - Sonstige Daten zur Weiterverarbeitung

- Datenbank-Server (postgreSQL-Datenbank):
  - Indoor-Ziel-/Wegpunkte (Registrierungsliste)
  - Indoor-Wegenetz (gekettete Adjazenzliste)
  - Outdoor-Wegenetz (Rohdaten, Adjazenzliste, Punktliste)
  - Universitätsgebäude
  - Verknüpfungslisten und ausgelagerte Listen

Die Gebäudepläne aus dem TUGonline sowie die generierten QR-Codes werden am Webserver gespeichert. Die Wegpunkte, Wegenetze und sonstigen Listen werden über den Datenbankserver verwaltet. Die Verwaltung der Indoor-Daten erfolgt ausschließlich über das WebGIS. Das bedeutet, dass eine Änderung in der Datenbank nur über den Aufruf eines PHP-Skriptes möglich ist. Die Daten des Outdoor-Bereiches werden hingegen fast zur Gänze mit der Geodatenbank-Verwaltungssoftware pgAdmin III aufbereitet.

Allgemein ist anzumerken, dass in der Datenbank die 3. Normalform nicht immer eingehalten wird, um bei der Programmierung eine gewisse Übersichtlichkeit zu gewährleisten.

5.4.1 Indoor-Daten

	id [PK] serial	name character varying	osm_id bigint	notation character varying	cost_meter boolean
1	1	Steyrergasse 30	31988058	Institute der Mathe	TRUE
2	4	Petersgasse 10-12	32348394	Biochemie	TRUE

Abbildung 45 – building_list (Eigene Darstellung)

Die Basistabelle der Daten, die sich auf das Innere eines Gebäudes bezieht, ist die Tabelle `building_list` (siehe Abbildung 45). In dieser Tabelle werden alle eingetragenen Gebäude miteinander verknüpft. Jedes Gebäude hat einen eindeutigen Primärschlüssel und einen eindeutigen Namen. Weitere Attribute verweisen auf die Geometrie des Gebäudes und geben Auskunft, ob die Kosten des Gebäudes in Meter oder Pixel angegeben sind.

Durch die Eintragung eines Gebäudes wird einerseits ein neuer Eintrag in der Basistabelle erstellt und andererseits werden für jedes Gebäude mehrere Tabellen angelegt. Der Name dieser Tabellen wird

aus der Identifikationsnummer *(id)* der Basistabelle und aus einer zusätzlichen Bezeichnung, die die Funktion der Tabelle beschreibt, generiert. Folgende Tabellen werden nach diesem Schema angelegt.

- `building_(id)`
- `building_(id)_registration`
- `building_(id)_edges`
- `building_(id)_nodes`
- `building_(id)_arcs`

	id [PK] integer	level_name character var	level integer
1	1	STEG	0
2	2	STK1	-1
3	3	STO1	1
4	4	STO2	2

Abbildung 46 – Stockwerksliste (Eigene Darstellung)

Die erste Tabelle dient dazu, alle Stockwerke des Gebäudes miteinander zu verknüpfen (siehe Abbildung 46). Die Attribute der Tabelle sind die Stockwerks-ID, der Stockwerksname, der aus dem Dateinamen des Stockwerksplans generiert wird und der Stockwerksnummer, die man beim Anlegen eines Stockwerks angeben muss.

In der Tabelle mit der Bezeichnung „registration“ werden alle Wegpunkte des Gebäudes eingetragen. Die Attribute der Tabelle sind in Tabelle 17 aufgelistet.

Tabelle 17 – Registrierungstabelle

Attribut	Beschreibung	Datentyp
id	Punkt-Identifizierung	Integer (PK)
type	Punkttyp	Integer (FK)
roomnumber	Raumnummer	Varchar
roomcode	Raumcode	Varchar
level	Stockwerk	Integer
usage	Nutzungsart	Integer (FK)
notation	Sonstige Bemerkung	Varchar
x_bk/y_bk	Bildkoordinaten	Integer
x_wk/y_wk	Weltkoordinaten	Double
id_down/id_up	Verknüpfung (Stiegen)	Integer
id_elevator	Verknüpfung (Fahrstühle)	Integer
id_neighbor_building/ **id_neighbor_point**	Verknüpfung (Nachbargebäude)	Integer Integer
mask	Maskierung (bzw. Ausblenden)	Integer
geom	Geometrie des Gebäudes (WK)	Geometrie(POINT, 4326)

Die Attribute „usage“ und „type“ sind Fremdschlüssel und verweisen jeweils auf eine Tabelle, in der die Nutzungsart bzw. die Punkttypen gespeichert sind. Das hat den Vorteil, dass diese Attribute leichter geändert oder erweitert werden können.

Nutzungsarten sind:

- Hörsaal
- Seminarraum
- EDV-Raum
- Labor
- Zeichensaal
- Studienarbeitsraum
- Sitzungszimmer
- Sekretariat
- Kopierraum
- Kaffeeautomat
- Mensa
- Bibliothek
- Werkstatt
- Sanitärraum

Punkttypen sind:

- Eingänge
- Stiegen
- Fahrstühle
- Wegpunkte
- Räume
- Sonstige
- Hindernisse

Das Wegenetz wird in der Tabelle mit der Bezeichnung „edges“ gespeichert. Dort werden die Geometrien stockwerksweise im WKT-Format gespeichert. Bei jeder Änderung am Wegenetz eines Stockwerkes wird das Geometrieattribut der entsprechenden Zeile aktualisiert.

Um mit dem erstellten Wegenetz routen zu können, wird aus der letzteren Tabelle eine gekettete Adjazenzliste erzeugt, die aus zwei Tabellen besteht. Diese werden nach jeder Aktualisierung (Speichern) des Wegenetzes neu angelegt. Die Knotentabelle (Bezeichnung: „nodes“) enthält alle Knoten des Gebäudes genau einmal und verweist mit den Bogenindizes auf die Bogentabelle (Bezeichnung: „arcs“), welche die Kosten zwischen den Knoten enthält. Eine genauere Beschreibung der geketteten Adjazenzliste ist in Abschnitt 4.6.2 zu finden. Da bei der Berechnung der geketteten Adjazenzliste neue Knoten-IDs berechnet werden, enthält die Knotentabelle zusätzlich noch die Knoten-ID der Registrierungstabelle als Fremdschlüssel-Attribut. In der Registrierungstabelle werden bei jeder Aktualisierung des Wegenetzes alle Wegpunkte entfernt und neu eingetragen.

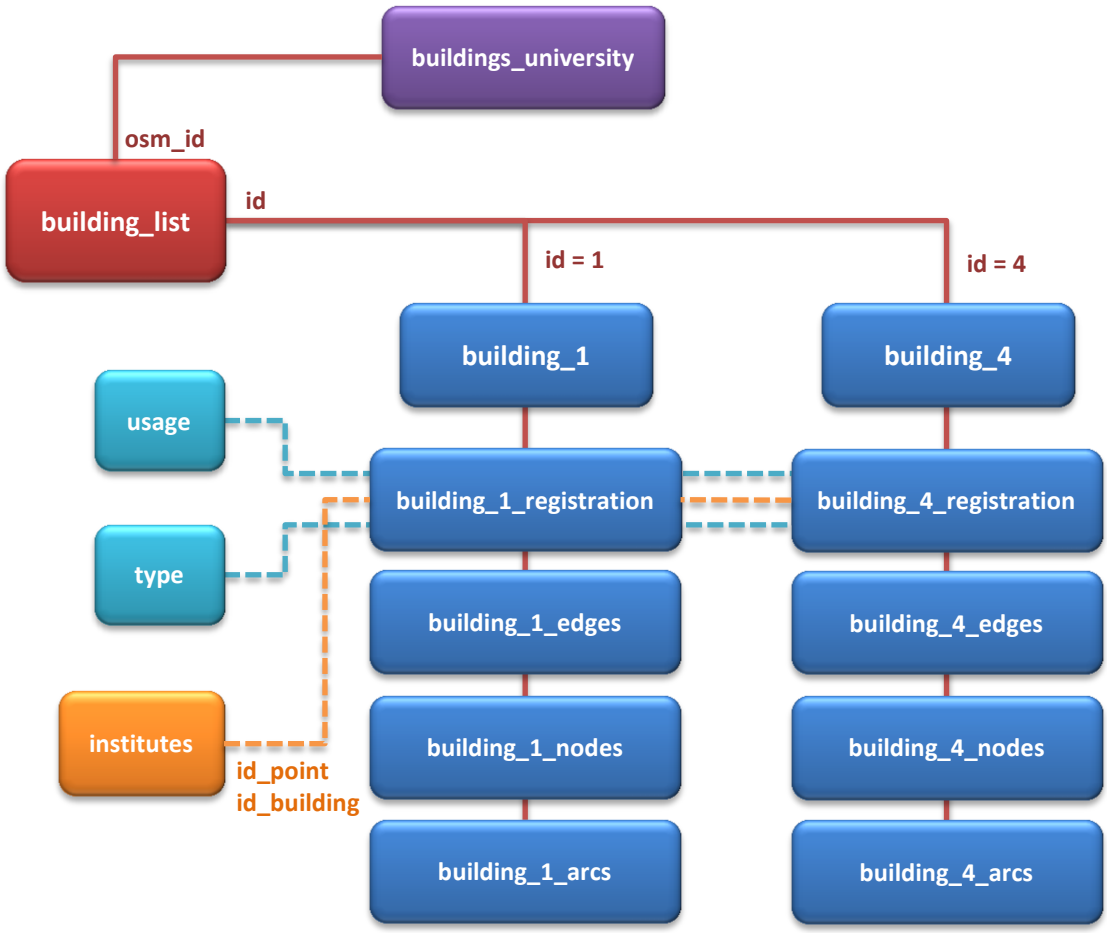

Abbildung 47 – Beziehungen der Datenbank (Eigene Darstellung)

In Abbildung 47 sind die wichtigsten Beziehungen der Tabellen der Datenbank grafisch dargestellt.

5.4.2 Outdoor-Daten

Die Tabellen der Daten, die sich auf den Outdoor-Bereich beziehen, bestehen aus:

- einer routingfähige Adjazenzliste (`roads_graz_pedestrian`)
- der daraus generierten Knotenliste (`roads_graz_pedestrian_vertices_pgr`)
- einer Liste der Universitäten von Graz inklusive deren Geometrien (`buildings_university`)

Die routingfähige Adjazenzliste des Wegenetzes, die in MATLAB generiert wurde (siehe 5.2.2), hat folgende Attribute:

Tabelle 18 – roads_graz_pedestrian

Attribut	Beschreibung	Datentyp
id	Punkt-Identifizierung	Integer (PK)
osm_id	*osm_id*	*Bigint*

osm_name	*Straßenname oder „footpath“*	*Varchar*
osm_source	*osm_id des Startknotens*	*Bigint*
osm_target	*osm_id des Zielknotens*	*Bigint*
clazz	Klasse	Integer (FK)
source	id des Startpunktes	Integer
target	id des Zielpunktes	Integer
cost	Kosten für Fußgänger (entspricht km)	Double
km	Distanz in Kilometer	Double
cost_vehicle	*Kosten für Fahrzeug (abhängig von km und kmh)*	*Double*
reverse_cost	*Kosten in rückwärtiger Richtung*	*Double*
kmh	*Geometrie des Gebäudes (WK)*	*Integer*
X1/Y1	Koordinaten des Startpunktes (λ bzw. φ, 4326)	Double
X2/Y2	Koordinaten des Zielpunktes (λ bzw. φ, 4326)	Double
disabled	= FALSE wenn für Personen mit körp. Beeintr. nicht passierbar	Boolean
cost_disabled	Kosten für Personen mit körperlicher Beeinträchtigung	Double
geom_text	Geometrie der Polylinie/Kante im WKT-Format	Varchar
geom_way	Geometrie der Polylinie/Kante im WKB-Format	Geometrie(LineString, 4326)

Die *kursiv* markierten Attribute werden in der Arbeit nicht benötigt, sind aber vollständigkeitshalber in der Tabelle enthalten. Das Fremdschlüssel-Attribut „clazz“ verweist auf eine Tabelle, in der die Bezeichnung der Klasse gespeichert ist.

Die Knotenliste, welche mit dem pgrouting-Befehl erzeugt wurde, enthält die Geometrie des Punktes im WKB-Format sowie einige Integer-Attribute, die aber erst bei weiteren pgrouting-Befehlen befüllt werden.

Die Tabelle `buildings_university` enthält die Attribute „id“, „osm_id“ und das Attribut „name“, das die Bezeichnung des Gebäudes enthält. Außerdem ist die Polygon-Geometrie des Gebäudes im WKB-Format unter dem Attribut „the_geom“ enthalten. Um auf die Tabelle zu referenzieren, enthält die Gebäudetabelle (`building_list`) das Fremdschlüssel-Attribut „osm_id“, das auf die Tabelle der Universitätsgebäude verweist.

5.4.3 Sonstige Listen

Es gibt noch weitere Tabellen, die bestimmte Aufgaben erfüllen. Die Tabelle `institutes` enthält die Attribute Gebäude- und Raum-ID der Sekretariate der Institute als zusammengesetzten Fremdschlüssel, mit dem auf einen bestimmten Raum eines Gebäudes verwiesen wird. Andere Tabellen wie `attributes` oder `colortable` enthalten Informationen über die schriftliche und farbliche Gestaltung der beiden Plattformen und stehen mit den restlichen Tabellen in keiner Beziehung.

5.5 Berechnung des optimalen Weges

Die Berechnung des kürzesten Weges zwischen zwei Punkten ergibt sich aus der Kombination von mehreren Berechnungen. Das liegt daran, dass die Wegenetzte der einzelnen Gebäude sowie das Outdoor-Wegenetz voneinander getrennt sind. Daher müssen die Berechnungen für das jeweilige Wegenetz separat durchgeführt werden. Folgende Varianten können auftreten:

- Routing innerhalb eines einzelnen Gebäudes (nur Indoor)
- Routing zwischen zwei Gebäuden (Outdoor + 2 x Indoor)
- Routing zwischen zwei benachbarten Gebäuden, die miteinander verbunden sind (2 x Indoor)
- Routing von einem Startpunkt im Outdoor-Bereich zu einem Zielpunkt im Gebäude (Outdoor + Indoor)

Da der Schwerpunkt der Anwendung im Auffinden von Räumen liegt, ist die Angabe der Zielpunkte auf den Indoor-Bereich (Gebäude) beschränkt. Bei der Berechnung des kürzesten Weges kommen zwei unterschiedliche Algorithmen zum Einsatz:

- Für den Indoor-Bereich wurde der Dijkstra-Algorithmus in PHP implementiert. Dieser nutzt die gekettete Adjazenzliste (Knoten- und Kantenliste) aus der Datenbank. Der Algorithmus wurde so optimiert, dass dieser abgebrochen wird, sobald der ausgewählte Zielpunkt erreicht wird (single-pair shortest path).
- Für die Wegenetzberechnung im Outdoor-Bereich wurde die für PostgreSQL bereitgestellte Erweiterung pgRouting verwendet. Da das Wegenetz des OSM-Datensatzes wesentlich mehr Knoten und Kanten hat, wurde mit dem A*-Algorithmus ein heuristischer Algorithmus gewählt. Die Berechnung mittels pgRouting hat noch den zusätzlichen Vorteil, dass die Berechnung direkt in der Datenbank ausgeführt wird.

Um den Outdoor- und Indoor-Bereich miteinander zu verknüpfen, wird von jedem Gebäudeeingang der nächst gelegene Knotenpunkt als Start- bzw. Zielpunkt des Outdoor-Wegenetzes gewählt. Dazu müssen zuvor jedoch die Koordinaten des Eingangs näherungsweise bestimmt werden. Sollte sich der Startpunkt im Outdoor-Bereich befinden, wird von diesem Punkt ebenfalls der nähest gelegene Knoten bestimmt. In der WebApp sind jedoch nur Startpunkte innerhalb von Graz zulässig.

Da die Wegenetze der einzelnen Gebäude und das des Outdoor-Bereiches voneinander getrennt sind, müssen mehrere Berechnungen durchgeführt und verglichen werden. Die Anzahl der Berechnungen ist von der Anzahl der Gebäudeeingänge abhängig. Dazu sind in Tabelle 19 mehrere Beispiele mit der Gesamtanzahl der Berechnungen aufgelistet. Die gesamte Rechendauer ist auch von der Anzahl der Knoten der Wegenetze abhängig. Die Kosten der Ergebnisse werden verglichen und die kürzeste Variante wird ausgewählt. Die Route mit dem kürzesten Weg im Outdoor-Bereich bestimmt den Eingang des Gebäudes bzw. die Eingänge der Gebäude. Schließlich wird, ausgehend vom Eingang, der kürzeste Weg innerhalb des Gebäudes zum Start- oder Zielpunkt berechnet. Im Fall einer Wegenetzberechnung zwischen zwei miteinander verbundenen Gebäuden fällt die Berechnung im Outdoor-Bereich gänzlich weg. Stattdessen werden die Wegenetzberechnungen der beiden Gebäude miteinander verglichen. Sollten sich Start- und Zielpunkt im selben Gebäude befinden, ist der Indoor-Algorithmus nur einmal auszuführen und der Outdoor-Bereich fällt ebenfalls weg.

Tabelle 19 – Berechnungskombinationen

$N_{Gebäudeeingänge}$		$N_{Berechnungen}$			Bemerkung
Start	Ziel	Indoor	Outdoor	Gesamt	
1	1	2	1	3	zwei voneinander getrennte Gebäude
1	2	2	2	4	
2	3	2	2×3	8	
4	5	2	4×5	22	
3	3	$3 + 3$	0	6	zwei miteinander verbundene Gebäude
-	-	1	0	1	ein einzelnes Gebäude
-	3	1	1×3	4	Start im Outdoor-Bereich

Der Dijkstra-Algorithmus wurde neben PHP auch in JavaScript und in pl/SQL implementiert, um zu testen, ob sich bezüglich Laufzeit Verbesserungen ergeben. Die Unterschiede waren jedoch minimal. Bei der Verwendung größerer Datenmengen ist die Verwendung von pl/SQL jedoch von Vorteil.

5.6 WebGIS

Beim WebGIS handelt es sich um eine desktopbasierte Plattform zur Anlegung und Verwaltung eines routingfähigen Wegenetzes. Die Anwendung wird über einen Webbrowser ausgeführt und bezieht die Daten vom Web- und Datenbankserver.

5.6.1 Allgemeine Beschreibung

Das WebGIS besteht aus mehreren Kartenfenstern und zahlreichen Bedienelementen, die meist auch durch Symbole gekennzeichnet sind. Zusätzlich werden auch Popups eingesetzt.

Um ein routingfähiges Wegenetz zu erstellen, muss zu aller erst ein Gebäude hinzugefügt werden, für das wiederum mindestens ein Stockwerksplan hochgeladen werden muss. Im Indoor-Kartenfenster können anhand dieser Stockwerkspläne Zielpunkte registriert werden (siehe Punkttypen Seite 68). Zwischen diesen Punkten kann man ein Wegenetz einzeichnen. Um die Punkte mit den Weglinien exakt zu verbinden, wurde mittels PHP und SQL ein eigener „Punkt-Fang“ realisiert. Es ist auch möglich, die Punkte und die Weglinien nachträglich zu editieren oder zu entfernen. Gebäude und Stockwerke lassen sich auch als Ganzes löschen.

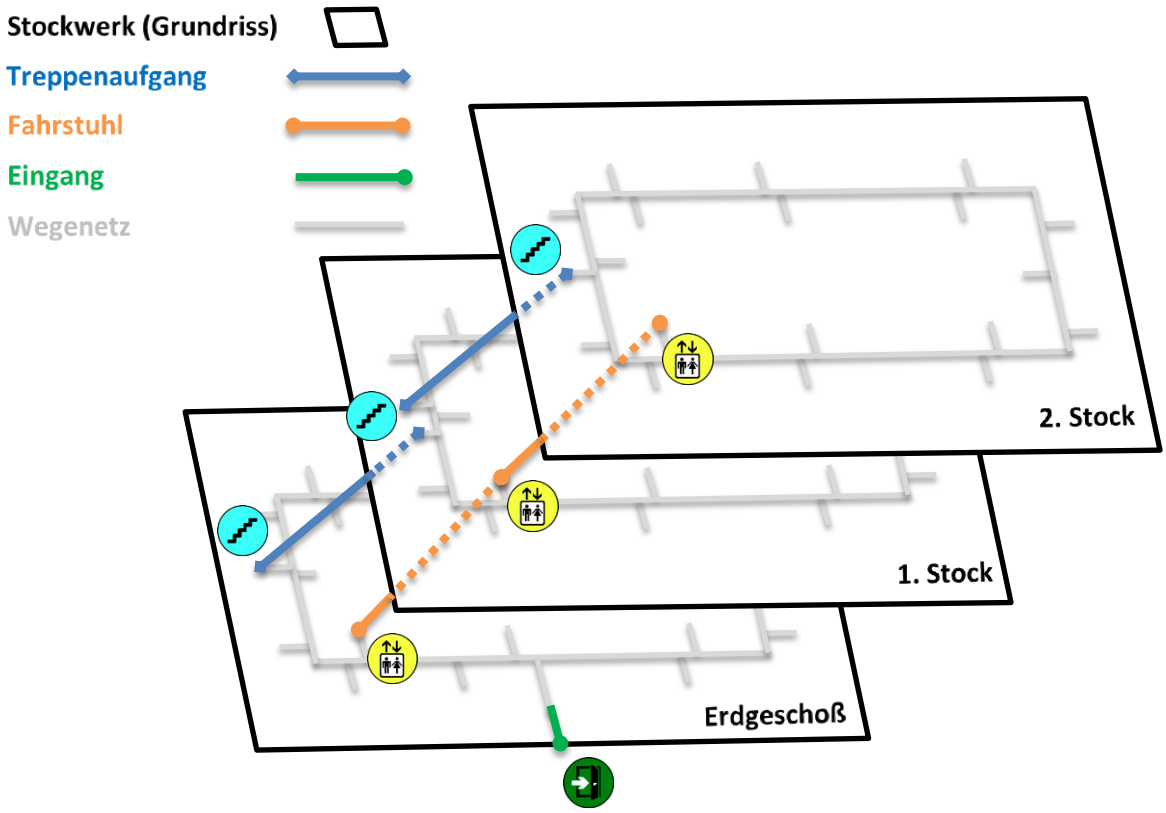

Abbildung 48 – Verknüpfung der Stockwerke (Eigene Darstellung)

Die Verbindung zwischen den Stockwerken ist über die Punkttypen „Fahrstühle“ und „Stiegen“ realisierbar (siehe Abbildung 48). Die Kanten zwischen den Verknüpfungspunkten bekommen je nach Typ entsprechende Kosten zugewiesen. Es ist auch möglich, benachbarte Gebäude über einen bestimmten Eingangstyp miteinander zu verbinden.

Nachdem alle Punkte und Stockwerke eines Gebäudes miteinander verknüpft sind, kann das routingfähige Wegenetz erstellen werden. Die daraus generierte gekettete Adjazenzliste besteht aus zwei Tabellen (siehe 4.6.2).

Zur leichteren Verwaltung der registrierten Punkte wird neben der Kartenoberfläche auch eine Tabelle angeboten. Die anzuzeigenden Attribute sind teilweise auswählbar und eine Sortierung ist ebenfalls möglich. Außerdem ist eine Filterung nach Attributen über ein Dropdown-Menü oder durch eine Texteingabe möglich. Die Tabellenoberfläche bietet auch die Möglichkeit, Punkte zu editieren oder zu löschen. Durch das Auswählen einer Zeile in der Tabelle wird in das Kartenfenster gewechselt und der entsprechende Punkt im Gebäudeplan rot hervorgehoben.

Ein weiteres Fenster stellt die Outdoor-Karte dar, die die OSM als Basiskarte nutzt. Mit dieser ist es möglich, die Indoor-Gebäudepläne mit der Outdoor-Karte zu verknüpfen. Dazu müssen die registrierten Eingänge des Indoor-Bereiches ebenfalls in der OSM registriert werden. Um die Identifizierung zu erleichtern, wird das ausgewählte Gebäude in der Karte rot hervorgehoben. Nach dem Platzieren eines Punktes in der OSM kann einer der Eingänge des Gebäudes ausgewählt werden. Zur eindeutigen Identifizierung ist es notwendig, im Punktattribut „notation“ des Eingangs Angaben darüber zu machen, nach welcher Himmelsrichtung der Eingang ausgerichtet ist und/oder ob es sich um einen Haupt- oder Nebeneingang handelt. Um eine Verknüpfung zwischen dem Gebäude und

dem Outdoor-Wegenetz zu ermöglichen, ist die Identifizierung von mindestens einem Eingang notwendig. Ab der Identifizierung von zwei Eingängen können die Kosten anhand eines geschätzten Maßstabs in Meter umgerechnet werden.

Das WebGIS bietet eine weitere Funktion, mit der man Hindernisse für Menschen mit körperlicher Beeinträchtigung in die OSM-Karte eintragen kann (z.B. Stiegen). Da anhand des ursprünglichen Datensatzes des Outdoor-Wegenetzes eine Unterscheidung nicht möglich ist, muss das Material nachträglich geändert werden. Dazu sind in der Tabelle zwei Attribute vorgesehen (siehe 5.4.2). In der OSM sind die Hindernisse als rot-weiß-strichlierte Wege eingezeichnet. Durch das Einzeichnen eines Hindernisses in die Karte werden die geschnittenen Pfade selektiert und rot visualisiert. Nach dem Speichern werden die entsprechenden Attribute der selektierten Kanten geändert. Ein Rückgängigmachen ist auf die gleiche Art und Weise ebenfalls möglich.

Um die Routingfunktion des Outdoor-Wegenetzes zu testen, können in der Karte zwei Punkte eingezeichnet werden. Anschließend wird der kürzeste Weg zwischen den Punkten berechnet und visualisiert. Dabei ist es auch möglich, eine körperliche Beeinträchtigung zu berücksichtigen.

Die für das Routing benötigten QR-Codes können im WebGIS generiert werden. Dazu wird für die Punkttypen „Räume“, „Eingänge“ und „Sonstige“ ein QR-Code erstellt und in einem entsprechenden Gebäudeverzeichnis am Webserver als *.jpg-Grafik gespeichert.

In früheren Versionen des WebGIS konnte auch die Routingfunktion des Indoor-Wegenetzes überprüft werden. Nach der Erstellung der WebApp wurde diese Möglichkeit jedoch überflüssig und die entsprechenden Codes wurden nicht mehr aktualisiert.

Zusammenfassend bietet das WebGIS folgende Funktionen an:

- Gebäude hinzufügen/entfernen
- Stockwerk hinzufügen/entfernen
- Registrierung von Punkten (insgesamt 7 Punkttypen)
- Wegenetz einzeichnen (inklusive Punktfang)
- Editieren/Löschen der Punkte und des Wegenetzes
- Herstellen einer Verbindung zwischen den Stockwerken
- Herstellen einer Verbindung zwischen benachbarten Gebäuden
- Generierung eines routingfähigen Wegenetzes
- Verwaltung der Punkte mittels Tabelle
- Verknüpfung zwischen Indoor- und Outdoor-Karte durch Identifizierung der Eingänge in der OpenStreetMap
- Identifizierung von Hindernissen im Outdoor-Wegenetz mithilfe der OSM, da diese für körperlich beeinträchtige Personen wichtig ist
- Testen des Outdoor-Wegenetzes
- Generierung der QR-Codes
- Überprüfung der Routingfunktion des Indoor-Wegenetzes (wurde nicht mehr aktualisiert)
- Zusätzliche Symbole für Schaltflächen, um diese besser zu veranschaulichen

5.6.2 Beschreibung der Funktionen des WebGIS anhand von Screenshots

Das WebGIS wird durch die Eingabe folgender Adresse aufgerufen:

http://fphotpc68.tu-graz.ac.at/student/master/ederm/webgis/main.html

Die Elemente der WebGIS-Oberfläche (siehe Abbildung 49) passen sich an die Fenstergröße des Browsers an. Die Fenstergröße des Browsers wurde entsprechend abgestimmt, um möglichst viel Information über die Screenshots vermitteln zu können.

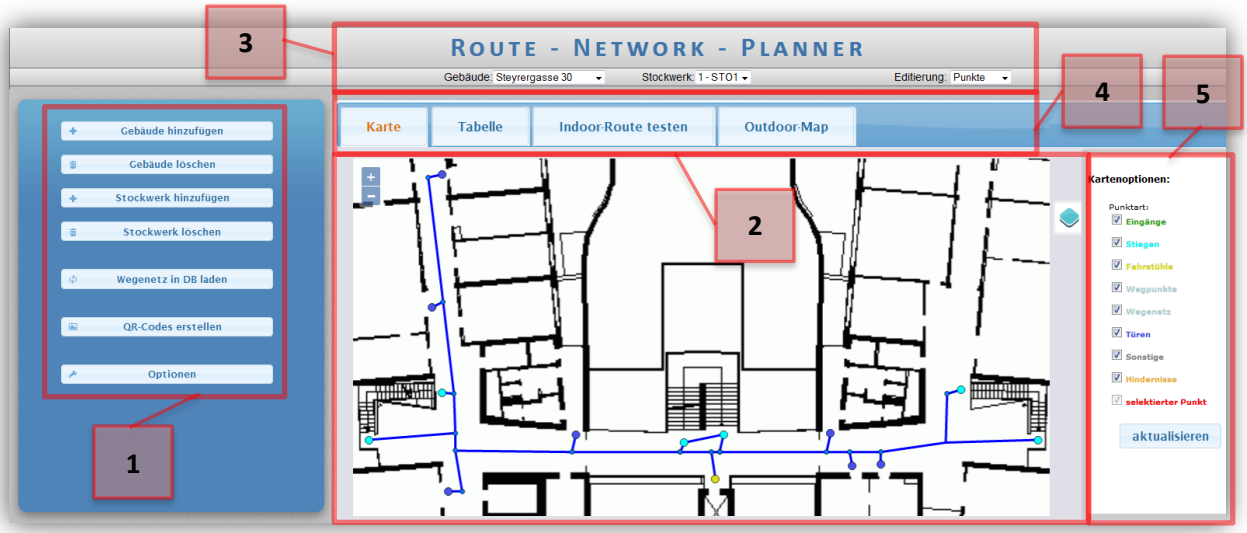

Abbildung 49 – WebGIS, Überblick (Eigene Darstellung)

Abbildung 49 zeigt die komplette Desktopoberfläche der Plattform. Diese ist in fünf Teile unterteilt:

- Hauptmenü (1):
- Hauptfenster (2):
- Top-Menü (3):
- Tab-Auswahl (4):
- Optionen (5):

Das Hauptmenü (1) besteht aus Schaltflächen zum Anlegen und Löschen der Gebäude sowie zum Speichern des Wegenetzes und Generieren der QR-Codes (siehe Abbildung 50). Außerdem beinhaltet es eine Schaltfläche, um das Optionen-Popup (5) zu aktivieren.

Abbildung 50 – WebGIS, Hauptmenü (Eigene Darstellung)

Abbildung 51 – WebGIS, Top-Menü (Eigene Darstellung)

Das Top-Menü (3) beinhaltet den Titel der Seite und eine Reihe von Dropdowns zur Bedienung des Hauptfensters (2) (siehe Abbildung 51).

Abbildung 52 – WebGIS, Tab-Auswahl (Eigene Darstellung)

Mit diesen Reitern (Tabs) der Tab-Auswahl (4) in Abbildung 52 kann man im Hauptfenster (2) zwischen den vier unterschiedlichen Fenstern wechseln:

- der Indoor-Karte (siehe Abbildung 53)
- der Tabelle zur Punktverwaltung (siehe Abbildung 54)
- einem Fenster zum Testen der Indoor-Route
- der Outdoor-Karte (siehe Abbildung 56 bis Abbildung 59)

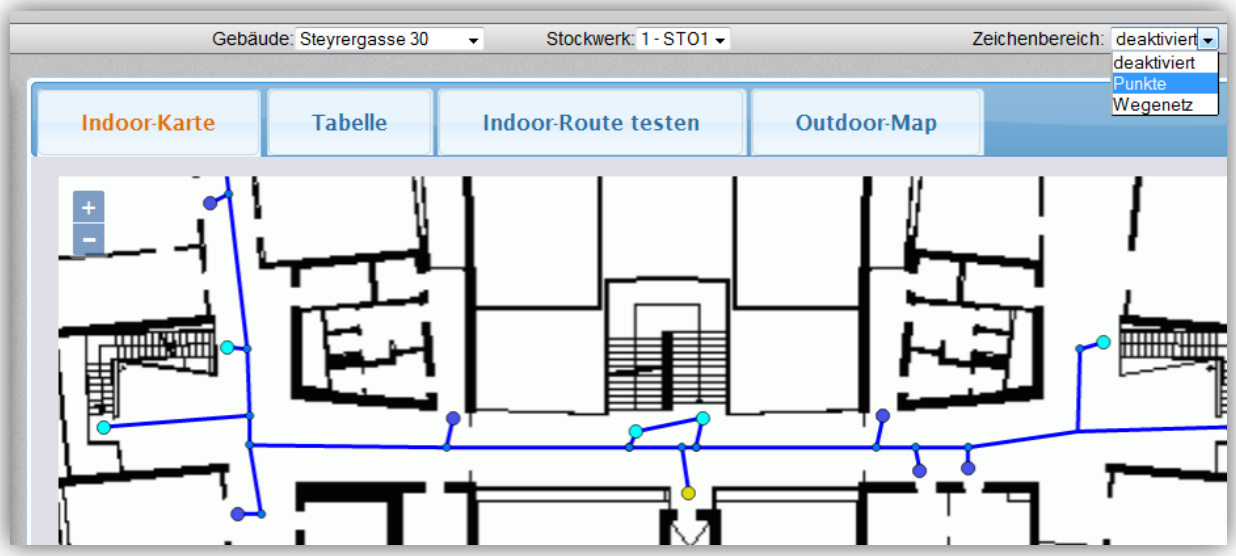

Abbildung 53 – WebGIS, Indoor-Karte (Eigene Darstellung)

Im Indoor-Kartenfenster (Abbildung 53) werden die registrierten Punkte und Wegenetze angelegt und visualisiert. Dafür bietet dieses Fenster folgende Funktionen an:

- Hinzufügen und Editieren der Punkte
- Zeichnen, Verschieben und Löschen von Verbindungslinien (Wegenetze)

Das Wechseln zwischen den Modi erfolgt über eine Dropdown-Auswahl, die sich rechts im Top-Menü befindet und mit „Zeichenbereich“ beschriftet ist.

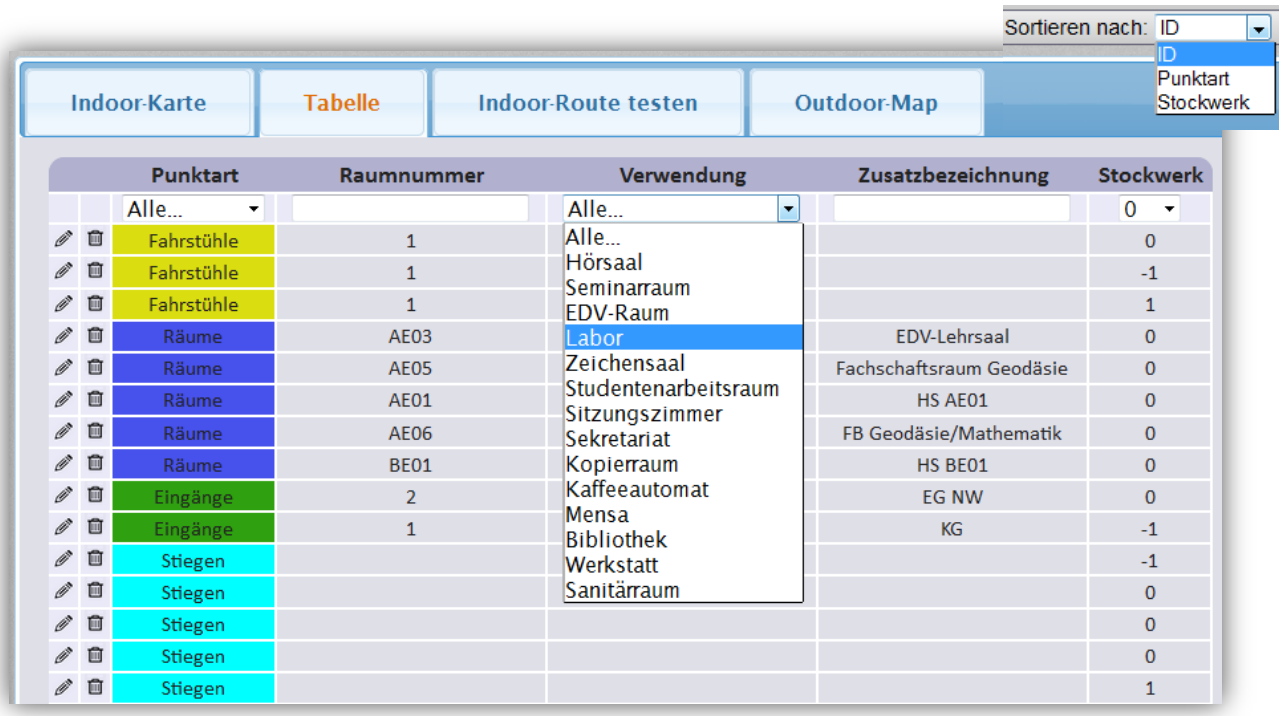

Abbildung 54 – WebGIS, Tabelle (Eigene Darstellung)

Die Tabelle zur möglichst einfachen Punktverwaltung ist in Abbildung 54 dargestellt und bietet folgende Möglichkeiten:

- Auswahl der anzuzeigenden Attribute (Spalten)
- Sortierung nach ID, Punktart oder Stockwerk
- Filterung nach bestimmten Attributen über Dropdown-Menüs oder Texteingabefelder (z.B. nach EDV-Räumen oder Räumen, die mit AE anfangen)
- Editieren/Löschen jedes Eintrages mit den Buttons auf der linken Seite
- Wechseln zu einem Punkt in der Karte („zoom-to“-Funktion) durch Klicken in eine Zeile

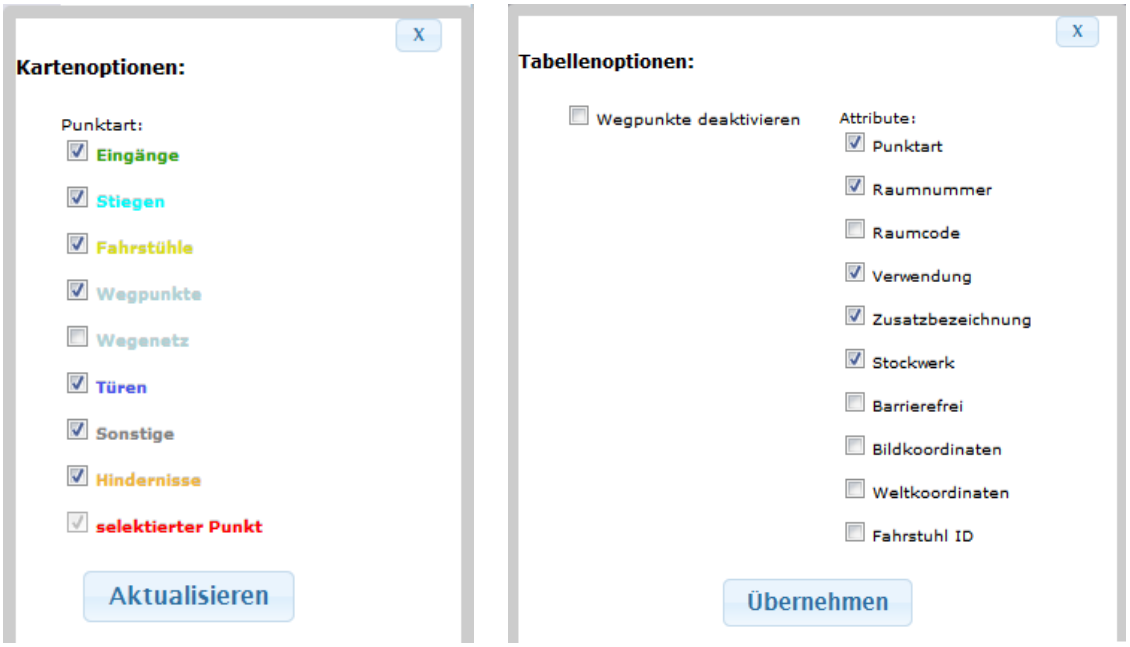

Abbildung 55 – WebGIS, Optionen (Eigene Darstellung)

Um bestimmte Darstellungsarten zu ändern, wird das Optionen-Popup aufgerufen, welches für Karten und Tabellen verfügbar ist (siehe Abbildung 55).

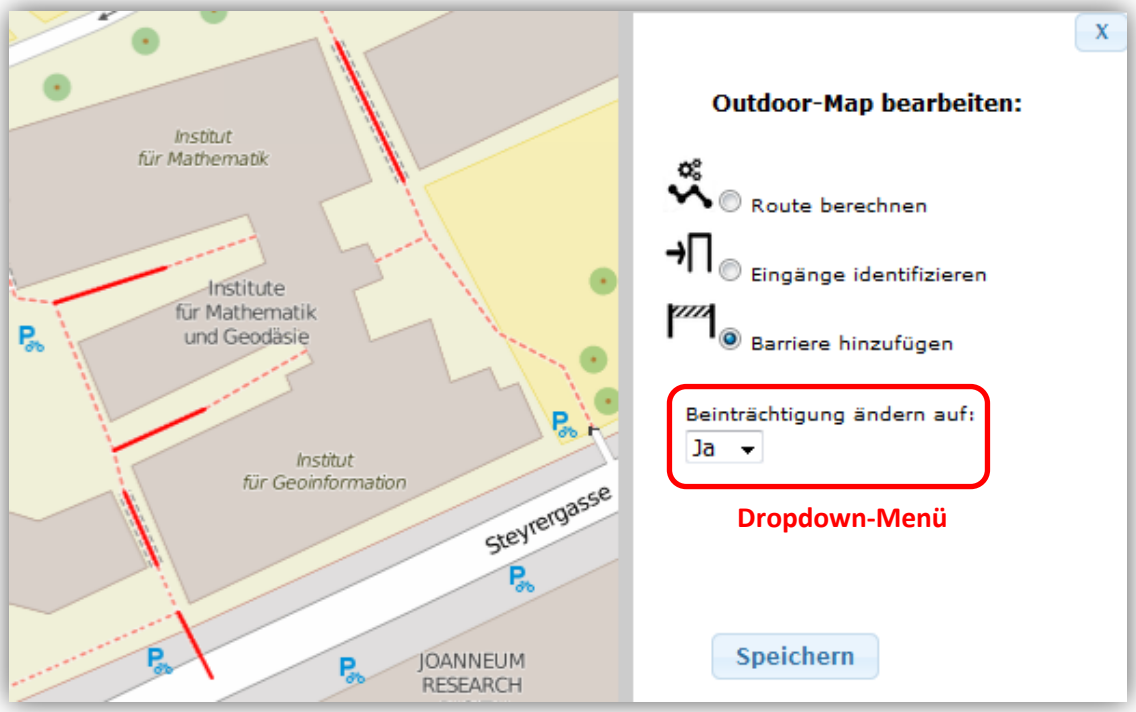

Abbildung 56 - WebGIS, Barriere hinzufügen (Eigene Darstellung)

Abbildung 56 zeigt eine der drei Anwendungen, die mit der Outdoor-Karte durchgeführt werden können. Es besteht die Möglichkeit, Barrieren (Hindernisse) einzutragen, die Menschen mit körperlicher Beeinträchtigung nicht bewältigten können (z.B. Rollstuhlfahrer). Dazu zeichnet man zwei

Punkte in der OSM-Karte ein. Die dadurch erzeugte Verbindungslinie stellt die Barriere dar. Alle Pfade, die diese Linie schneiden, werden anschließend selektiert und in der Karte visualisiert. Nach dem Speichern wird der entsprechende Eintrag in der Datenbank geändert. Alle bereits eingetragenen Barrieren werden ebenfalls in der Karte angezeigt. Möchte man eine Barriere wieder entfernen, kann diese wieder selektiert werden. In diesem Fall muss vor dem Speichern das Dropdown-Menü auf Nein geändert werden. Aus dem Vektor-Datensatz der OSM konnte bereits der Großteil der Stiegen als Hindernisse identifiziert werden.

Abbildung 57 – WebGIS, Eingänge identifizieren (Eigene Darstellung)

Abbildung 57 zeigt die Möglichkeit, wie ein Eingang identifiziert wird. In diesem Beispiel kann der Anwender des WebGIS die Karte so interpretieren, dass der Schnittpunkt des strichlierten Fußweges und der Kante des ausgewählten Gebäudes demnach der Eingang sein muss. Daher wird dort ein grüner Punkt platziert. Da die Karte genordet ist, handelt es sich im Beispiel um den Südeingang (EG Sued). Durch das Bestätigen (Speichern) werden die Koordinaten des platzierten Punktes für den ausgewählten Eingang in der Datenbank gespeichert.

Abbildung 58 – WebGIS, Route berechnen/nicht beeinträchtigt (Eigene Darstellung)

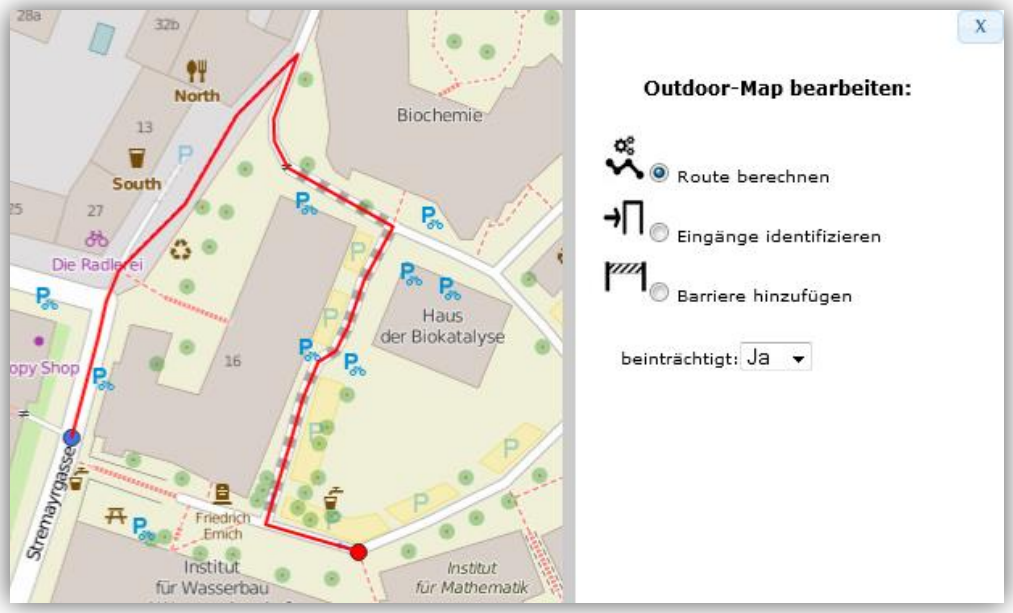

Abbildung 59 – WebGIS, Route berechnen/beeinträchtigt (Eigene Darstellung)

Die dritte Anwendung der Outdoor-Karte dient zum Überprüfen der Routenberechnung im Outdoor-Bereich. Dazu sind zwei Punkte in der Karte einzuzeichnen. Beim ersten Klicken in die Karte wird ein roter Punkt (Start) platziert und beim zweiten Mal ein blauer Punkt (Ziel). Da das Routen nur entlang des Outdoor-Wegenetzes möglich ist, wird zu jedem Punkt der nächstliegende Knoten gesucht. Zwischen diesen Knoten wird anschließend über den pgrouting-Befehl der kürzeste Weg berechnet und in der Karte dargestellt. Mit Hilfe eines Dropdown-Menüs kann man in der Berechnung auch eine Beeinträchtigung berücksichtigen. Die beiden Varianten sind in Abbildung 58 und Abbildung 59 dargestellt. In Abbildung 58 ist zu erkennen, dass der kürzere Weg auf Grund eines Hindernisses (Stiege) vermieden wird. Stattdessen wird ein längerer Weg empfohlen.

5.6.3 Vorteile der semi-automatisierten Wegenetzerstellung

Die finale Version der Wegenetzerstellung im WebGIS basiert auf einer semi-automatisierten Variante. Die Vorteile sind:

- Integration der Beurteilung des Anwenders
- der Erzeuger des Wegenetzes ist besser in der Lage bestimmte Strukturen zu interpretieren
- Unabhängigkeit von einem Kartenstandard (z.B. Tür als ¼ Kreis darstellen oder wie in der Arbeit offen)
- Unabhängigkeit von zusätzlicher Software bzw. zusätzlichen Algorithmen
- weniger Schnittstellen

Als Nachteil kann genannt werden, dass ein höherer Zeitaufwand bei der Erstellung erforderlich ist. Da aber die Realisierung einer automatisierten Methode aufgrund der vielfältigen Karten und dessen Interpretation sehr komplex ist, nimmt auch diese Methode viel Zeit in Anspruch.

5.7 WebApp

Die im Rahmen der Diplomarbeit entworfene WebApp ist eine für mobile Geräte optimierte Plattform zur Suche von Räumen an der TU Graz. Als Hilfestellung wird dem Anwender anhand einer Karte der Weg vom Standort zum Zielpunkt angezeigt.

5.7.1 Allgemeine Beschreibung

Die WebApp ist auf einem WebServer der TU Graz gespeichert und kann mit Hilfe mobiler Webbrowser auf Smartphones und Tablets aufgerufen werden. Es handelt sich dabei um einen standortbezogenen Dienst (LBS, siehe 4.4). Die Routenfindung wird dem Benutzer durch die Einblendung der schnellsten Route in einer Karte ermöglicht. Dazu werden sowohl Gebäudepläne (Indoor) als auch Straßenkarten (Outdoor) eingesetzt. Um eine Suche zu starten, sind vom Anwender zuerst Start- und Zielpunkt anzugeben. In Tabelle 20 wird gezeigt, welche Eingabemöglichkeiten dem Nutzer angeboten werden. Das ist auch aus der Grafik in Abbildung 36 zu entnehmen.

Tabelle 20 – Ortungsmethoden

Methode	Startpunkt	Zielpunkt
Einscannen eines QR-Codes	ja	ja
Manuelle Auswahl: Eingabe des Raumes über Dropdown-Menüs und Eingabefelder	ja	ja
Ortung mittels Geolocation-API (Methoden: IP-Adresse, Zellortung, GPS oder WLAN)	ja	nein
Ortung mittels Geocoding-API (Eingabe der Adresse)	ja	nein
Angabe eines „Point of Interest“	nein	ja

Neben der Routenberechnung ist es auch möglich, eine reine Standortbestimmung durchzuführen. Dazu sind dieselben Ortungsmethoden möglich wie bei der Angabe des Startpunktes.

Während der Berechnung erscheint eine Fortschrittsanzeige (engl. progress bar), die dem Anwender mitteilt, dass die Route gerade berechnet wird. Nach dem Laden der Karte und der Layers verschwindet die Anzeige wieder. Die Karte, die nach der Berechnung als erstes angezeigt wird, ist der Gebäudeplan. Welches der Gebäude bzw. welches Stockwerk angezeigt wird, hängt vom Start- bzw. Zielpunkt ab. Meist ist es das Stockwerk, in dem sich der Gebäudeeingang befindet.

Um leicht zwischen Start und Ziel zu wechseln, sind in der App eigene Schaltflächen platziert. Die OSM (Outdoor-Karte) ist ebenfalls leicht über einen Reiter (Tab) erreichbar. Um im Gebäude zwischen den Stockwerken zu wechseln, sind ebenfalls Schaltflächen platziert. Außerdem ist es auch möglich, durch eine Interaktion mit der Karte das Stockwerk zu wechseln. Dazu ist auf ein Stiegen- oder Fahrstuhl-Feature zu klicken. Aus den Start- und Zielpunkt-Features erhält man Rauminformationen, welche in einem Popup angezeigt werden. Durch Anklicken eines Eingang-Features wird in den Outdoor-Bereich gewechselt. Dabei wird darauf geachtet, dass der Kartenausschnitt auf das aktuelle Gebäude zentriert ist. Falls auch ein Weg im Outdoor-Bereich berechnet wurde, wird dieser ebenfalls visualisiert. Jene Gebäude, die von der Berechnung des kürzesten Weges betroffen sind, werden in der OSM-Karte grafisch hervorgehoben. Durch Anklicken eines Gebäudes gelangt man in dessen Indoor-Bereich. Falls sich die Position des Anwenders verändert hat, kann man diese auf Wunsch aktualisieren, ohne den Zielpunkt erneut anzugeben.

Die App bietet auch die Funktion an, bestimmte Parameter für die Wegberechnung zu ändern. Es ist möglich, die Anzahl der Stockwerke anzugeben, ab der Fahrstühle bevorzugt werden sollen bzw. ob auf Fahrstühle überhaupt verzichtet werden soll. Außerdem kann man angeben, ob man zu Fuß oder mit dem Fahrrad unterwegs ist. Eine körperliche Beeinträchtigung kann ebenfalls berücksichtigt werden. Diese Einstellungen bleiben dem Anwender erhalten, da diese im Browser als Cookie gespeichert werden. Beim erneuten Aufruf der WebApp werden alle nutzerspezifischen Informationen wieder geladen.

Die berechneten Kosten des Wegenetzes im Outdoor-Bereich bzw. im Gebäude werden in Minuten angegeben. Für die Umrechnung werden geschätzte Größen für die Geschwindigkeit verwendet. Im Outdoor-Bereich wird neben der Zeit, die ein Fußgänger benötigt, auch die Zeit für einen Radfahrer angegeben. Außerdem wird bei der Verwendung der Geolocation-API auch die Genauigkeit der Positionsbestimmung angegeben, um dem Benutzer anzuzeigen, wie sehr man der Ortung vertrauen kann.

Bei der Gestaltung der App wurde darauf Wert gelegt, die Oberfläche nicht zu überladen und dem Anwender genug Möglichkeiten zu geben, schnell zwischen den Karten zu wechseln. Außerdem wurde versucht, mit einfachen Symbolen die Orientierung des Benutzers zu erleichtern und dem Design der WebApp einen eigenen Charakter zu verleihen.

Zusammenfassend bietet die WebApp folgende Funktionen an:

- Raumsuche im Indoor- und Outdoor-Bereich
- manuelle Angabe der Position (Dropdown, Eingabefeld)
- Unterstützung von mehreren Ortungsmethoden
- Suche nach „Points of Interests“
- schnelle Berechnung des kürzesten Weges
- Möglichkeit, nur den Standort zu bestimmen
- Visualisierung der Route im Indoor- und Outdoor-Bereich
- Visualisierung der betroffenen Gebäude im Outdoor-Bereich

- Unterstützung mehrerer Fälle von Routingkombinationen (innerhalb eines Gebäudes, zwischen zwei Gebäuden, zwischen Outdoor-Punkt und Gebäude)
- leichtes Wechseln zwischen Stockwerken, Gebäuden und Outdoor/Indoor-Bereich
- Angabe von Informationen über Start- und Zielpunkt mittels Popup
- Möglichkeit, die momentane Position zu aktualisieren
- Einstellbarkeit der Anzahl der Stockwerke, ab der Fahrstühle bevorzugt werden
- Verwendung von Fahrstühlen
- Auswahl der Fortbewegungsart
- Möglichkeiten der Berücksichtigung einer körperlichen Beeinträchtigung bei der Wegberechnung
- Speichern der nutzerspezifischen Informationen
- Angabe der geschätzten Zeit für Fußgänger und Radfahrer
- Angabe der Genauigkeit bei der Positionsbestimmung (nur bei Geolocation)
- übersichtliche Nutzeroberfläche mit beschreibenden Symbolen

5.7.2 Beschreibung der Funktionen der WebApp anhand von Screenshots

Die WebApp wird durch den Aufruf folgender Adresse geladen:

`http://fphotpc68.tu-graz.ac.at/student/master/ederm/webapp/main_mobile.html`

Die dargestellten Screenshots stammen jeweils vom selben Smartphone (Kyocera Torque KC-S701, Android), wobei als Browser Chrome verwendet wurde.

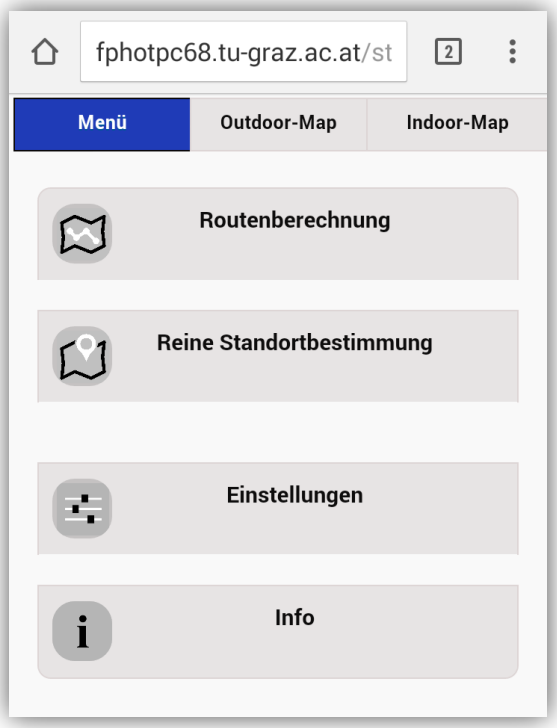

Abbildung 60 – Hauptmenü (Eigene Darstellung)

Abbildung 61 – Auswahl_1 (Eigene Darstellung)

In Abbildung 60 wird das Hauptmenü abgebildet. In diesem kann der Benutzer wählen, ob er eine Route berechnen oder nur den Standort wissen möchte. Außerdem ist es möglich, bestimmte Einstellungen zu ändern und bestimmte Informationen über die WebApp zu bekommen.

Wenn man sich für die Routenberechnung entscheidet, wird ein Popup geöffnet, in dem man aufgefordert wird, Start- und Zielpunkt auszuwählen. Durch ein Symbol wird mitgeteilt, ob jener Punkt bereits ausgewählt wurde oder nicht. In Abbildung 61 wird ein rotes Fragezeichen angezeigt, als Hinweis darauf, dass Start- und Zielpunkt noch unbekannt sind.

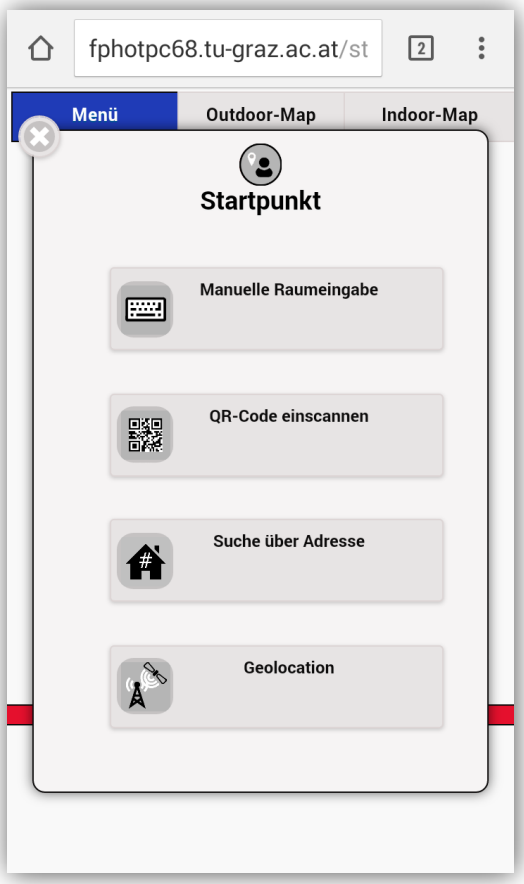

Abbildung 62 – Auswahl der Ortungsmethode (Eigene Darstellung)

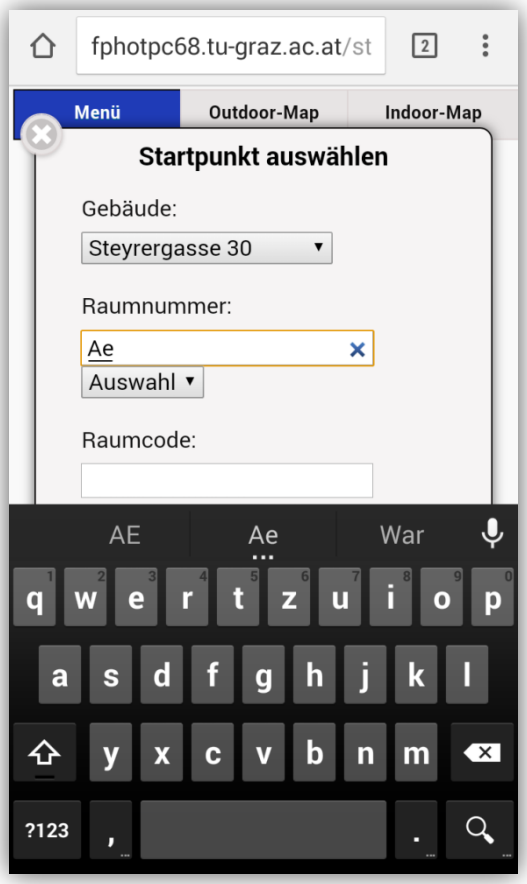

Abbildung 63 – Manuelle Raumangabe (Eigene Darstellung)

Nach der Wahl des Startpunktes wird man aufgefordert, die Ortungsmethode auszuwählen (siehe Abbildung 62). Bei der manuellen Raumangabe ist zunächst das Startgebäude über ein Dropdown-Menü zu selektieren (siehe Abbildung 63). Der Benutzer hat die Möglichkeit, nach einem der drei Attribute (Raumnummer, Raumcode oder Raumbezeichnung) über eine Texteingabe zu suchen. Dabei kann auch nach Anfangsbuchstaben gesucht werden, wobei die Groß/Kleinschreibung keine Rolle spielt. Nach der Texteingabe taucht unter dem entsprechenden Textfeld ein weiteres Dropdown-Menü auf, in dem man alle möglichen Treffer angezeigt bekommt (siehe Abbildung 64). Nach der Auswahl der gesuchten Raumnummer und dem Bestätigen durch „Suchen“ gelangt man wieder zum 1. Popup-Auswahlfenster zurück. Da der Startpunkt bereits bekannt ist, wird jetzt neben dieser Auswahl ein grünes Häkchen angezeigt (siehe Abbildung 65).

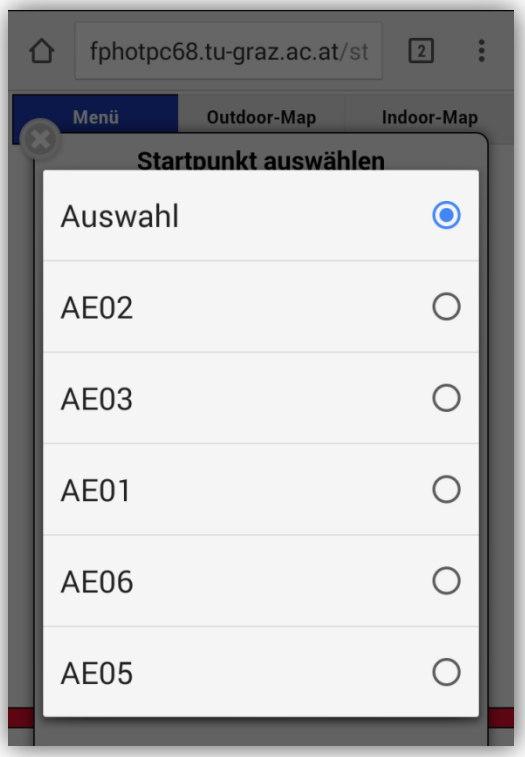

Abbildung 64 – Manuelle Raumangabe, Dropdown (Eigene Darstellung)

Abbildung 65 – Auswahl_2 (Eigene Darstellung)

Bei der Methode „QR-Code einscannen“ öffnet sich ein Popup-Fenster, in dem in Echtzeit eine Videoaufnahme des mobilen Gerätes eingeblendet wird (siehe Abbildung 66). Um den Media-Stream zu ermöglichen, wird eine spezielle API benötigt (getUserMedia/Stream). Die API wird zurzeit nur von wenigen Webbrowsern unterstützt (caniuse.com - getUserMedia/Stream API 2015). Nachdem die Kamera des Gerätes auf den QR-Code gerichtet wird, versucht die API den Code scharf zu stellen, um ihn korrekt zu interpretieren. Anschließend wird das Ergebnis mit dem WebCodeCam-jQuery-Plug-In ausgewertet und das Popup wieder geschlossen. Die Informationen des Plug-In beziehen sich auf (jplugins.directory - webcodecam 2015) und (atandrastoth.co.uk - codereader 2015).

Abbildung 66 – QR-Code einscannen (Eigene Darstellung)

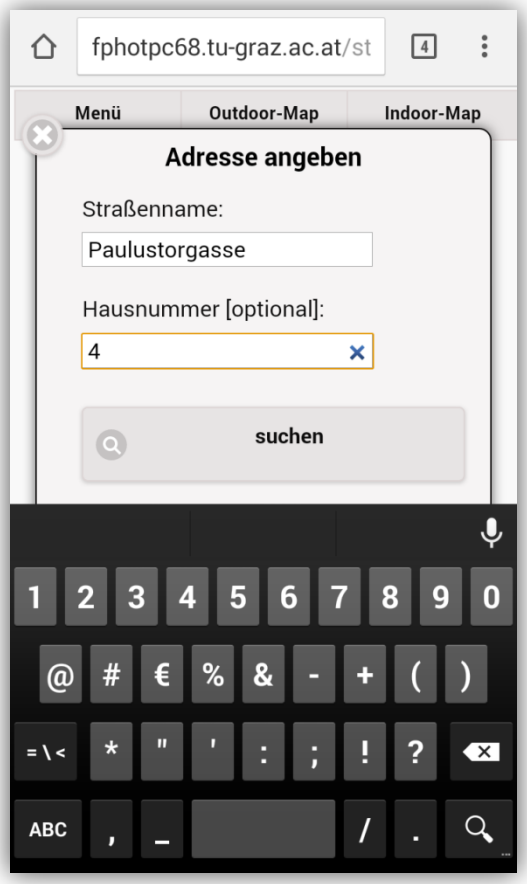

Abbildung 67 – Adressangabe (Eigene Darstellung)

Abbildung 68 – Angabe des Zielpunktes (Eigene Darstellung)

Eine weitere Methode, um den Standort anzugeben, ist die Angabe der Adresse. Dazu sind der Straßenname und optional die Hausnummer anzugeben (siehe Abbildung 67).

Bei der Bestimmung des Standortes mittels „Geolocation“ (siehe 4.17.4) kommt je nach der Sicherheitseinstellung des mobilen Gerätes eine Aufforderung, dass die Position freizugeben ist.

Für die Angabe des Zielpunktes stehen drei Methoden zu Verfügung (siehe Abbildung 68).

Abbildung 69 – Points of Interest (Eigene Darstellung)

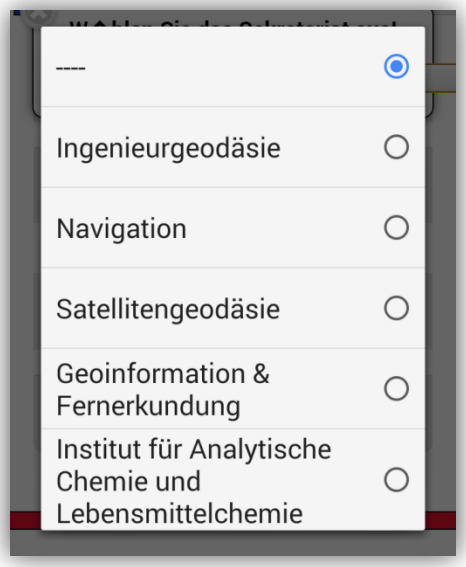

Abbildung 70 – POI, Sekretariat (Eigene Darstellung)

Die manuelle Raumangabe und die Angabe mittels QR-Codes erfolgen gleich wie bei der Angabe des Startpunktes. Zusätzlich ist es auch möglich, nach „Point of Interest“ zu suchen (siehe Abbildung 69). Das sind bestimmte Punkte, die für den potentiellen Nutzer zweckmäßig sind. Nach der Auswahl eines POI wird jener Punkt dieser Kategorie ausgewählt, der sich am nächsten zum Startpunkt befindet. Bei der Auswahl der Kategorie „Sekretariat“ erscheint ein weiteres Dropdown-Menü, in dem man das gesuchte Institut bzw. die Arbeitsgruppe auswählen kann (siehe Abbildung 70).

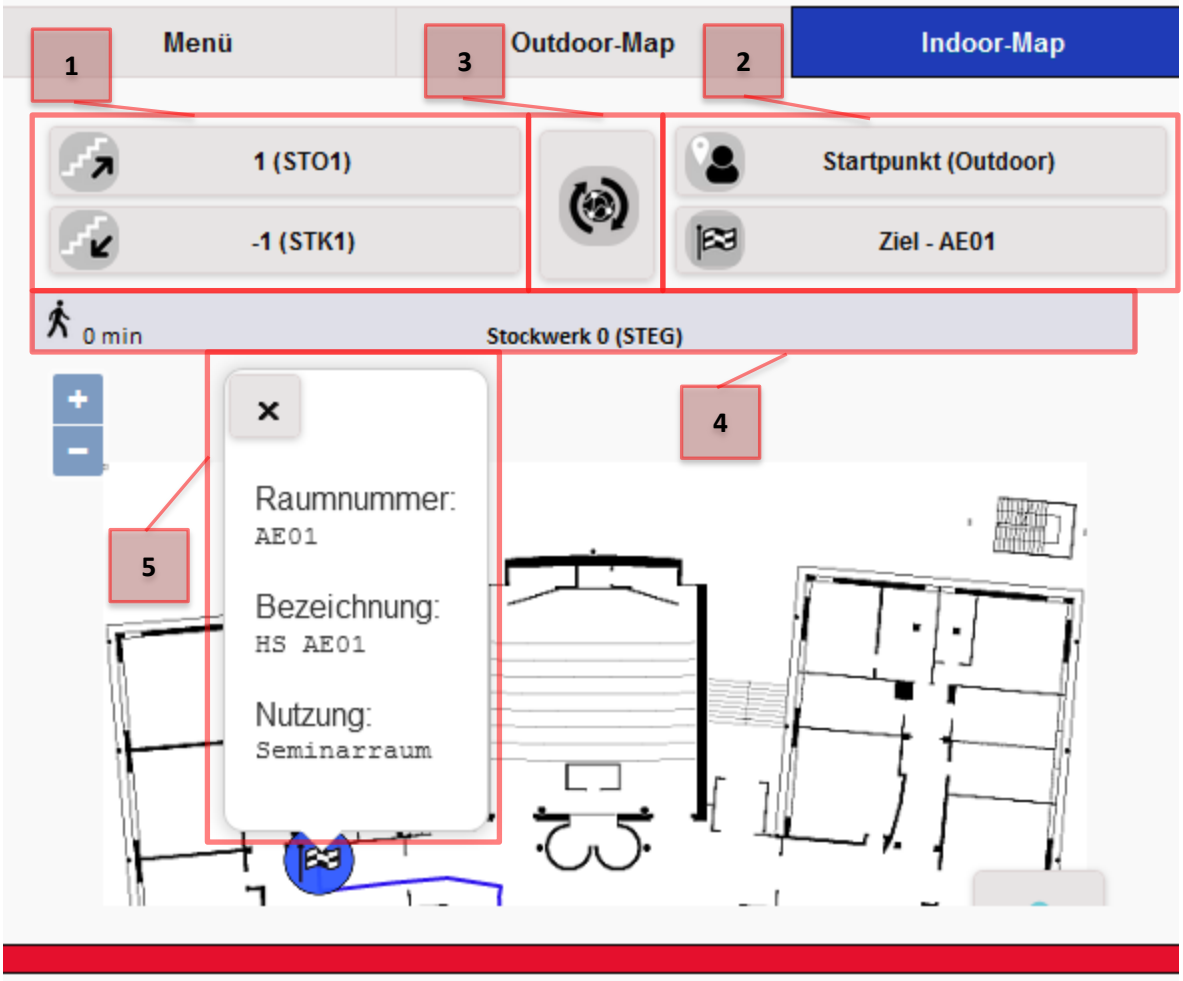

Abbildung 71 – Indoor-Oberfläche, Screenshot vom Tablet (Eigene Darstellung)

Nach der Auswahl von Start- und Zielpunkt wird in das Kartenfenster (Indoor-Map) gewechselt. Dort sind fünf Knöpfe platziert, die den Zweck haben, zwischen den Gebäuden und Stockwerken zu wechseln (siehe Abbildung 71). Die linken Schaltflächen (1) ermöglichen es, zum jeweiligen Nachbarstockwerk zu wechseln. Die rechten Bedienfelder (2) wechseln zwischen Start- und Zielpunkt. Die Schaltfläche in der Mitte (3) öffnet das Popup-Fenster aus Abbildung 62, in dem der Startpunkt aktualisiert werden kann. Der darunter liegende Informationsbalken (4) gibt die benötigte Zeit vom Eingang des Startgebäudes zum Eingang des Zielgebäudes in Minuten an. Außerdem zeigt dieser die Stockwerksnummer und Bezeichnung an. Die eigens kreierten Kartensymbole werden in Tabelle 21 beschrieben.

Durch das Anklicken eines Start- oder Zielpunkt-Features öffnet sich in der Karte ein Popup (5). Es gibt die Rauminformationen des jeweiligen Punktes an.

Tabelle 21 – Kartensymbole

Symbol		Beschreibung
Karte	Menü	
		Startpunkt
		Zielpunkt
		Stiege/Treppenaufgang
		Fahrstuhl
		Eingang
		Position aktualisieren

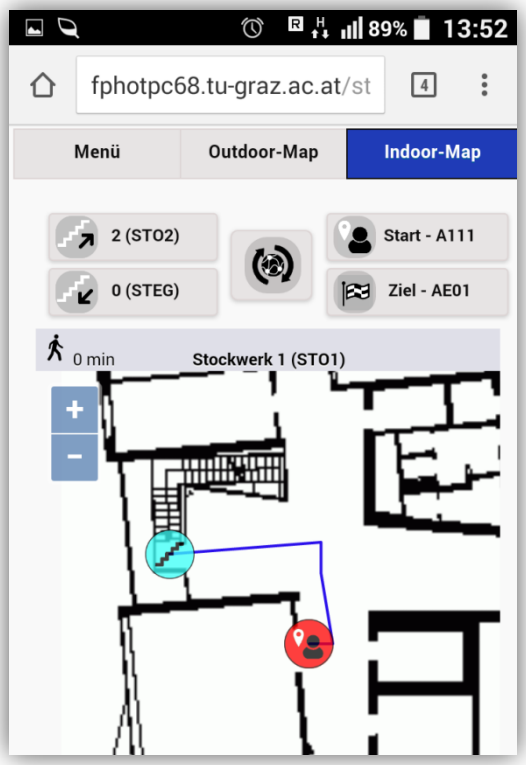

Abbildung 72 – Indoor-Map, Start (Eigene Darstellung)

Abbildung 73 – Indoor-Map, Ziel (Eigene Darstellung)

Abbildung 72 und Abbildung 73 zeigen die Ergebnisse im Indoor-Bereich an, wobei die jeweiligen Stockwerkspläne als Grundkarte dienen. In diesem Fall handelt es sich um ein Routing zwischen zwei Gebäuden. Der berechnete kürzeste Weg (Route) wird dabei als blaue Linie dargestellt.

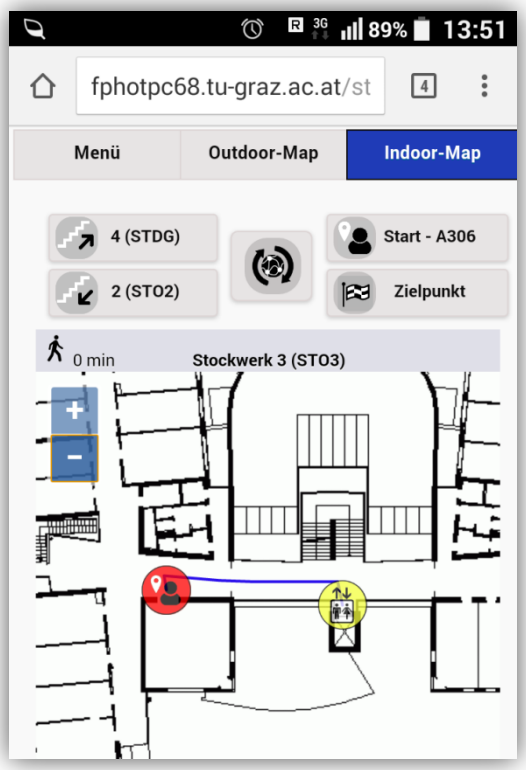

Abbildung 74 – Indoor-Map, Startpunkt/Fahrstuhl (Eigene Darstellung)

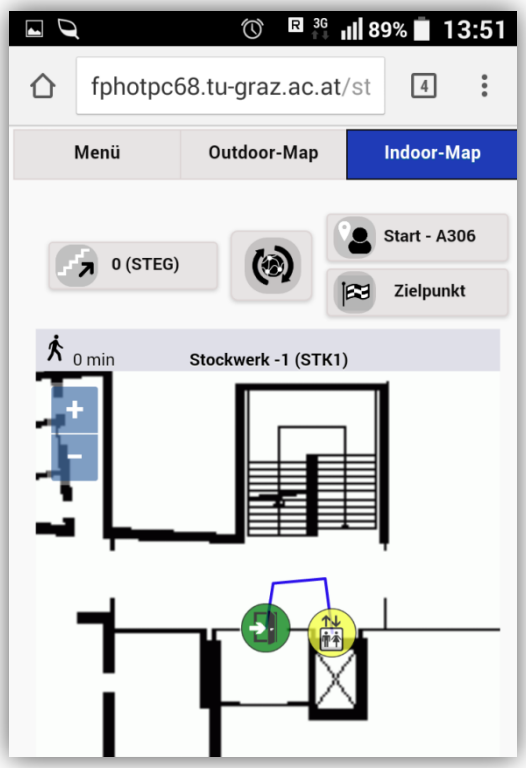

Abbildung 75 – Indoor-Map, Fahrstuhl/Eingang (Eigene Darstellung)

In Abbildung 74 und Abbildung 75 sieht man ein Beispiel mit Fahrstuhlbenutzung vom 3. Stock in das erste Untergeschoss. Von dort wird das Gebäude über einen Gebäudeeingang verlassen.

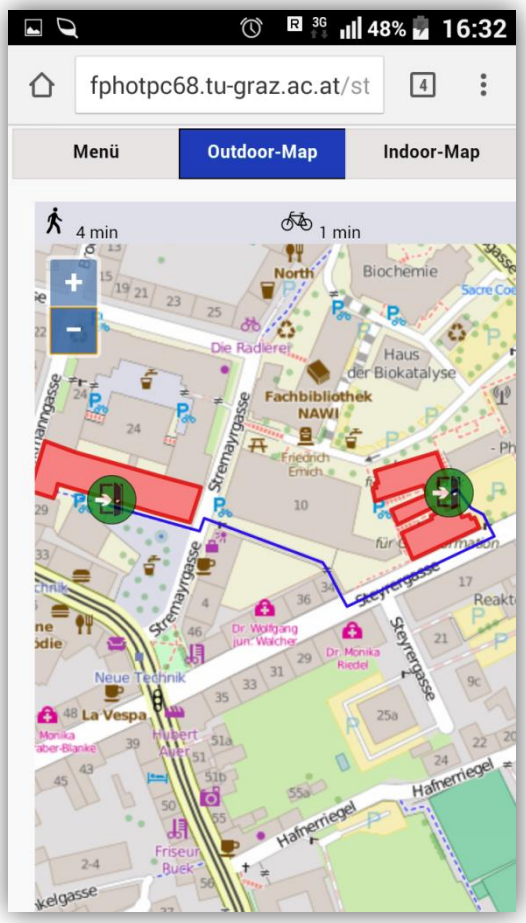

Abbildung 76 – Outdoor-Map (Eigene Darstellung)

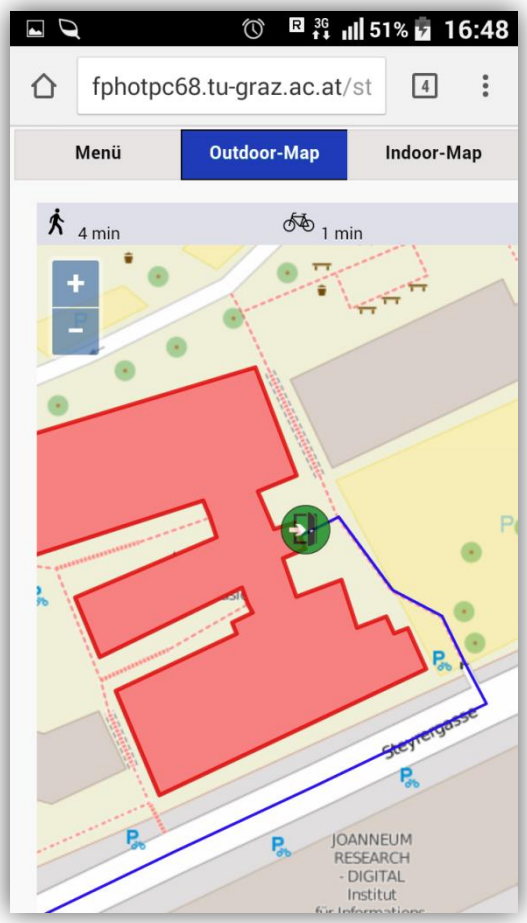

Abbildung 77 – Outdoor-Map, Start (Eigene Darstellung)

Abbildung 76 und Abbildung 77 zeigen das Ergebnis im Outdoor-Bereich an, wobei die OSM-Karte als Grundkarte verwendet wird. Der kürzeste Weg wird ebenfalls blau dargestellt. Die grünen Symbole stellen Start- und Zielpunkt dar. Durch das Heranzoomen des Startpunktes wird in der Karte auch das Startgebäude besser sichtbar (siehe Abbildung 77).

In der Outdoor-Karte wird im Informationsbalken, neben der benötigten Zeit für Fußgänger, auch die Zeit für Fahrradfahrer angegeben.

Abbildung 78 – Einstellungen (Eigene Darstellung)

Über das Menü kann das Popup „Einstellungen“ geöffnet werden (siehe Abbildung 78). Mit zwei Dropdown-Menüs können die Fortbewegungsart (siehe Abbildung 79) und die Verwendung von Fahrstühlen bestimmt werden. Sollte als Fortbewegungsart „Fußgänger“ oder „Radfahrer“ gewählt werden, kann man im zweiten Dropdown einstellen, ob oder ab welcher Anzahl von Stockwerken man Fahrstühle verwenden möchte (siehe Abbildung 80). Sollte man „Beeinträchtigt“ als Fortbewegungsart auswählen, werden im berechneten Weg alle Hindernisse des Outdoor-Bereichs sowie alle Stiegen des Indoor-Bereichs vermieden. Die Wahl des zweiten Dropdown wird dadurch hinfällig. Sollte man „Radfahrer“ ausgewählt haben, werden nur die Hindernisse der Outdoor-Karte vermieden. Im Einstellungs-Popup kann der Anwender die Berechnung mit den geänderten Eingaben erneut durchführen. Um eine neue Suche zu starten, gibt es die Möglichkeit, alle bisherigen Daten zu löschen („Neue Suche starten“). Außerdem ist es möglich, die nutzerspezifischen Einstellungen für eine spätere Verwendung zu speichern („Änderung übernehmen“).

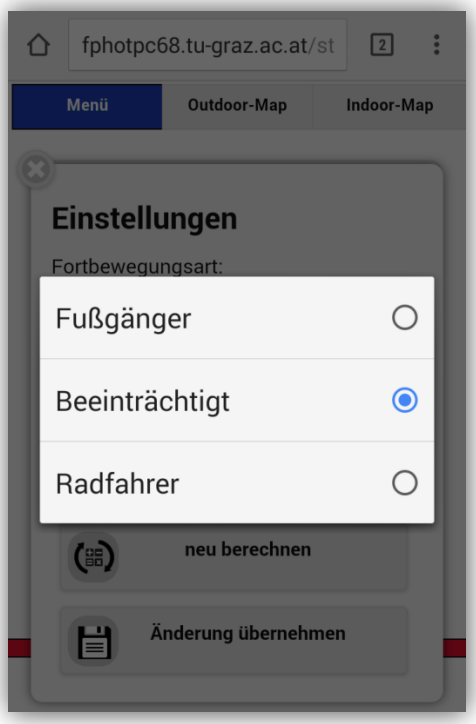

Abbildung 79 – Einstellungen, Fortbewegungsart (Eigene Darstellung)

Abbildung 80 – Einstellungen, Fahrstühle (Eigene Darstellung)

6. Validierung

Die Funktionalitäten der WebApp wurden in der Steyrergasse 30 und der benachbarten Hundewiese überprüft. Dazu wurden an den Gebäudeeingängen und an einigen Räumen QR-Codes platziert. Diese wurden auf selbstklebende Etiketten[3] gedruckt (siehe Abbildung 81).

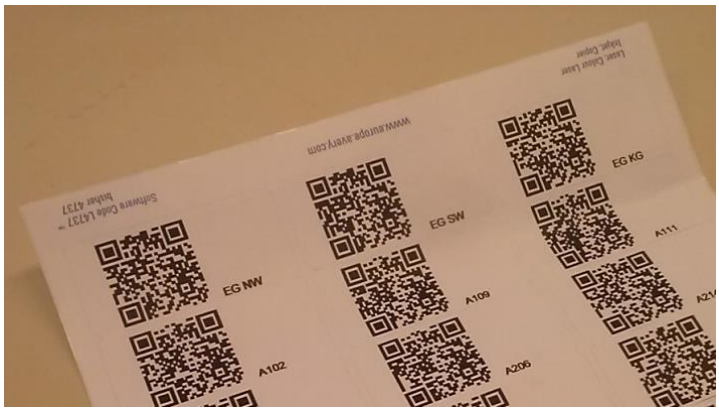

Abbildung 81 – Etiketten mit QR-Codes (Eigene Darstellung)

Einige Tests erfolgten im Freien, um z.B. die Ortung mittels GPS zu prüfen. Insgesamt wurden sechs Smartphones und ein Tablet verwendet. Die Ergebnisse der Tests sind in Tabelle 22 aufgelistet, dessen Grafiksymbole auf (de.freepik.com - Symbole 2015) basieren.

Tabelle 22 – Liste der verwendeten Geräte

Typ	OS			Browser		Bedienung	Darstellung		Σ
LG Optimus L5	Android 4.0	1x800MHz	4.0‘‘	Android	5 MP	fehlerhaft	OK	nicht möglich	✗
Apple iPhone 5S	iOS	2x1.3GHz	4.0‘‘	Safari	8 MP	OK	OK	nicht möglich	✗
Lenovo A7-50 A3500H	Android 4.4	4x1.3 GHz	7.0‘‘	Firefox	5 MP	OK	OK	OK	✓
Samsung Galaxy S2	Android 4.0	2x 1.2 GHz	4.3‘‘	Firefox	8 MP	OK	OK	OK	✓
Samsung Galaxy S4	Android 4.4	4x 1.9GHz	5.0‘‘	Firefox	13 MP	OK	OK	OK	✓
Kyocera Torque KC-S701	Android 4.4	4x1.4 GHz	4.5‘‘	Chrome	8 MP	OK	OK	OK	✓
Nokia Lumia 530	Windows 8.1	4x1.2 GHz	4.0‘‘	Internet Explorer	5 MP	OK	OK	nicht möglich	✗

[3] MS-Word Formatvorlage: L4737REV

Anmerkung:

- Beim Test mit dem LG Optimus L5 reagierte die WebApp träge auf die Betätigung der Schaltflächen.
- Das Einlesen der QR-Codes war mit dem Default-Android Browser, Safari (iOS) sowie dem Internet Explorer nicht möglich, da die notwendige Stream-API nicht unterstützt wird. Dieses Problem kann jedoch durch die Installation des Firefox- oder Chrome-Browsers behoben werden (mobilehtml5.org - Browser-Kompabilität 2015).
- Der Test mit dem Tablet war zufriedenstellend. Dennoch ist Folgendes zu bemängeln: Da die Kartendarstellung der WebApp auf das Smartphone-Hochformat angepasst ist, wird bei einem Tablet die Karte nur in einem kleinen Fenster dargestellt. Die Karte reicht etwa bis zur Hälfte und das Potential des größeren Bildschirms kann nicht voll ausgenutzt werden. Außerdem machte die Kamera beim Einscannen der QR-Codes keine gute Figur, da hier das Scharfstellen bei schlechten Lichtverhältnissen zu viel Zeit in Anspruch nahm (kein Blitz vorhanden).
- Manche mobilen Geräte verwenden zum Einlesen des QR-Codes standardmäßig die Frontkamera. Da sich mit dieser nur mühsam Codes einscannen lassen, sollte standardmäßig die rückseitige Kamera ausgewählt werden.

7. Schlussbetrachtung

In der Diplomarbeit wurden zwei Plattformen erstellt, die in Kombination ein gutes Gesamtkonzept bieten, um Wegenetze zu erstellen und diese auf mobilen Geräten anzuwenden. Die umgesetzten Anwendungsmöglichkeiten des entwickelten WebGIS und der WebApp sind sehr umfangreich. Im Laufe der Arbeit wurden zahlreiche Ideen gesammelt, die aufgrund der begrenzten Zeit nicht alle realisiert werden konnten. Die Erweiterungsmöglichkeiten werden in Kapitel 9 aufgelistet. Die Ziele, die im Vorfeld bzw. am Beginn der Arbeit gesetzt wurden, konnten jedoch alle erreicht werden.

8. Anwendungsmöglichkeiten

Die in der Arbeit umgesetzten Plattformen können überall dort zur Anwendung kommen, wo Indoor-Karten eingesetzt werden. Vorrausetzung für die Installierung einer solchen Anwendung ist, dass für jedes Stockwerk Rasterkarten vorhanden sind und dass das Gebäude in der OpenStreetMap eingetragen ist. Der sehr geringe Installationsaufwand ermöglicht eine schnelle Umsetzung der Anwendung.

Beispiele für Anwendungen:

- Universitäten
- Sonstige öffentliche Gebäude
- Gebäude und Gelände von Unternehmen
- Einkaufszentren
- Museen/Messehallen

Da auf eine kontinuierliche Positionsbestimmung verzichtet wurde, ist die Umsetzung lediglich für großflächige Gebäude weniger geeignet. Beispiele dafür sind große Flughäfen und Lagerhallen.

9. Erweiterungsmöglichkeiten

Allgemein:

- Die QR-Codes, die für die passive Positionsbestimmung verwendet werden, sind zu individualisieren. Nur dann ist es möglich, diese im Anwendungsfall auch leicht zu identifizieren. Das Auffinden kann sonst sehr mühsam sein. Möglich wäre eine größere Darstellung, die Verwendung eines Rahmens, um den Code hervorzuheben, die Integration eines Logos oder sogar die Verwendung von extravaganten QR-Codes.
- Für eine praktische Umsetzung der Arbeit an der TU Graz könnte eine Verknüpfung mit der TUGonline-Datenbank hergestellt werden, die die Rauminformationen enthält und auch noch zusätzliche nützliche Informationen liefern kann.
- Auf Datensicherheit sollte Rücksicht genommen werden. Diese Thematik wurde in der Arbeit vernachlässigt.
- Die Integration zusätzlicher „Points of Interest“ wie z.B. Restaurants oder Geldautomaten.

WebGIS:

- Um die Zielpunkte korrekt zu registrieren und die Wegenetze optimal einzuzeichnen, ist es empfehlenswert, die Eintragung vor Ort durchzuführen. Dies könnte man mit einer mobilen Plattform des WebGIS realisieren, die auf eine Bedienung mittels Tablet optimiert ist.
- Nicht alle Funktionalitäten von Openlayers konnten korrekt umgesetzt werden. Die Realisierung der Snap-Funktionalität von OpenLayers 3 machte aufgrund der noch etwas mangelhaften Beschreibung Probleme. Stattdessen musste eine eigene komplexere Lösungsmethode entwickelt werden.
- Eine GIS-Anwendung mit umfangreichen Anwendungsmöglichkeiten wird zwangsläufig sehr komplex. Aufgrund der begrenzten Zeit konnten nicht alle Probleme vertiefend behandelt werden.
- Das Speichern des gezeichneten Wegenetzes ist noch etwas langsam (in Abhängigkeit von der Anzahl der Verbindungen) und könnte noch optimiert werden.
- Von Vorteil wäre, wenn die QR-Codes der Räume und Eingänge automatisiert in ein Word-Dokument gespeichert würden. Diese Datei sollte auch eine kurze Beschreibung enthalten, um die jeweiligen Codes voneinander unterscheiden zu können. In der Arbeit wurden die QR-Codes am Datenbankserver nur als jpg-Datei gespeichert und mussten einzeln in ein entsprechendes Dokument eingefügt und bearbeitet werden.
- Großteils wird der berechnete Weg optisch ansprechend dargestellt. In einigen wenigen Fällen ist dies noch zu optimieren.

WebApp:

- In der Arbeit wurde nicht berücksichtigt, dass die WebApp auf unterschiedlichen mobilen Geräten nicht einheitlich dargestellt wird. Dies hat folgende Gründe:
  - unterschiedliche Webbrowser
  - Art des Gerätes (Smartphone, Tablet, PC)
  - unterschiedliche Displayauflösung
  - unterschiedliches Betriebssystem des mobilen Gerätes

  Eine Anpassung wäre daher zu empfehlen.

- Aufgrund der unterschiedlichen Kombinationsmöglichkeiten bei der Routenberechnung (siehe Tabelle 19) war es äußerst schwierig, auch eine gute Nutzerfreundlichkeit zu realisieren. Das liegt daran, dass jede Kombination eine andere Anforderung an die Karte und die sonstigen Elemente stellt. Das führte zu einem teilweise unübersichtlichen Programm-Code, für den bei einer weiteren Verwendung unbedingt eine umfangreiche Dokumentation notwendig ist.
- Durch die Verwendung der Inertialsensoren (Beschleunigungs-/Drehratensensor) des mobilen Gerätes ist es möglich, die Orientierung des Gerätes zu berechnen (Theorie: Abschnitt 4.1.2, Umsetzung: Abschnitt 10.1.10). Dem Benutzer könnte man die Richtung anzeigen, die man am Startpunkt einschlagen soll. Als Referenz wird dabei die Ausrichtung des Smartphones beim Einscannen des QR-Codes angenommen. Da die eingebauten Sensoren der Smartphones ungenau sind und die Genauigkeit mit der Entfernung stark abnimmt, ist eine Routenführung (Guidance) mit diesen Sensoren nur am Beginn der Navigation sinnvoll.

10. Anhang

10.1 Ergänzung zu OpenLayers 3 (OL3)

Der Inhalt dieses Abschnitts basiert auf der englischsprachigen Ausgabe des Buches (GRATIER, SPENCER, and HAZARD 2015). Allgemeines zu OpenLayers ist im Abschnitt 4.16 zu finden.

10.1.1 Client-seitig

Wenn man von clientseitigen Anwendungen spricht, bezieht man sich dabei auf den Computer des Benutzers, insbesondere dessen Webbrowser. OpenLayers befindet sich auf der Client-Seite und die Hauptaufgaben bestehen darin, die Kartenbilder von einem Kartenserver zu laden und zu verarbeiten. OpenLayers funktioniert auf beinahe allen Webbrowsern.

10.1.2 Client/Server-Model

Folgende Darstellung (Abbildung 82) zeigt den Kern des Client/Server-Model, auf dem alle Web-Applikationen aufbauen. Im Fall einer Web-Map-Applikation kommuniziert der Map-Client (Open-Layers) mit dem Map-Server (WMS, OSM, etc.).

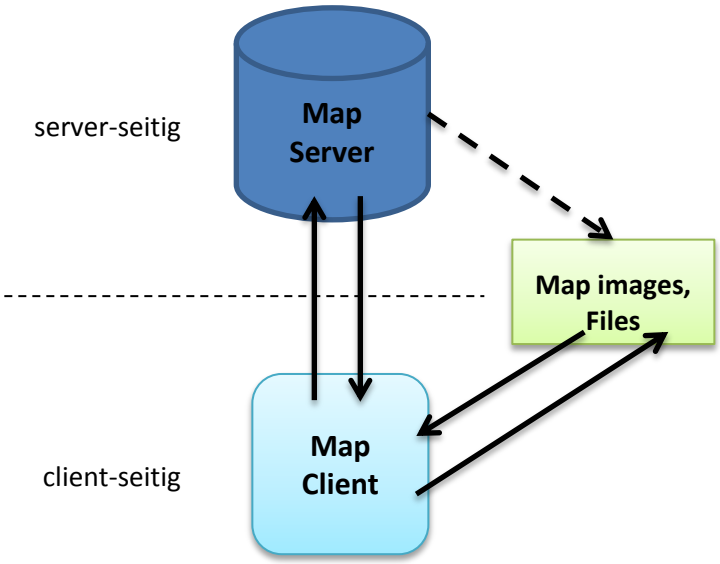

Abbildung 82– Client-Server Schnittstelle, eigene Darstellung nach (GRATIER, SPENCER, and HAZARD 2015)

Die Kommunikation wird von OpenLayers übernommen und mittels AJAX (**A**synchronous **J**ava**S**cript + **X**ML, siehe 4.17.2) realisiert. Das Grundkonzept stellt sich so dar, dass jedes Mal, wenn mit der Karte interagiert wird, OpenLayers eine Anforderung an den Map-Server sendet. Anschließend setzt OpenLayers alle erhaltenen Kartendaten stückweise zusammen, sodass diese wie eine große nahtlos zusammengehörende Karte dargestellt wird.

10.1.3 Map-Server

Der Map-Server (oder auch Map-Service) bietet die Karte selbst an. Es gibt unterschiedliche Map-Server „backends“:

- Map-Server Verwendung mit Hilfe von WMS- und WFS-Standards (z.B.: GeoServer, Mapserver, etc.)
- Proprietäre „backends“, bereitgestellt von Bing Maps oder Esri's ArcGIS Online
- „backends“, basierend auf OpenStreetMap, Stamen Maps oder MapQuest Maps

Für die Nutzung von Kartendaten sind unter Umständen Zugangsdaten erforderlich.

10.1.4 Objekt-orientiertes Design

Die OpenLayers-Bibliothek stattet den Webentwickler mit nützlichen Komponenten aus, um Web-Mapping-Applikationen zu erstellen. Um den Prinzipien eines objektorientierten Designs (siehe dazu auch Abschnitt 4.15) zu entsprechen, werden diese Komponenten als Klassen bezeichnet.

In OpenLayers sind eine Menge Klassen enthalten, die wichtigste davon ist die Map-Klasse (`ol.Map`). Instanzen dieser Klasse stehen im Mittelpunkt jeder OpenLayers-Anwendung. Diese Objekte sind Instanzen der Map-Klasse, welche aber auch Instanzen anderer Klassen nutzen, um eine interaktive Karte zu realisieren. Fast die Hälfte aller Klassen steht daher in einer direkten oder indirekten Verbindung mit der Map-Klasse.

Das folgende Diagramm (Abbildung 83) zeigt die wichtigsten direkten Beziehungen dieser Klasse.

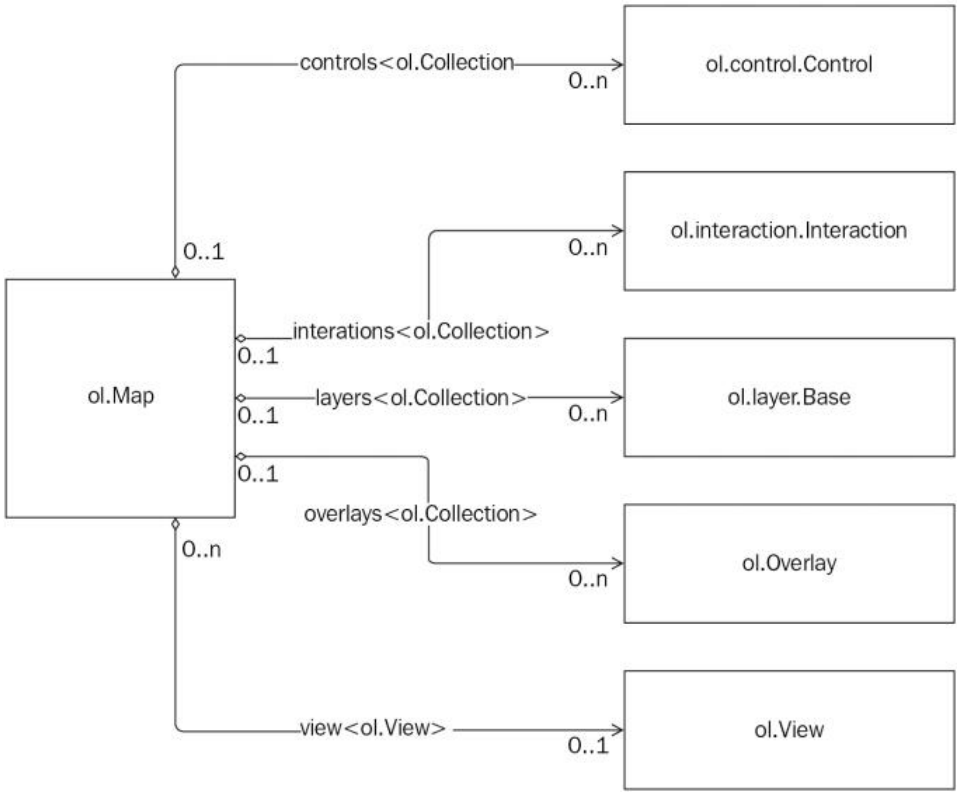

Abbildung 83 – Die wichtigsten Map-Klassen (GRATIER, SPENCER, and HAZARD 2015)

Beschreibung der Klassen:

- Die Layer-Klasse (`ol.layer.Base`) ist die Basis Klasse für Klassen, die es ermöglichen, Hintergrundkarten zu rendern (z.B. OSM-Karte, Satellitenfoto, Google-Map usw.).
- Die Control-Klasse (`ol.control.Control`) ist die Basisklasse, die es dem Benutzer ermöglicht, mit der Karte zu interagieren. Controls haben sichtbare User-Interface-Elemente (z.B. Buttons, Slides usw.).
- Die Overlay-Klasse (`ol.Overlay`) ist ein interaktives visuelles Element wie ein „Control", das aber an eine bestimmte geografische Position gebunden ist. Diese Klasse wird dazu verwendet, um die Basiskarte mit weiteren Layern zu überlagern.
- Die Interaction-Klasse (`ol.interaction.Interaction`) ist eine Basisklasse, die es dem Benutzer ebenfalls ermöglichen soll, mit der Karte zu interagieren. Im Unterschied zur Control-Klasse gibt es kein sichtbares User-Interface-Element. Beispiel dafür ist die Methode DragPan, welche es ermöglicht, mit der Maus die Karte zu bewegen.

10.1.5 Events

Events bzw. Ereignisse können z.B. durch ein Map-Objekt ausgelöst werden. Diese Ereignisse kann man mit Hilfe der `on()` und `once()` Methoden definieren. Es wird unterschieden zwischen Browser-Events, Map-Events oder Render-Events.

Beispiele für Events:

- `click/singleclick`: Dieser Event wird bei jedem einzelnen Klick in der Karte ausgelöst.
- `dblclick`: Dieser Event wird bei einem Doppelklick ausgelöst.
- `pointerdrag`: Dieser Event wird ausgelöst, wenn die Maus bewegt wird, während eine der Maustasten gedrückt ist.
- `pointermove`: Dieser Event wird ausgelöst, sobald sich die Maus bewegt.
- `moveend`: Dieses Ereignis wird ausgelöst, nachdem die Karte neu gerendert wurde, z.B. nach dem Schwenken oder Zoomen.
- `postrender`: Dieser Event wird ausgelöst, wenn die Karte bzw. alle Layer gerendert wurden.
- `precompose`: Dieser Event wird ausgelöst, wenn die Karte gerendert wird, kurz bevor alle Layer gezeichnet werden.

10.1.6 Map-Renderer

OpenLayers zeichnet die Karten mit Hilfe von Renderer, wobei es in OL3 drei Renderer gibt, von denen jeder bestimmte Fähigkeiten und Einschränkungen hat.

- Canvas-Renderer:

  Das in HTML5 eingeführte Canvas-Element ermöglicht es, mit JavaScript in einem mit Höhen- und Breitenangaben beschriebener Bereich zu zeichnen. Innerhalb von HTML5 wurde dieses Element standardisiert. Beim Internet Explorer ist die Canvas-Unterstützung erst ab der Version 9+ gegeben.

- WebGL-Renderer:

  Dieser leistungsfähige Renderer ermöglicht dem Webbrowser den Zugriff auf die hardwarebeschleunigten 2D- und 3D-Grafik-Renderer, welche normalerweise nur der Desktop-Software (Spiele) vorbehalten ist. Vor allem bei mobilen Browsern ist die Unterstützung noch mangelhaft.

- DOM-Renderer:

  Der DOM Renderer zeichnet den Karteninhalt mit Hilfe von HTML-Elementen (`<img>` und `<div>` Tags) und verwendet Stylesheets (CSS), um die Inhalte zu positionieren. Dieser Renderer wird auch von den meisten älteren Browser unterstützt. Einige der interessanten Fähigkeiten des Canvas-Renderers werden jedoch nicht unterstützt. DOM-Renderer haben keine gute Performance.

10.1.7 Projektionen

In OpenLayers werden zahlreiche Projektionen für verschiedene Anwendungen unterstützt. Eine wichtige Projektion, die auch von den kommerziellen Kartenanbietern Google Maps, Bing Maps sowie OpenStreetMap verwendet wird, ist die Mercatorprojektion (siehe 4.12.3).

Um die zahlreichen Projektionen voneinander zu unterscheiden, gibt es unterschiedliche Klassifizierungssysteme, wobei OpenLayers die EPSG-Codes nutzt. Die Standard-Projektion wird mit dem Code EPSG:4326 angegeben, welcher auch unter der Bezeichnung WGS 84 (sphärisches Rechtssystem, Länge: -180° bis 180 und Breite: -90° bis 90°) bekannt ist. Die kartesische Mercator-Projektion dieses Referenzsystems (WGS 84) wird mit dem Code EPSG:3857 (bzw. EPSG:900913 für Google) definiert.

10.1.8 Layer

Es gibt zwei verschiedene Arten von Layern, die sich wie folgt unterscheiden:

- Raster:

  Raster-Layer sind Bilder (PNG oder JPEG), die serverseitig oder statisch erzeugt werden. Es gibt auch noch eine weitere Unterteilung in tiled- oder untiled-Layer (siehe 4.10). Bei einem Raster-Layer wird das Rasterbild in der Karte angezeigt (z.B. ein Satellitenbild). Alle Objekte, wie z.B. Gebäude oder Straßen, die sich in diesem Bild befinden, sind miteinander vereinigt. Für die weitere Verarbeitung von Raster-Daten sind für die Objekte keine Metadaten vorhanden.

- Vektor

  Vektor-Layer sind Features, welche durch geografische Koordinaten beschrieben werden. Diese verändern beim Zoomen nicht ihre Größe und lassen ihr Aussehen (`ol.style`) leicht manipulieren. Bei der Verwendung von Vektor-Layern kann man aus den Feature-Objekten zusätzliche Informationen gewinnen. Außerdem können Vektor-Layer dazu verwendet werden, um Geometrien zu zeichnen. Auch das Editieren dieser Layer ist möglich. Das Erzeugen des Vektor-Layers und die Interaktionen erfolgen dabei client-seitig.

Folgendes Diagramm (Abbildung 84) gibt einen Überblick, wie die Layers in OpenLayers 3 organisiert sind.

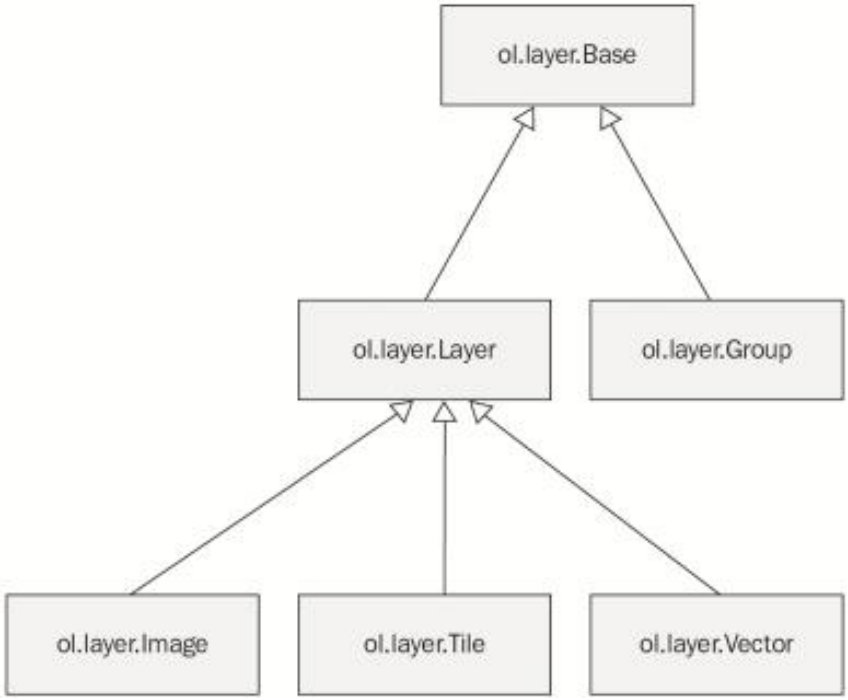

Abbildung 84 – Layer in OL3 (GRATIER, SPENCER, and HAZARD 2015)

In Abbildung 85 sind Raster-Layer (Satellitenbild) und Vektor-Layer (Gebäude = rot, Straßen = weiß, Parkplätze = grau, Grünflächen = grün, Bäume = hellgrün) zum Vergleich dargestellt.

Abbildung 85 – Vergleich zwischen Raster-Layer und Vektor-Layers (GRATIER, SPENCER, and HAZARD 2015)

10.1.9 Layer-Quellen

In OpenLayers 3 wird, wie schon zuvor erwähnt, zwischen den drei Arten von Layern (Image, Tile und Vector) unterschieden. Zusätzlich gibt es eine Unterscheidung nach der Quelle (englisch: source, `ol.source`), die angibt, aus welcher Datenquelle diese Daten kommen. Folgendes Diagramm (Abbildung 86) zeigt die möglichen Layer-Quellen, die es in OpenLayers 3 gibt.

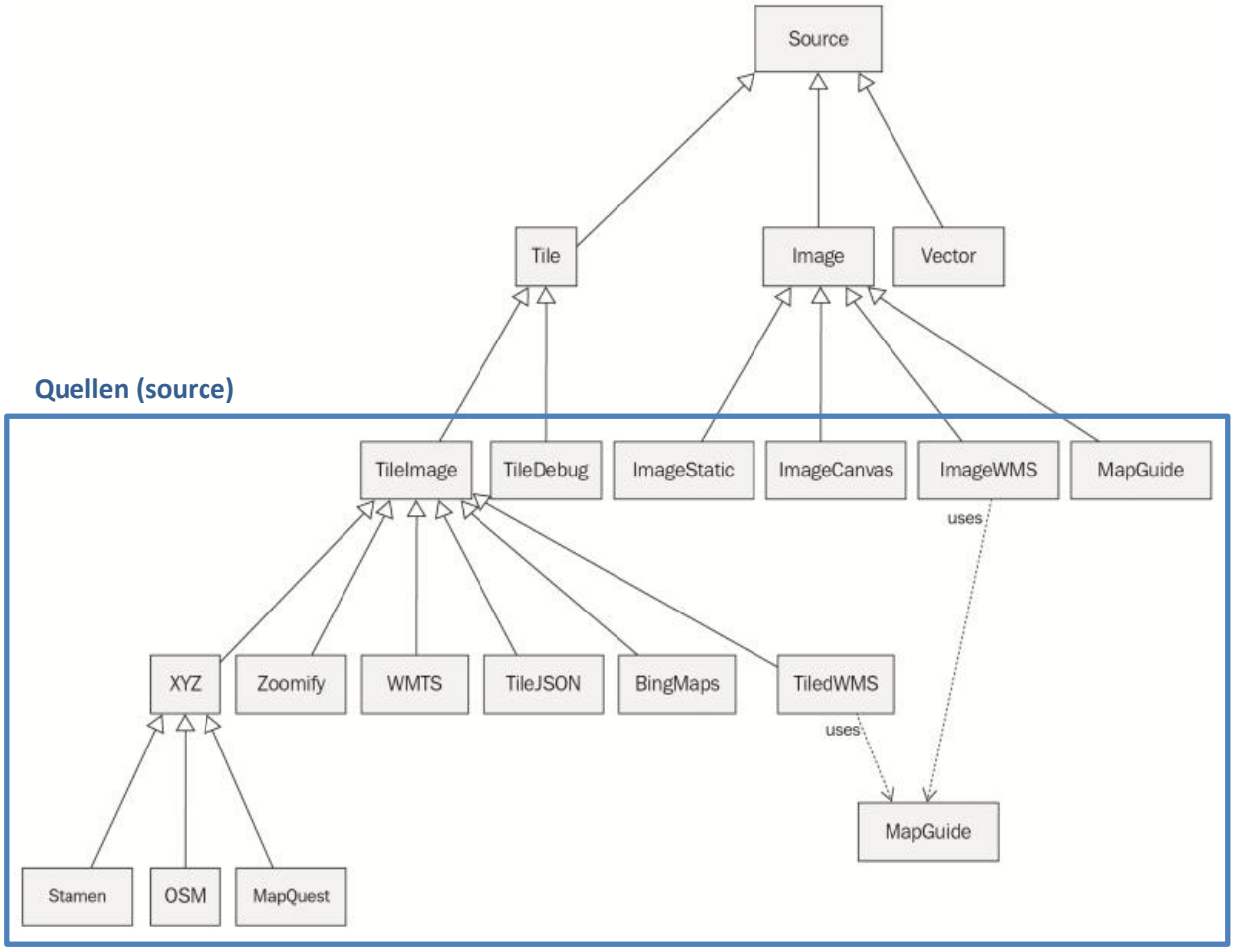

Abbildung 86 – Quellen von Layern in OL3 (GRATIER, SPENCER, and HAZARD 2015)

Beispiel für einen OpenStreetMap-Konstruktor:

```
ol.source.OSM
```

10.1.10 DeviceOrientation-Klasse

Immer mehr Smartphones und Tablets erlauben die Hardware-Unterstützung einiger integrierter Sensoren, um es zu ermöglichen, die Orientierung des Gerätes zu verfolgen. In HTML5 gibt es die DeviceOrientation-API, um den Zugang zu diesen Informationen zu bekommen. So wie die Geo-location-API stellt OpenLayers 3 dafür die `ol.DeviceOrientation`-Klasse bereit.

Die Ausrichtung des Gerätes bezieht sich auf die aktuelle Orientierung des Geräts relativ zu einer Ausgangsstellung. Für Smartphones und Tablets ist die Ausgangsstellung dadurch definiert, dass das Gerät auf einem Tisch in der horizontalen Ebene liegt, wobei das Display nach oben schaut und die

obere Kante des Gerätes nach Norden zeigt. Für Computer ist die Ausgangsstellung dieselbe, außer dass der Bildschirm um 90 Grad nach oben gedreht ist und in einer vertikalen Ebene liegt.

Die Orientierung des Gerätes wird durch die drei Drehwinkel Alpha, Beta und Gamma aufgezeichnet, deren Bezug relativ zur Ausgangsstellung ist. Die Definition der Achsen ist in Abbildung 87 dargestellt. Um die DeviceOrientation-Klasse mit dem Smartphone zu testen, wurde das Beispiel von (openlayers.org - DeviceOrientation-Klasse 2015) adaptiert.

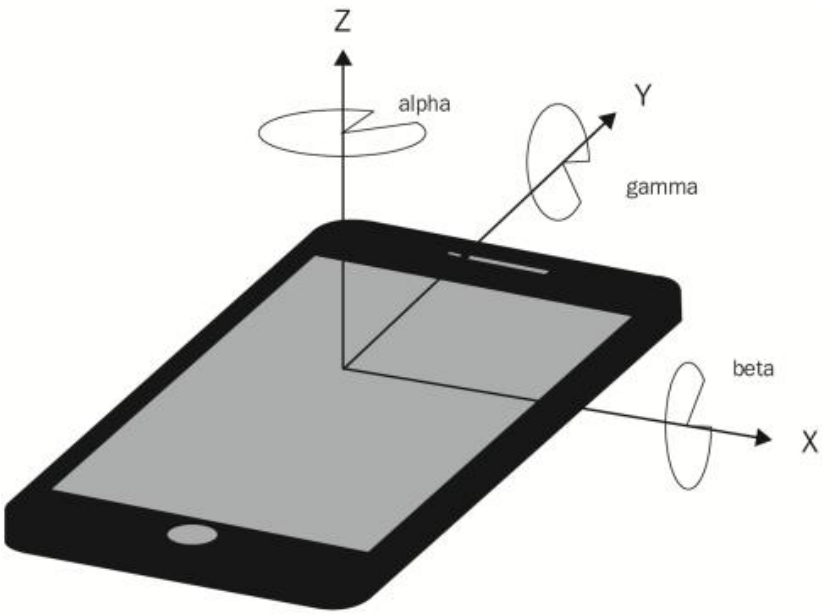

Abbildung 87 – Die drei Drehachsen eine Smartphones (GRATIER, SPENCER, and HAZARD 2015)

11. Literaturverzeichnis

11.1 Bücher

BARTELME, Norbert. 2005. *Geoinformatik - Modelle, Strukturen, Funktionen*. Springer.

BAUER, Günther. 2009. *Architekturen für Web-Anwendungen: Eine praxisbezogene Konstruktions-Systematik*. Springer.

BINDER, Franz 2006. "Mathematik und Logik." Uni Linz. (http://www.algebra.uni-linz.ac.at/Students/Win/fg07s/lec/planar.pdf)

DIESTEL, Reinhard. 2006. *Graphentheorie*. Springer-Verlag Heidelberg.

GRATIER, Thomas, Paul SPENCER, and Erik HAZARD. 2015. *OpenLayers 3: Beginner's Guide*. Packt Publishing.

HOFMANN-WELLENHOF, Bernhard, Klaus LEGAT, and Manfred WIESER. 2003. *Navigation Principles of Positioning and Guidance*. Springer-Verlag.

JANSEN, Marc and Till ADAMS. 2010. *OpenLayers - Webentwicklung mit dynamischen Karten und Geodaten*. OpenSource Press.

WIESER, Manfred. 2013. *LV Navigation Systems - Nummer 508.302*.

11.2 Webseiten, Zeitungsartikel und pdf-Dokumente

1php.de - PHP-Einführung und Grundlagen. 2015. (http://1php.de/).

ADOLF, D., M. FLIEGE, and C. LAUN. 2006. "Studienarbeit im Fach Multimedia und Webtechnologien." goessner.net - AJAX. (http://goessner.net/download/learn/mwt/ws2005/presentations/Ajax.pdf).

anitagraser.com - osm2po-Quickstart. 2011. (http://anitagraser.com/2011/12/15/an-osm2po-quickstart/).

arcgis.com - Kartenprojektionen. 2015. (http://resources.arcgis.com/de/help/main/10.1/index.html#//003r0000000q000000).

artalis.de - GIS. 2015. (http://www.artalis.de/pages/wissenswertesglossar/mathematische-grundlagen/gis.php).

artalis.de - Mathematische Grundlagen. 2015. (http://www.artalis.de/pages/wissenswertesglossar/mathematische-grundlagen.php).

artalis.de - Projektionen. 2015. (http://www.artalis.de/website/media/fertige%20Bilder/Projektionen.png).

atandrastoth.co.uk - codereader. 2015. (http://atandrastoth.co.uk/main/pages/plugins/codereader/).

BARTHEL, S., R. EID-SABBAGH, S. MÜLLER, and C. TINNEFELD. 2005. “hpi.de/fileadmin/user_upload - AjaxAusarbeitung.” (http://hpi.de/fileadmin/user_upload/fachgebiete/meinel/lecturenotes/webprogrammierung/AjaxAusarbeitung.pdf).

browsercheck.pcwelt.de - Geolokalisierung. 2015. (http://www.browsercheck.pcwelt.de/de/dokumentation/geolokalisierung-so-bestimmen-sie-ihren-standort).

caniuse.com - getUserMedia/Stream API. 2015. (http://caniuse.com/#feat=stream).

cartogis.de - Was ist GIS? 2015. (http://cartogis.de/parallel/geom/gistxt.htm#Was%20GIS).

clipartbest.com - Weltkarte. 2015. (http://www.clipartbest.com/cliparts/LTK/jBL/LTKjBL5Ta.gif).

“c't zu Postgresql v7.0.2.” 2000. (http://www.heise.de/download/postgresql-363829.html).

CZYCHOLL, Harald. 2015. “welt.de -.Navi in Gebäuden.” (http://www.welt.de/wirtschaft/webwelt/article116376073/WLAN-Netz-soll-Navi-in-Gebaeuden-moeglich-machen.html).

datenbanken-verstehen.de - Datenbankmodel. 2015. (http://www.datenbanken-verstehen.de/datenbanken/datenbank-grundlagen/datenbankmodell).

datenbanken-verstehen.de - Fremdschlüssel. 2015. (http://www.datenbanken-verstehen.de/datenbanken/datenmodellierung/fremdschluessel/).

datenbanken-verstehen.de - Hierarchisches Datenbankmodell. 2015. (http://www.datenbanken-verstehen.de/datenbanken/datenbank-grundlagen/datenbankmodell/hierarchisches-datenbankmodell/).

datenbanken-verstehen.de - Netzwerkdatenbankmodell. 2015. (http://www.datenbanken-verstehen.de/datenbanken/datenbank-grundlagen/datenbankmodell/netzwerkdatenbankmodell/).

datenbanken-verstehen.de - Normaliserung. 2015. (http://www.datenbanken-verstehen.de/datenbanken/datenmodellierung/normalisierung/).

datenbanken-verstehen.de - Objektorientiertes Datenbankmodell. 2015. (http://www.datenbanken-verstehen.de/datenbanken/datenbank-grundlagen/datenbankmodell/objektorientiertes-datenbankmodell/).

datenbanken-verstehen.de - Primärschlüssel. 2015. (http://www.datenbanken-verstehen.de/datenbanken/datenmodellierung/primaerschluessel/).

datenbanken-verstehen.de - Relationales Datenbankmodell. 2015. (http://www.datenbanken-verstehen.de/datenbanken/datenbank-grundlagen/datenbankmodell/relationales-datenbankmodell/).

datenbanken-verstehen.de - SQL Einführung – Was ist SQL ? 2015. (http://www.datenbanken-verstehen.de/datenbanken/sql-tutorial/sql-einfuehrung/).

de.freepik.com - Symbole. 2015. (http://de.freepik.com).

de.wikipedia - Funkzelle. 2015. (https://de.wikipedia.org/wiki/Funkzelle).

deep-map.com - Deep-Map. 2015. (http://www.deep-map.com/de/).

deep-map.com - Funktionen. 2015. (http://www.deep-map.com/de/funktionen.html).

dev.w3.org - Geolocation API Specification. 2015. (http://dev.w3.org/geo/api/spec-source.html).

developers.google.com - geocoding. 2015.
(https://developers.google.com/maps/documentation/geocoding/intro).

docs.pgrouting.org - pgr-create-topology. 2015.
(http://docs.pgrouting.org/dev/src/common/doc/functions/create_topology.html#pgr-create-topology).

docs.pgrouting.org - pgRoutingDocumentation.pdf. 2015.
(http://docs.pgrouting.org/2.0/en/pgRoutingDocumentation.pdf).

docs.qgis.org - Vorwort. 2015.
(http://docs.qgis.org/2.8/de/docs/user_manual/preamble/foreword.html).

electronics360.globalspec.com - NFC-Tag. 2015.
(http://electronics360.globalspec.com/images/assets/147/5147/NFC_tag.jpg).

etools.fernuni.ch - MATLAB. 2015.
(http://etools.fernuni.ch/matlab/matlab2/de/html/unit_why_matlab.html).

fossgis.de - OGC Web Services in JSON. 2015.
(http://www.fossgis.de/konferenz/2015/programm/events/814.de.html).

futurezone.at - TU-Wien Navigations-App. 2015. (http://futurezone.at/apps/tu-wien-entwickelt-navigations-app-die-ohne-gps-auskommt/131.169.631).

GASSMANN, Johanna. 2014. “flyacts.com - Native App vs. mobile Web-App – Ein praktischer Vergleich.” (http://www.flyacts.com/blog/native-app-vs-mobile-web-app-ein-praktischer-vergleich/).

geofabrik.de - Download. 2015. (http://download.geofabrik.de/).

giswiki - WKB. 2015. (http://giswiki.org/wiki/WKB).

giswiki - WKT. 2015. (http://giswiki.org/wiki/Well_Known_Text).

giswiki.org - WMS. 2015. (http://giswiki.org/wiki/Web_Map_Service).

gitta.info - Dijkstra Algorithm: Short terms and Pseudocode. 2015.
(http://www.gitta.info/Accessibiliti/de/html/Dijkstra_learningObject1.html).

google.com - indoormaps. 2015. (https://www.google.com/maps/about/partners/indoormaps/).

handyortung-und-schutz.de - GSM-Ortung. 2015. (http://www.handyortung-und-schutz.de/ortung/gsm-ortung-zellortung).

HENGSTBACH, Axel. 2011. “computerbild.de - QR-Codes: lesen, erzeugen, verstehen.” (http://www.computerbild.de/artikel/cb-Tipps-Wissen-QR-Codes-Barcode-EAN-6122468.html).

html-seminar.de - Web App versus Native App. 2015. (http://www.html-seminar.de/web-app-versus-native-app.htm).

iis.fraunhofer.de - Adaptive Systemsoftware. 2015. (http://www.iis.fraunhofer.de/content/dam/iis/de/doc/lv/los/lokalisierung/SatNAV/Adaptive%20Systemsoftware_deutsch.pdf).

iis.fraunhofer.de - rssi/awiloc®. 2015. (http://www.iis.fraunhofer.de/de/ff/lok/tech/feldstaerke/rssi.html).

India-Indoor. 2015. (http://cdn1.tnwcdn.com/wp-content/blogs.dir/1/files/2014/03/India-Indoor.png).

indoor-ortung.de - Ortung mit Bluetooth. 2015. (http://indoor-ortung.de/technik/ortung-mit-bluetooth).

inf-schule.de - Der Algorithmus von Dijkstra. 2014. (http://www.inf-schule.de/algorithmen/graphen/wegeingraphen/station_dijkstra).

it-infothek.de - Objektorientierte Programmierung. 2003. (http://www.it-infothek.de/wirtschaftsinformatik/semester-2/programmierung-2-java-3.html#progr_2_301).

itwissen.info - LBS. 2015. (http://www.itwissen.info/definition/lexikon/location-based-service-LBS-Ortsbezogener-Dienst.html).

itwissen.info - Objektrelationale Datenbank. 2015. (http://www.itwissen.info/definition/lexikon/Objektrelationale-Datenbank-object-relational-database.html).

itwissen.info - OGC. 2015. (http://www.itwissen.info/definition/lexikon/open-GIS-consortium-OGC.html).

jaxa.jp - GPS. 2015. (http://global.jaxa.jp/countdown/f18/overview/gps_e.html).

jplugins.directory - webcodecam. 2015. (http://jplugins.directory/webcodecam/?show=demo).

kartenkunde-leichtgemacht.de - Kartennetzentwurf. 2015. (http://kartenkunde-leichtgemacht.de/handbuch.php?page=Kartennetzentwurf).

kartenkunde-leichtgemacht.de - Koordinatensysteme. 2015. (http://kartenkunde-leichtgemacht.de/handbuch.php?page=Koordinatensysteme).

kowoma - GPS. 2015. (http://www.kowoma.de/gps).

kowoma - Kartenprojektionen. 2007. (http://www.kowoma.de/gps/geo/Projektionen.htm).

KRAMER, Andre. 2013. “Quadratisch, praktisch, Code.” *c't*, July. http://www.heise.de/ct/ausgabe/2013-7-Erfinderische-und-praktische-Anwendungen-fuer-QR-Codes-2326094.html.

"Leitfaden – Hardware, Software, IT-Sicherheit V2.0." 2015. pdf, Geoinformatik, TU München. http://www.rtg.bv.tum.de/files/Kompetenzpools/Mobile%20GIS/2014/Leitfaden_Mobile_GIS_V20.pdf.

live.osgeo.org - pgrouting_quickstart. 2015. (http:///de/quickstart/pgrouting_quickstart.html).

LOULIER, Benjamin. 2011. "Using smartphones for indoor navigation." West Lafayette, Indiana. http://docs.lib.purdue.edu/cgi/viewcontent.cgi?article=1058&context=techmasters.

m9.ma.tum.de - Der A*-Algorithmus. 2014. (https://www-m9.ma.tum.de/graph-algorithms/spp-a-star/index_de.html).

m9.ma.tum.de - Der Dijkstra-Algorithmus. 2014. (https://www-m9.ma.tum.de/graph-algorithms/spp-dijkstra/index_de.html).

m9.ma.tum.de - Graphalgorithmen. 2014. (http://www-m9.ma.tum.de/Allgemeines/GraphAlgorithmen).

mathe.tu-freiberg - Graphentheorie. 2015. (http://www.mathe.tu-freiberg.de/~hebisch/cafe/graphentheorie.html).

mathworks - Documentation. 2015. (http://de.mathworks.com/help/matlab/).

mathworks.com - GUI. 2015. (http://www.mathworks.com/matlabcentral/mlc-downloads/downloads/submissions/38235/versions/1/screenshot.jpg).

MEYER, Steffen. 2008. "Positionsbestimmung per WLAN statt GPS.". http://www.heise.de/ct/artikel/Positionsbestimmung-per-WLAN-statt-GPS-1898179.html.

mobilehtml5.org - Browser-Kompabilität. 2015. (http://mobilehtml5.org/).

molily.de - JavaScript. 2015. (http://molily.de/js/aufgaben.html).

mozilla.org - HTML. 2015. (https://developer.mozilla.org/de/docs/Web/HTML).

navvis.lmt.ei.tum.de - about. 2015. (http://www.navvis.lmt.ei.tum.de/about/).

navvis.lmt.ei.tum.de - NAVVIS. 2015. (http://www.navvis.lmt.ei.tum.de/).

nd.edu - Statistik. 2015. (http://www3.nd.edu/~jeff/Teaching/CBE60478/RefineryVAR/html/RefineryVAR_03.png).

olanis.de - Kartenprojektionen. 2015. (http://www.olanis.de/de/01-10-02-kartenprojektionen-und-koordinatentransformationen.html).

openbook.rheinwerk-verlag - Objektorientierte Programmierung. 2015. (http://openbook.rheinwerk-verlag.de/oop/).

openlayers.org - DeviceOrientation-Klasse. 2015. (http://openlayers.org/en/v3.8.2/examples/device-orientation.html).

osgeo.org. 2015. (http://www.osgeo.org/).

osm2po.de. 2015. (http://osm2po.de/).

panoramatec.com - indrz-campus. 2015.
(http://panoramatec.com/upload/background/products/Folder_Campus.pdf).

pbs.twimg.com - vanillaNAV. 2015. (https://pbs.twimg.com/media/CJdOL2eUkAAdcWm.jpg).

“pgRouting Manual.” 2013. (http://docs.pgrouting.org/2.0/en/pgRoutingDocumentation.pdf).
http://docs.pgrouting.org/2.0/en/pgRoutingDocumentation.pdf.

pgrouting.org - Project. 2015. (http://pgrouting.org/).

php.net - PHP-Handbuch. 2015. (https://secure.php.net/manual/de).

play.google.com - TU Graz Raumsuche. 2015.
(https://play.google.com/store/apps/details?id=com.team4win.tugroom&hl=de).

plazz-entertainment.com - Native App vs. Web App vs. Mobile Seite. 2014. (http://www.plazz-entertainment.com/allgemein/native-app-vs-web-app-vs-mobile-seite-%E2%80%93-vor-und-nachteile-der-mobilen-angebote-5741/).

portal.tugraz.at - tugonline. 2015. (http://portal.tugraz.at/portal/page/portal/zid/tugonline).

postgresql.de - das freie objektrelationale Open Source Datenbanksystem. 2015.
(http://www.postgresql.de/).

postgresql.org - PL/pgSQL: SQL prozedurale Sprache. 2013.
(https://web.archive.org/web/20100315055655/http://www.postgresql.org/files/documentation/books/pghandbuch/html/plpgsql.html).

praxistipps.chip.de - NFC - Was ist das? 2015. (http://praxistipps.chip.de/nfc-was-ist-das_12294).

python-kurs.eu - OOP-Klassen. 2015. (http://www.python-kurs.eu/python3_klassen.php).

qgis-anwendertreffen.de - Über QGIS/Ziele. 2015. (http://www.qgis-anwendertreffen.de/).

quantenwelt.de - Almanach. 2015. (http://www.quantenwelt.de/technik/GPS/almanach.html).

reeg.junetz.de - Datenbanksystem. 2015. (http://reeg.junetz.de/DSP/node6.html).

ree-wifi-service.de - Passive Indoor-Navigation. 2015. (http://www.free-wifi-service.de/indoor-navigation-per-wifi-wlan/passive-indoor-navigation/).

“RFID Radiofrequenz-Identifikation.” 2010.
https://www.datenschutz.rlp.de/downloads/oh/info_RFID.pdf.

roomaps.com - ROOMAPS. 2015. (http://www.roomaps.com/de/technology/).

roomaps.com - Technology. 2015. (http://www.roomaps.com/de/technology/).

sejox.de - Geschwindigkeit eines Fußgängers. 2015. (http://www.sejox.de/2011/03/23/google-maps-routenplaner-geschwindigkeit-eines-fusgangers/).

selfhtml5.org - geolocation. 2015. (http://www.selfhtml5.org/2014-html5-features/standort-im-browser-ermitteln-per-html5-geolocation-api/).

selfhtml5.org - HTML5 Features. 2015. (http://www.selfhtml5.org/html5-features/).

seos-project.eu - Satellitenbahnen. 2015. (http://www.seos-project.eu/modules/GPS/images/satellitenbahn.gif).

signalinc.com - Choosing the Right Technology for Your Mobile App Strategy. 2013. (http://www.signalinc.com/choosing-the-right-technology-for-your-mobile-app-strategy/).

sql-und-xml.de - Grundbegriffe und Konzepte von Datenbank-Systemen. 2015. (http://www.sql-und-xml.de/sql-tutorial/datenbank-grundbegriffe.html).

Stelzel-Morawietz, Peter. 2015. “Routenplanung in geschlossenen Räumen.” *Android Welt*, February, pp. 20/21.

teialehrbuch.de - PostgreSQL - Werkzeuge. 2015. (https://www.teialehrbuch.de/Kostenlose-Kurse/Datenbankentwicklung-mit-PostgreSQL-9/2.1.2-PostgreSQL-Werkzeuge.html).

trafficmaxx.de - Geschwindigkeit eines Radfahrers. 2015. (http://www.trafficmaxx.de/blog/google/fahrradwege-bei-google-maps).

vanillanav.com. 2015. (http://www.vanillanav.com/Home/Index).

w3schools - JavaScript Cookies. 2015. (http://www.w3schools.com/js/js_cookies.asp).

w3schools.com - geolocation. 2015. (http://www.w3schools.com/html/html5_geolocation.asp).

web.mit.edu - Pathfinding using A*. 2002. (http://web.mit.edu/eranki/www/tutorials/search/).

wiki.openstreetmap.org - QGIS. 2015. (http://wiki.openstreetmap.org/wiki/QGIS).

wiki.selfhtml - CSS. 2015. (http://wiki.selfhtml.org/wiki/CSS).

wiki.selfhtml - DOM. 2015. (http://wiki.selfhtml.org/wiki/JavaScript/Objekte/DOM).

wiki.selfhtml - JavaScript. 2015. (http://wiki.selfhtml.org/wiki/JavaScript/Einf%C3%BChrung).

wiki.selfhtml - JSON. 2015. (http://wiki.selfhtml.org/wiki/JavaScript/JSON).

wiki.selfhtml - PHP. 2015. (http://wiki.selfhtml.org/wiki/PHP).

wiki.selfhtml.org - Geolocation. 2015. (https://wiki.selfhtml.org/wiki/JavaScript/API/Geolocation).

wikibooks - Einführung in SQL. 2015. (https://de.wikibooks.org/wiki/Einf%C3%BChrung_in_SQL).

wikipedia - Konforme Abbildungen. 2015. (https://de.wikipedia.org/wiki/Konforme_Abbildung).

wikipedia - QR-Code. 2015. (https://de.wikipedia.org/wiki/QR-Code).

wikipedia - QR-Code, svg. 2015. (https://de.wikipedia.org/wiki/QR-Code#/media/File:QR_Code_Struktur_Beispiel.svg).

wikipedia - WLAN-basierte Ortung. 2015. (https://de.wikipedia.org/wiki/WLAN-basierte_Ortung).

wirtschaftslexikon.gabler.de - Datenbanken. 2015. (http://wirtschaftslexikon.gabler.de/Definition/datenbank.html).

12. Abbildungsverzeichnis

13. Tabellenverzeichnis