

Fakultät für Informatik

Studiengang Informatik

„GeoVisualisierung von technischen Netzparametern im DOCSIS Umfeld“

Bachelor Thesis

von

Martin Daxlberger

Datum der Abgabe: 30.09.2017

Erstprüfer: Prof. Dr. Beneken

Zweitprüfer: Prof. Dr. Mühlbauer

ERKLÄRUNG

Ich versichere, dass ich diese Arbeit selbständig angefertigt, nicht anderweitig für Prüfungszwecke vorgelegt, keine anderen als die angegebenen Quellen oder Hilfsmittel benützt sowie wörtliche und sinngemäße Zitate als solche gekennzeichnet habe.

Rosenheim den, tt.mm.jjjj

Martin Daxlberger

**Kurzfassung (Abstract)**

Text

Schlagworte

**Inhaltsverzeichnis**

[Abbildungsverzeichnis iii](#_Toc492385970)

[Tabellenverzeichnis v](#_Toc492385971)

[Abkürzungsverzeichnis vii](#_Toc492385972)

[1 Einleitung 1](#_Toc492385973)

[1.1 Motivation 1](#_Toc492385974)

[1.2 Aufgabenstellung 1](#_Toc492385975)

[1.3 Aufbau der Arbeit 2](#_Toc492385976)

[2 Grundlagen 3](#_Toc492385977)

[2.1 Grundlagen Netzwerktechnik 3](#_Toc492385978)

[2.1.1 SNMP 3](#_Toc492385979)

[2.1.2 DOCSIS 3](#_Toc492385980)

[2.2 Grundlagen Geografische Informationssysteme (GIS) 4](#_Toc492385981)

[2.2.1 Geoinformatik 4](#_Toc492385982)

[2.2.2 Geokodierung 4](#_Toc492385983)

[2.2.3 GML und GeoJson 5](#_Toc492385984)

[2.2.4 Netzwerkanalysen 6](#_Toc492385985)

[3 Konzept 7](#_Toc492385986)

[3.1 Anforderungsanalyse 8](#_Toc492385987)

[3.2 Fachliches Konzept 8](#_Toc492385988)

[3.2.1 Serverseite 8](#_Toc492385989)

[3.2.2 Clientseite 9](#_Toc492385990)

[3.2.3 Datenspeicherung 10](#_Toc492385991)

[3.3 Technologiestack 10](#_Toc492385992)

[3.3.1 Webstack 11](#_Toc492385993)

[3.3.1 Geoinformatik 11](#_Toc492385994)

[3.3.2 Sonstiges 12](#_Toc492385995)

[4 Implementierung 13](#_Toc492385996)

[4.1 Webserver und angewandte Geokodierung 13](#_Toc492385997)

[4.2 Startseite und Geojson-View 14](#_Toc492385998)

[4.3 Kartendarstellung 15](#_Toc492385999)

[4.4 Styling 19](#_Toc492386000)

[4.5 Tests 20](#_Toc492386001)

[4.5.1 Tests während der Entwicklung 20](#_Toc492386002)

[4.5.2 Praxistests 20](#_Toc492386003)

[5 Ergebnisse 22](#_Toc492386004)

[5.1 Endresultat des Projekts 22](#_Toc492386005)

[5.2 Probleme beim Erstellen der Software 22](#_Toc492386006)

[5.3 Stellungnahme des Auftraggebers 22](#_Toc492386007)

[6 Zusammenfassung und Ausblick 23](#_Toc492386008)

[6.1 Zusammenfassung 23](#_Toc492386009)

[6.2 Künftige Features 23](#_Toc492386010)

[6.3 Integration/Zusammenlegen mit anderen Systemen 24](#_Toc492386011)

[A Erstes Kapitel des Anhangs 26](#_Toc492386012)

[Literaturverzeichnis 28](#_Toc492386013)

# Abbildungsverzeichnis

[Abbildung 1 Originalbild 1](file:///C:\Users\stefanie.urchs\Desktop\Allgemeines\Vorlage-Bachelorarbeit\Vorlage-Bachelorarbeit.docx#_Toc363021901)

[Abbildung 2 erweitertes Bild 2](#_Toc363021902)

# Tabellenverzeichnis

[Tabelle 1.1: Testtabelle 2](#_Toc440029735)

# Abkürzungsverzeichnis

|  |  |
| --- | --- |
|  |  |
| DOCSIS  GIS | Data Over Cable Service Interface Specification  Grundlagen Geografische Informationssysteme |
| JS  AG  OGC  XML  JSON | Java Script  Auftraggeber  Open Geospatial Consortium  Extensible Markup Language  JavaScript Object Notation |

# Einleitung

In diesem Kapitel werden Motivation, Aufgabenstellung und Aufbau dieser Bachelorarbeit behandelt.

## Motivation

Die Firma Komro betreibt seit einiger Zeit (Details herausfinden) ein Kabelnetzwerk für das Stadtgebiet Rosenheim. Später erfolgten eine Ausweitung des Gebiets, sowie eine Erweiterung der Angebotenen Produkte um Internet und Internettelefonie. Beim Betrieb einer Netzstruktur ist es für einen Kabelnetzbetreiber sehr von Vorteil, auch bei vielen Anschlüssen in einem Gebiet die Übersicht zu behalten. Dies ist wichtig um schnell auf Störungen reagieren und deren Ursachen erkennen und beheben zu können. Zu wissen ob nur ein einzelner Anschluss betroffen ist oder gar alle die an einem Verteilerpunkt hängen kann dabei eine große Rolle spielen.

Es ist auch nur schwer nachvollziehbar ob eine andere Geographische Gegebenheit zu einer schlechteren Verbindung führt. Ob Beispielsweise andere Netze die Signale stören wie etwa von einer Sendeantenne für LTE oder sich der Hausanschluss sehr weit entfernt vom restlichen Netz befindet.

Auch für die Netzausdehnung ist eine gute Planung für das Setzen von Verteilerpunkten oder das Ersetzen solcher ein schwieriges Unterfangen.

Hier kommt die Geoinformatik mit ins Spiel. Ist das Telekommunikationsnetz innerhalb einer Karte dargestellt, ist es leicht wichtige Informationen hervorzuheben. Beispielsweise Störungen, die einen kompletten Straßenblock betreffen sind auf Anhieb leicht zu erkennen.

Aber auch für die Marktforschung kann eine Kartendarstellung von Vorteil sein. Gebiete in denen sich nur wenige Anschlüsse, trotz hoher potentieller Kundenzahl, befinden können schnell ermittelt werden.

## Aufgabenstellung

Im Rahmen dieser Bachelorarbeit soll eine GeoVisuelle Darstellung einer Sammlung von Modemdaten implementiert werden. Genauer geht es um einzelne Netzparameter aus dem Docsis-Umfeld. Sie umfasst Alles von der Anforderungsanalyse über die Technologieauswahl, bis hin zur konkreten Implementierung und Dokumentation eines Prototyps.

Je eine Datenbank für die Modemdaten, sowie für die Adressdaten der Kunden ist bereits gegeben. Diese stehen bislang allerdings noch nicht miteinander in Verbindung und müssen erst noch „gemapped“ werden. Anschließen soll das Resultat in eine Anschauliche Kartendarstellung gebracht werden. Ein mögliches Ergebnis dieser Projektarbeit könnte es sein die Modemdaten zu nutzen um in Echtzeit eine farbliche Änderung an den Modempunkten auf der Karte zu bewirken. Die Anwendung soll möglichst so gestaltet werden, dass es keine Performanceprobleme im Zusammenhang mit der Modemdatenbank gibt. (Dieser Abschnitt muss nach der Anforderungsanalyse nochmal gewaltig überarbeitet werden)

## Aufbau der Arbeit

Das erste Kapitel stellt eine Einleitung in das Thema dar. Es werden Motivation und Aufgabenstellung der Arbeit erläutert. Kapitel 2 beschäftigt sich mit den Grundlagen und stellt bereits existierende Methoden zur Kartendarstellung vor. In Kapitel 3 werden die Anforderungen genauer spezifiziert und das Konzept zur anschließenden Implementierung dargestellt. Im 4. Kapitel geht es um die Implementierung der Anwendung. Die Ergebnisse des Projekts werden in Kapitel 5 dargelegt. Das letzte Kapitel besteht aus einer Zusammenfassung und einem Ausblick auf die Zukunft der Anwendung.

# Grundlagen

In diesem Kapitel werden ein paar grundlegende Begriffe erklärt, die für diese Bachelorarbeit wichtig sind und nicht unbedingt jedem bekannt sind.

## Grundlagen Netzwerktechnik

Es folgen einige wichtige Grundbegriffe aus der Netzwerktechnik.

### SNMP

SNMP heißt ausgeschrieben Simple Network Management Protocol, was so viel wie einfaches Netzwerkverwaltungsprotokoll bedeutet. Der Kern SNMPs ist eine einfache Sammlung an Funktionalitäten (und den Informationen die diese Sammeln), welches Administratoren die Fähigkeiten verleiht den Status SNMP-basierter Geräte zu ändern. Beispielsweise lässt sich mit SNMP ein Interface eines Routers runterfahren oder man kann sich die Geschwindigkeit des Ethernets anzeigen lassen. SNMP ist auch dazu in der Lage die Temperatur eines Switches zu überwachen und zu warnen wenn diese zu heiß wird.

SNMP wird normalerweise mit dem verwalten von Routern in Verbindung gebracht, allerdings kann es auch für viele andere Gerätetypen verwendet werden. Während SNMPs Vorgänger SGMP entwickelt wurde um Internetrouter zu verwalten, kann man mit SNMP auch für Unis Systeme, Windows Systeme, Drucker, Modem Racks, Energiegeneratoren und noch mehr benutzt werden. Jedes Gerät auf dem Software läuft von der sich SNMP Informationen abrufen lassen kann verwalten werden. Dies ist nicht nur auf physische Geräte beschränkt und gilt auch für Software wie Webserver und Datenbanken.[[1]](#footnote-1)

### DOCSIS

„Data Over Cable Service Interface Specification“ oder kurz DOCSIS ist ein Standard, der die Anforderungen für Datenübertragungen in einem Breitbandkabelnetz festlegt. Der wichtigste Anwendungsbereich von DOCSIS besteht in der schnellen Übertragung von Daten über bestehende Kabelfernsehnetze.

## Grundlagen Geografische Informationssysteme (GIS)

Geografische Informationssysteme, Geoinfomationssysteme oder auch Räumliche Informationssystem beschreiben alle dasselbe und werden mit GIS abgekürzt.

Eine genaue Definition für ein GIS ist „Im Mittelpunkt der Geoinformatik stehen mit den Geoinformationssystemen raumbezogene Informationssysteme, die im Gegensatz zu den übrigen Informationssystemen Geoobjekte der realen Welt modellieren und diese in ein digitales Informationssystem abbilden … Das Besondere bei Geoinformationssystemen ist, dass Geoobjekte darüber hinaus Geometrie und Topologie als implizite und untrennbare Bestandteile aufweisen! Die Verarbeitung derartiger raumbezogener Informationen erfordert spezielle Werkzeuge bzw. Funktionen, die von den übrigen Informationssystemen nicht bereitgestellt werden.“[[2]](#footnote-2)

Eine Alternative Definition wäre: „Ein Geoinformationssystem dient der Erfassung, Speicherung, Analyse und Darstellung aller Daten, die einen Teil der Erdoberfläche und die darauf befindlichen technischen und administrativen Einrichtungen sowie geowissenschaftliche, ökonomische und ökologische Gegebenheiten beschreiben.“

Es gibt je nach Quelle noch viele weitere Definitionen die sich in der Regel nur in Kleinigkeiten unterscheiden.

>HSDA-Modell, EVAP-modell<

Eine spezielle Formen des GIS sind Web-GIS und Internet-GIS. Während ein Web GIS ein GIS ist welches das WWW nutzt, reicht es für ein Internet-GIS lediglich irgendeinen Internetdienst zu nutzen.

>die zu entwickelnde Software kann als Teil eines Web-GIS gesehen werden, um selbst als eines zu gelten, werden zu wenige Benutzerinteraktionen mit den Geodaten begangen<

### Geoinformatik

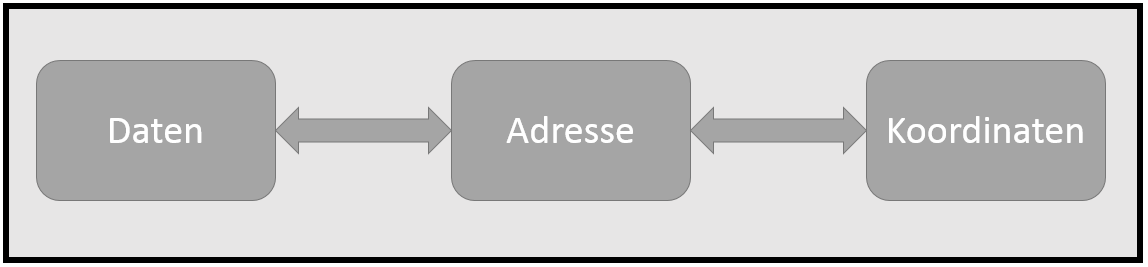
Der Bereich der Geoinformatik wird sehr oft mit den Geoinformationssystemen gleichgesetzt. Die Geoinformatik umfasst jedoch deutlich mehr als nur GIS. Auch Fernerkundung und digitale Bildbearbeitung sind Bestandteile der Geoinformatik, um nur zwei Beispiele zu nennen. Diese Bereiche sind jedoch für das Projekt nicht relevant und werden nicht weiter betrachtet.

### Geokodierung

Geokodierung bzw. Geocoding bedeutet so viel wie, die erdgebundenen geografischen Informationen digital so darzustellen, das man sie mithilfe eines Computers in einer Karte anzeigen kann.

Unter dem Vorgang der Georeferenzierung, Geokodierung, Geotagging oder Verortung versteht man die Zuweisung raumbezogener Informationen, der Georeferenz, zu einem Datensatz.

Für das Projekt besonders wichtig ist hierbei die Adresskodierung. Unter Zuhilfenahme geokodierter Adressen (Punkte die sowohl Koordinaten als auch eine Postanschrift haben) lässt sich ein direkter Raumbezug von Daten herstellen.



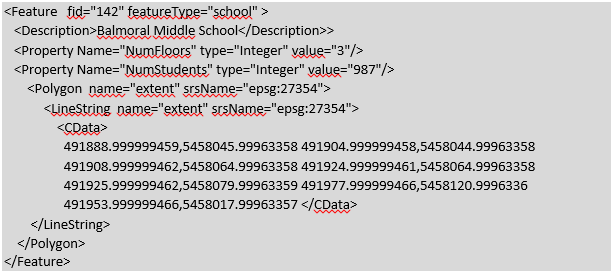
2.I Datenbezug Adresskodierung

Der Hauptnutzen solcher Datensätze besteht im Bereich der Navigationssoftware oder bei Katasterverwaltungen.

### GML und GeoJson

„Innerhalb der Open Geospatial Consortium (OGC) hat das Feature Geometry Model, das als ein abstraktes, implementierungsunabhängiges, konzeptionelles Datenmodell die räumlichen Eigenschaften von Geoobjekten beschreibt, eine zentrale Bedeutung.“[[3]](#footnote-3)

„Die Geography Markup Language (GML), die auf dem Feature Geometry Model beruht, stellt eine XML-basierte Beschreibung von Geodaten dar … Das OGC beschreibt Geoobjekte als sog. Features mit den Bestandteilen "Element Property", die allgemeine Informationen zum Geoobjekt enthalten, und mit den Bestandteilen "Geometrie Property", modelliert durch geometrische Basistypen (Geometrische Primitive) ,,Point", ,,LineString", "LinearRing" oder ,,Polygon" sowie durch komplexere, aggregierte Mengen dieser Objekte.“[[4]](#footnote-4)



2.II Beispiel für ein Feature in GML

Zu GML gibt es ein entsprechendes Gegenstück für die JavaScript Object Notation (JSON), das GeoJSON. Dies ist besonders bei Web-GIS sehr verbreitet, da hier auf Clientseite in der Regel sehr viele auf JS beruhende Frameworks verwendet werden.



2.III Beispiel für ein GeoJSON

In der obigen Abbildung sieht man ein Beispiel für eine GeoJSON. Ein valides GeoJSON hat immer den type FeatureCollection und ein Features Array innerhalb dessen einzelne Features aufgelistet werden. Ein Feature wiederum besteht immer aus einem Geometryfeld im dem die Koordinaten und der Typ des Geoobjects angegeben sind und aus einem Propertiesfeld in beliebige Daten stehen können.

### Netzwerkanalysen

GIS werden oft für Netzwerkanalysen verwendet. Die wichtigsten Gebiete hierbei sind Versorgungsnetze, wie Wasser/Abwasser oder Strom, Verkehrsnetze aller Straßen und Schienen und wie im Fall dieser Bachelorarbeit Telekommunikationsnetze.

Die hier verwendeten Daten lassen leider kaum auf ein Netz im Sinne eines Graphen schließen. Es gibt kaum Informationen über die Kanten. Die Bestandsdaten enthalten lediglich Informationen über die Knoten. Allerdings ließe sich über zugehörigen Modems zu einer CMTS das Versorgungsgebiet approximieren.

>Ich hätte es gerne drin, bin allerdings bisher nicht sonderlich zufrieden mit diesem Abschnitt<

# Konzept

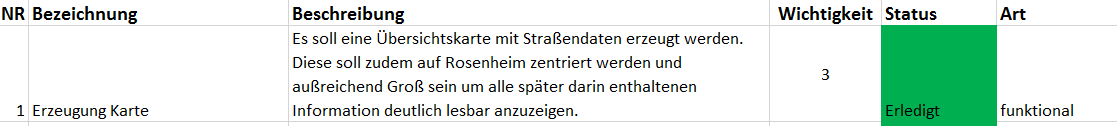
Das Konzept teilt sich in die Anforderungsanalyse, das fachliche Konzept und der Erarbeitung eines Technologiestacks. Desweitern trennen wir beim fachlichen Konzept die Anwendung zwischen der Serverseite und der Clientseite, sowie der Datenspeicherung.

## Anforderungsanalyse

Zu allererst wird mithilfe der von der Komro gegebenen Informationen und Vorgaben eine Anforderungsanalyse erstellt. In dieser werden alle Anforderungen aufgelistet, nach funktionalen und nicht funktionalen unterteilt und anschließend nach ihrer Wichtigkeit bewertet.

* **Muss-Anforderungen** beschreiben die Mindestanforderungen an die Software und umfassen deren wesentliche Merkmale sowie Kernfunktionen, die zwingend zu erfüllen sind. Wegen ihrer hohen Relevanz werden die Muss-Anforderungen dreifach gewichtet.
* **Soll-Anforderungen** beschreiben Merkmale deren Vorhandensein vorteilhaft, aber nicht zwingend erforderlich ist. Sie erweitern den Basisumfang der Software um Funktionalitäten, die einen relevanten Mehrwert bieten, und werden daher zweifach gewichtet.
* **Optionale-Anforderungen** beschreiben Anforderungen an die Software, deren Erfüllung zwar wünschenswert, aber von untergeordneter Bedeutung ist. Aufgrund der niedrigen Relevanz werden die optionalen Anforderungen einfach gewichtet. Diese Anforderungen werden oft in einer späteren Iteration der Software höher priorisiert.

Es folgt eine Beispielhafte Darstellung einer Anforderung. Die Komplette Liste mit allen während dieser Arbeit erfassten und bearbeiteten Anforderungen befindet sich im Anhang.



3.I Beispiel für eine Anforderung

Auf Grundlage dieser Anforderungen und einer ausgiebigen Recherche wurde das Konzept für den Softwareprototypen erstellt. Die Anforderungen wurden im Laufe des Projektes falls notwendig angepasst und erweitert.

## Fachliches Konzept

Das Grundkonzept der Anwendung kann als Web-GIS gesehen werden. Zwar erfüllt die im Laufe dieser Arbeit erstellte Anwendung nicht alle Bedingungen dafür, jedoch in Kombination mit den genutzten APIs schon. Dementsprechend wird die Anwendung als Webanwendung designt. Das heißt wir unterteilen die Applikation in ein Client-Server Produkt. Desweiterem teilt sich die Serverseite noch einmal zwischen Websever, auf dem neben der Seitenauslieferung an den Client auch die Anwendungslogik implementiert wird und dem Map-Server der die Kartendaten an die Clientseite liefert und auch für die Geokodierung zuständig ist.

### Serverseite

Um die zur Verfügung stehenden Adressdaten für eine Kartendarstellung nutzen können, müssen erst noch ein paar Schritte unternommen werden. Als erstes erfolgt eine Zuordnung der einzelnen Modems über ihre (MAC-Adresse?) zu den jeweiligen Kundendaten. Aus diesen ergibt sich die Adresse an der das Modem steht.

Die allermeisten Services zur Kartendarstellung fordern allerdings Geokoordinaten zur Lokalisation. Hier kommt ein so genannter Geocoder zum Einsatz. Dies ist eine Software die mithilfe einer Datenbank Zuordnungen zwischen Adressdaten und Geokoordinaten vornehmen kann. Nominatim ist eine verbreitete und freie Softwarelösung für so etwas. Es gibt auch viele kommerzielle Angebote im Bereich Geocoding. Hier sollte man beachten das Geocoder und der Mapserver denselben Datenstand nutzen. Andernfalls kann es gerade bei Neubauten zu Unregelmäßigkeiten kommen. Besonders wenn man den Geocoder selbst betreibt sollte man Regelmäßig die Kartendaten auf den neusten Stand bringen. Die meisten Anwendungen basieren entweder auf den Daten von Googlemaps oder Openstreetmap.

Nun haben wir zu unseren Modemdaten und der Adresse die entsprechenden Geokoordinaten. Bevor wir diese in unsere Karte einspeisen können müssen wir die Daten noch in ein vom Mapframework akzeptiertes Datenformat übertragen. Die meisten Clientseitigen Mapframeworks basieren auf JS und verwenden deshalb das GeoJSON-Format.

Das umwandeln der Daten in die entsprechende Form ist nicht sonderlich schwierig und man kann sich leicht selbst eine eigene Funktion dafür schreiben. Sollte am Format etwas nicht in Ordnung sein, wirft das Kartenframework einen entsprechenden Fehler und lasst sogar erkennen an welcher Stelle ein Fehler vorliegt.

Das fertige GeoJSON kann nun vom Server bereitgestellt werden um dann von der Clientseite der Software aufgerufen und in die Karte geladen werden. Es gäbe auch Geocoding Lösungen die sich Clientseitig nutzen lassen. Bei der Menge an Anfragen würde dies allerdings zu einer sehr schlechten Performance der Webanwendung führen. Zudem wäre es eine Ressourcenverschwendung bei jedem Aufruf erneut alle Adressen neu zu Koordinaten umzuformen. Zudem kann mit dieser serverseitigen Lösung die Geokodierung mit geringem Aufwand automatisiert werden.

### Clientseite

Die Clientseite wird Standardmäßig als MVC-Pattern (Model View Controller) realisiert. Sprich es existiert eine View die der Anzeige dient, dies ist in der Regel ein HTML-Template. Im Model werden die Daten abgelegt welche eine Verbindung zur View aufweisen und in der Regel von der Serverseite kommen bzw. an diese geschickt werden. Im Controller wird die Clientseitige Logik untergebracht. Hier wird dies konkret in einer JavaScript-Datei umgesetzt.

Mithilfe eines Kartenframeworks wird eine interaktive Karte von Rosenheim erzeugt. Dazu holt sich das das Framework Kartendaten von einem Server des Anbieters.

In diesem Fall werden Restaufrufe nötig um sich das serverseitig erstellte GeoJSON aus 3.1.1 in den Client zu laden. Mithilfe des GeoJSONs werden dann Punkte an die Stelle der Modems in die Karte gesetzt. Dann wird noch eine Sidebar erstellt in der die Modems ebenfalls aufgelistet werden.

Die Modems sollen zudem anhand ihrer aktuellen DOCSIS-Daten entsprechend eingefärbt werden. Um dies zu realisieren wird eine Funktion benötigt die in regelmäßigen Abständen die aktuellen Daten vom Server holt. Auf hierfür wird ein Restaufruf verwendet.

### Datenspeicherung

Ein wichtiger Punkt bei Geoinformationssystemen ist die Datenspeicherung. Hier wäre es natürlich ideal, wenn man Geoobjekte als solches einfach abspeichern könnte. Dies ist indirekt sogar möglich. Je nach Datenbank gibt es Möglichkeiten Geometriedaten als Datentyp zu wählen. Oft kann man auch eine Komplette JSON-datei in einer Datenbank sichern. Die kann jedoch zu schlechter Performance führen, wenn man nur einzelne Features haben möchte. Da jedoch für die zu entwickelnde Anwendung das komplette JSON notwendig ist, sollte dies kein Problem darstellen.

Des Weiteren müssen die vorhandenen Datenbanken angesprochen werden. Da Informationen aus mehreren Datenbanken benötigt werden, wird hier eine durch einen Join erzeugte View hilfreich sein. Da für die erstellten Json-Dateien eine eigene Datenbank erstellt wird ist lediglich ein Lesezugriff auf die Bestandsdaten notwendig.

## Technologiestack

Es gibt für das Programmieren einer Webapplikation viele Wege die ans Ziel führen. Im Folgenden werden Technologien für die verschiedenen Teilbereiche der Software vorgestellt und in Bezug auf den Anwendungsfall bewertet.



3.II wichtigste Technologien im Überblick

### Webstack

Ein voller Technologiestack für eine Webanwendung umfasst in der Regel immer die Standardsprachen HTML, CSS, JavaScript für die Clientseite und eine serverseitige Programmiersprache mit zugehörigen Webframework. Dazu kommen dann je nach Anwendungsfall noch weitere Frameworks und Bibliotheken.

Als Programmiersprache wurde in diesem Fall Python über Beispielsweise PHP oder ASP.Net gewählt. Diese findet bei der Komro schon oft Verwendung und ist aufgrund der Schreibweisen eine sehr angenehme und leicht zu lernende Sprache. Die am weitest verbreiteten Webframeworks für Python sind Flask und Django. Flask ist das leichtgewichtigere der beiden und bringt trotzdem alle notwendigen Funktionalitäten mit, daher ist es für diese Anwendung zu bevorzugen. Es bringt sogar einen eigenen Webserver mit. Zur Paketverwaltung wird pip verwendet. Der Name „pip“ ist ein rekursives Akronym und steht für „pip installs packages“.

Um die doch recht komplexe Clientseite leichter Programmieren zu können wird statt nur nativem JavaScript auch noch das Framework Qjuery verwendet. Neben den andern Vorteilen die es bietet wird es hier vor allem zum Manipulieren von HTML-Dokumenten und der Handhabung von Events eingesetzt.

Es gibt unzählige Datenbanken die sich grundsätzlich für den Einsatz in einer Webapplikation eignen. Genauer betrachtet wurden MySQL, MongoDB und PostgreSQL. PostgreSQL, auch Postgres genannt eignet sich hierbei am besten. Es verfügt sowohl über die Möglichkeit direkt Jsondateien abzuspeichern als auch über Geometrie-Datentypen mit dessen Hilfe sich ganz einfach Geoobjekte verwalten lassen. Zudem befinden sich die Bestandsdaten auch in Postgres-Datenbanken, was die Verwendung dieser erleichtert.

### Geoinformatik

Um effektiv mit Geoinformationen arbeiten und diese darstellen zu können, werden einige Frameworks benötigt.

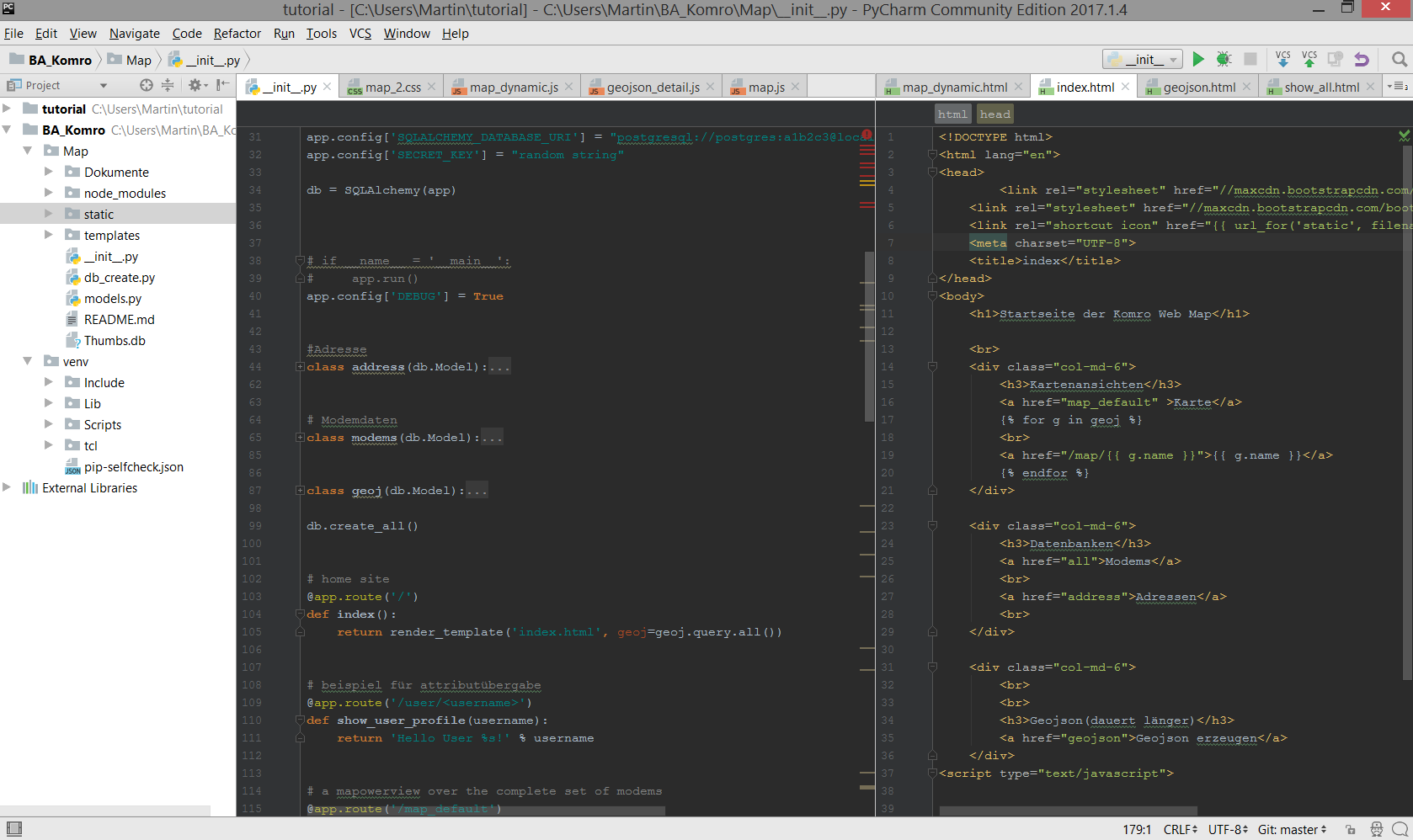
Geopy wird Serverseitig vor allem für die Geokodierung verwendet. Mit Geopy kann mit nur wenig Aufwand zwischen verschiedenen Geocode-APIs gewechselt werden. Es ist ein sehr leichtgewichtiges Framework extra für Python.

Auch für die clientseitige Frontenddarstellung der Kartendaten ist ein entsprechendes Framework notwendig. Die Auswahl des richtigen wird durch die große Anzahl verschiedener sehr erschwert. Zunächst werden daher erst einmal alle Technologien verworfen die nicht Open-Source bzw. nicht für den Kommerziellen Einsatz geeignet sind ohne dafür Abgaben an die Ersteller zu leisten. Daher fliegen alle GoogleMaps basierte Ansätze gleich aus dem Rennen. Diese sind zwar nicht zwingend kostenpflichtig es ist aber nur schwer einzuschätzen ob die frei verfügbaren Funktionalitäten ausreichend sind. Es wird deshalb ein Ansatz mit Open-Street-Map gewählt, da es neben Google die genausten Kartendaten bietet.

Hier wurde aus allen verschiedenen Technologien Mapbox ausgewählt. Es liefert die gewünschten Kartendaten. Sprich auch wenn man den Anwendungsserver über ein Internes Netzwerk erreicht, wird dennoch zwingend eine Internetverbindung vorausgesetzt. Mapbox bietet ein Vielzahl an vorgefertigter Interaktionsmöglichkeiten mit der Karte, beispielsweise Zoom und Vollbildmodus. Zudem enthält es Ansätze für eigene Funktionalität. Wie etwa Events für Mausinteraktionen. Außerdem bringt es noch eine eigene Geocode-API mit. Diese kann in der frühen Entwicklungszeit zum Testen recht praktisch sein. Zudem bietet Mapbox neben API und dem rendern von Karten auf noch viele andere Möglichkeiten die zwar nicht für diese Arbeit benötigt werden, aber in Zukunft möglicherweise für die Komro von Nutzen sein werden. Man Beispielsweise kann seine eigenen Kartenstyles erstellen, verwalten und für Benutzer freigeben. Es gibt auch gesonderte APIs für Mobile Entwicklung.

### Sonstiges

Durch den recht komplexen Programmcode und verschiedenen Technologien die in diesem Projekt zusammenarbeiten müssen kommt man um eine gute IDE nicht herum. Deshalb wurde als Entwicklungsumgebung PyCharm von der Firma Jetbrains verwendet. Diese bietet alle notwendigen Vorzüge einer modernen IDE, unterstützt alle verwendeten Programmiersprachen und erfordert aufgrund der Vorerfahrung mit anderen IDEs des Hersteller (Webstorm, Phpstorm, …) keine Einarbeitungszeit.



3.III PyCharm IDE Benutzeroberfläche

Zudem wird zum Darstellen der Anwendung noch ein Browser benötigt. Hierbei sollte dieser am besten über eingebaute Entwicklerwerkzeuge besitzen. Das ist insbesondere für das Debuggen der Clientseite notwendig. Als Standardbrowser für die Entwicklung wird Google Chrome verwendet. Dieser erfüllt alle Anforderungen und ist recht weit verbreitet. Da die spätere Anwendung der Software auch in anderen Browser erfolgen können soll, werden auch Tests in Beispielsweise dem Mozilla Firefox vollzogen.

Später soll die Software im Live-Betrieb auf einen Linux-Server installiert werden. Hier wird empfohlen noch einen Webserver als Proxy-Server davor zu setzen.

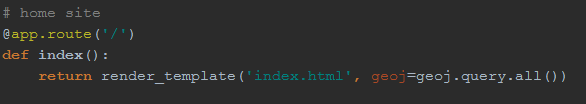
Zusätzlich wurde noch Git zur Versionskontrolle verwendet. Git ist nicht nur nützlich wenn mehrere Personen mit möglichst wenig Abhängigkeit voneinander an einem Projekt arbeiten wollen, mithilfe der Versionierung lässt sich auch ganz einfach bei eingeschlichenen Bug ein funktionierender Zustand wieder herstellen. Zudem diente es während dieser Arbeit als Backup für Code und Dokumente falls es Probleme mit dem Arbeitsrechner geben sollte.

# Implementierung

In diesem Kapitel wird die konkrete Implementierung der Anwendung behandelt. Dabei wird sich oft auf die eingesetzten Technologien bezogen.

## Webserver und angewandte Geokodierung

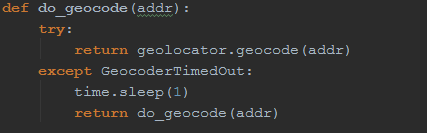
Als erstes wird die Serverkomponente der Anwendung betrachtet. Zuallererst werden hier die allgemeinen Einstellungen der Webapplikation und das Routing vorgenommen. Dies wird durch das Webframework Flask vereinfacht.



4.I Routing mit Flask

Des Weiteren werden hier auf die Datenverwaltung mithilfe von einer Postgresdatenbank und der SQL-Alchemy Bibliothek vorgenommen. Neben den bestehenden Datenbanken der Komro wird noch zusätzlich eine Datenbank angelegt in welcher die später erzeugten Geojson-Dateinen abgelegt werden können. Diese besteht lediglich aus einer ID, für Verwaltungszwecke, der eindeutigen Bezeichnung der Datei und der Datei selbst. Postgres bietet hierfür extra einen eigenen Typ zur Speicherung von Json-Daten an.

Auch die Geokodierung wird auf dem Server gehandhabt, da ein Clientlösung hierfür zwar möglich wäre, allerdings Performancetechnisch nicht sehr optimal ist. Um die Adressen geokodieren zu können wird die API von Nominatim verwendet. Dieser muss lediglich eine Adresse in Form eines Strings übergeben werden und sie liefert ein Objekt zurück. Man kann verschiedene Optionen verwenden um den Inhalt bzw. das Format des Objekts zu beeinflussen. Dieses enthält neben einem vollständigen Adressstring unter anderem die zugehörigen Koordinaten. Leider ist es nicht möglich der API mehrere Adressen auf einmal zu übergeben. Daher muss man hier selbst die Geokodierung cachen und häufig auftretende Timeout-Fehler mit einen Try-Catch-Block abfangen.



4.II Aufruf der Geokodierung mit Timeout Fehlerbehandlung

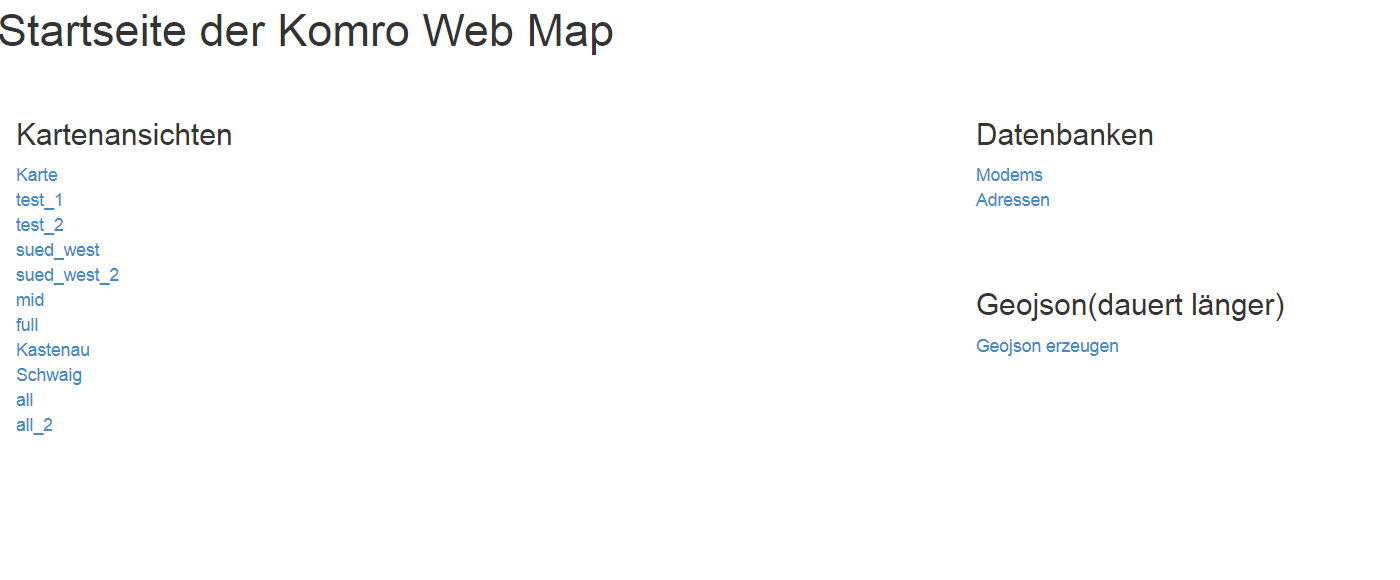
Nominatim bietet als Open-Source-Software die Möglichkeit sich einen eigenen Server einzurichten und nicht die Öffentlichen APIs nutzen zu müssen. Bei ausreichenden Ressourcen bzw. nur geringer Auslastung kommt es nur noch selten zu Timeouts. Bei größeren Mengen an zu kodierenden Adressen kann es sonst zu erheblichen Performanceproblemen kommen.

Da die von der Nominatim-API zurückgegebenen Objekte noch nicht dem Json-Format entsprechen muss man dies noch manuell umwandeln. Bei dieser Gelegenheit können auch die Propertys mit entsprechenden Daten aus der Modemdatenbank befüllt werden. Sobald all dies abgeschlossen ist kann die Json-Datei in der Datenbank abgelegt werden. Hier muss noch eine Restschnittstelle geschaffen werden um einen Zugriff vom Client aus zu ermöglichen.

## Startseite und Geojson-View

Bevor es an die eigentlichen Kartenansichten geht, benötigt es noch ein paar andere Views. Als erstes eine Startseite der Webapplikation. In dieser werden alle verfügbaren Karten-Views als Links aufgelistet und es gibt eine Weiterleitung zur Geojson-View. Die Links der Karten-Views werden als dynamische Liste generiert. Diese Basiert auf den in der Datenbank bestehenden Geojson-Daten. Diese Links sind von der URL her wie folgt aufgebaut: …/map/<name> Hierbei steht die Variable <name> für den Namen der entsprechenden Geojson-Datei.

<Screenshot der fertigen Startseite hier einfügen.>



4.III Startseite

In der Geojson-View werden neben einer Auflistung bereits der bestehenden Geojson-Dateien in der Datenbank Optionen geboten diese Daten auf den neusten Stand zu bringen, zu löschen oder neue anzulegen. Eine manuelle Änderung der Daten innerhalb der Geojsons ist über dieses Webinterface nicht vorgesehen.

<ausführliche beschreiben sobald die Implementierung wirklich steht>

## Kartendarstellung

Der wichtigste Kern der Anwendung die Darstellung der Karte. Dies wird mithilfe des Karten-Frameworks Mapbox bewerkstelligt. Hier werden sich von Mapbox die Kartendaten über das Internet geholt um diese dann anzugeigen. Mapbox bietet neben der Möglichkeit eigene Kartenstiele zu erstellen auch verschiedenen Standardmöglichkeiten an. Im konkreten Fall werden die Standardstiele „streets-v10“, sowie „satelite-streets-v10“ benützt. Neben dem Stiel werden auch ein Punkt auf dem sich die Karte zentrieren soll und die Ausgangsstufe des Zooms bei der Initialisierung der Karte angegeben. Um die Karte darzustellen muss im HTML noch ein Container mit der ID der Karte erstellt werden. Um eine fehlerfreie Darstellung zu garantieren muss mittels CSS die genaue Position und Größe der Karte angegeben werden.

Aufgrund der Open-Source-Lizenz enthält die Karte eine so genannte „AttributionControll“, in der das Mapbox und Open-Street-Map Copyright enthalten sind, sowie ein Link auf eine Feedback Seite. Diese lässt sich an eine beliebige Position innerhalb der Karte verschieben. Durch die Verwendung der Standardstiele kommt die Karte auch noch mit einem integrierten Mapbox-Logos. Dieses ist leider fest integriert und lässt sich daher nicht verschieben.

Weiterhin werden einige Standardkontrollmöglichkeiten für die Karte erstellt. Hierzu zählen Zoom per Mausrad oder den dafür erstellten Knöpfen, eine Ausrichtung der Karte nach Norden, ein Vollbild Modus sowie ein umschalten zwischen einer Straßenkarte und Satellitendaten. Während die anderen Funktionen Standardmäßig in einer Mapbox-Karte enthalten sind und in der Regel nur noch aktiv geschaltet werden müssen, muss die Implementierung für den Kartenwechsel selbst vorgenommen werden. Hierbei liegt die eigentliche Schwierigkeit darin, dass beim Wechsel der Kartenbasisdaten auch der Layer mit den Modems verworfen wird. Dieser muss daher erneut gezeichnet werden.

4.IV Mapbox-Karte mit Kontrollfunktionen

Um die Positionsdaten der Modems clientseitig zu erhalten, wird mithilfe eines Restaufrufs, die entsprechende Json-Datei aus der Datenbank ausgelesen. Diese Json-Datei wird einer entsprechenden Funktion des Kartenframeworks übergeben, welches dann Punkte an die Positionen der Modems setzt. Um die richtige Json-Datei zu erhalten, kann der Name aus der URL ausgelesen werden.

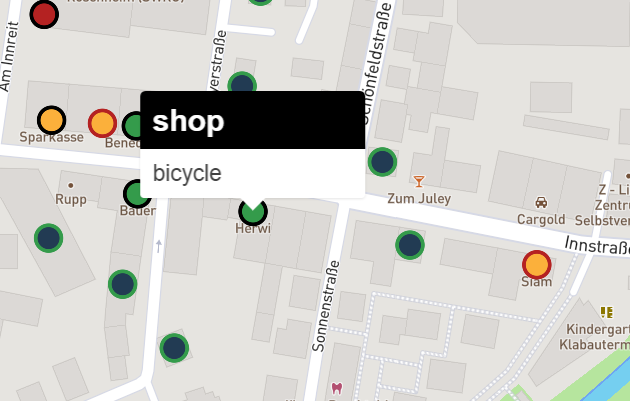
Die Punkte, welche die Modems Darstellen, werden entsprechend der zugehörigen Messdaten eingefärbt. Hierbei kann man bei einer weiteren Funktion des Kartenframeworks den Typ „categorical“ angeben. Dann werden noch ein „property“ und ein Array von „stops“ angegeben. Es wird dann bei jedem Modem der Wert des jeweiligen „propertys“ mit den in den „stops“ angegebenen werten verglichen und entsprechend den dort zugeordneten Farbwerten eingefärbt.



4.V Beispiel für einen categorical Typ

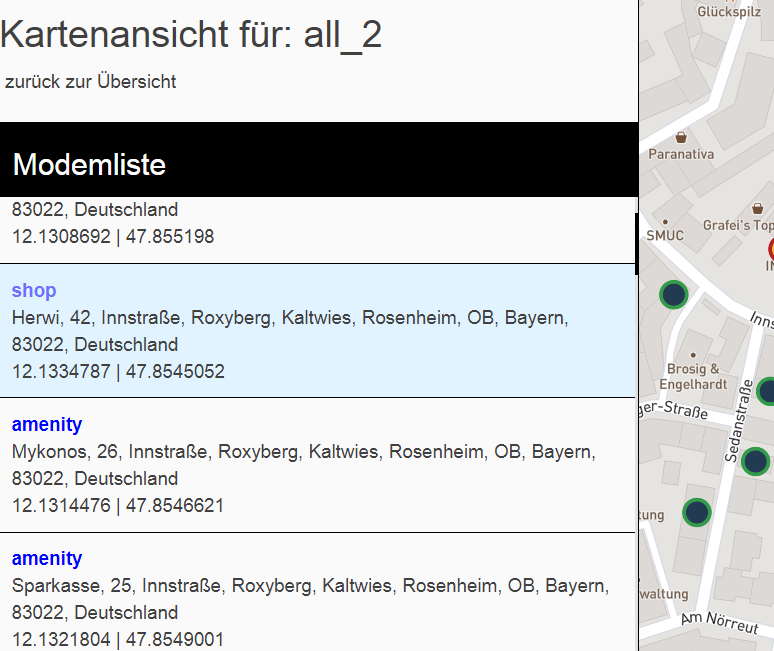
Die Größe der Punkte kann auch auf „stops“ beruhend eingestellt werden. Hierbei wird die Größe der Punkte anhand des Zoomlevels festgemacht.

Diese Messdaten werden beim Klick auf den Kartenpunkt innerhalb eines Popups dargestellt. Auch dieses Popup wird entsprechend Farblich markiert.



4.VI Darstellung eines offenen Popups

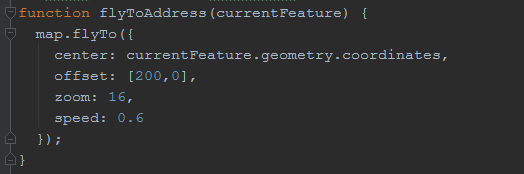
Die Karten-View teilt sich inhaltlich in eine Kartendarstellung sowie eine Sidebar auf. Diese stehen im Verhältnis von 2/3 Karte zu 1/3 Sidebar. Um weiterhin einen funktionierenden Vollbildmodus zu gewährleisten muss die Karte über den kompletten Seitenbereich gehen. Man legt lediglich die Sidebar eine Ebene über die Karte. Die Sidebar enthält im oberen Teil eine Überschrift die Auskunft über den aktuellen Datensatz gibt, sowie einen Navigationslink mit dessen Hilfe man zurück zur Startseite kommt.



4.VII Sidebar

Im unteren Teil befindet sich eine Auflistung der in der Karte angezeigten Modems. Zu den Modems sind einige statische Informationen wie die Bezeichnung und die zugehörige Adresse aufgeführt. Diese werden auch mithilfe des Geojsons befüllt. Mit einen Klick auf die Bezeichnung wird das Element in der Liste aktiv gesetzt und hervorgehoben. Dies führt dazu, dass sich die Karte auf die Zugehörige Position zentriert und sich das Popup des Punktes öffnet. Selbiges Passiert auch beim Klick auf einen der Punkte, allerdings wird hier noch zusätzlich die Liste der Modems auf den entsprechenden Eintrag gescrollt. Hierfür die die scrollTop() Funktion verwendet. Diese würde eigentlich an das obere Ende der Liste scrollen. Allerdings kann man ein Offset als Parameter übergeben. Hierfür nimmt man das entsprechende Offset des ausgewählten Listenelements.

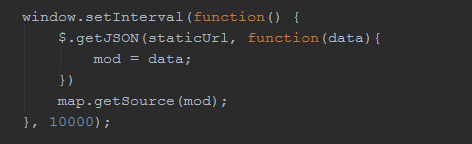
Für die Zentrierung der Karte bietet Mapbox die eine flyTo() Funktion an. Dieser kann neben dem eigentlichen Zielpunkt auch noch werte für einen Offset und die Zoomtiefe mitgegeben werden. Das Offset wird so eingestellt, dass es sich so weit nach rechts verschiebt, dass das Zentrum in der Mitte des sichtbaren Kartenausschnitts liegt. Die Geschwindigkeit der Animation kann ebenfalls eingestellt werden. Diese sollte entsprechend der Distanz zwischen den einzelnen Modems eher langsamer als in der Standardeinstellung gewählt werden, da es sonst sehr ruckelt.



4.VIII Codebeispiel für die flyTo() Funktion

Die Popups können entweder mit einem Klick auf die Karte, außerhalb des Popups, geschlossen werden. Beim Klick auf ein anderes Modem wird das bisherige Popup geschlossen bevor das neue geöffnet wird. Leider ließ sich für dieses Verhalten das Standardverhalten des Popups mit dem „closeOnClick: true“ Statement nicht realisieren. Rein auf die Karte bezogen würde es funktionieren, allerdings bezieht die die Sidebar nicht mit ein und führte so zu einem nicht gewünschten verhalten. Demnach muss „closeOnClick: false“ verwendet werden und das Schließen der Popups selbst implementiert werden. Eine Kombination aus „closeOnClick: true“ und einer eigenen Implementierung nur für die Sidebar führt zu Fehlermeldungen und anschließend zu einem Fehlerhaften verhalten der Funktionalität.

Um die Daten der Modems zu aktualisieren ohne die Seite neu laden zu müssen wird eine Funktion geschrieben, die alle 10 Sekunden einen Restaufruf welcher das aktuellste GeoJSON zurückliefert. Dieses wird dann in die Karte geladen, woraufhin Mapbox für eine Überschreibung der Modempunkte sorgt. Die Dauer zwischen den Restaufrufen kann sehr einfach umgestellt werden, falls sich die Karte öfter aktualisieren soll.



4.IX Funktion zum Updaten der Kartendaten unter Verwendung von Jquery

Diese Implementierung (vgl. Abb. 4.IX) führt dazu, dass auch die Modemliste in der Sidebar geupdatet wird, da diese sich auf die Variable „mod“ bezieht.

## Styling

Am Ende der Implementierungsphase rückt langsam das Design der Frontend-Darstellung der Anwendung in den Vordergrund. Hierbei ist das Cassadian Style Sheet, kurz CSS, das wichtigste Hilfsmittel. Damit kann jedes HTML-Element auf der Gestaltungsebene verändert werden. Angefangen bei der Positionierung auf der Seite über Farb- und Textgestaltung bis hin zum Festlegen der Darstellungsebene.

Eine weitere wichtige Funktionalität ist das so genannte Responsive Design. Dabei geht es darum gezielte Designänderungen je nach Darstellungsgröße der Seite zu erwirken. Die ist vor allem beim Einsatz von mobilen Geräten relevant. Es bietet sich oft an diese Arbeit an ein schon fertiges CSS Beispielsweise von Twitter Bootstrap auszulagern. Dies kann einfach wie ein eigenes CSS in das HTML-Template eingebunden werden. Dann muss man lediglich den HTML-Elementen Klassen aus dem CSS zuweisen und erhält das gewünschte Ergebnis. Die Anwendung ist grundsätzlich nicht für Mobilgeräte konzipiert, da auch mit Responsive Design auf einen Mobiltelefon einfach zu wenig Platz für eine ordentliche Darstellung vorhanden ist. Deshalb wird sich hierbei lediglich auf größere Tablets/Laptops und PC-Bildschirme fokussiert.

Bei der Farbgestaltung der Modems wurde sich an ein Klassisches Ampel Schema gehalten. Grün für alles in Ordnung, Gelb für es gibt Probleme und Rot für starke Probleme/Totalausfall. Eine derartig einfache und bekannte Farbunterteilung führt zu einem sofortigen Verständnis für die Karte. Desweiterem müsste man bei einem komplexeren Schema eine Legende für die Farben miteinbauen, welche wiederum Platz verbraucht und damit zusätzlich die Übersichtlichkeit einschränkt.

Abschließend werden noch Anpassungen am Cursor vorgenommen. Dieser soll, je nach dem über welchem Element er sich befindet eine andere Form haben. Zum Vergleich normalerweise sieht dieser in einer Browseranwendung wie ein normaler Zeiger bzw. Pfeil aus. Standardmäßig ändert sich dieser wenn man ihn innerhalb der Karte bewegt allerdings zu einer Hand. Dies deutet daraufhin, dass bei gedrückter Maustaste sich die Karte bewegen lässt. Das ist soweit ganz gut, nur soll der Cursor sowohl beim Bewegen über einen Modempunkt als auch beim Bewegen über den Bezeichner in der Modemliste in eine Hand mit ausgestrecktem Zeigefinder verändern. Dies ist der Standardzeiger für einen Link und deutet auf eine Interaktionsmöglichkeit hin. Also nutzen wir Eventhandler um das Bewegen der Maus über einen Modempunkt zu fangen und entsprechend dann die Form des Cursors zu verändern.

## Tests

In diesem Kapitel wird das Vorgehen beim Test der Software geschildert und die Notwenigkeit hierfür erläutert. Im ersten Abschnitt werden die Tests vorgestellt, welche während der Entwicklung durchgeführt wurden. Der zweite Abschnitt berichtet über die Praxistests, welche nach Abschluss der Implementierung durchgeführt wurden.

Ein weiteres wichtiges Sicherheitsmerkmal einer Webanwendung wäre die Validierung von Eingabedaten der Nutzer. Dieses Thema wird in dieser Arbeit jedoch vernachlässigt, da die Nutzer bisher noch keine Möglichkeit hat Informationen einzugeben und diese an den Server zu senden.

### Tests während der Entwicklung

Damit man auch sicher sein kann, dass ein Einzelteil der Software auch genau das tut, was von ihm erwartet wird sind Tests notwendig. Bereits während der Entwicklung ist es daher nötig mit Modultests auch Unittests genannt zu arbeiten. Hierbei werden einzelne Module auf ihre korrekte Funktionalität überprüft. Diese Tests können in Python mit PyUnit durchgeführt werden.

Auf der Frontendseite ist es leider nicht ganz so einfach. Visuelle Tests machen erst wirklich Sinn sobald sich am visuellen nichts mehr ändert. Deshalb sind diese Tests oft erst ganz am Ende der Implementierung möglich. Aber auch dann muss sehr stark darauf geachtet werden, ob man auch wirklich den eigenen Code testet und nicht nur die Funktionalitäten des entsprechenden Frameworks.

Ein weiterer wichtiger Punkt der getestet werden muss ist die Performance. Dies sollte auch schon früh in der Entwicklungsphase passieren, da man dann noch leichter einen anderen Ansatz oder gar eine andere Technologie wählen kann. Beim Erstellen der Kartenapplikation hätte es auch die Möglichkeit gegeben die Geokodierung der Adressen clientseitig ausführen zu lassen. Durch die Geschwindigkeitslimitierungen bei nicht kommerziellen APIs hätte das jedoch zu unvertretbaren Ladezeiten geführt und wurde als Ansatz verworfen. Selbst bei einer kommerziellen oder eigenen Geokodierung mit hoher Geschwindigkeit hätte es bei einer größeren Anzahl an Modems zu Problemen geführt.

### Praxistests

Gerade bei einer Webanwendung, die von vielen unterschiedlichen Endgeräten aus genutzt werden soll, sind Praxistests von großer Wichtigkeit. Mit deren Hilfe lassen sich vor allem Darstellungs- und Performanceprobleme sehr leicht feststellen. Diese lassen sich jedoch erst durchführen wenn die Software auf einem geeigneten Server läuft und damit von außen zugänglich ist. Dies erfolgt erst nach Abschluss dieser Bachelorarbeit, da lediglich ein erster Prototyp entwickelt wurde.

Dennoch wurde ein Praxistest im kleinen Rahmen auf verschiedenen Geräten und Browsern durchgeführt und aufgetretene Probleme behoben.

# Ergebnisse

In diesem Kapitel werden die Ergebnisse des Projekts, die aufgetretenen Probleme, sowie das Feedback des Auftraggebers dargestellt.

## Endresultat des Projekts

Im Laufe dieser Arbeit wurde neben diesem Dokument auch die beschriebene Software erstellt. Diese enthält die in der Anforderungsliste festgehaltene Funktionalität und wurde der Komro in Form eines Git-Repositorys übergeben. Auch ein Großteil der zusätzlich entstandenen Dokumente liegt dem Code bei.

## Probleme beim Erstellen der Software

Problematisch im Laufe der Entwicklung war vor allem der Cache des Browsers. Hierfür gibt es mit einem Tastenkürzel (STRG + F5 je nach Browser auch anders) einen Neuladen bei dem alle Daten neu angefordert werden.

Ein beim Styling aufgetretenes Problem war die designtechnische Anpassung der Scroll-Leiste. Da dies zwar im Chrome Browser von Google funktioniert jedoch nicht im Mozilla Firefox. Die würde sich möglicherweise mithilfe eines Plug-Ins ähnlich anpassen lassen, allerdings wäre das Ergebnis immer leicht anders. Da es nur ein kleiner visueller Unterschied ist und keinerlei Auswirkung auf die Verwendbarkeit der Software hat kann dieses Problem vernachlässigt werden.

## Stellungnahme des Auftraggebers

Hier kommt eine hoffentlich sehr positive Rückmeldung von Seiten der Komro. Es wird dargelegt was ihnen mein Projekt gebracht hat und warum sie darüber derart froh sind.

>Zum Ende könnte man hier ein derartiger Satz stehen<

Ein erster Schritt ist getan. Das System muss nun in der Praxis getestet, gegebenenfalls verbessert und um zusätzliche Funktionen erweitert werden.

# Zusammenfassung und Ausblick

In diesem Kapitel werden die bisherigen zusammengefasst. Zudem wird über weitere geplante Features und die Integration der Software in ein bestehendes System berichtet.

## Zusammenfassung

Im Rahmen dieser Arbeit wurden das Design und die Implementierung einer Webanwendung für die Kartendarstellung von Docsis basierten Modemdaten entwickelt.

Zu Beginn wurde ein Konzept für die Anwendung erstellt. Hierfür werden zunächst die Anforderungen an das System analysiert. Aus diesen ergibt sich recht schnell, dass es eine Art Webbasiertes Geoinformationssystem werden soll. Während es fachlich relativ eindeutige Vorgaben gibt muss bei der Technologiefindung sehr viel ausprobiert und getestet werden. Nicht nur bei der Wahl der Serverseitig eingesetzten Programmiersprachen und Frameworks, sondern auch bei den Frameworks und APIs für die Geokodierung und Kartendaten gibt es sehr viel Auswahl.

Bei der Implementierung liegt der Hauptfokus vor allem an der clientseitigen Darstellung. Insbesondere die Darstellung der Modems innerhalb der Karte und der Liste stehen dabei im Vordergrund. <> Einen anderen wichtigen Teilbereich stellt die serverseitige Geokodierung dar. Hierbei steht die erfolgreiche Zuordnung von Adresse zu Koordinaten sowie das formatieren der Daten in ein Jsonformat im Mittelpunkt.

Dann folgen schließlich noch das Styling sowie das Testing. <>

## Künftige Features

Die Webanwendung ist zwar funktionell könnte allerdings noch um einige Features erweitert werden.

Die Geokodierung der Daten wird bisher noch manuell ausgelöst. Dieser Vorgang könnte automatisiert werden und die Daten zu festen Uhrzeiten auf den neusten Stand gebracht werden.

Eine weitere Möglichkeit die Anwendung zu erweitern wäre eine Optimierung des Designs für mobile Geräte. Für kleinere Tablets und größere Smartphones könnte man Beispielsweise aus der Sidebar ein Klappmenü machen.

Für die Modemliste könnten gegebenenfalls auch Filter oder Sortiermöglichkeiten sehr praktisch sein. So kommt man auch bei einer längeren Liste bzw. größeren Anzahl an Modems schnell an die gesuchten Daten.

Es steht für die Anbindung richtiger Live-Daten eine Schnittstelle bereit. Es existiert zudem ein Programm welches über SNMP die Modemdaten ausliest und in eine Datenbank speichert. Die könnte man so erweitern, damit dieser Code ebenfalls die aktuellen Daten in das entsprechende GeoJson schreibt.

Der über SNMP erfolgende Aufruf der Modemdaten kann mit einigem Aufwand parallelisiert werden. Dadurch würde ein schnellerer Durchlauf durch alle Modems möglich und man könnte die Daten deutlich aktueller halten.

Eine Möglichkeit um die Performance der Clientseite zu erhöhen ist es die Größe der Map einzuschränken. Hierfür können beim Kartenaufruf so genannte Bonds verwendet werden. Mit diesen kann man zwei Grenzpunkte in Form von Koordinaten angeben.

## Integration/Zusammenlegen mit anderen Systemen

Beim Erstellen der Anwendung wurde insbesondere bei der Technologiewahl darauf geachtet, dass sich diese möglichst einfach in die bestehende Softwareumgebung der Komro integrieren lässt. Es besteht bereits eine Applikation mit der Live-Daten des Rauschabstands dargestellt werden können.

# A Erstes Kapitel des Anhangs

Tabellen, Abbildungen und Ähnliches, die Sie nicht direkt in der Arbeit unterbringen möchten.

Wenn Sie keinen Anhang benötigen, dann bitte einfach rausnehmen.

# Literaturverzeichnis

**Gedruckte Quelle:**

de Lange, Norbert: Geoinformatik in Theorie und Praxis, 3. Auflage. Berlin 2013(eBook)

Mauro, Douglas R.: Essential SNMP, 2nd Edition. Sebastopol, Calif. 2005

**Website:**  
Nachname Autor, Vorname Autor: Name der Quelle - Untertitel der Quelle. Link (Zugriff am Datum)

Beispiele:

Faulhaber, Leopold: Alternatives Wörterbuch – Kreditor. <http://www.awb1.ch/dat/k/kreditor.php> (Zugriff am 10.04.2012)

N.N.: Handelsgesetzbuch. <http://www.gesetze-im-internet.de/bundesrecht/hgb/gesamt.pdf> (Zugriff am 06.04.2012)  
(Achtung Gesetze haben keinen Autor!)

1. Essential SNMP, Kapitel 1.1 frei übersetzt [↑](#footnote-ref-1)
2. S. 337 [↑](#footnote-ref-2)
3. Norbert de Lange, Seite 254 [↑](#footnote-ref-3)
4. ^ S. 243 [↑](#footnote-ref-4)