# OpenStreetMap-in-a-Box Version 2

## freie Karten-Webservices für Desktops und Mobiles,

## Teil B




| | |
|---|---|
| **Projekt:** | OpenStreetMap-in-a-Box (2) Teil B |
| **Autor:** | Marco Busarello, HSR, Abteilung Informatik |
| **Firma:** | University of Applied Sciences Rapperswil |
| **Abteilung:** | Informatik |
| **Betreuer:** | Prof. Stefan Keller, HSR, Abt. Informatik |
| **Externer Betreuer:** | Reto Senn, bitforge AG Ltd., Rapperswil |

# Inhaltsverzeichnis

# Abbildungsverzeichnis



# Tabellenverzeichnis

# 1 Abstract


## Beschreibung

Das Ziel dieser Arbeit ist OpenStreetMap-in-a-Box, welches in einem Vorgängerprojekt entwickelt wurde, verschiedenen Verbesserungen zu unterziehen wie auch neue Anwendungsmöglichkeiten dieser Software aufzuzeigen.

Die Arbeit wurde in drei Teile unterteilt:

1. Verbesserung der Kartendarstellung:
Im ersten Teil wird die Kartendarstellung, also das Rendering der in OSM-in-a-Box vorhandenen Geo-Daten, überarbeitet und verbessert.

2. Webseite:
Es wird eine Webseite erstellt, welche die im ersten Teil gemachten Verbesserungen an der Kartengrafik zeigt. Mit Hilfe des von OSM-in-a-Box konfigurierten Geoservers wird eine Suchfunktion entwickelt und auf der Webseite demonstriert.

3. Android Applikation:
Im dritten Teil der Arbeit wird die verbesserte Version von OSM-in-a-Box genutzt, um eine lauffähige Mobile-Applikation auf Basis des Betriebssystems Android entwickelt. Die Software bietet dem Benutzer in etwa die gleichen Möglichkeiten wie die zuvor implementierte Webseite, also die Darstellung der Karte und eine entsprechende Suchfunktion und zusätzlich noch die Darstellung des eigenen Standorts über GPS.

## Anforderungen

Folgende Anforderungen sind wichtig:

- Das Rendering der Karte soll einem hohen Standard entsprechen. Bei der Optimierung dienen die Karten von OSM und Google Maps als Referenz. Der Virtuelle Server muss so konfiguriert werden, dass das entwickelte Rendering auch entsprechend dargestellt werden kann. Insbesondere die Darstellung der Schriften muss hier beachtet werden.

- Die Webseite soll neben der sauberen Darstellung der Karte eine gut nutzbare Suchfunktion bieten. Die Webseite ist eine Art „Showcase“ für OSM-in-a-Box.

- Der Suchalgorithmus soll performant sein und die Suchresultate sollen in sinnvoller strukturierter Reihenfolge zurückgeliefert werden.

- Die Android Software soll dem Benutzer in etwa die gleichen Möglichkeiten wie die zuvor implementierte Webseite bieten. Zusätzlich soll noch die Darstellung des eigenen Standorts über GPS angezeigt werden können.

## Lösungen

Die für die Grafikkonfiguration nötigen Dateien (Styled Layer Descriptor und Symbology Encoding, SLD) sind grösstenteils von Grund auf neu implementiert und teilweise sinnvoll angepasst worden. Der Virtuelle Server wurde neu konfiguriert, u.a. dass die vorgegebenen Schriften richtig dargestellt werden können.
Die Suchfunktion nutzt den Webservice des Geoservers (sog. Web Feature Server). Die Anfrage liefert ein GeoJSON-Objekt mit allen Suchresultaten zurück und wird mit Hilfe der Javascript-Library JQuery ausgewertet und auf der Webseite dargestellt. Für die Anfrage wurden Views in der PostgreSQL Datenbank entwickelt.

Die Android Applikation nutzt für die Darstellung und Bedienung der Karte eine Java Library von Nutiteq. Mit Hilfe dieser Library können die Grundfunktionalitäten der Karte relativ einfach implementiert werden. Für die Suchfunktion sind eigene Funktionen für die Anfrage an den WFS des Geoservers und für die Auswertung und Darstellung des Resultates mplementiert worden. Am Ende der Arbeit liegt ein lauffähiger Prototyp vor. Der Prototyp demonstriert die Möglichkeiten der Nutzung von OSM-in-a-Box im Bereich der Mobilkommunikation.

# 2 Management Summary

## Problemstellung

OpenStreetMap-in-a-Box liegt in einer Version vor, welche eine kompakte Java basierte Lösung für die Nutzung der OpenStreetMap Daten bietet. Die Architektur ist bereits sehr ausgereift und kann beliebig erweitert werden. Es werden noch nicht alle Daten von OpenStreetMap importiert und es gibt noch einige wichtige Kartendaten, die dem Benutzer von OpenStreetMap-in-a-Box zur Verfügung gestellt werden sollten. Um dies zu erreichen kann das bestehende Mapping um neue Elemente erweitert werden und gegebenenfalls auch die Datenbankstruktur ausgebaut werden, um ganz neue Gruppen wie z.B. Gebäude zu importieren.
Das Rendering der Kartengrafik bei OpenStreetMap-in-a-Box wurde bei der ersten Version in einer eher einfachen Art und Weise umgesetzt. Für eine gute Nutzung der Karten müssen diese gut lesbar sein, was nur durch ein ausgereiftes Styling Konzept erreichbar ist. Ein zentraler Punkt bei der Nutzung von Bildschirmbasierten Karten ist, dass man dem Benutzer die Informationen gut strukturiert präsentieren kann. Er sollte nicht mit einer Flut von Daten überfordert werden, trotzdem aber die Möglichkeit haben, die ganze Fülle an vorhandenen Daten abzurufen. Bei der Umsetzung müssen somit gezielt Daten in Abhängigkeit der Zoomstufe der Karte weggelassen werden. Durch heranzoomen kann man diese Daten dann dem Benutzer trotzdem zur Verfügung stellen. Im weiteren sind Farbkontraste und die Wahl des Schriftstils und der Schriftgröße wichtige Gestaltungselemente für die Präsentation von Kartendaten.
Die Darstellung der Karte wird vom Web Map Service (WMS) des Geoservers übernommen. Er interpretiert dabei die SLD Dateien und setzt die dort definierten Regeln entsprechend um. Damit dies auch einwandfrei funktioniert, muss aber auch die Infrastruktur des Servers, auf dem der Geoserver läuft, entsprechend konfiguriert werden. Das Testen der Applikation hat ergeben, dass der Linux Server vor allem bei der Darstellung der Schriften Probleme hat und somit die Konfiguration angepasst werden muss. Zuerst muss also das Problem eingegrenzt werden und anschließend muss der Server entsprechend konfiguriert werden.

Sobald das Styling und die Serverkonfiguration abgeschlossen sind, soll das Resultat auch demonstriert werden. Nur so kann der Benutzer von OpenStreetMap-in-a-Box sich auch vorstellen, auf welche Weise er die Software nutzen kann. Die Möglichkeiten der Nutzung von OpenStreetMap-in-a-Box können natürlich nicht alle aufgezeigt werden, aber zumindest zwei in der heutigen Zeit zentrale Nutzungsgebiete, wie die Verwendung in einer Webseite und auf einem Mobiltelefon sollten gezeigt werden. Die Anforderungen an diese beiden Lösungen sind grundsätzlich ähnlich. Die Karte muss dargestellt werden und gut bedienbar sein. Für die Bedienbarkeit ist bei beiden Lösungen eine gute Suchfunktion unverzichtbar. Erst durch die Möglichkeit nach Ortschaften, Straßen, Flüssen etc. suchen zu können kann eine Karte sinnvoll eingesetzt werden.
Der Algorithmus der Suchfunktion ist sehr zentral. Auch hier gilt es den Benutzer wenn möglich nicht mit irrelevanten Daten zu überfordern und ihm eine sinnvolle Auswahl von Daten zur Verfügung zu stellen. Es muss also genau definiert sein, welche Daten durchsucht werden sollen und welche dieser Daten auch als Suchresultate erscheinen sollen. Es muss festgelegt werden, wie die Suchresultate dem Benutzer präsentiert werden sollen und auf welche Weise er diese dann auf der Karte sichtbar machen kann.

Die technischen Möglichkeiten einer Webseite und die eines Mobiltelefons unterscheiden sich ebenfalls ein wenig. Bei Mobiltelefonen gibt es heute bereits einige Möglichkeiten, die bei einer Webseite keinen Sinn machen würden. Beispiele dafür sind die Nutzung von GPS oder eines eingebauten Kompass. Um auch diese zusätzlichen Nutzungsgebiete demonstrieren zu können sollte dies bei der Implementation der Mobil-Applikation einbezogen werden. Aus zeitlichen Gründen ist es auch hier nicht möglich alles zu demonstrieren, daher wird der Fokus auf die Nutzung von GPS gesetzt. Weitere Möglichkeiten und Ideen werden dann in der Dokumentation erwähnt.

## Ansatz

Zum Beginn des Projektes wurde die bestehende Applikation OpenStreetMap-in-a-Box lokal installiert und getestet. Dabei wurde einerseits die Installation, die Bedienung und auch die resultierende Anwendung der Karte auf der bereits bestehenden Webseite der Vorgängerarbeit betrachtet. Die hier gewonnenen Erkenntnisse konnten bereits für die später folgende Verbesserung der Kartengrafik verwendet werden. Neben dem Testen der Software wurde die vorliegende Architektur und der Programmcode studiert, um das Verhalten der Software zu verstehen.

Als Grundlage für den ersten Teil der Arbeit galt die ETH Studie „Kartenkritik an OpenStreetMap“. Die wichtigsten Erkenntnisse dieser Studie wurden gesammelt und galten danach als Grundlage für die Verbesserung der Kartengrafik. Während dann die SLD Dateien für das Rendering erneuert oder angepasst wurden, ist gleichzeitig die Serverarchitektur getestet worden um die Probleme bei der Darstellung der Schriften zu finden und diese anschließend zu beheben. Beim Styling der Karte dienten OpenStreetMap und Google Maps als Referenzkarten. Nach dem Testen der Resultate des neuen Stylings wurde das Design der Webseite entwickelt und mit deren Implementation begonnen. Beim Suchalgorithmus wurden die bestehenden Suchfunktionen von OpenStreetMap und Google Maps getestet und diese Erkenntnisse in den Aufbau der Suchfunktion einbezogen.

Für den dritten Teil der Arbeit, die Android Applikation, wurden zuerst die Anforderungen in Zusammenarbeit mit Herr Reto Senn erarbeitet. Anschliessend galt es sich möglichst schnell in die Entwicklungsumgebung von Android einzuarbeiten und mit der Implementation zu beginnen.

Das Testing konnte bei der vorliegenden Arbeit nicht im klassischen Sinne der Softwareentwicklung durchgeführt werden. Aus zeitlichen und technischen Gründen musste bei der Verbesserung der Kartengrafik und bei der Implementation der Webseite ein sehr pragmatischer Ansatz des Testings umgesetzt werden. Ein ausführliches Testing der Kartengrafik wäre durch die sehr grosse Anzahl an verschiedenen Kartenelementen, welche einerseits auf der ganzen Karte der Schweiz variieren und andererseits auch pro Zoomstufe der Karte unterschiedlich sind, nicht im Rahmen dieser Arbeit möglich gewesen. Bei der Suchfunktion wurden einige wichtige Black Box Testcases geschrieben um die Grundfunktionalität des Suchalgorithmus testen zu können.

Während des Projektes wurden wöchentliche Meetings mit dem Betreuer Professor Stefan Keller festgelegt. In diesen Meetings wurde jeweils der aktuelle Stand präsentiert, Probleme besprochen und das weitere Vorgehen festgelegt.

## Resultate

Das Resultat der Arbeit ist eine Webseite, die die verbesserte Kartengrafik demonstriert und eine gut bedienbare Suchfunktion bereitstellt. Die Seite beinhaltet folgende Elemente:

- Darstellung der Karte über den WMS Aufruf des Geoservers
- Bedienelemente der Karte (Zoom, Paning, Permalink)
- Zusätzliche Geo-Information der Karte (Mouse Position, Skalierung der Karte)
- Eingabefeld für die Suche (immer sichtbar)
- Anzeigebereich für die Suchresultate (ein- und ausschaltbar)
- Visualisierung der Suchresultate durch korrektes Zoomen der Karte und Darstellung eines Markers an der entsprechenden Position

Die folgende Abbildung [Abbildung 1] fasst die gemachten Arbeiten aus Sicht der Serverarchitektur zusammen.

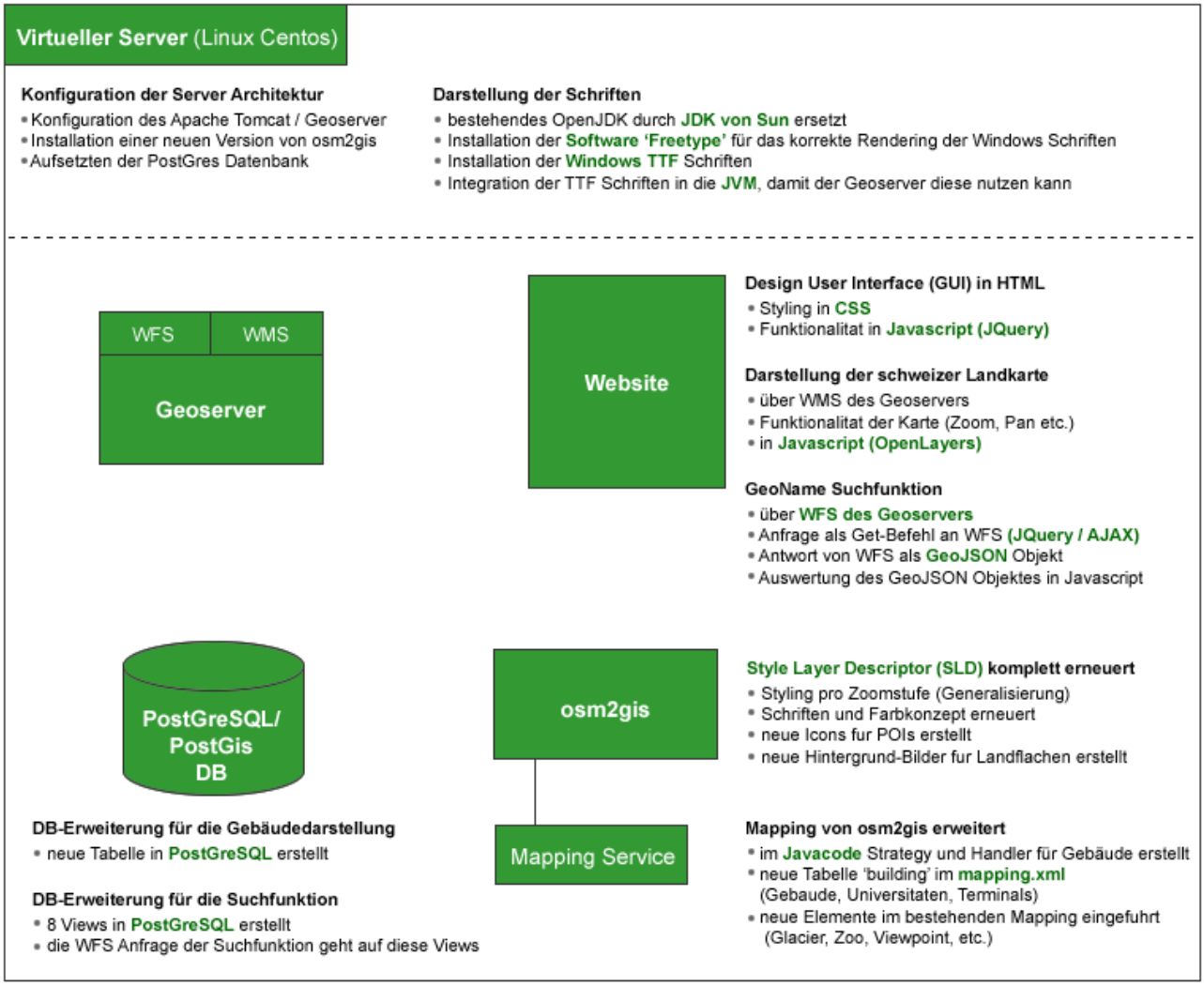


*Abbildung 1: Übersicht Serverarchitektur / Webseite*

Im Weiteren liegt am Ende der Arbeit auch eine installierbare Software, der Karten-Viewer für Android vor.

Dieser bietet folgende Funktionalität:

- Darstellung der Karte über den WMS Aufruf des Geoservers
- Bedienelemente der Karte (Zoom, Paning)
- Suchfunktion über Menu aufrufbar, Darstellung der Suchresultate in separater View als Liste
- Darstellung des aktuellen Standortes über GPS

Beim Rendering der Karte sind auf gewissen Zoomstufen noch Verbesserungen anzubringen. Betroffen sind vor allem die Darstellung von Polygonen wie beispielsweise Waldflächen. Der genaue Ursprung dieser Unschönheiten beim Rendering konnte bis zur Abgabe der Arbeit nicht ermittelt werden. Das Problem kann verschiedene Ursachen haben, einerseits kann es an der Serverkonfiguration liegen oder am SLD Styling selber, es könnte aber auch an den Daten in der Datenbank oder an der Konfiguration des Geoservers liegen. Um die Ursache herauszufinden ist ein ausführliches Testing notwendig was zeitlich im Rahmen dieses Projektes nicht möglich gewesen ist und daher im Anschluss an diese Arbeit gemacht werden muss.


*Abbildung 2: Pois R Us - Screenshots*

## Zukünftige Arbeiten

Wie bereits erwähnt sollten die Unschönheiten beim Rendering bald behoben werden. Hierfür müssten wenn möglich mehrere Personen eingesetzt werden, die die aktuelle Konfiguration testen und das Problem so eingrenzen können.
Bei der Kartengrafik sollte das Mapping noch zusätzlich erweitert werden. Es gibt noch einige sogenannte POI (Point of Interest), die zur Zeit noch nicht gemapped werden. Auch gewisse öffentliche Verkehrsmittel wie beispielsweise Luftseilbahnen werden noch nicht dargestellt und sollten noch einbezogen werden, da diese Verkehrsmittel gerade in der Schweiz mit ihren verschiedenen Bergregionen wichtig sind.

Bei der Webseite (und auch bei der Android Applikation) könnte man noch verschiedene Kartenlayer erzeugen, die dann bei Bedarf ein- und ausgeschaltet werden können. Durch die bestehende Architektur des Geoservers und den Hilfsmitteln wie OpenLayers für die Webseite oder der Nutiteq Library bei der Android Applikation sollte der Aufwand dafür relativ klein sein.

Die Android Applikation liegt am Ende des Projektes lediglich als Prototyp vor. Die Zeit reichte nicht aus, eine ausführlich getestete Software zu entwickeln und auch bezüglich der Funktionalität konnten nur die Grundfunktionen implementiert werden. Es gäbe hier noch viele weitere Funktionalitäten, die eine solche Applikation bieten kann. Beispielsweise könnte mit Hilfe des integrierten Kompass bei Android die Karte bei Bedarf entsprechend ausgerichtet werden. Schön wäre auch die Darstellung von POIs in Abhängigkeit der aktuellen Position. So könnte der Benutzer beispielsweise Restaurants oder Einkaufsmöglichkeiten in der Nähe suchen.

Die Anpassungen, welche an der OSM-in-a-Box Software gemacht worden sind, werden im Anschluss an dieses Projekt zusammengefasst und an die Gruppe, die das Refactoring von OSM-in-a-Box parallel zu diesem Projekt machen, weitergegeben. Die Anpassungen können somit in das Refactoring einbezogen werden.

# 3 Aufgabenstellung

## 3.1 Einführung

In einer vorangegangenen Bachelorarbeit mit dem Namen „OpenStreetMap-in-a-Box“ wurde eine Server-Applikation mit Datenbank, Geo-Webservices und einer Webseite erstellt. Im Wesentlichen dient OpenStreetMap-in-a-Box der Bereitstellung von Hintergrund-Karten für interaktive (Web-)Kartenapplikationen im Web und auf Mobiles. Im Gegensatz zu bekannten Kartendiensten wie Google Maps bietet diese Lösung zwei entscheidende Vorteile: 1. freie Kontrolle über die Verfügbarkeit des Services (da lokal), 2. die Möglichkeit, eine an individuelle Bedürfnisse angepasste Kartengrafik zu definieren. Zudem sind die Daten oft detaillierter und aktueller. Dies ist möglich durch die GPL-artige Lizenz der Daten. Die Software selber steht unter LGPL. Diese Arbeit soll übernommen und evolutionär erweitert werden.

Das Szenario für die Nutzung als Webservice bleibt gleich. Dazu kommt die Nutzung von OpenStreetMap-Daten durch einen Mobile Viewer. Weitere Anforderungen sollen ggf. innerhalb der Arbeit erarbeitet werden.

## 3.2 Aufgabenstellung

Die Arbeit soll mind. folgende Tätigkeiten und Software umfassen:

1. Server-Applikation:
    - Aktueller Stand (v.a. osm2gis) testen und ggf. (nach-)dokumentieren und wo sinnvoll refaktorisieren (z.B. beim OSM API)
    - Verbessern der Kartengrafik (1. Alternative zu Grundkarte Mapnik und Google Maps, 2. für interaktive Campus-Karte ohne Gebäude)
2. Website (am Beispiel Schweiz): Erweiterung durch
    - Suche nach Ort (Geonamen-Suche)
    - Mobiles Web: Bei Aufruf der Website durch einen Mobile Browser soll eine angepasste Website erzeugt werden (Layout und ggf. Kartengrafik, serverseitige Erkennung). *Anmerkung: Dieser Punkt wurde nach Absprache mit Herr Prof. Stefan Keller nicht umgesetzt, um die Zeit zugunsten der Androi Applikation nutzen zu können*
3. Android-Applikation:
    - Darstellung der OpenStreetMap-Daten via Tiles-API (online)
    - Suche nach Ort (Geonamen-Suche)
    - Darstellung des eigenen Standorts mittels GPS-Sensor (und falls vorhanden IndoorWPS)

Lieferdokumente (deutsch):

- Installationsanleitung (Server- und Android-App) wo nötig (englisch).

- Website:
  - Migration an definitiven Ort
  - Demo der Android-App. im Web (stichwortartig und bebildert).
- Gesamter compilier-bereiter Sourcecode (englisch) inkl. Programmdokumentation sowie als Download gekennzeichnetes Zip-File mit ausführbarem Bytecode.
- Technischer Bericht und SW-Engineering-Dokumentation
- Allfällige Dokumente gemäss Vorgaben der Abt. I. (z.B. Plakat, ‚Abstract').

## 3.3 Hinweise

- Die Arbeitsweise ist agil, wo sinnvoll mit Unit-Tests. Es wird Wert auf ausgetestete Software und einfache Installation gelegt.
- Eine Bedienungsanleitung ist nur in begründeten Fällen gefordert.

## 3.4 Randbedingungen, Infrastruktur, Termine und Beurteilung

- Randbedingungen Hardware/OS:
  - Server: Linux
  - PDA-Client: Android-Hardware und SDK
- Software:
  - Java gem. Android und Android SDK
  - JavaEE gem. Tomcat 6
  - ANT, Eclipse IDE
  - Web-Client: Mind. getestet mit Internet Explorer >6 und Firefox >3
- Termine und Beurteilung: Gemäss Angaben auf www.i.hsr.ch.

Rapperswil, 2. Oktober 2009

Der Betreuer Der Studierende

# 4 Teil I: Technischer Bericht

## 4.1 Problemstellung

### 4.1.1 Server-Applikation

Die Darstellung der OpenStreetmap-in-a-Box Karten basiert auf sogenannten Styled Layer Deskriptoren (SLD). Bei der Installation wird ein Geoserver auf Tomcat deployed, welcher solche SLDs für die darzustellenden Kartenelemente enthält. Innerhalb dieser XML ähnlichen Dateien sind Regeln definiert, welche der Geoserver beim Rendering der Karte umsetzt und somit das Aussehen der Karte bestimmen. Je nach Zoomstufe ändert sich auch das Ausmass der dargestellten Kartenelemente. Je näher man heranzoomt, desto mehr Elemente werden dargestellt. Die aktuelle Implementation von OpenStreetMap-in-a-Box liefert sehr einfache SLDs wodurch die Karte noch nicht optimal genutzt werden kann. Die Möglichkeiten von SLD werden nicht ausgeschöpft. Zudem werden die Schriftdefinitionen nicht richtig umgesetzt, wodurch die Beschriftungen von Ortschaften und anderen Objekten schlecht lesbar sind. Das Mapping, bei dem die Daten von OpenStreetMap auf die verschiedenen Datenbanktabellen von OpenStreetMap-in-a-Box abgebildet werden, deckt nur einen Teil der zur Verfügung stehenden Daten ab. Es sollte noch um einige grundlegende Elemente wie beispielsweise die Fläche von Gebäuden erweitert werden.

### 4.1.2 Webseite

Die von OpenStreetMap-in-a-Box zur Verfügung gestellten Kartendaten können über den Web Map Service (WMS) des Geoservers gerendert werden und somit beispielsweise auf einer Webseite dargestellt werden. Um die Karte auf dieser Webseite auch wirklich nutzen zu können, muss dem Benutzer eine Suchfunktion zur Verfügung gestellt werden. Der Benutzer möchte nach Ortschaften, Gewässer, Strassen etc. suchen und diese in der Karte betrachten. Es muss also über einen entsprechenden Service, nach einem Namen gesucht werden können. Die Antwort einer solchen Suchanfrage muss die gefundenen Resultate mit den entsprechenden Koordinaten und Attributen enthalten. Der Geoserver, welcher bei OpenStreetMap-in-a-Box mitgeliefert wird, bietet bereits einen Web Feature Service (WFS) an, welcher es ermöglicht, Abfragen auf der Datenbank des Geoservers zu machen. Die Suchfunktion könnte aber auch über andere Services durchgeführt werden, falls diese bessere Resultate liefern oder die Suchanfrage über einen anderen Service besser umsetzbar ist.

### 4.1.3 Android Applikation

Android ist ein Betriebssystem bzw. eine Software-Platform für mobile Geräte auf der Basis eines Linux-Kernels. Es handelt sich um eine freie Software die sich daher für die Entwicklung einer Mobiltelefon Applikation anbietet. Es gibt bereits verschiedene Karten-Viewer für dieses Betriebssystem, unter anderem steht eine API von Google Maps zur Verfügung, mit welcher Karten von Google Maps verwendet werden können. Für die Darstellung von WMS basierten Kartengrafiken wurden noch kaum Applikationen entwickelt und es gilt nun herauszufinden, auf welche Weise ein solcher Service am Besten implementiert werden kann und ob es bereits Schnittstellen gibt, die eine solche Implementation vereinfachen würden. Moblitelefone bieten neben dem Zugriff auf das Internet heutzutage auch meistens weitere Services wie GPS oder einen Kompass an. Solche Technologien sollen bei der Entwicklung der Applikation wenn möglich miteinbezogen werden.

## 4.2 Ziel der Arbeit

### 4.2.1 Server-Applikation

Es soll ein Konzept für das SLD basierte Rendering der Karte erarbeitet und umgesetzt werden. Als Grundlage für die Verbesserung der Kartengrafik dient eine Studie der ETH Zürich, welche die Darstellung bei OSM untersucht hat und Stärken und Schwächen aufdeckt. Basierend auf den Erkenntnissen dieser Arbeit sollen die Schwächen der Kartengrafik von OpenStreetMap-in-a-Box aufgezeigt werden und entsprechende Verbesserungen vorgenommen werden. Das Aussehen der Karte sollte sich an bereits vorhandenen online Karten wie OpenStreetMap und Google Maps orientieren. Für die Umsetzung steht ein Virtueller Linux Server zur Verfügung. Auf diesem Server soll die verbesserte Version von OpenStreetMap-in-a-Box installiert werden. Dies beinhaltet auch die Konfiguration des Servers, damit dieser die Daten auch entsprechend verarbeiten kann.

### 4.2.2 Webseite

Auf dem virtuellen Server soll eine Webseite erstellt werden, welche die verbesserte Kartengrafik zeigt. Dazu muss eine Benutzeroberfläche entwickelt werden, welche die Bedienung der Karte möglichst einfach macht. Die Benutzeroberfläche soll das Navigieren auf der Karte selber ermöglichen und zudem eine Suchfunktion bieten. Bei der Suchfunktion gilt es zu evaluieren, welcher Dienst für die Umsetzung am besten geeignet ist. Da diese Webseite die Möglichkeiten von OpenStreetMap-in-a-Box demonstrieren soll, sind Benutzerfreundlichkeit und Aussehen der Webseite ebenfalls sehr zentral.

### 4.2.3 Android Applikation

Es soll ein lauffähiger Prototyp entwickelt werden, der über einen WMS Aufruf des Geoservers die von OpenStreetMap-in-a-Box zur Verfügung gestellte Karte darstellen kann und alle Grundfunktionalitäten einer Karte wie Zoom und Paning bietet. Wie auch bei der Webseite soll eine Suchfunktion verfügbar sein, welche analog zu der bereits entwickelten Suchfunktion der Webseite funktioniert. Um die weiteren Möglichkeiten bei der Nutzung von Mobiltelefonen zu demonstrieren soll auch GPS genutzt werden, um die aktuelle geographische Position des Benutzers auf der Karte darstellen zu können.

## 4.3 Die wichtigsten Anforderungen

Für die Webseite und die Android Applikation gelten bis GPS dieselben Funktionalen Anforderungen:

| Funktionalität | Beschreibung | Priorität | Abhängigkeit |
|---|---|---|---|
| **Kartendarstellung über WMS** | Über eine HTTP WMS Anfrage an den Geoserver werden die Map Tiles empfangen und dargestellt. | Hoch | Web Map Service Geoserver (WMS) |
| **Bedienung der Karte (Zoom, Pan)** | Die Karte bietet die wichtigsten Bedienelemente. Es kann herein- und herausgezoomt werden und in alle Richtungen navigiert werden. | Hoch | Web Map Service Geoserver (WMS) |
| **Suchfunktion** | Mit Hilfe eines Eingabefeldes kann nach verschiedenen Elementen auf der Karte gesucht werden. | Hoch | Web Feature Service Geoserver (WFS) |
| **Anzeigen eines Suchresultates auf der Karte** | Durch Klick auf ein Suchresultat zeigt die Karte das Resultat an. Die Zoomeinstellung wird abhängig vom Typ des Resultates (City, Town, Waterway etc.) angepasst. Ein Marker visualisiert den Ort. | Hoch | - |
| **Darstellung aktueller Standort** | Der aktuelle Standort des Benutzers kann auf der Karte dargestellt werden. | Hoch | GPS des Mobiltelefons |

## 4.4 Ergebnisse

### 4.4.1 Server-Applikation

#### SLD Dateien

Die SLD Dateien sind von Grund auf erneuert worden. Die meisten vorhandenen Dateien wurden gänzlich neu definiert, einige Dateien wurden lediglich verbessert und ergänzt.

| FeatureType | Kommentar |
|---|---|
| **osm_place** | • SLD komplett erneurt<br>• Schriftgrösse, -farbe, -umrandung (halo) abhängig von Zoomstufe und Typ<br>• Spacing bei der Beschriftung → Durch entsprechendes Spacing wurde sichergestellt, dass die Beschriftungen der Ortschaften nicht zu nahe beisammen liegen und somit die Lesbarkeit beinträchtigt wird |
| **osm_road** | • SLD komplett erneurt<br>• Schriftgrösse, -farbe, -umrandung (halo) abhängig von Zoomstufe und Typ<br>• Gruppierung der Strassenbeschriftung → Eine Strasse besteht in der Datenbank aus mehreren Segmenten. Jedes oder fast alle Segmente besitzen das Attribut Name und eine normale Beschriftung hätte zur Folge, dass unzählige Labels entlang der Strasse erscheinen würden. Abhilfe schafft hier das Grouping, wodurch gleiche Namen entlang einer Strasse gruppiert werden und nur einmal angezeigt werden. <VendorOption name="group">yes</VendorOption> |
| **osm_railwaystation** | • SLD komplett erneuert<br>• Bahnhöfe werden neu mit sinnvollen Icons visualisiert. Für die unterschiedlichen Typen (Bahnhof, Tramstation etc.) wurde ein entsprechendes Icon in verschiedenen Grössen eingeführt, welches abhängig von der Zoomstufe angezeigt wird. Dies verbessert die intuitive Interpretation der Karte. |
| **osm_railway** | • SLD komplett erneuert<br>• Darstellung der Bahnlinien ähnlich wie bei Google Maps. Die alte Darstellung, die der von OpenStreetMap entsprach, war zu dominant und führte vor allem bei grösseren Bahnhöfen dazu, dass diese Punkte zu stark ins Auge fielen. |
| **osm_water**<br>**osm_waterway** | • SLD komplett erneuert<br>• kräftigeres Blau für Seen und Flüsse<br>• Beschriftung von Flüssen eingeführt |
| **osm_landuse** | • SLD komplett erneuert<br>• Farbe der Wälder wurde abgeschwächt, damit ein ruhigeres Geamtbild der Karte entsteht.<br>• Es wurden zahlreiche neue Flächen über das Mapping und entprechende Regeln im SLD eingeührt (siehe Mapping Erweiterungen). |
| **osm_poi** | • SLD wurde mit weiteren POIs ergänzt (siehe Mapping Erweiterungen) und für eine grosse Anzahl bestehender POIs wurden neue, modernere Icons eingeführt. |
| **osm_pofw**<br>(place of worship) | • Die bestehende SLD Datei wurde etwas besser strukturiert, inhaltlich aber so belassen wie sie ist. |
| **osm_building** | • SLD neu erstellt<br>• Durch das erweiterte Mapping und die Erzeugung einer neuen Tabelle in der Datenbank können nun auch Gebäude dargestellt werden. Dafür wurde eine komplett neue SLD Datei erstellt. |

*Tabelle 1: Änderungen SLD Dateien*

## Darstellung der Schriften

Der Linux Server wurde so konfiguriert, dass die Schriften sauber dargestellt werden. Das Rendering der Schriften hat zuvor nicht funktioniert, da die in den SLD definierten Windows Schriften nicht auf dem Server verfügbar waren. Durch die Installation der Windows Schriften auf dem Linux Server und deren Integration in die Java Umgebung des Server konnte dieses Problem behoben werden. Damit die Windows Schriften auch tatsächlich von der Java Umgebung interpretiert werden können, wurde das installierte OpenJDK durch das offizielle JDK von Sun ersetzt.

## Mapping Erweiterungen

Das bestehende Mapping von OpenStreetMap-in-a-Box wurde um verschiedene neue Elemente erweitert. Zudem wurden der Java Code und die PostgreSQL Dateien so angepasst, dass eine weitere Datenbanktabelle für Gebäude erzeugt wird. Es sollten noch zahlreiche weitere Elemente in das Mapping einbezogen werden, was aber den Rahmen dieser Arbeit sprengen würde. Es wurden nur einige wichtige neue Elemente einbezogen um das Mapping zu testen und die Möglichkeiten diesbezüglich aufzuzeigen.

| Tabelle | OSM Key - Name | Beschreibung |
|---|---|---|
| **osm_building** | building=yes<br>aeroway=terminal<br>amenity=university<br>building=university | Neue Tabelle für die Darstellung von Gebäuden erzuegt. Flughafengebäude und Universitätsgebäude werden unterschieden, um diese durch entprechende SLD Regeln anders darstellen zu können. |
| **osm_road** | aeroway=runway<br>aeroway=taxiway | Start- und Landebahnen, sowie die zugehörigen Verbindungs-Strasen wurden neu eingeführt. |
| **osm_landuse** | aeroway=helipad<br>aeroway=apron<br>landuse=village_green<br>amenity=parking<br>leisure=stadium<br>leisure=pitch<br>sport=tennis<br>tourism=zoo<br>surface=grass<br>landuse=commercial<br>landuse=vineyard<br>natural=glacier<br>place=island | Diverse Landflächen, wie Gletscher, Weingut, Parking, Sportanlagen, Helikopterlandeplätze etc. wurden eingeführt |
| **osm_place** | place=country<br>place=state<br>place=region | Ermöglicht Beschriftung und vor allem auch die Suche nach Ländern und Regionen |
| **poi** | aeroway=terminal<br>aeroway=helipad<br>power=generator<br>tourism=viewpoint<br>railway=level_crossing | Aussichtspunkte sind bei Karten sehr typische Elemente und wurden daher in die Liste der POIs aufgenommen. |

*Tabelle 2: Mapping Erweiterungen*

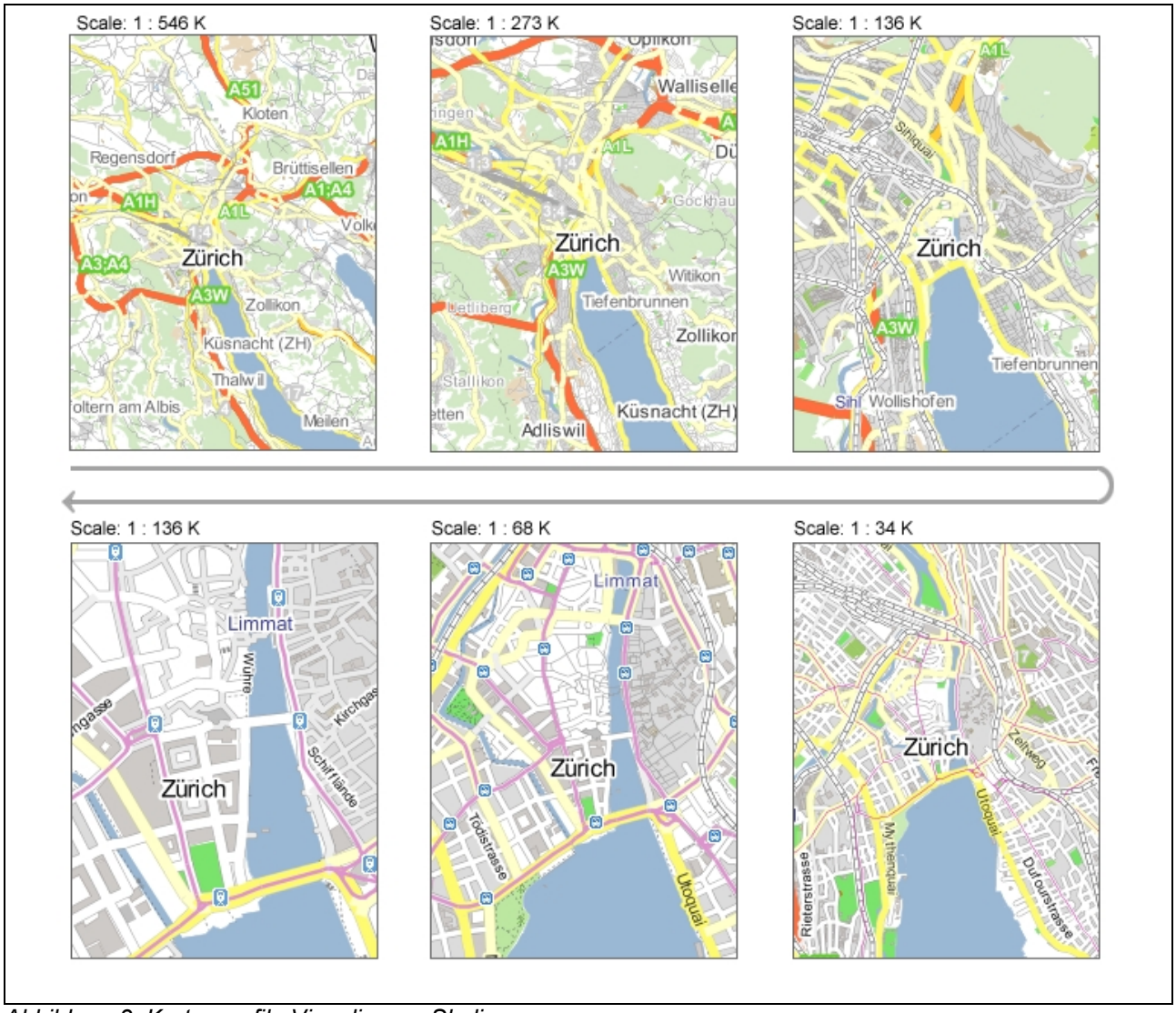


*Abbildung 3: Kartengrafik Visualierung Skalierung*

### 4.4.2 Webseite

Am Ende der Arbeit liegt eine Internetseite vor, die das neue Aussehen der Karte demonstriert und eine auführliche Suchfunktion bietet. Die Suchresultate können ein- und ausgeblendet werden und die Resultate bieten je nach Typ unterschiedliche Informationen. Beispielsweise werden bei POIs falls in der Datenbank vorhanden, Links angezeigt, die auf die Internetseite des POI führen. Bei Ortschaften wird wenn möglich angezeigt, in welchem Gebiet (z.B. Kanton, Stadt in der Nähe etc.) sich die Ortschaft befindet. Diese Attribute helfen dem Benutzer die einzelnen Resultate schneller unterscheiden zu können. Wenn ein Suchresultat angeklickt wird, zeigt die Karte den entsprechenden Ort mit einer Stecknadel an. Auch hier wird nach Typen unterschieden und die Zoomstufe beim Anzeigen des Resultates angepasst. Folgende Technologien wurden für den Aufbau der Seite eingesetzt:

- HTML für das Grundgerüst der Seite
- CSS für das Aussehen der HTML Elemente
- Javascript (JQuery) für die dynamische Funktionalität wie die ein- und ausschaltbaren Suchresultate

Die Internetseite wurde für alle gängigen Browser optimiert (Internet Explorer, Firefox, Opera)

Die Suchfunktion ist basierend auf folgenden Technik implementiert worden:

- Get Anfrage an WFS des Geoservers über Javascript (JQuery)
- Eingrenzung des Anfrage über OGC Filter Queries
- WFS Abfrage auf generierten Lookup Views auf der PostgreSQL Datenbank
- Antwort als GeoJSON Objekt
- Auswertung der GeoJSON Objekte in Javascript
- Darstellung der Suchresultate über Javascript und CSS

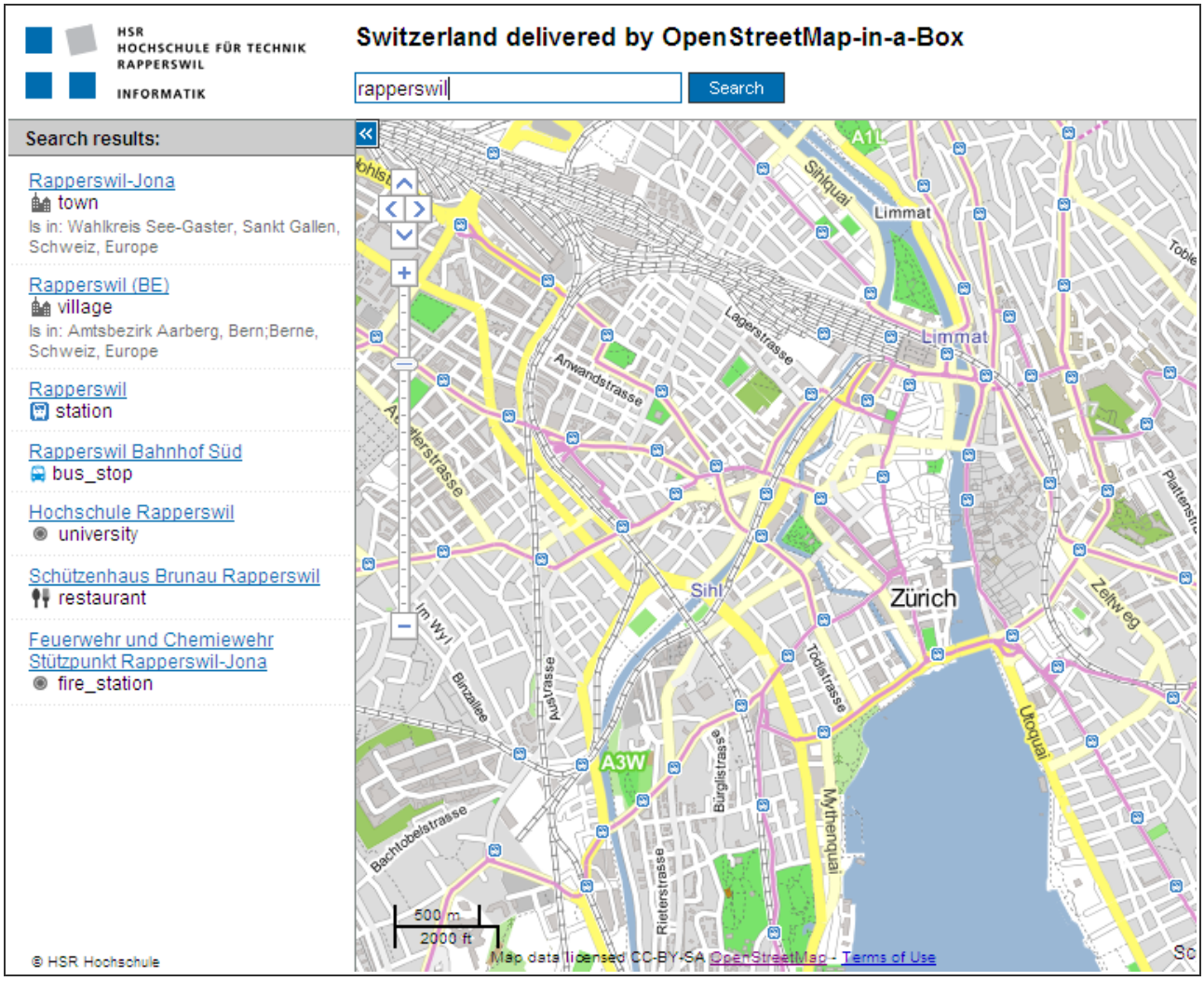


*Abbildung 4: Webseite*

### 4.4.3 Android Applikation

Die Android Applikation liegt als Prototyp vor. Durch die Nutzung der Java Library von Nutiteq ist es möglich, die Karte über einen WMS Aufruf darzustellen und zu bedienen. Die API von Nutiteq bietet zudem Möglichkeiten, die aktuelle Position über GPS zu ermitteln und diese auf der Karte darzustellen, was über den Menu-Punkt „My Location“ machbar ist. Über das Menu kann zudem die Suchfunktion aufgerufen werden, welche eine Anfrage an den WFS des Gesoservers schickt und die ausgewerteten Resultate in einer Liste darstellt. Durch Anklicken eines Resultates wird dieses auf der Karte mit einer Stecknadel dargestellt. Auch hier wird die Zoomstufe je nach Typ des Resultates entsprechend angepasst. Durch die Platzbeschränkungen bei einer Mobilapplikation konnte die Suche nicht gleich implementiert werden wie es bei der Webseite möglich war. Die Applikation besteht aus drei Sichten, der Karte, der Sucheingabe und der Darstellung der Suchresultate.


*Abbildung 5: Pois R Us Mobiltelefon - Screenshot*

## 4.5 Schlussfolgerungen

Die drei verschiedenen Teile dieses Projektes konnten erfolgreich umgesetzt werden. Der Aufwand für die einzelnen Teile war relativ schwierig einzuschätzen, da in allen Bereichen zuerst nach geeigneten Technologien gesucht werden musste. Die Verbesserung der Kartengrafik war sehr zeitintensiv und durch die Fülle an vorhandenen Informationen in einer Karte ist ein war kein ausführliches Testen der Anpassungen möglich. Die Serverinstallation auf Linux Centos ist für folgende Projekte nicht empfehlenswert, da die Konfiguration relativ aufwendig ist und auf dem Internet im Vergleich zu anderen Linux Versionen relativ wenig Informationen zu finden sind.

## 4.6 Ausblick

Die Kartengrafik sollte stetig weiter verbessert werden. Es gibt noch sehr viele Features von OpenStreetMap, die noch nicht importiert werden und es sollte daher das Ziel sein, den Detailgrad der Karte noch weiter auszubauen, um dem Benutzer möglichst viele Informationen bieten zu können. Die Erweiterung der importierten Daten und somit auch des Mappings von osm2gis hat natürlich auch entsprechende Erweiterungen an den SLD Dateien zur Folge.

Bei der Webseite wäre es sicherlich Interessant, die Möglichkeit Kartenlayer ein- und auszuschalten auch noch zu zeigen. Der Suchalgorithmus kann ebenfalls noch weiterentwickelt werden. Es wäre denkbar, dass ähnlich wie bei der Suchmaschine von Google bei Tippfehlern ein Korrekturvorschlag erscheint. Genauso wäre es hilfreich, wenn bereits während der Eingabe eines Such-Strings Vorschläge erscheinen, die zu der bereits gemachten Eingabe passen.

Der Prototyp der Android Applikation sollte als Grundlage dienen, um einen ausgereiften Karten-Viewer für Mobiltelefone zu entwickeln. Im Gegensatz zu der Webseite, welche vor allem im Detail noch erweitert werden kann, gibt es bei der Android Applikation noch einige Funktionalitäten die eingebaut werden können. So zum Beispiel die Ausrichtung der Karte anhand des integrierten Kompass.

# 5 Teil II: Projektdokumentation

## 5.1 Verbesserung der Kartengrafik

### 5.1.1 Analyse der aktuellen Kartendarstellung

**Studie „Kartengrafik an OpenStreetMap“ (ETH Zürich, Institute of Cartography)**

Die Studie analysiert den aktuellen Stand der Kartendarstellung von OpenStreetMap. Die Analyse wird anhand eines für Bildschirmbasierte Andwendungen entworfenen Kriterienkatalog gemacht. Dieser Kriteriekatalog wird als Grundlage für die Analyse der Karte von OpenStreetMap-in-a-Box genommen. Nachfolgend sind die wichtigsten Kriterien aufgelistet und erklärt, auf welche Weise diese Kriterien bei der Umsetzung der Verbesserungen einfliessen:

| Kriterium | Beschreibung |
|---|---|
| **Allgemeines** | **Studie ETH**<br>• Macht die Karte einen guten Eindruck?<br>• Lädt die Karte zum Betrachten und Lesen ein?<br><br>**OpenStreetMap-in-a-Box**<br>• Der Gesamteindruck der Karte soll das Interesse des Benutzers wecken. Die folgenden Kriterien fliessen alle in diesen Punkt ein. |
| **Grafische Bilddichte** | **Studie ETH**<br>• Ist sie dem Masstab sowie der Bildschirmgrösse und -auflösung angepasst oder wirkt die Karte überladen?<br><br>**OpenStreetMap-in-a-Box**<br>• OSM wirkt oft etwas überladen → bei der neuen Gestaltung wird Wert darauf gelegt, Information in bestimmten Zoomstufen bewusst wegzulassen<br>• Die Daten, die von OSM zur Verfügung stehen, sind an gewissen Stellen noch lückenhaft und es kann daher v.a. in ländlichen Gebieten zu einer eher mangelnden Bilddichte kommen → die zugrunde liegenden Daten können bei dieser Arbeit nur beschränkt beeinflusst werden, es kann jedoch durch ein verbessertes Mapping der OSM Daten eine verbesserte Bilddichte erreicht werden. |
| **Grafische Differenzierbarkeit** | **Studie ETH**<br>• Wird Gleiches immer gleich und Verschiedenes immer verschieden dargestellt?<br>• Sind die Kartenelemente am Bildschirm klar voneinander unterscheidbar?<br>OpenStreetMap-in-a-Box<br><br>**OpenStreetMap-in-a-Box**<br>• Gleiche Elemente werden grundsätzlich bereits überall gleich dargestellt<br>• OSM weist aber klare Schwächen bei der Unterscheidbarkeit auf. Ein gutes Beispiel dafür sind Autobahnen, die blau dargestellt sind und leicht mit Flüssen zu verwechseln sind. Bei der Umsetzung wird grossen Wert darauf gelegt, dass die unterschiedlichen Elemente klar unterscheidbar sind. |
| **Lesbarkeit** | **Studie ETH**<br>• Wird die verminderte Auflösung des Bildschirmes bei der Gestaltung der Karte berücksichtigt?<br>• Ist die Karte bei einem mittleren Abstand zwischen Augen und Bildschirm von 60 bis 80 cm ohne grössere Mühe lesbar?<br><br>**OpenStreetMap-in-a-Box**<br>• Der Mindestabstand von 60 – 80 cm wird bei der Neugestaltung miteinbezogen. |

| | |
|---|---|
| | Die grosse Anzahl an verschiedenen Kartenelemente macht es unmöglich, die Lesbarkeit aller Elemente aus dieser Distanz zu gewährleisten. Es wird darauf geachtet, dass die wichtigsten Elemente pro Zoomstufe auf diese Distanz lesbar sind. |
| **Kartografische Generalisierung** | **Studie ETH**<br>• Ist der Generalisierungsgrad stets dem Massstab angepasst?<br>• Wird gemäss den kartografischen Regeln generalisiert?<br>• Welche Objekttypen leiden besonders unter schlechter Generalisierung?<br><br>**OpenStreetMap-in-a-Box**<br>• Die Generalisierung von OSM kann noch verbessert werden. Es fällt auf, dass gewisse Elemente beim heranzoomen ziemlich ruckartig eingeblendet werden. Hier kann noch stärker darauf geachtet werden, dass die verschiedenen Elemente etwas „weicher“ in Erscheinung treten, indem sie bereits bei grösseren Zoomstufen angedeutet werden und in der Grösse (Linienbreite, Grösse von Icons etc.) und Farbe (schwächere Farben bei grossen Zooms) besser dem Generalisierungsgrad angepasst werden. |
| **Minimaldimensionen** | **Studie ETH**<br>• Werden die Anforderungen an die minimalen Strichstärken und Mindest-abstände für Bildschirme eingehalten?<br>• Bei welchen Elementen werden die Minimaldimensionen unterschritten?<br><br>**OpenStreetMap-in-a-Box**<br>• Es fällt auf, dass die Strichstärke von Strassen nur wenig variiert. Bei grossen Zoomstufen sind die Strassen mit relativ dünnen Linien dargestellt und somit eher schlecht erkennbar. Autobahnen und andere wichtige Strassen könne eine wichtige Hilfe für die Orientierung auf einer Karte sein und sollten bereits bei grossen Zoomstufen klar erkennbar sein. |
| **Farben** | **Studie ETH**<br>• Ist auf dem Bildschirm ein ausreichender Farbkontrast gewährleistet oder gibt es unerwünschte Mischeffekte?<br>• Besteht ein verständlicher und sofort einleuchtender Zusammenhang zwischen der Farbe und dem Kartenobjekt?<br>• Sind die Farben auf verschiedenen Bildschirmen gleich dargestellt?<br><br>**OpenStreetMap-in-a-Box**<br>• Die Farbkontraste sind bei OSM nicht optimal. Die Flächen von Wäldern sehr kräftig und bringen Unruhe in die Karte. Vorwiegend bei grossen Zoomstufen sollten die Wälder weniger kräftig sein, damit andere wichtige Elemente wie die Beschriftungen besser zur Geltung kommen.<br><br>• Die Farbwahl bei den Strassen ist wie bereits erwähnt eher schlecht. Es sollte hier darauf geachtet werden, dass die Strassen keine Farben verwenden, die bereits bei Flächen oder anderen wichtigen Elementen verwendet werden. Für die Strassen sollte ein völlig neues Farbkonzept entworfen werden, da diese, wie es der Name schon sagt, bei OpenStreetMap-in-a-Box sehr zentral sind.<br><br>• Allgemein sollte mehr Wert darauf gelegt werden, dass kräftige Farben für die pro Zoomstufe wichtigen Elemente verwendet werden und eher schwächere, Pastell-Farben für Flächen verwendet werden, um den Kontrast zu erhöhen |
| **Symbolisierung** | **Studie ETH**<br>• Können die Symbole eindeutig den bezeichneten Objekten zugeordnet werden?<br>• Lassen die Symbole leicht auf den Objekt-Typ schliessen?<br>• Werden Punktsignaturen in sinnvollem Masse eingesetzt? |

| | |
|---|---|
| | **OpenStreetMap-in-a-Box**<br>• Die Symbolisierung der POIs bei kleiner Zoomstufe ist bei OSM bereits ziemlich gut. Gewisse Symbole wirken teilweise vom Stil her etwas veraltet, lassen aber leicht auf den Objekttyp schliessen.<br><br>• Haltestellen von öffentlichem Verkehr jeglicher Art werden bei OSM mit Quadraten verschiedener Grösse dargestellt. Diese Symbolisierung ist nicht optimal gewählt, da mit einem Quadrat grundsätzlich alles gemeint sein kann. Google Maps verwendet hier klare Symbole welche sogar bis zu einem gewissen Grad die Art des Verkehrsmittels aufzeigen. OSM beschriftet die Haltestellen bereits bei relativ grosser Zoomstufe, wodurch die Karte manchmal etwas überladen wirkt. Hier sollte sowohl bei der Wahl der Symbole, als auch bei der Beschriftung der Ansatz von Google Maps verfolgt werden<br><br>• Ein sehr guter Punkt von OSM gegenüber Google Maps ist, dass die Symbolisierung bei nahen Zoomstufen auch Flächen wie Wald, Zoo, Weingut etc. zusätzlich mit Hintergrundsymbolen erkennbar gemacht werden. Dieser Ansatz sollte weiterverfolgt werden, da er die Unterscheidbarkeit stark verbessert. |
| **Schrift** | **Studie ETH**<br>• Unterstützt die Schrift den Betrachter oder stört sie eher?<br>• Sind die Schriftart und der Schriftschnitt mit ihren Eigenschaften für Bildschirm-karten geeignet?<br>• Ist die Grösse angemessen und sind Abstufungen bezüglich der Objektklassen ersichtlich?<br>• Ist die Schrift bestmöglich platziert und massstabsabhängig generalisiert?<br>• Wie verhält es sich mit der Freistellung der Schrift?<br><br>**OpenStreetMap-in-a-Box**<br>• Die Darstellung der Schriften ist bei OpenStreetMap eines der grossen Probleme. Die Studie weist darauf hin, dass bei Bildschirmbasierten Karten ausschliesslich serifenlose Schriften verwendet werden sollten. Dies ist bei OpenStreetMap-in-a-Box nicht der Fall, was aber an der Konfiguration des Servers liegt. Natürlich ist es zentral dieses Problem zu beheben, damit die Schrift lesbar wird.<br><br>• Bei der Schriftgrösse empfiehlt die Studie die Schriftgrösse von mindesten 12 Punkten einzuhalten. In der Praxis ist diese Minimalgrösse jedoch nicht absolut einhaltbar. Weder OSM noch Google Maps halten sich bei allen Beschriftungen an diese Mindestgrösse und es ist durch die Grosse Anzahl an unterschiedlichen Elementen auch nicht möglich, sich immer an diese Richtline zu halten. Bei der Gestaltung der Karte wird jedoch darauf geachtet, dass pro Zoomstufe die wichtigsten Elemente immer mindestens eine Schriftgrösse von 12 Punkt haben. Unwichtigere Elemente können teilweise in kleinerer Schriftgösse dargestellt sein, werden aber beim heranzoomen grösser und somit auch gut lesbar.<br><br>• Die Freistelllung der Schriften wurde bei OpenStreetMap-in-a-Box noch gar nicht umgesetzt. Sie ist jedoch ein wichtiges Element um den Benutzer nicht mit Information zu überfordern. Wenn bei weitem Zoom nicht darauf geachtet wird, dass die Beschriftung der Ortschaften freigestellt wird, können die Namen der Ortschaften kaum mehr gelesen werden, da es auf kleiner Fläche sehr viele, nahe beisammen liegende Ortschaften geben kann. |

*Tabelle 3: Kartenkriterien ETH Studie - OpenStreetMap-in-a-Box*

### 5.1.2 Neues SLD Konzept

Als Grundlage für das neue SLD Konzept wurden die bestehenden SLD Dateien analysiert um zu erkennen, ob bestehende Strukturen übernommen werden können und ob die Ursache der Probleme bei der Schriftdarstellung bereits hier ersichtlich sind. Die Analyse sowie auch die spätere Umsetzung der Änderungen war sehr aufwendig, da der aktuelle Stand und auch die Umsetzung selber jeweils auf den verschiedenen Zoomstufen der Karte und an verschiedenen geographischen Lagen getestet werden musste. Durch das aufwendige Testen hat sich schnell herausgestellt, dass es sich lohnt, die SLD Dateien von Grund auf neu aufzubauen. Durch Kommentare in den Dateien wurde versucht, eine übersichtlichere Struktur zu erzeugen und somit spätere Erweiterungen und Anpassungen zu vereinfachen. Da es im Rahmen dieser Arbeit nicht möglich gewesen wäre das Rendering der gesamten Schweizer Karte zu testen, wurde der Fokus auf städtische Gebiete gelegt. In diesen Gebieten ist die Dichte an vorhandenen Kartenelementen viel höher und somit die Lesbarkeit der Karte am besten überprüfbar. Um die ländlichen Regionen nicht zu vernachlässigen wurden aber auch Tests in diesen Regionen durchgeführt.

Bei der Umsetzung des neuen SLD Konzeptes wurde der Fokus auf folgende Kriterien gesetzt:

- **Kontrast**
  Flächenfarben sollen weniger kräftige Farben haben damit sich Ortschaften, Strassen und POIs, welche in kräftigen Farben dargestellt werden, besser hervorgehoben werden.
- **Farbe**
  Die Farbwahl ist für die Lesbarkeit der Karte entscheidend. Es sollen Farben verwendet werden, welche dem Benutzer von anderen Karten bereits vertraut sind und somit gleich mit den entsprechenden Elementen assoziiert werden.
- **Schrift**
  Bei der Beschriftung der Ortsnamen und Strassennamen ist es wichtig, dass wichtigere Elemente bereits an der Schriftgrösse und Schriftfarbe erkennbar sind. Es sollen nur serifenlose Schriften eingesetzt werden. Die Schriftgrösse von 12 Punkt sollte so selten wie möglich unterschritten werden.
- **Generalisierung**
  Der Aufbau der SLD Regeln hält sich an die verfügbaren Zoomstufen der Karte. Je näher man heranzoomt, desto mehr Elemente werden sichtbar. Es wird Wert darauf gelegt, dass dass Erscheinen dieser neuen Elemente möglichst fliessend ist. Um dies zu ermöglichen können Farben von Landflächen beim heranzoomen etwas stärker werden, Symbole und Schriften größer werden, Linien von Strassen und Flüssen stärker und breiter werden und allgemein Elemente Schritt für Schritt zugeschaltet werden

### 5.1.3 Styled Layer Deskriptoren

Styled Layer Deskriptor (SLD) ist ein XML Schema, das vom Open Geospatial Consortium (OGC) definiert worden ist. Es dient dazu, das Aussehen von Vektor- und Rasterbasierten Kartengrafiken zu definieren. Der Web Map Service (WMS) von Geoserver interpretiert die Definitionen in solchen SLD Dateien und generiert daraus die Grafik.

Beispiele für eine SLD Regel (Gletscher Darstellung für Zoomstufe 1:50 Millionen bis 1:20 Millionen):

```
<Name>OSM_Landuse</Name>
<UserStyle>

<FeatureTypeStyle>

	<Rule>
		<Name>Glacier 50M - 20M</Name>

		<ogc:Filter>
			<ogc:PropertyIsEqualTo>
				<ogc:PropertyName>typ</ogc:PropertyName>
				<ogc:Literal>glacier</ogc:Literal>
			</ogc:PropertyIsEqualTo>
		</ogc:Filter>

		<MinScaleDenominator>20000000</MinScaleDenominator>
		<MaxScaleDenominator>50000000</MaxScaleDenominator>

		<PolygonSymbolizer>
			<Fill>
				<CssParameter name="fill">#D1F1F1</CssParameter>
			</Fill>
			<Stroke>
				<CssParameter name="stroke">#C6EDED</CssParameter>
				<CssParameter name="stroke-width">1</CssParameter>
			</Stroke>
		</PolygonSymbolizer>

		<TextSymbolizer>
			<Label>
				 <ogc:PropertyName>name</ogc:PropertyName>
			</Label>
			<Font>
				<CssParameter name="font-size">11</CssParameter>
				 <CssParameter name="font-family">Arial</CssParameter>
			</Font>
			<Fill>
				...
			</Fill>
		</TextSymbolizer>

	</Rule>

</FeatureTypeStyle>

</UserStyle>
</NamedLayer>
```

Im Beispiel erkennt man die wichtigsten Elemente einer SLD Regel:

- **Filter**
  Über das OGC Filter Encoding Schema können logische Operatoren definiert werden und somit genau bestimmt werden, für welche Features der Karte eine Regel gültig ist
  → Hier wird eine Regel für das Feature vom Typ 'Glacier' (Gletscher) definiert
- **Min- und MaxScaleDenominator**
  Der MinScaleDenominator definiert ab welcher Zoomeinstellung die Regel in Kraft tritt
  Der MaxScaleDenominator definiert bis zu welcher Zoomeinstellung die Regel gilt
- **PolygonSymbolizer**
  Da es sich beim Gletscher um eine darzustellende Fläche handelt, wird hier ein PolygonSymbolizer definiert. Hier wird also die Füllfarbe und die Rahmenfarbe, sowie die Rahmendicke definiert. Je nach Kartenobjekt-Typ (Punkt, Linie, Fläche) wird hier ein anderer sogenannter Symbolizer eingesetzt
  - PointSymbolizer
  - LineSymbolizer
  - PolygonSymbolizer
- **TextSymbolizer**
  Um ein Feature zu beschriften wird ein TextSymbolizer eingesetzt. Der „Label"-Tag definiert den Inhalt des Labels. Hier wird mit PropertyName = name festgelegt, dass der in der Datenbank vorhandene Name des Gletschers angezeigt werden soll. Durch zusätzliche Attribute wie „Font", „Fill" oder „Halo" können Schriftart, Schriftfarbe oder Schriftumrandung festgelegt werden.

### 5.1.4 Java Code und Mapping von OpenStreetMap-in-a-Box

Für die Darstellung von Gebäuden wurde der bestehende Code von OpenStreetMap-in-a-Box so angepasst, dass eine neue Datenbanktabelle für die Speicherung der Gebäude Flächen verfügbar ist.

Folgende Anpassungen wurden am Java Code gemacht:

- Im Paket 'db' wurden neue Methoden und Handler für Elemente vom Typ Building eingeführt:
  Beim Interface RelationHandlingStrategy.java wurde eine neue Methode eingeführt und in den Klassen, welche das Interface verwenden überschrieben:

  → `public void addBuilding(Relation relation, HashMap<String, StringBuffer> statements, GeomStrategy geom);`

- Die Methode 'createStatement' in der Klasse RelationHandling.java wurde für den Aufruf der neuen Strategy Methode erweitert.

```
→ …} else if(relation.gisMapping.table_name.equals(DBConstants.BUILDING)){
      relationStrategy.addBuilding(relation, statements,
      strategy.get(relation.relationtype));
}
```

- Bei der Klasse DBConstants.java wurde eine entsprechende Konstante für Buildings eingeführt
- In der Klasse DataInitialisation.java wurde Building zu den Relation Tables hinzugefügt

Zudem wurde die SQL-Datei osm_create.sql bzw. dbrefresh.sql, welche die Tabellen auf der Datenbank erzeugen und updaten, mit den entsprechenden PostgreSQL Anweisungen erweitert.

Im Verzeichnis 'geserver_data' Verzeichnis wurde der neue FeatureType 'osm_ds_building' und die dazugehörige SLD Datei erstellt. Es wurden auch die Konfigurationsfiles service.xml und catalog.xml

angepasst.

Schliesslich wurde in der Mapping Datei mapping.xml das neue Mapping für die Gebäude eingeführt.

Wie man sieht müssen für die Einführung einer neuen Tabelle bei OpenStreetMap-in-a-Box an vielen Stellen im Code und in anderen Dateien Anpassungen gemacht werden. In einem Projekt, welches parallel zu diesem läuft, wird der Code von OpenStreetMap-in-a-Box einem ausführlichen Refactoring unterzogen. Wenn dieses Refactoring abgeschlossen ist, sollten solche Anpassungen nur noch zentral in der Mapping-Datei gemacht werden müssen.

### 5.1.5 Anpassungen Server Architektur

Das Problem mit der Darstellung der Schriften hatte eigentlich zwei unterschiedliche Ursachen. Einige SLD Dateien verwendeten bei den Schriftdefinitionen das Literal 'Sans' als Font Type, was gemäss CSS Definition nicht verwendet werden darf. Diese Fehler wurden behoben, aber die Schriften konnten trotzdem nicht korrekt dargestellt werden. Der Grund dafür war, dass die verwendete Windows Schrift 'Arial' auf dem Linux Server nicht vorhanden war. Um dies zu beheben musste der Linux Server entsprechend konfiguriert werden:

1. Normalerweise ist es relativ einfach, Windows Schriften auf Linux zu installieren. Bei der Linux Version von Centos ist dies etwas aufwendiger. Die Lösung dafür ist die Nutzung einer kleinen Software namens Freetype, welche auf dem System installiert wird. Die Software ermöglicht das korrekte Rendering der Schriften. Nach der Installation der Software müssen nur noch die Windows Schriften heruntergeladen und installiert werden.
2. Auch nach korrekter Installation der Windows Schriften auf dem Server wurden diese immer noch nicht richtig auf der Webseite dargestellt. Der Grund dafür ist, dass der Geoserver nur diese Schriften darstellen kann, welche auch der zugrundeliegenden Java Virtual Machine bekannt sind. Die zuvor installierten TTF (True Type Fonts) Dateien müssen hierfür in das 'fonts' Verzeichnis der Java Runtime Environment (JRE) kopiert werden.
3. Centos verwendet per Default OpenJDK, ein Open Source JDK, welches nicht gleich aufgebaut ist, wie das offizielle JDK von Sun. Geoserver empfiehlt auf seiner offiziellen Webseite, dass für eine optimale Performance das JDK von Sun verwendet werden sollte. Somit wurde das bestehende OpenJDK mit dem von Sun ersetzt und die Schriften dieser JVM bekannt gemacht.

## 5.2 Implementation Webseite

### 5.2.1 Funktionale Anforderungen

| Funktionalität | Beschreibung | Priorität | Abhängigkeit |
|---|---|---|---|
| **Kartendarstellung über WMS** | Über eine HTTP WMS Anfrage an den Geoserver werden die Map Tiles empfangen und dargestellt. | Hoch | Web Map Service Geoserver (WMS) |
| **Bedienung der Karte (Zoom, Pan)** | Die Karte bietet die wichtigsten Bedienelemente. Es kann herein- und herausgezoomt werden und in alle Richtungen navigiert werden. | Hoch | Web Map Service Geoserver (WMS) |
| **Suchfunktion** | Mit Hilfe eines Eingabefeldes kann nach verschiedenen Elementen auf der Karte gesucht werden. | Hoch | Web Feature Service Geoserver (WFS) |
| **Anzeigen eines Suchresultates auf der Karte** | Durch Klick auf ein Suchresultat zeigt die Karte das Resultat an. Die Zoomeinstellung wird abhängig vom Typ des Resultates (City, Town, Waterway etc.) angepasst. Ein Marker visualisiert den Ort. | Hoch | - |

*Tabelle 4: Funktionale Anforderungen - Webseite*

### 5.2.2 Nicht-Funktionale Anforderungen

**Konnektivität zum Geoserver**

- Über einen Webbrowser muss eine Verbindung zum Internet aufgebaut werden können.

### 5.2.3 Design Mockup

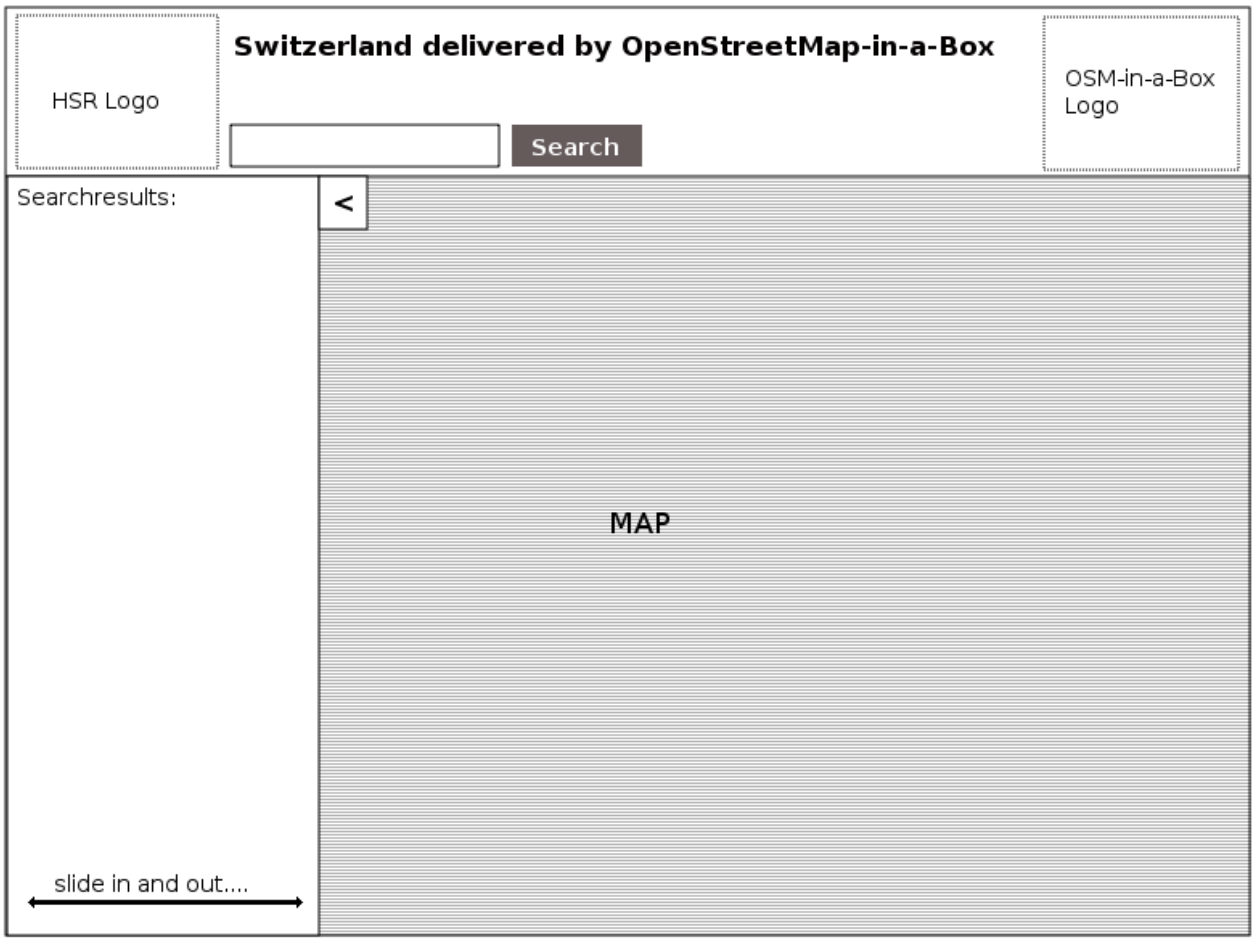


*Abbildung 6: Design Mockup Webseite*

### 5.2.4 Technologien

**OpenLayers**
OpenLayers ist eine Javascript Library, welche für die dynamische Darstellung von Karten auf Webseiten genutzt werden kann. Mit Hilfe dieser Library wird die OpenStreetMap-in-a-Box Karte auf der Seite dargestellt und die entsprechenden Kontrollfunktionen wie Zoom und Paning zur Verfügung gestellt.

**JQuery**
JQuery ist eine Javasript Library welche die Programmierung von Javascript Funktionen vereinfacht. Unter anderem bietet diese Library gute Unterstützung um DOM Objekte zu animieren. Mit Hilfe dieser Funktionen wurde auch die Anzeige der Suchresultate umgesetzt, welche bei Bedarf ein- und ausgeblendet werden kann.

**WFS Geoserver / GeoJSON**
Bei der Frage welcher Service für die Suche verwendet werden soll, war die Tatsache, dass der interne WFS von Geoserver ausschliesslich auf den vorhandenen Daten Abfragen macht, von entscheidender Bedeutung. Andere Services wie der Gazetteer von OpenStreetMap oder der WFS von GeoNames (http://www.geonames.org) haben den Nachteil, dass diese auf einer eigenen Datenbank Abfragen machen und es daher möglich wäre, dass ein Suchresultat auf der Karte von OpenStreetMap-in-a-Box gar nicht

vorhanden ist. Beim Service von GeoNames wäre zudem nicht gewährleistet, dass die Koordinaten eines Suchresultates genau mit den Koordinaten der Daten in der eigenen PostgreSQL Datenbank übereinstimmen. Die Folge wäre, dass der Marker, der beim Klick auf ein Suchresultat die Position des Resultates anzeigt, nicht genau dort ist, wo er eigentlich sein sollte.

Der WFS von Geoserver hat gegenüber den anderen Services den Nachteil, dass die Abfragemöglichkeiten bei einer Suchanfrage stark eingeschränkt sind. Bereits bei relativ einfachen Bedingungen wie beispielsweise das Ausschliessen von doppelten Resultaten durch eine Distinct-Abfrage kommt man bei der aktuellen Version von Geoserver an seine Grenzen. Als Lösungen für dieses Problem können jedoch spezielle Views auf der Datenbank generiert werden, welche die Daten bereits in der gewünschten Form liefern. Diese Views werden dann dem Geoserver bekannt gemacht, wodurch eine Suchanfrage auf diesen Views gemacht werden kann. Somit kann der Nachteil der eingeschränkten Abfragemöglichkeiten umgangen werden und ist bei der Wahl des Services nicht mehr zentral.

### 5.2.5 Suchfunktion

#### Suchalgorithus

Um einen geeigneten Suchalgorithmus zu entwickeln wurden zuerst das Suchverhalten von Google Maps und OpenStreetMap getestet. Es stellte sich schnell heraus, dass eine einfache Wildcard-Suche hier nicht sinnvoll ist. Bei einer solchen Suchanfrage würden alle Einträge in der Datenbank zurückgegeben, welche den Suchstring in irgendeiner Weise enthalten, also auch als Teilstring eines Wortes. Beispielsweise würde bei einer Suchanfrage mit dem Suchstring „dorf“ jeder Datenbankeintrag, der das Wort „dorf“ enthält zurückgegeben. Von den unzähligen Möglichkeiten wie z.B. „Urdorf“, „Dübendorf“ wäre es für den Benutzer kaum möglich genau diese Ortschaft, die er gesucht hat herauszufiltern. In diesem Beispiel würde der Benutzer erwarten, dass die Ortschaft namens „Dorf“, die es tatsächlich gibt, in den Suchresultaten aufgelistet ist, und dies möglichst weit oben in der Liste der Resultate.

Es muss also davon ausgegangen werden, dass der Benutzer weiß wie die Ortschaft oder die Strasse die er sucht heißt. Nur so ist es möglich, die Anzahl der Suchresultate in einem sinnvollen Rahmen zu halten. Um doch einen gewissen Freiraum bei der Eingabe zu ermöglichen, reicht es wenn das eingegebene Wort als ganzes gefunden wird, d.h. bei der Eingabe von Rapperswil, welches in der Datenbank unter dem Namen „Rapperswil-Jona“ gespeichert ist, wird die Suche trotzdem erfolgreich sein. Bei der Eingabe „Langnau“ wird somit auch „Langnau am Albis“ zurückgegeben.

Die Abfrage wird Case-Insensitive durchgeführt.

#### Suchanfrage

Die HTTP Anfrage an den WFS des Geoservers besteht aus der URL des Servers und verschiedenen Parametern in Form von Key – Value Paaren:

| **Url** | http://<URL des Servers>/geoserver/wfs | |
|---|---|---|
| | | |
| **Key** | **Value** | **Beschreibung** |
| **request** | 'getfeature' | Request Typ:<br>• getfeature → fordert Objekte des Feature Types an |
| **typename** | <feature type> | Bestimmt den Feature Typ auf dem die Anfrage gemacht wird. Der Feature Typ muss beim Geoserver vorhanden sein. Bei osm2gis wird für jede Tabelle in der Datenbank ein entsprechender Feature Typ erstellt. Für die Suchanfrage werden noch zusätzliche Views auf der Datenbank erstellt, damit die Suche auf diesen Views gemacht werden kann (siehe 'Datenbank Views – Lookup-Tabellen') |
| **service** | 'wfs' | Service Typ |

| | | |
|---|---|---|
| **version** | '1.1.0' | Version:<br>• 1.0.0 oder 1.1.0. Bei Fertigstellung der Arbeit ist 1.1.0 die letzte Version |
| **outputformat** | json | Format der Antwort. per Default werden wird die Antwort eines WFS Request bei Geoserver im GML (Geographic Markup Language) verschickt. Um die Auswertung der Daten zu erleichtern wird hier GeoJSON verwendet. |
| **filter** | <ogc:Filter> | Filter für die Eingrenzung der Antwort. Die Eingrenzung wird über OGC Filter im XML Format gemacht (siehe nachfolgendes Beispiel) |

*Tabelle 5: WFS Anfrage Parameter*

## Suchfilter

Beispile für einen OGC Filter mit einer 'Or' Verknüpfung um mehrere Bedingungen einzubeziehen:

```
<Filter>
        <Or>
        <PropertyIsLike wildCard="*" singleChar="." escape="!" matchCase="false">
                <PropertyName>name</PropertyName>
                <Literal>*-basel</Literal>
        </PropertyIsLike>
        <PropertyIsLike wildCard="*" singleChar="." escape="!" matchCase="false">
                <PropertyName>name</PropertyName>
                <Literal>basel-*</Literal>
        </PropertyIsLike>
        </Or>
</Filter>
```

| OGC Element | Erklärung |
|---|---|
| **Filter** | Definiert den Anfang einer Filterdeklaration |
| **Or** | Logisches OR um mehrere Bedingungen zu verknüpfen |
| **PropertyIsLike** | Bedingung des Filters. 'PropertyIsLike' entspricht einer Wildcard-Suche. Andere Möglichkeiten wären 'PropertyIsEqual', 'PropertyIsGreaterThan' etc.<br>Parameter:<br>• wildcard:<br>Zeichen wird als Wildcard verwendet<br>• singlechar:<br>Zeichen wird als Platzhalter für einen beliebiges Zeichen verwendet<br>• escape:<br>Zeichen um die zuvor definierten wildcard- und singlechar-Zeichen trotzdem als Literal nutzen zu können<br>• matchCase:<br>Ermöglicht Case-Insensitive Suche indem auf 'false' gesetzt. |
| **PropertyName** | Definiert auf welchem Attribut des Feature Typ die Abfrage gemacht werden soll. Es handelt sich dabei um den Spaltenname der Datenbanktabelle, auf der die Abfrage gemacht wird. |
| **Literal** | Nach dem hier definierten Suchbegriff wird gesucht. Im Beispiel wird nach Namen gesucht, welche „*-basel“ oder „basel-*“ enthalten. D.h. Dass hier Resultate wie „Klein-Basel“, „Basel-Land“ oder „Basel-Stadt“ zurückgegeben werden. |

*Tabelle 6: XML Elemente eines OGC Filters*

Auf die gezeigte Weise muss also basierend auf dem eingegebenen Suchstring für jeden Feature Typ, also für jede Datenbanktabelle bzw. View, auf der gesucht werden soll, ein Filter erzeugt werden. Für jeden Feature Typ wird somit eine HTTP Anfrage an den Server geschickt und ausgewertet. Die folgende Grafik zeigt, wie ein ein solcher Filter aufgebaut wird, um den definierten Suchalgorithmus umzusetzen.

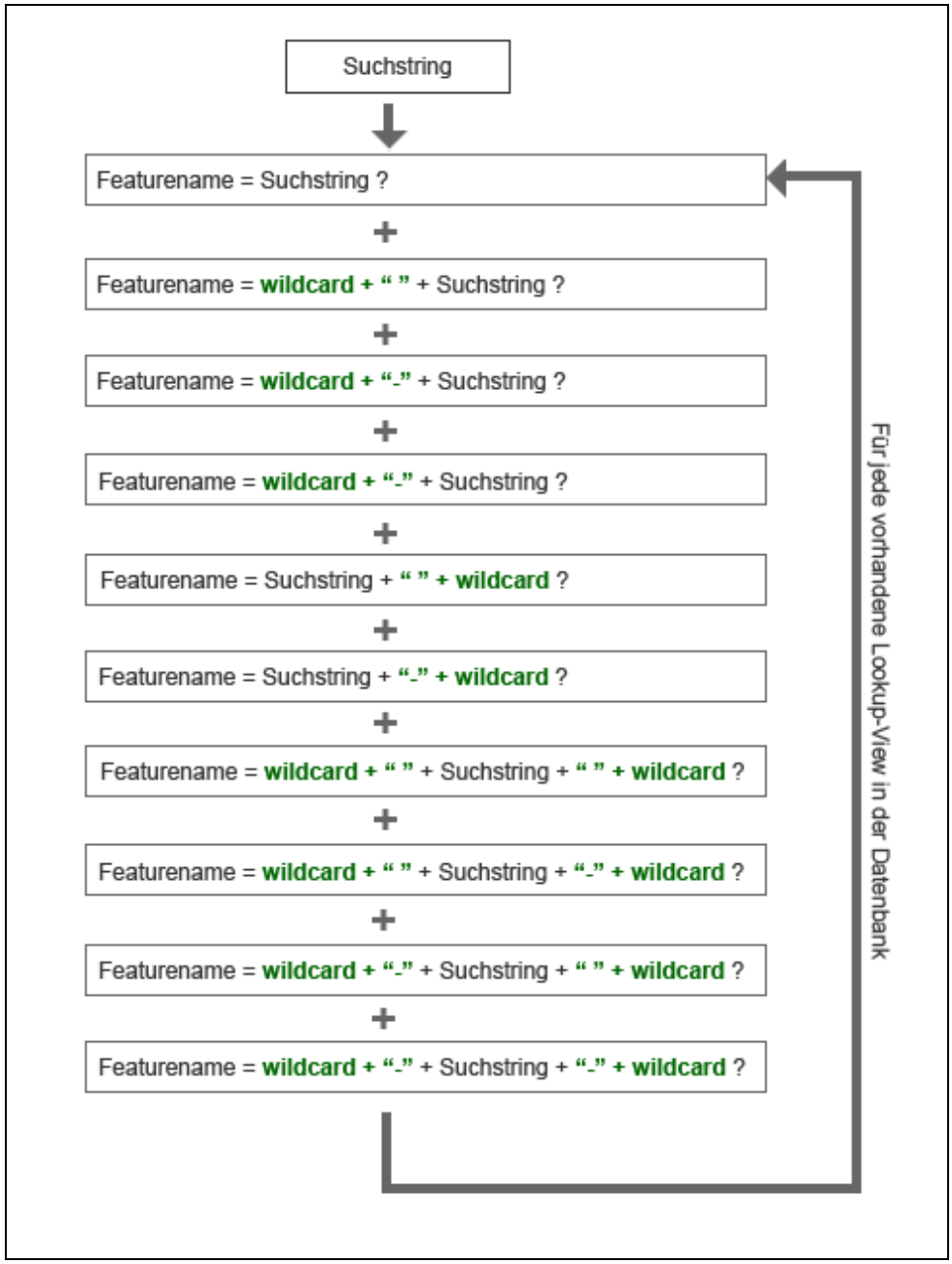


*Abbildung 7: Aufbau der Filter für den Suchalgorithmus*

Mit dem so entstehenden Filter werden also alle Datenbankeinträge gesucht, die entweder den Suchstring selber als Name haben, oder vor bzw. nach dem Suchstring eine Leerzeichen oder einen Bindestrich haben.

## Datenbank Views – Lookup-Tabellen

Die Möglichkeiten der OGC Filter sind nicht so gut wie man es sich von einer gewöhnliche Datenbankanfrage gewöhnt ist. Es ist beispielsweise nicht möglich, doppelte Resultate in der Suchanfrage herauszufiltern. Bei einer Datenbankanfrage wäre dies über einen „SELECT DISTINCT“ oder über „GROUP BY“ sehr einfach möglich. Bei der vorliegenden Suchfunktion ist es jedoch unverzichtbar, bei gewissen Anfragen doppelte Resultate auszuschliessen. Eine Strasse wird Beispielsweise in der Datenbank als eine Sammlung von Strassensegmenten gespeichert. Für die jede Strasse sind also sehr viele Einträge in der Datenbank, welche meistens auch denselben Namen besitzen. Bei einer Suchanfrage würden also all diese Einträge zurückgegeben und als Resultat aufgelistet. Auch eine sortierte Rückgabe des Resultates ist zur Zeit noch nicht möglich. Das Attribut 'SORT BY' sollte zwar seit kurzem verfügbar sein, konnte aber während dieser Arbeit noch nicht erfolgreich getestet werden. Sortierung ist ebenfalls sehr wichtig bei der Suchfunktion, da die Reihenfolge, in der dem Benutzer die Resultate präsentiert werden, ebenfalls von Bedeutung ist. Sucht man Beispielsweise mit dem Suchstring „rapperswil“, dann wird unter anderem „Rapperswil-Jona“ und „Rapperswil (BE)“ gefunden. Rapperswil bei Bern ist ein kleines Dorf, Rapperswil-Jona hingegen eine Stadt. Es macht also Sinn wenn man bestimmte Resultate höher Priorisieren kann als andere, denn es ist zu erwarten dass bei einer Suche, die mehrere Ortschaften liefert, zuerst die grösseren Städte und erst dann die Dörfer aufgelistet werden.

Als Lösung für diese Einschränkungen bei den OGC Filtern können auf der Datenbank Views, sogenannten Lookup-Tabellen erstellt werden. Diese muss der Geoserver kennen, sie müssen zu den Feature Typen des Servers hinzugefügt werden.

Für die Suchfunktion wurden Views basierend auf folgenden Kriterien erstellt:

- Wenn die Tabelle Elemente enthält, die aus mehreren Segmenten zusammen gesetzt werden, werden doppelte Einträge herausgefiltert. Dies gilt für:
  - Strassen → Tabelle 'road'
  - Flüsse → Tabelle 'waterway'
- Für jede View wird eine Sortier-Reihenfolge festgelegt. Die Views werden absteigend nach der Wichtigkeit des Resultat-Typs (Typ ist hier z.B. bei Ortschaften 'city', 'town', 'village' etc) sortiert.
- Einträge die keinen Namen haben (leere Einträge können vor allem bei Strassensegmenten und Flusssegmenten vorkommen) werden herausgefiltert.
- Jede Tabelle besitzt eine Spalte namens 'keyvalue'. Diese Spalte hat einen spezielle PostGreSQL Typ namens 'hstore'. In solchen Spalten können mehrere Key – Value – Paare gespeichert werden. OpenStreetMap-in-a-Box nutzt dies um diverse Zusatzinformationen zu speichern, die in den Daten von OpenStreetMap vorhanden sind. Beispielsweise haben viele POIs wie z.B. Restaurants als Zusatzinformation die Webseite des Restaurants gespeichert. Bei Ortschaften werden Information zur Umgebung gespeichert, also z.B. bei der Ortschaft „Rapperswil-Jona“, dass es sich um das Rapperswil im Kanton St. Gallen handelt.
  Bei der aktuellen Suche werden genau die beiden beschriebenen Beispiele miteinbezogen und auf den Views wird der entsprechende Wert aus der 'keyvalue' Spalte ausgelesen und in der View dargestellt, falls er vorhanden ist.

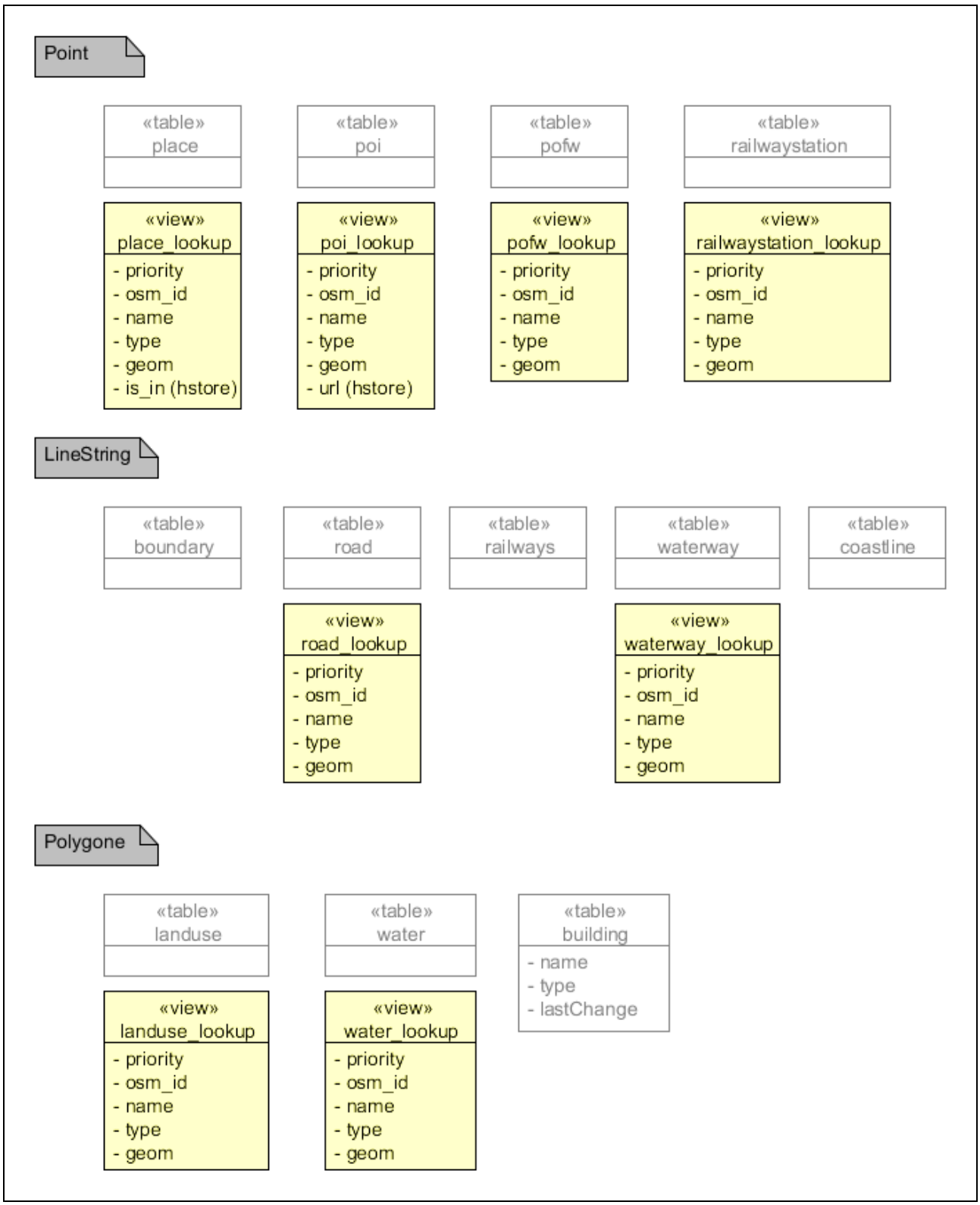


*Abbildung 8: Domainmodell - Datenbank Views*

### 5.2.6 Integration der Views in OSM-in-a-Box

Um dem Benutzer von OSM-in-a-Box die Implementation einer Suchfunktion zu erleichtern, sind die erstellten Views in die Software integriert worden. Dazu wurde die SQL Datei, welche die Tabellen von OSM-in-a-Box erstellt mit den entsprechenden SQL Anweisungen erweitert und die Konfiguration des Geoservers angepasst.

Die folgende SQL Anweisung zeigt dies am Beispiel der Tabelle place:

```
CREATE VIEW place_lookup AS
    SELECT CASE WHEN typ = 'country' then 1
                WHEN typ = 'city' then 2
                WHEN typ = 'town' then 3
                WHEN typ = 'suburb' then 4
           WHEN typ = 'village' then 5
                WHEN typ = 'state' then 6
                WHEN typ = 'region' then 7
                WHEN typ = 'hamlet' then 8
                ELSE 1000
           END AS index,
           osm_id, name, typ, (keyvalue->'is_in') AS is_in, geom
    FROM place
    WHERE name != ''
    ORDER BY index;

INSERT INTO geometry_columns(
            f_table_catalog, f_table_schema, f_table_name, f_geometry_column,
coord_dimension, srid, "type")
VALUES ('', 'public', 'place_lookup', 'geom', 2, 4326, 'POINT');
```

Das INSERT Statement ist notwendig, damit der Geoserver die View auch verarbeiten kann. Der Geoserver kann nur Tabellen verarbeiten, die auch eine Spalte vom Typ 'geometry' haben. Die Tabelle 'geometry_columns' führt sozusagen Buch über die Tabellen des Geoservers.

Nun muss noch für jede View im Geoserver Verzeichnis ein Feature Type erstellt werden, indem ein entprechender Eintrag im Verzeichnis 'featureType' gemacht wird.

### 5.2.7 Verarbeitung der Suchresultate

Defaultmäßig werden Suchanfragen an den WFS als GML (Geometry Markup Language) Objekte zurückgeliefert. Diese Dateien enthalten die Resultate in einer XML Struktur, welche relativ mühsam ausgewertet werden muss. JSON ist ein neueres, schlankeres Datenaustauschformat, welches als Alternative zu XML verwendet werden kann und insbesondere bei der Auswertung Vorteile bietet. JSON Dateien beinhalten Name/Wert Paare, wobei die Werte auch Listen von Werten sein können. Somit kann bei der Auswertung direkt über den Attributnamen auf den Wert zugegriffen werden. Falls ein Attribut eine Liste von Werten enthält, können diese über eine einfache Schlaufe ausgelesen werden, genau gleich wie dies sonst bei Arrays gemacht wird.

Geoserver bietet Plugins für die Verwendung von GeoJSON, eine Variante von JSON welche speziell für die Verwendung von geographischen Datenstrukturen entwickelt wurde. Da diese Plugins nicht standardmässig in Geoserver enthalten sind, müssen diese Plugins heruntergeladen und installierte werden, was jedoch durch die gute Struktur von Geoserver sehr einfach ist.

## 5.3 Implementation Android Applikation „Pois R Us“

### 5.3.1 Funktionale Anforderungen

| Funktionalität | Beschreibung | Priorität | Abhängigkeit |
|---|---|---|---|
| **Kartendarstellung über WMS** | Über eine HTTP WMS Anfrage an den Geoserver werden die Map Tiles empfangen und dargestellt. | Hoch | Web Map Service Geoserver (WMS) |
| **Bedienung der Karte (Zoom, Pan)** | Die Karte bietet die wichtigsten Bedienelemente. Es kann herein- und herausgezoomt werden und in alle Richtungen navigiert werden. | Hoch | Web Map Service Geoserver (WMS) |
| **Suchfunktion** | Mit Hilfe eines Eingabefeldes kann nach verschiedenen Elementen auf der Karte gesucht werden. | Hoch | Web Feature Service Geoserver (WFS) |
| **Darstellung aktueller Standort** | Der aktuelle Standort des Benutzers kann auf der Karte dargestellt werden. | Hoch | GPS des Mobiltelefons |

*Tabelle 7: Funktionale Anforderungen - Pois R Us*

### 5.3.2 Nicht-Funktionale Anforderungen

**Konnektivität zum Geoserver**

- Das Mobiltelefon, auf die Applikation läuft, muss eine Verbindung zum Internet aufbauen können um die WMS und WFS Anfragen zu verschicken.

**Externe Bibliotheken**

- Für die Kartendarstellung kann eine externe Java Bibliothek verwendet werden um die Grundlegende Funktionalität der Karte zu implementieren.Vor der Implementation muss abgeklärt werden, ob es bereits solche Bibliotheken für Android gibt.
- Um die Abhängigkeiten möglichst gering zu halten sollten ansonsten möglichst wenig externe Bibliotheken verwendet werden.

**Auslieferung**

- Die Applikation kann über die ausgelieferte .apk Datei auf dem Android Mobiltelefon installiert werden. Die .apk Datei kann über das Internet heruntergeladen werden
- Eine spezielle Installationsanleitung ist nicht nötig, da es sich bei der Installation um ein Standardverfahren von Android handelt.

### 5.3.3 Evaluation für die Implementation der WMS Anfrage

Ein zentraler Punkt für die Implementation von „Pois R Us“ ist die Implementation der WMS Anfrage. Android ist eine relativ neue Systemumgebung und daher gibt es noch wenige vergleichbare Applikationen auf dem Markt. Karten-Viewer gibt es zwar bereits von Google Maps und auch von OpenStreetMap, beide Versionen arbeiten jedoch nicht mit einem WMS Aufruf um die Karten darzustellen.

Bei der Evaluation wurden drei verschiedene Lösungsansätze genauer betrachtet:

1. Anleitung für einen einfachen WMS Client für Android – AndroidGPSBlog
2. Vespucci – omseditor4android: Der erste OpenStreetMap Editor für Android
3. Mobile GMaps – Freie Sofware für die Darstellung von Karten auf Mobiltelefonen

Die Evaluation ergab folgendes Resultat:

#### AndroidGPSBlog

Webseite: http://androidgps.blogspot.com/2008/09/simple-wms-client-for-android.html

Auf der Webseite gibt es eine Anleitung wie ein WMS Client auf Android implementiert werden kann. Sie enthält auch Code-Beispiele der wichtigsten Klassen, die für die Implementation notwendig sind. Es wurde eine Testapplikation erstellt, die diesen Ansatz umsetzt.

Der Ansatz dieser Lösung ist folgender: Basierend auf der bereits vorhandenen Android API von Google Maps wird der Viewer aufgebaut. Es wird also einfach eine Karte von Google Maps erzeugt, um danach einen neuen Layer auf diese Karte zu legen, dessen Inhalt über eine WMS Anfrage generiert wird. Somit navigiert man eigentlich auf der Karte von Google Maps, nur dass man den eigenen WMS Layer darübergelegt hat und somit dieser sichtbar ist.

Schon der Ansatz dieser Lösung zeigte schnell, dass sie nicht für „Pois R Us“ in Frage kommt. Es handelt sich hier eher um einen Workaround, indem die API von Google Maps für die Darstellung und Navigation der Karte „missbraucht“ wird. Zudem hat sich bei den Tests ergeben, dass die Performance dieser Lösung relativ schlecht ist und nicht den Anforderungen von „Pois R Us“ entsprechen würde.

#### Vespucci

Webseite: http://code.google.com/p/osmeditor4android/wiki/Overview

Der Android Editor von Vespucci liegt zur Zeit in der Version 0.6.1 vor. Es handelt sich hier um einen Editor für OSM. Dabei werden die Daten über die von OSM zur Verfügung gestellten XML-Dateien importiert und können dann bearbeitet werden. Die zugrundeliegende Map ist also nicht eine gerenderte Karte wie bei einer WMS Anfrage, sondern nur die Darstellung dieser XML Daten (Nodes, Ways, Relations). Die Möglichkeit zur Darstellung einer gerenderten Karte ist nicht vorhanden. Der vorliegende Code kann zwar für den Aufbau einer eigenen Applikation hilfreich sein, speziell für die Navigation auf einer Karte (Zoom, Pan etc.), ist aber sehr spezifisch auf der Verarbeitung der XML Daten aufgebaut und eine einfache Anpassung für die Verwendung von WMS Anfragen wäre sehr aufwendig. Es wäre vermutlich sinnvoller, die Applikation selber neu aufzubauen und diesen Code bei der Implementation als Referenz herbei zuziehen. Der zeitliche Rahmen dieses Projektes erlaubt es nicht, eine solche Appliktion von Grund auf neu zu erstellen, wodurch dieser Lösungsansatz ebenfalls nicht in Frage kommt.

#### Mobile Gmaps

Webseite: http://www.mgmaps.com

Mobile Gmaps ist eine frei verfügbare Software welche unter der „Attribution - NonCommercial - NoDerivs Creative Commons Lizenz“ erhältlich ist. Die Software ermöglicht die Darstellung von Karten auf Mobiltelefonen und bei genauerer Betrachtung hat sich herausgestellt, dass auch WMS Aufrufe möglich sind. Leider ist die Unterstützung für solche WMS Aufrufe noch nicht für Android erhältlich. Gemäss Einträgen auf der Seite soll dies für das Jahr 2009 geplant sein.

Auf der Seite wird auch erwähnt, dass die Estnische Firma „Nutiteq“ einen neue Software auf Basis von Mobile Gmaps entwickelt hat.

Webseite: http://www.nutiteq.com/

Nutiteq bietet tatsächlich eine Bibliothek für Android an, welche alle Anforderungen entsprechen würde. Es können Karten über WMS Aufrufe dargestellt werden und die Bibliothek bietet zahlreiche Funktionen um die Karte zu bedienen. Die Bibliothek ist auf den ersten Blick aber nur über ziemlich teure Lizenzen erhältlich. Bei genauerem Hinschauen hat sich jedoch herausgestellt, dass die Bibliothek für nicht-kommerziellen Gebrauch unter der GPL Lizenz genutzt werden darf.

#### Fazit

Die ersten beiden Lösungen sind für den Einsatz in diesem Projekt unbrauchbar. Die Lösung von Nutiteq ist optimal geeignet und sollte es ermöglichen, einen Prototypen der Applikation „Pois R Us“ in der vorhandenen Zeit zu entwickeln. Als Folge davon wird auch „Pois R Us unter der GPL Lizenz entwickelt.

### 5.3.4 Aufbau einer Android Applikation in Java

Nachfolgend werden die wichtigsten Komponenten von Android kurz erklärt. Dabei werden speziell diese Komponenten und Abläufe erklärt, die für die Implementation von „Pois R Us“ gebraucht werden können.

#### Activity

Eine Activity baut ein funktionales User Interface auf. Eine Activity ist also eine Ansicht in der Applikation, die beispielsweise eine Liste von Elementen, Bilder oder auch eine Karte darstellt. Eine Android Applikation besteht immer aus mindestens einer Activity. Sie kann aber auch mehrere Activities enthalten, wenn von einer Sicht zur anderen gewechselt werden soll. Es kann also sein dass eine Activity eine Liste mit einer Auswahl präsentiert und durch das Klicken auf einen Listeneintrag wird eine andere Activity aufgerufen und somit sichtbar gemacht.

Activities bei „Pois R Us“:

Die Applikation kann aus zwei Activites aufgebaut werden. Die Haupt-Activity stellt die Karte dar. Daneben braucht es noch eine Activity für die Suchanfrage, da eine Suchanfrage im Stile wie sie auf der Webseite implementiert ist, auf dem kleinen Bildschirm eines Mobiltelefones nicht umsetzbar ist. Der Platz würde nicht ausreichen, um eine gut bedienbare Suche direkt auf der Kartenansicht zur Verfügung zu stellen. Es wird also eine zweite Activity gestartet um die Suchresultate zu präsentieren. Für die Eingabe der Suche könnte man ebenfalls eine eigene Activity erstellen, Android hat jedoch ein eigene SearchManager Klasse, eine Art Serviceklasse, welche die Implementation einer Suchfunktion in der eigenen Software unterstützt.

Es gibt noch 3 weitere wichtige Komponenten bei Android, dies sind Services, Broadcast Receivers und Content Providers. Da diese Komponenten bei Pois R Us nicht vorkommen, wird darauf verzichtet diese näher zu erklären.

#### Intent: Aktivierung von Komponenten

Content Receiver werden aktiviert, wenn sie von einem Content Resolver angesprochen werden. Die anderen 3 Komponenten bei Android, also auch die Activities, werden von asynchronen Nachrichten, sogenannten Intents aktiviert. Wenn die Haupt Activity zum ersten mal gestartet wird, geschieht dies durch einen Start Intent. Danach werden neue Activities indem ein Intent Objekt an die Context.startActivity() Methode gegeben wird. Dieser Intent kann dann von der aufgerufenen Activity gelesen werden. Dies ist daher sehr wichtig, weil einem Intent Informationen mitgegeben werden können, die dann von der aufgerufenen Activity verwendet werden können.

Intents bei „Pois R Us“:

Die Haupt-Activity von „Pois R Us“ ruft die Search Activity auf. Dies geschieht eigentlich indirekt, da

eigentlich die Lokale Suche von Android aufgerufen wird, welche dem den Suchstring dem Intent mitgibt wodurch die die aufgerufenen Activity einen Suchlauf basierend auf der Eingabe machen kann. Nachdem die Resultate der Suche in einer Liste dargestellt worden sind, wird durch den Klick auf ein Suchresultat wieder die MapActivity aufgerufen. Der Intent dieses Aufrufes enthält die Informationen über das ausgewählte Suchresultat, wodurch das Resultat auf der Karte dargestellt werden kann.

## SearchManager – Global- und Local Search

Die Klasse SearchManager bietet nützliche Funktionen um eine Suchfunktion in einer Applikation einzubauen. Der SearchManager muss einfach entsprechend eingerichtet werden. Die Funktion onSearchRequested() startet eine Suche über den SearchManager aus einer Activity heraus. Defaultmäßig wird eine sogenannte „Global Search“ gestartet, welche eine Suche auf der Google Suchmaschine ausführt. Wenn eine solche globale Suche ausgelöst wird, wird automatisch der Browser des Mobiltelefons gestartet und in den Vordergrund gebracht. Auf dem Browser werden dann die Suchresultate angezeigt.

Es soll aber natürlich auch eine Suche auf applikationseigenen Daten gemacht werden können. Für eine sogenannte „Local Search“ muss zuerst in der Activity, welche die Suche auslöst mittels:

```
setDefaultKeyMode(DEFAULT_KEYS_SEARCH_LOCAL);
```

dem SearchManager mitgeteilt werden, dass auf eigenen Daten gesucht werden möchte. Danach muss im Manifest noch angegeben werden, welche Activity bei der lokalen Suche aufgerufen werden soll (siehe Manifest.xml)

## Manifest.xml

Im Manifest wird der Applikation mitgeteilt, welche Komponenten existieren. Für jede Komponente können zusätzliche Informationen über deren Verhalten angegeben werden. Im Weiteren wird hier auch die minimale SDK Version angegeben für welche die Software entwickelt worden ist und es können Zugriffsrechte auf bestimmte Ressourcen für die Applikation angegeben werden.

Manifest von „Pois R Us“:

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="ch.hsr.poisrus.activity" android:versionCode="1"
    android:versionName="1.0">

    <application android:icon="@drawable/icon" android:label="@string/app_name">

        <activity android:name=".MapActivity" android:label="Pois R Us"
            android:launchMode="singleInstance">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
            <meta-data android:name="android.app.default_searchable"
                android:value=".SearchActivity" />
        </activity>

        <activity android:name=".SearchActivity" android:label="Search Results"
            android:launchMode="singleTask">
            <intent-filter>
                <action android:name="android.intent.action.SEARCH" />
                <category android:name="android.intent.category.DEFAULT" />
            </intent-filter>
            <meta-data android:name="android.app.searchable"
                android:resource="@xml/searchable" />
        </activity>

    </application>
    <uses-sdk android:minSdkVersion="3" />
    <uses-permission android:name="android.permission.INTERNET"></uses-permission>
    <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"></uses-permission>

</manifest>
```

*Abbildung 9: Manifest.xml - Pois R Us*

Erklärung der wichtigsten Punkte im Manifest.xml von „Pois R Us“:

- **<activity>** Die Applikation besteht aus zwei Activities, der MapActivity und der SearchActivity. Bei den Metadaten der Activity (<meta-data>) ist ersichtlich, dass die Activites als 'searchable' deklariert sind. Der value-Parameter bei der MapActivity teilt dem SearchManager von Android mit, dass beim Aufruf der lokalen Suchfunktion die SearchActivity aufgerufen werden soll. Der resource-Parameter bei der SearchActivity gibt den Namen der XML Datei an, welche für eine Activity definiert werden muss, damit sie für die lokale Suche verwendet werden kann. In dieser XML Datei kann unter anderem angegeben werden, was für ein Text im Suchfeld stehen soll und was für ein Label das Suchfeld haben soll.
- **<uses-sdk>** Hier wird die minimale SDK angegeben, für welche die Android Applikation entwickelt wurde. Die Zahl 3 entspricht der Android SKD Version 1.5.
- **<uses-permission>** Um einer Applikation Zugriff auf das Internet zu ermöglichen, muss dies unter den Permissions ausdrücklich definiert werden. Da „Pois R Us“ auch GPS verwendet, um die

aktuelle Position des Benutzers anzuzeigen, muss auch der Zugriff auf GPS erlaubt werden.

## Layout

Das Layout einer Activity kann entweder direkt im Java Code aufgebaut werden, schöner ist es aber, wenn die Layouts in einer XML Datei definiert werden und im Code über setContentView() das entsprechende Layout aufgerufen wird.

### Values

Konstanten wie z.B. der Name der Applikation, oder auch die URL der WMS Anfrage, welche in der MapActivity verwendet wird, können zentral in einer XML Datei namens strings.xml definiert werden.

### 5.3.5 Domainmodell

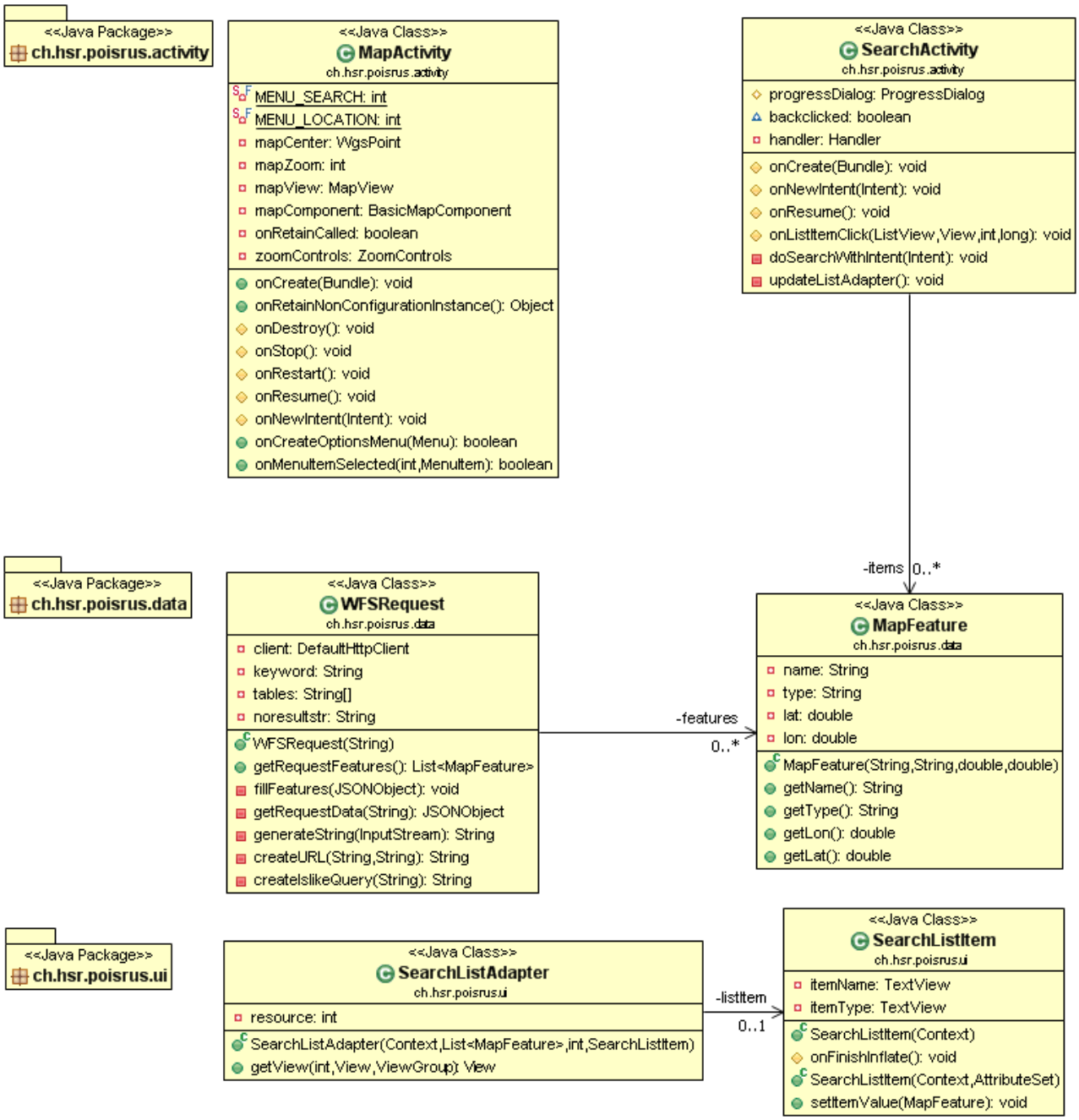


*Abbildung 10: Domainmodell - Pois R Us*

### 5.3.6 Sequenzdiagramm Suchfunktion – WFS Request

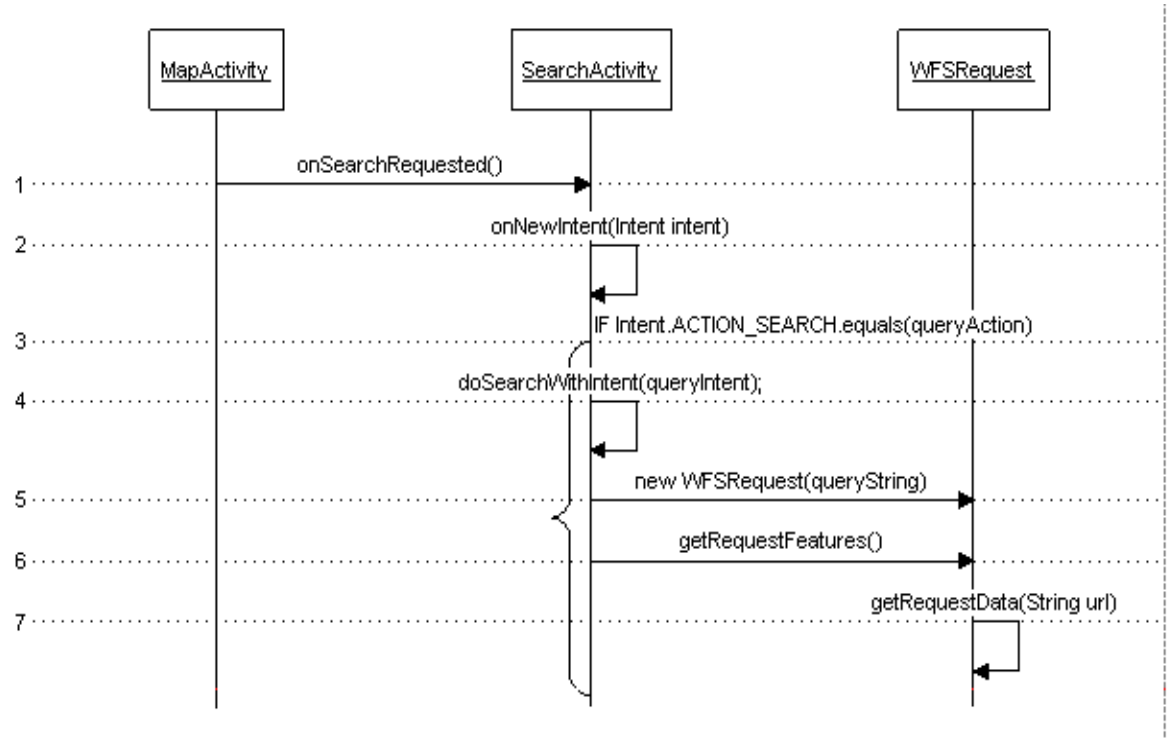


*Abbildung 11: Sequenzdiagramm Suchfunktion - WFS Request*

**Erklärung:**

1. Der Menupunkt 'Search' wurde ausgewählt und Android eigene Funktion 'onSearchRequest()' wird aufgerufen. Die Lokale Suche wird aufgerufen und startet die SearchActivity.
2. Nachdem ein Suchbegriff eingegeben wurde und die Suche ausgelöst worden ist, wird auf der SearchActivity onNewIntent(Intent intent) aufgerufen. Dies ist eine überschriebene Methode der Klasse Activity, von der jede Activity abgeleitet ist. In der überschriebenen Methode wird der neue Intent gesetzt, welcher die eingegebene Suchanfrage als String enthält.
3. Es wird geprüft, ob die Suche ausgelöst wurde und somit der neue Intent gesetzt wurde.
4. Falls Suche ausgelöst wurde und folglich der neue Intent gesetzt wurde, wird über doSearchWithIntent(queryIntent) die Suche gestartet.
5. doSearchWithIntent(queryIntent) erzeugt ein neues WFSRequest Objekt mit dem eingegebenen Suchstring als Parameter.
6. Auf dem erzeugten WFSRequest wird die Methode getRequestFeatures() aufgerufen.
7. Die getRequestFeatures() Methode der SearchActivity ruft auf dem WFSRequest Objekt die getRequestData(String url) Methodeauf, welche die HTTP Anfrage an den Geoserver schickt, das erhaltene JSON Objekt auswertet und für jedes gefunde Suchresultat ein MapFeature erstellt und in eine Liste abfüllt. Diese Liste wird an die aufrufende SearchActivity zurückgeliefert und die Suchresultate werden in einer Liste dargestellt.

## 5.4 Projektmanagement

### 5.4.1 Projektorganisation

### 5.4.2 Meilensteine

| **Meilenstein 1:** | |
|---|---|
| • Projekt- und Zeitplan erstellt<br>• Osm2gis getestet<br>• Kartengrafik optimiert<br>→ neue bzw. überarbeitete SLD Dateien erstellt<br>• Server konfiguriert, damit die Schriften richtig dargestellt werden<br>• OSM-in-a-Box Java Code und Mapping für die Darstellung von Gebäuden und weiteren Elementen angepasst | |
| • Projektplan<br>• Zeitplan<br>• Kartengrafikanalyse<br>• Dokumentation der Verbesserungen | |

| **Meilenstein 2:** | |
|---|---|
| • Webseite erstellt<br>• Suchfunktion implementiert<br>• OSM-in-a-Box angepasst, damit Gebäude und weitere Elemente<br>• PostgreSQL Datenbank mit den Views für die Suche erweitert | |
| • Anforderungsanalyse<br>• Design PostgreSQL Datenbank<br>• Design User Interface Webseite | |

| **Meilenstein 3:** | |
|---|---|
| • Android Applikation „Pois R Us“ implementiert<br>• Geonamen Suche<br>• Eigener Standort ermitteln und darstellen | |
| • Anforderungsanalyse<br>• Design User Interface Android | |

| **Meilenstein 4:** | |
|---|---|
| • Fertigstellung und Dokumentation der gemachten Arbeiten (Kartegrafik Verbesserung, Webseite, Android Applikation) | |
| • Abstract<br>• Management Summary<br>• Technischer Bericht<br>• Erfahrungsbericht<br>• Plakat<br>• Gesamtdokumentation | |

### 5.4.3 Projektplan

Der ausführliche Projektplan liegt der CD bei.

### 5.4.4 Arbeitspakete

**Projekt Management**

| Projektplan Excel | |
|---|---|
| Beschreibung | Das Excel für den Projektplan erstellen und für die Verwendung vorbereiten. Später wird immer für die nächste Iteration vorgeplant. |
| Abhängigkeiten | - |

| Milestones | |
|---|---|
| Beschreibung | Definieren der Meilensteine des Projektes. Erstellen eines Dokumentes mit der Beschreibung dieser Meilensteine |
| Abhängigkeiten | - |

| Excel Zeiterfassung | |
|---|---|
| Beschreibung | Nachführen des Excel Sheets gemäss der Zeiterfassung. |
| Abhängigkeiten | Projektplan Excel |

| Review Projektplan | |
|---|---|
| Beschreibung | Review des Projektplans Excel und Word. |
| Abhängigkeiten | Projektplan Excel<br>Projektplan Dokument |

**Requirements**

| Anforderungsspezifikation | |
|---|---|
| Beschreibung | Erstellen der Anforderungsspezifikation mit funktionalen und nicht funktionalen Anforderungen. |
| Abhängigkeiten | - |

| Review Anforderungsspezifikation | |
|---|---|
| Beschreibung | Review der Anforderungsspezifikation. |
| Abhängigkeiten | Anforderungsspezifikation |

**Analyse**

| Analyse Vorgängerprojekt | |
|---|---|
| Beschreibung | Analyse des Vorgängerprojektes. Codestudium und Architekturstudium. |
| Abhängigkeiten | - |

| Evaluation Kartengrafik Optimierung | |
|---|---|
| Beschreibung | Einarbeiten in das Verfahren der Kartengrafik Renderings und den unterschiedlichen Gestaltungsmöglichkeiten bei Online Karten. |
| Abhängigkeiten | |

| Analyse SLD Vorgängerprojekt | |
|---|---|
| Beschreibung | Anhand des Wissens über das Rendering der Kartengrafik die vorliegenden SLD Dateien des Vorgängerprojektes analysieren. |
| Abhängigkeiten | Evaluation Kartengrafik Optimierung |

| Domainanalyse Android | |
|---|---|
| Beschreibung | Dokument für die Domainanalyse erstellen. Domainmodell mit einem entsprechenden Tool erstellen und in der Dokumentation integrieren. |
| Abhängigkeiten | - |

| Review Domainanalyse Android | |
|---|---|
| Beschreibung | Domainanalyse überarbeiten. |
| Abhängigkeiten | Domainanalyse |

| Definition der Implementationsarbeiten | |
|---|---|
| Beschreibung | Es wird durch ein Brainstorming definiert, welche Teile für die Webseite und für die Android Applikation umgesetzt werden sollen. |
| Abhängigkeiten | - |

**Design Analyse**

| Design PostgreSQL Datenbank | |
|---|---|
| Beschreibung | Neues Design bzw. Erweiterung des bestehenden PostgreSQL Datenbank. |
| Abhängigkeiten | - |

| Design User Interface Webseite | |
|---|---|
| Beschreibung | Mockup für das User Interface der Webseite erstellen. Auf Basis diese Mockups wird das User Interface der Seite implementiert. |
| Abhängigkeiten | - |

| Design User Interface Android | |
|---|---|
| Beschreibung | Mockup für das User Interface der Android Applikation entwerfen. Falls verschiedenen Screens nötig sind die Abhängigkeit aufzeigen. |
| Abhängigkeiten | - |

**Implementation**

| SLD | |
|---|---|
| Beschreibung | Umsetzung des neuen SLD Konzepts. Die SLD Dateien völlig neu aufbauen oder falls ausreichend anpassen. |

| | |
|---|---|
| Abhängigkeiten | Evaluation Kartengrafik Optimierung<br>Analyse SLD Vorgängerprojekt |

| OpenLayers | |
|---|---|
| Beschreibung | Darstellung und Bedienung der Karte auf der Webseite mit Hilfe der Javascript Library von OpenLayers. |
| Abhängigkeiten | - |

| Webseite (CSS, HTML, Javascript) | |
|---|---|
| Beschreibung | Implementation der Webseite. Aufbau der Seite mittels HTML und CSS. Funktionalität mittels Javascript. |
| Abhängigkeiten | - |

| Mapping osm2gis | |
|---|---|
| Beschreibung | Das Mapping von osm2gis erweitern damit Gebäude und weitere Elemente auf der Karte dargestellt werden können. |
| Abhängigkeiten | Evaluation Kartengrafik Optimierung |

| Anpassungen Java Code osm2gis | |
|---|---|
| Beschreibung | Anpassungen am Java Code von osm2gis vornehmen, damit das Mapping und das neue Design der PostgreSQL Datenbank umgesetzt werden kann. |
| Abhängigkeiten | Design PostgreSQL Datenbank<br>Mapping osm2gis |

| Geoserver Konfiguration | |
|---|---|
| Beschreibung | Konfiguration des Geoservers damit die neuen SLD Dateien gerendert werden und die neuen Daten des angepassten Mappings übernommen werden. |
| Abhängigkeiten | Design PostgreSQL Datenbank<br>Mapping osm2gis<br>SLD |

| Server Konfiguration | |
|---|---|
| Beschreibung | Konfiguration des Linux Server damit die Schriften richtig dargestellt werden können. |
| Abhängigkeiten | - |

| Suchfunktion Algorithmus | |
|---|---|
| Beschreibung | Entwicklung eines geeigneten Suchalgorithmus für die Suchfunktion. |
| Abhängigkeiten | - |

| Suchfunktion DB Views | |
|---|---|
| Beschreibung | Views auf der PostgreSQL Datenbank erstellen um die Suchresultate in gewünschter Form darstellen zu können. |
| Abhängigkeiten | Suchfunktion Algorithmus<br>Design PostgreSQL Datenbank |

| Android Kartendarstellung | |
|---|---|
| Beschreibung | Durch den Einsatz einer geeigneten Library die Kartendarstellung auf Android implementieren. |
| Abhängigkeiten | |

| Android GPS | |
|---|---|
| Beschreibung | Den GPS Service von Android nutzen und die Lokalisierung und Darstellung des aktuellen Standortes implementieren. |
| Abhängigkeiten | - |

| Android Suchfunktion | |
|---|---|
| Beschreibung | Eine Suchfunktion so auf Android implementieren, wie sie bereits auf der Webseite implementiert wurde. |
| Abhängigkeiten | Suchfunktion Algorithmus |

**Qualitätsmassnahmen**

| Technikstudium osm2gis | |
|---|---|
| Beschreibung | Einlesen in die verschiedenen Technologien von osm2gis. |
| Abhängigkeiten | - |

| Technikstudium Android | |
|---|---|
| Beschreibung | Einlesen und Kennenlernen der Entwicklungsumgebung und den Aufbau von Android. |
| Abhängigkeiten | - |

| Codereview SLD und Webseite | |
|---|---|
| Beschreibung | Review der SLD Dateien, welche erneuert wurden. Review des Javascript Codes und der HTML und CSS Dateien der Webseite. |
| Abhängigkeiten | - |

| Codereview Android | |
|---|---|
| Beschreibung | Review des Android Codes, welcher implementiert wurde. |
| Abhängigkeiten | |

**Dokumentation**

| Abstract | |
|---|---|
| Beschreibung | Abstract schreiben. |
| Abhängigkeiten | - |

| Broschüre | |
|---|---|
| Beschreibung | Broschüre anhand der Vorlage erstellen. |

| | |
|---|---|
| Abhängigkeiten | - |

| Management Summary | |
|---|---|
| Beschreibung | Management Summary schreiben. |
| Abhängigkeiten | - |

| Technischer Bericht / Projektdokumentation | |
|---|---|
| Beschreibung | Technischer Bericht und Projektdokumentation schreiben. |
| Abhängigkeiten | - |

| Plakat | |
|---|---|
| Beschreibung | Erstellen des Plakates für die Präsentation der Arbeit. |
| Abhängigkeiten | |

| Gesamtdokumentation | |
|---|---|
| Beschreibung | Gesamtdokumentation schreiben. |
| Abhängigkeiten | - |

**Sitzungen**

| Wöchentliche Sitzungen | |
|---|---|
| Beschreibung | 1.5 Stunden pro Woche Sitzung. |
| Abhängigkeiten | - |

| Sitzungsprotokolle | |
|---|---|
| Beschreibung | Sitzungsprotokolle erstellen. |
| Abhängigkeiten | |

### 5.4.5 Infrastruktur

Als Infrastruktur dienten folgende Komponenten:

- Linux Server:
  - http://sinv-56020.edu.hsr.ch
  - Tomcat 5.5
  - PostgreSQL 8.3
  - PostGIS 1.4
- Ein Developer-Wiki:
  - http://dev.ifs.hsr.ch/osminabox/
- SVN-Repository:
  - http://dev.ifs.hsr.ch/svn/osminabox/branches/osmiab

### Entwicklungsumbegung

Als Entwicklungsumgebung wird ein eigener Laptop sowie der Heimarbeitsplatz verwendet. Der Code und die Dokumente werden zentral im SVN Repository revisioniert.

### Entwicklungssoftware

Die Entwicklung wird mit folgender Software durchgeführt:

- Adobe Dreamweaver
- Adobe Photoshop
- PostgreSQL 8.3
- PostGIS 1.4
- Apache Tomcat 5.5
- Geoserver 1.7.5
- Eclipse Gallileo
- OpenOffice 3.0.1
- UMLet 10.2

## 5.5 Projektmonitoring

### 5.5.1 Soll-Ist-Zeit-Vergleich

| Überpakete | Soll | Ist | Differenz |
|---|---|---|---|
| **Projekt Management** | 42.5 | 36 | -6.5 |
| **Requirements** | 7.5 | 5 | -2.5 |
| **Analyse** | 77 | 84.5 | 7.5 |
| **Design Analyse** | 15 | 22 | 7 |
| **Implementation** | 230 | 275 | 45 |
| **Qualitätsmassnahmen** | 42 | 47.5 | 5.5 |
| **Dokumentation** | 129 | 123 | -6 |
| **Sitzungen** | 16 | 15.5 | -0.5 |
| | | | |
| **Total** | 559 | 608.5 | 49.5 |

### 5.5.2 Codestatistik

Aufgrund der Art des Projektes wird darauf verzichtet eine ausführliche Codestatistik mit entsprechenden Metrik-Tools zu erzeugen. In den ersten beiden Teilen der Arbeit wurde vorwiegend mit XML und Javascript gearbeitet. Bei Javascript ist eine solche Statistik nicht sehr aussagekräftig. Da es sich bei der Java Implementation der Android Applikation nur um einen Prototypen handelt, wird auch hier auf eine ausführliche Statistik verzichtet.

| **Javascript Code** | |
|---|---|
| Anzahl Zeilen | 393 |
| **Java Code** | |
| Anzahl Pakete | 3 |
| Anzahl Klassen | 6 |
| Anzahl Zeilen | 843 |

### 5.5.3 Sitzungsprotokolle

Es wird darauf verzichtet, die Sitzungsprotokolle hier einzufügen. Die Sitzungsprotokolle sind auf der CD enthalten.

# 6 Anhang

## 6.1 Erfahrungsbericht

Das Thema des Projektes hat mich von Anfang an sehr interessiert. Dadurch, dass das Projekt in drei unterschiedliche Teile aufgeteilt wurde, konnte ich in relativ kurzer Zeit sehr viele neue Erfahrungen sammeln. Die Verbesserung der Kartengrafik bereitete mir Freude, da die Resultate der Arbeit Schritt für Schritt sichtbar wurden. Da ich auch in der Freizeit und in meinem Beruf, den ich neben dem Studium ausübe, oft grafische Arbeiten mache, faszinierte mich die Möglichkeit, der Karte ein komplett neues Aussehen zu geben. Die Umsetzung mit den SLD Dateien ware jedoch sehr zeitintensiv und nachdem ich bereits nach wenigen Tagen relativ gute Resultate bei den Anpassungen feststellte, dachte ich dass dieser Teil der Arbeit am wenigsten Probleme bereiten würde. Es hat sich dann im Verlauf der Arbeit aber herausgestellt, dass es ziemlich schwierig ist, bei all den Anpassungen an der Kartengrafik immer den Überblick zu behalten. Ich bin mit dem Resultat der verbesserten Kartengrafik zufrieden, auch wenn ich die aufgetretenen Probleme beim Rendering gerne behoben hätte. Die Konfiguration des Linux Servers hat mir am Anfang ziemlich Sorgen gemacht, da ich bis jetzt sehr selten auf Linux gearbeitet habe. Aus diesem Grund war diese Arbeit für mich sehr lehrreich.

Die Implementation der Suchfunktion war sehr abwechslungsreich. Ich konnte konnte dabei neue Technologien wie den Geoserver kennenlernen und gleichzeitig mein Wissen aus dem Studium beim erstellen der Datenbank Views einsetzen.

Ich habe mich bereits am Anfang der Arbeit auf den dritten Teil, die Android Applikation gefreut. Die Entwicklung einer Sofware für Mobiltelefone faszinierte mich auch wenn ich noch keine Erfahrungen in diesem Bereich hatte. Leider ist relativ viel Zeit durch die Evaluation für eine geeignete Umsetzung eines Karten-Viewers für Android verloren gegangen. Dadurch blieb nur sehr wenig Zeit für die Implementation selber. Die kurze Erfahrung die ich mit Android machen konnte hat aber mein anfängliches Interesse noch gesteigert und mir eine gute Grundlage für die Software Entwicklung für Mobile Geräte gegeben.

Ich habe das Projekt alleine, also nicht wie gewöhnlich in einer Gruppe gemacht. Ich muss sagen, dass es nicht ganz einfach war, das ganze Projekt alleine durchzuführen. Oft wäre ich froh gewesen, wenn ich gewisse Entscheide und Probleme mit einem Partner hätte besprechen können. Der Vorteil ist sicherlich der, dass ich jedes Detail des Projektes sehr gut kenne und mich mit allen Technologien wirklich ausführlich beschäftigen musste.

## 6.2 Inhalt der CD

| Pfad | Dateien |
|---|---|
| Dokumente | Gesamtdokumentation.pdf<br>Gesamtdokumentation.odt<br>Plakat.odpx |
| Dokumente/Projektplan | Projektplan.ods<br>Zeiterfassung.ods |
| Javadoc | Javadoc |
| Pois R Us 1.0 | Installationsdatei Android Applikation |
| Sitzungsprotokolle | Alle Sitzungsprotokolle |
| Source | Sourcecode osminabox 1.1<br>Sourcecode poisrus<br>Sourcecode website (html, css, javascript) |

## 6.3 ANHANG B Glossar und Abkürzungsverzeichnis

| Term | Eklärung |
|---|---|
| Feature Type | Ein Feature Type bezeichnet eine Gruppe von Kartenelementen. Beispiele sind Strassen, Ortschaften, Flüsse, Seen etc. |
| Feature | Ein Feature bezeichnet ein spezifisches Kartenelemente eines bestimmten Feature Types. Ein Beispiel eines Feature des Feature Type Ortschaft ist Stadt oder Dorf. |
| Halo | Ein Halo ist ein Styling Begriff. Er definiert die Umrandung der einer Schrift. |
| OGC | Open Geospatial Consortium |
| OSM | Das OpenStreetMap Projekt |
| SLD | Styled Layer Descriptor und Symbology Encoding |
| WFS | Web Feature Service |
| WMS | Web Map Service |

## 6.4 ANHANG C Literatur- und Quellenverzeichnis

### Bücher und Artikel

(1) Frederik Ramm und Jochen Topf, «OpenStreetMap», 2. Version 2009

(2) Sfefan Zollinger, «Kartenkritik an OpenStreetMap», ETH Studie 2008

### Links und Informationen

(3) OpenLayers: http://openlayers.org/

(4) JQuery: http://jquery.com/

(5) GeoJSON: http://geojson.org/

(6) GeoServer: http://www.geoserver.org

(7) GeoServer Users: http://old.nabble.com/GeoServer---User-f1194.html

(8) GeoServer Dev: http://old.nabble.com/GeoServer---Dev-f1195.html

(9) Open Geospetial Consortium OpenGIS standards and specification, http://www.opengeospatial.org/standards

(10) SLD documentation, http://www.opengeospatial.org/standards/sld

(11) PostgreSQL 8.3 documentation, http://www.postgresql.org/docs/8.3/interactive/index.html

(12) PostGIS 1.5.3 documentation, http://postgis.refractions.net/documentation/

(13) GeoWebCache documentation, http://geowebcache.org/trac

(14) Nutiteq: http://www.nutiteq.com/search/node/gpl

## 6.5 ANHANG D Eigenhändigkeitserklärung

Ich erkläre hiermit,

- dass ich die vorliegende Arbeit selber und ohne fremde Hilfe durchgeführt habe, ausser derjenigen, welche

explizit in der Aufgabenstellung erwähnt ist oder mit dem Betreuer schriftlich vereinbart wurde,

- dass ich sämtliche verwendeten Quellen erwähnt und gemäss gängigen wissenschaftlichen Zitierregeln

korrekt angegeben habe.

Ort, Datum:

Name, Unterschrift: