

Wohnortfinder Zürioberland

Studienarbeit

Abteilung Informatik

HSR Hochschule für Technik Rapperswil

Frühjahrssemester 2014

Autoren: Laurin Murer, Remo Liebi

Betreuer: Prof. Stefan Keller

Projektpartner: Zürioberland Wohnen

Erstellung des Dokuments:	1. März 2014
Letzte Aktualisierung des Dokuments:	28. Mai 2014
Impressum:	
Auflagenzahl:	1
Jahr:	2014
Autoren:	Remo Liebi, Laurin Murer
Alle Rechte vorbehalten	

Inhaltsverzeichnis

INHALTSVERZEICHNIS	3
0.1 ABBILDUNGSVERZEICHNIS	5
0.2 TABELLENVERZEICHNIS	6
1 AUFGABENSTELLUNG	7
2 KURZFASSUNG STUDIENARBEIT	11
3 ABSTRACT	11
4 MANAGEMENT SUMMARY	12
4.1 AUSGANGSLAGE	12
4.2 ZIELSETZUNG	12
4.3 VORGEHEN, TECHNOLOGIE	12
4.4 ERGEBNISSE	13
4.5 AUSBLICK	14
5 TECHNISCHER BERICHT DER ARBEIT	15
5.1 EINFÜHRUNG	15
5.1.1 VISION	15
5.1.2 ZIELE	15
5.1.3 ABGRENZUNGEN	15
5.2 AUSGANGSLAGE	16
5.2.1 EXCEL TABELLE	16
5.2.2 OSM-DATEN	16
5.3 EVALUATION	16
5.3.1 FRAMEWORK	16
5.3.2 HEATMAP	17
5.3.3 WOHNUNGSANZEIGE	18
5.3.4 DATENGRUNDLAGE	18
5.3.5 JAVA-SCRIPT	18
5.4 REALISIERUNG	19
5.4.1 DATENHALTUNG	19
5.4.2 HEATMAP DARSTELLUNG	19
5.4.3 SERVICES UND SCHNITTSTELLEN	19
5.4.4 DATENARTEN	20
5.5 ERGEBNISSE	20

6	SW-PROJEKTDOKUMENTATION	23
6.1	VISION	23
6.2	ANFORDERUNGSSPEZIFIKATION	23
6.2.1	ANFORDERUNG AN DIE ARBEIT	23
6.2.2	USE CASES	23
6.2.3	SYSTEM-SEQUENZDIAGRAMME	24
6.2.4	WEITERE FUNKTIONEN	25
6.2.5	NICHT FUNKTIONALE ANFORDERUNGEN	26
6.3	ANALYSE UND DESIGN SPEZIFIKATIONEN	26
6.3.1	DOMAIN MODELL	26
6.3.2	OBJEKTKATALOG	27
6.3.3	ARCHITEKTUR	28
6.3.4	UI DESIGN	29
6.3.5	AUSWAHL DER ANGEZEIGTEN KRITERIEN	30
6.4	IMPLEMENTATION UND TEST	32
6.4.1	IMPLEMENTATION	32
6.4.2	AUTOMATISCHE TESTVERFAHREN,	37
6.4.3	MANUELLE TESTVERFAHREN	40
6.5	RESULTATE UND WEITERENTWICKLUNG	40
6.5.1	RESULTATE	40
6.5.2	MÖGLICHKEITEN DER WEITERENTWICKLUNG	40
6.5.3	WEITERES VORGEHEN	41
6.6	PROJEKTMANAGEMENT	41
6.6.1	SPRINT	41
6.6.2	PROTOTYPEN, RELEASES, MEILENSTEINE	42
6.6.3	TEAM, ROLLEN UND VERANTWORTLICHKEITEN	42
6.6.4	AUFWANDSCHÄTZUNG, ZEITPLAN, PROJEKTPLAN	42
6.6.5	RISIKEN	43
6.7	PROJEKT MONITORING	44
6.7.1	SOLL-IST-ZEIT-VERGLEICH	44
6.7.2	CODESTATISTIK	45
6.8	SOFTWAREDOKUMENTATION	45
6.8.1	INSTALLATION	45
6.8.2	TUTORIAL / BENUTZERHANDBUCH	45
7	PROJEKT	48
7.1	SITZUNGEN	48

7.2	PLANUNG DES PROJEKTS	48
7.2.1	SPRINTS	48
7.2.2	ZEITABRECHNUNG	48
7.3	PERSÖNLICHE BERICHTE	49
7.3.1	REMO LIEBI	49
7.3.2	LAURIN MURER	49
7.4	UNVORHERGESEHENES	49
7.4.1	AUSFALL HSR SERVER	49
7.5	INHALT DER CD	49
8	GLOSSAR UND ABKÜRZUNGSVERZEICHNIS	52
9	LITERATUR- UND QUELLENVERZEICHNIS	53
10	LISTE DER VERWENDETEN LIBRARIES	54
10.1	PYTHON	54
10.2	JAVA SCRIPT	54
10.3	POSTGRES	54
10.4	WEITERE	54
11	RECHTLICHES	55
11.1	NUTZUNGSRECHTE	55
11.2	EIGENSTÄNDIGKEITSERKLÄRUNG	56
0.1	Abbildungsverzeichnis	
Abbildung 4-1:	Stand Wohnortfinder am Projektende	13
Abbildung 4-2:	Übersicht Ablöseprozess der Adressliste (Details siehe 6.4.1.3)	14
Abbildung 5-1:	Ergebnis des Vergleichs der Technologien	16
Abbildung 5-2:	Beispielansicht des Heatmap Plugin	17
Abbildung 5-3:	Beispielansicht des Data Visualization Framework	18
Abbildung 5-4:	Vergleich der Server	21
Abbildung 5-5:	Screenshot des Flat Finders	22
Abbildung 6-1:	Haupt-Use Cases vom WF	23
Abbildung 6-2:	System Sequenz Diagramm vom Use Case 6.2.2.2.1 Guter Wohnort finden	25
Abbildung 6-3:	POI-Daten Klassen Diagramm	26
Abbildung 6-4:	Überblick über die restlichen Klassen	27
Abbildung 6-5:	Mockup vom WF Version I	29
Abbildung 6-6:	Mockup vom WF Version II	29
Abbildung 6-7:	Mockup vom WF Version III	29
Abbildung 6-8:	Finaler Designentwurf des WF	29
Abbildung 6-9:	Printscreen vom WF am Projektende	30

Abbildung 6-10: Datenbankschema (ausser OSM-Daten)	33
Abbildung 6-11: OSM-Datenbankschema	35
Abbildung 6-12: Handling der ZOW-Daten	37
Abbildung 6-13: Erfolgreiche Nosetests	38
Abbildung 6-14: Erfolgreichen Jasmine-Testlauf	39
Abbildung 6-15: Jasmine-Testlauf mit einem fehlgeschlagenen Test	39
Abbildung 6-16: Liste von weiteren Funktionen aus dem Jira	40
Abbildung 6-17: Liste von zwingend zu erledigenden Tasks	42
Abbildung 6-18: Liste von weiteren, fakultativen Tasks die auch im Projekt erledigt wurden	43
Abbildung 6-19: Schätzgenauigkeit	44
Abbildung 6-20: Aufteilung der Tasks	45
Abbildung 6-21: Backend des WF während dem Bearbeiten eines Kriteriums	47
Abbildung 7-1: Zeitaufwand während dem Projektablauf	48

0.2 Tabellenverzeichnis

Tabelle 1: Liste von Kriterien für den Benutzer	31
Tabelle 2: Grösse der OSM-Tabellen	34
Tabelle 3: Risikoanalyse WF	44
Tabelle 4: Übersicht über Total geänderte Zeilen	45



1 Aufgabenstellung

Wohnortfinder für das Zürcher Oberland

Studienarbeit von Laurin Murer und Remo Liebi

Abteilung Informatik, Frühjahrssemester 2014

Ausgangslage

Wer hat sich nicht schon gewünscht, mittels eines einfachen Tools seinen idealen Wohnort herauszufinden? Oder mit ein paar Klicks die nächsten Einkaufsmöglichkeiten, ÖV-Haltestellen oder Schulen auszumachen? Wir möchten für die potentiellen Neuzuzüger sowie auch für die Einwohnerinnen und Einwohner eine optimale Lösung finden, wie ihr Bedürfnis nicht mehr nur ein Wunsch ist. Für die Entwicklung eines geeigneten Programms möchten wir gerne für Studierende die Möglichkeit schaffen, sich einzubringen und den Wohnortfinder zu gestalten.

Im Auftrag der Standortförderungskommission Zürcher Oberland betreibt Pro Zürcher Berggebiet die Geschäftsstelle von natürli Zürioberland Wohnen. Ziel ist es, das Zürcher Oberland als Wohnort attraktiv zu gestalten und gegen aussen zu vermarkten. Regelmässig werden Anlässe zu Wohnthemen durchgeführt und eine Auskunftsstelle betrieben. (siehe auch: <http://www.zuerioberland-wohnen.ch>)

Aufgabenstellung

Der Wohnortfinder soll es den Usern der Website Wohnen ermöglichen, mittels ein paar Klicks den für sie persönlich idealen Wohnort im Zürcher Oberland zu finden. Ebenfalls soll es möglich sein, mittels Eingabe seines Standortes herauszufinden, in welcher Entfernung die nächsten Einkaufsmöglichkeiten, Bankomaten, öffentlichen Gebäude, etc. zu finden sind. Die Datenbank (Excel), welche als Grundlage dient, soll einfach zu pflegen und für alle Bereiche anwendbar sein (wie z. B. auch für die Suchfunktionen, die auf der Webseite Wohnen zu finden sind).

Aufgaben

Es sollte zwei Suchmöglichkeiten geben:

a) Wohnortsuche (Vorschlag)

1. Der Website-Besucher gibt in den Rubriken an, wie weit die einzelnen „Leistungsträger“ von seinem zukünftigen Ort entfernt sein dürfen.
2. Das Programm sucht entsprechend den Eingaben den perfekten Wohnort für den Website-Benutzer.
 - Als Beispiel dafür eignet sich der Localisator des Statistischen Amtes des Kantons Zürich
 - Die Funktion sollte wenn möglich mit Wohnungs-Such-Webseiten (z.B. comparis.ch) verbunden werden, damit dem Besucher gleich verschiedene Ausschreibungen/Inserate der Region präsentiert werden.

b) Wohnortbewertung (Vorschlag)

1. Der Website-Besucher gibt möglichen zukünftigen oder aktuellen Standort ein (Adresse).
2. Pro Rubrik kann der User, muss aber nicht, verschiedene Wünsche eingeben (z.B. welche Schulen innerhalb von wie vielen Kilometern sein müssen, etc.).
3. Die Angaben in jeder Rubrik werden gespeichert.
4. Anhand hinterlegten Adressen und eines Distanzberechnungsdienstes (z.B. Google Maps oder OpenStreetMap) werden die Distanzen berechnet.
5. Es werden alle möglichen Faktoren innerhalb des eingegebenen Radius aufgeführt. Der User erkennt daran, ob der Wohnort für ihn geeignet ist.

Als Output soll eine digitale Kartenausgabe mit Legende angegeben werden. Diese Karte kann auch gedruckt werden. Die einzelnen Themen werden weiter zu Unterthemen auf der Homepage verlinkt, sowie mit den Gemeindeportraits (siehe: <http://www.zuerioberland-wohnen.ch/wohnregion/gemeinden/>). Sobald mehr als z.B. 10 Vorschläge vorliegen, muss ein weiteres Kriterium eingegeben werden, das die Anzahl der Resultate reduziert. Die Übersichtlichkeit soll gewährleistet sein.

Bisher wird eine Adressdatenbank mit allen möglichen Leistungsträgern in Excel geführt. Eine einfache Pflege und Handhabung ist wünschenswert.

Am Schluss soll ein kurzes Benutzerhandbuch erstellt werden, um die Datenpflege vereinfachen.

Erwartete Resultate

Qualitative Resultate: Es soll mit Beendigung der studentischen Arbeit eine Softwarelösung vorliegen, die anwendungsbereit und benutzerfreundlich ist. Die Software „Wohnortfinder“ sollte nach Abgabe der Arbeit auf der Website Wohnen aufgeschaltet werden können. natürli Zürioberland Wohnen erhofft sich von den Studierenden weitere Vorschläge für Funktionen welche im Wohnortfinder sinnvoll sind. Dazu gehört eine fundierte Recherche bereits vorhandener ähnlicher Tools/Softwares.

Zudem wichtig:

- Die Softwarelösung muss auch über Smart-Phones und Tablets angewendet werden können.
- Die Aktualisierung der verwendeten Daten soll kontinuierlich und möglichst automatisch stattfinden.
- Für die visuelle Darstellung wird die Zusammenarbeit mit den Webdesignern von der Bitfrucht AG vorausgesetzt.
- Die Kompatibilität zwischen der Datenbank und den visuellen, gestalterischen Möglichkeiten der Website Wohnen soll gewährleistet sein.
- Die Suchmaske soll möglichst anwenderfreundlich (z.B mit Symbolen) gestaltet werden. Die Kriterien dafür werden mit der Betreuungsperson von Zürioberland Wohnen ausgearbeitet.

Quantitative Resultate: Im Minimum sollten die oben aufgeführten Anwendungen von den Usern der Website ausgeführt werden können. Sämtliche zur Verfügung stehenden Daten aus dem Perimeter für die entsprechenden Rubriken sollten generiert werden.

Der Perimeter umfasst die Gemeinden des Projektpartners (siehe <http://www.zuerioberland-wohnen.ch/wohnregion/portrait/>).

Mögliche Rubriken und Kriterien der POIs: Gemäss Absprache mit Projektpartner und Betreuer.

Vorgaben/Rahmenbedingungen

- Die Nutzungsrechte werden separat geregelt.
- Die Studenten entscheiden sich nach Rücksprache mit Projektpartner und Betreuer für die Programmiersprache/Framework und SW-Entwicklungsmethodik. Die Meilensteine werden mit dem Betreuer und dem Projektpartner vereinbart.
- Code, Kommentare und Versionsverwaltung sind in Englisch. Der Code ist grundsätzlich mehrsprachig. Alles andere ist deutsch.
- Ansonsten gelten die Rahmenbedingungen, Vorgaben und Termine des Studiengangs bzw. der HSR.



- Die Betreuungsperson von natürli Zürioberland Wohnen ist jeweils wochentags für die Studierenden erreichbar. Termine können jeweils Mo. – Fr. tagsüber abgemacht werden. natürli Zürioberland Wohnen kann für die Studierenden einen Arbeitsplatz zur Verfügung stellen. Es besteht eine Adressdatenbank in Excel-Format mit über 6'000 Adressen von Firmen, öffentlichen Behörden, Vereinen, etc.

Inhalt der Dokumentation

- Die fertige Arbeit muss folgende Inhalte haben:
 1. Abstract, Management Summary, Aufgabenstellung (Original eingescannt)
 2. Technischer Bericht
 3. Projektdokumentation
 4. Anhänge (Literaturverzeichnis, CD-Inhalt; Eigenständigkeitserklärung)
- Die Abgabe ist so zu gliedern, dass die Inhalte klar erkenntlich und auffindbar sind (einheitliche Nummerierung).
- Die Zitate sind zu kennzeichnen, die Quelle ist anzugeben.
- Verwendete Dokumente und Literatur sind in einem Literaturverzeichnis aufzuführen (nicht ausschliesslich Wikipedia-Links auflisten).
- Dokumentation des Projektverlaufes, Planung etc.
- Weitere Dokumente (z.B. Kurzbeschreibung) gemäss Vorgaben des Studiengangs und gemäss Absprache mit dem Betreuer.

Form der Dokumentation

Bericht (Struktur gemäss Beschreibung; gebunden); alle Dokumente und Quellen der erstellten Software auf CD; CD's sauber angeschrieben.

- Für Betreuer: Bericht (1 Exemplar). CD (1 Exemplar).
- Für Projektpartner: Bericht (1 Exemplar). CD (1 Exemplar).
- Für HSR: CD (1 Exemplar).

Bewertungsschema

Es gelten die üblichen Regelungen zum Ablauf und zur Bewertung der Studienarbeit des Studiengangs Informatik mit besonderem Gewicht auf moderne Softwareentwicklung wie folgt:

- Projektorganisation (Gewichtung ca. 1/5)
- Bericht, Gliederung, Sprache (Gewichtung ca. 1/5)
- Inhalt inkl. Code (Gewichtung ca. 2/5)
- Gesamteindruck inkl. Kommunikation mit Partner (Gewichtung ca. 1/5)

Beteiligte

Diplomanden/innen

Siehe Titel.

Projektpartner

natürli Zürioberland Wohnen
c/o Pro Zürcher Berggebiet
Alice Trachsel
Heinrich Gujerstrasse 20
Postfach
8494 Bauma
052 396 50 93
alice.trachsel@proz.ch
www.zürioberland-wohnen.ch

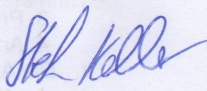


Betreuung HSR

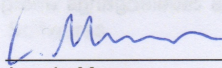
Verantwortlicher Dozent: Prof. Stefan Keller, Geometa Lab am IFS-HSR (sfkeller@hsr.ch).

Unterschriften

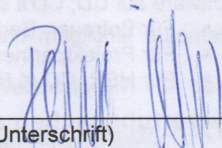
Stefan Keller

Rapperswil, 21.3.2014 
(Ort, Datum, Unterschrift)

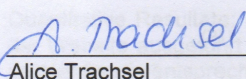
Laurin Murer


Rapperswil, 21.3.2014
(Ort, Datum, Unterschrift)

Remo Liebi

Rapperswil 21/3/14 
(Ort, Datum, Unterschrift)

Alice Trachsel


Rapperswil, 21.3.2014
(Ort, Datum, Unterschrift)

2 Kurzfassung Studienarbeit

Abteilung	Informatik
Namen der Studierenden	Laurin Murer Remo Liebi
Studienjahr	FS2014
Titel der Studienarbeit	Wohnortfinder Zürioberland
Themengebiet	Internet-Technologien und -Anwendungen
Projektpartner	Zürioberland Wohnen, Bauma
Institut	Institut für Software

3 Abstract

Die Wohnungssuche in der Schweiz kann sich als sehr umständlich gestalten und jeder Wohnungssuchende hat andere Ansprüche an seinen perfekten Wohnort.

Der Wohnortfinder als Webapplikation bietet eine Plattform, auf welcher der Wohnungssuchende seine individuellen Kriterien definieren kann und so die idealste Wohnregion im Zürcher Oberland ermitteln kann. Das Backend (API und CMS) wurde mit Hilfe des Python Microframeworks Flask umgesetzt, wobei für die persistente Datensicherung auf das Datenbanksystem PostgreSQL zurückgegriffen wird. Weiter ist eine REDIS-Datenbank für die Verwaltung von Tasks im Hintergrund eingesetzt. Als JavaScript-Framework wird AngularJS verwendet, welches dem Entwickler eine saubere Strukturierung des Codes ermöglicht und so die Wartbarkeit des Gesamtsystems gewährleistet. Die Kriterien werden hierbei einerseits direkt von OpenStreetMap heruntergeladen, andererseits werden zusätzliche – für OpenStreetMap nicht relevante – Daten in separaten Tabellen der PostgreSQL-Datenbank abgelegt. Bei der Entwicklung der Benutzeroberfläche wurde vor allem auf das Leaflet Plugin und die Stylesheets, die von Zürioberland Wohnen zur Verfügung gestellt wurden, gesetzt. Durch das Testingframework Jasmine wird mittels Behavior-Driven Tests die Lauffähigkeit und Korrektheit des JavaScript-Codes sichergestellt. Im Backend wird das Python Testingframework Nostest eingesetzt.

4 Management Summary

4.1 Ausgangslage

Die Wohnungssuche in der Schweiz kann sich als sehr umständlich gestalten und jeder Wohnungssuchende hat andere Ansprüche an seinen perfekten Wohnort.

Das Ziel dieser Studienarbeit ist es, diese Lücke für das Zürcher Oberland zu schliessen und eine Webapplikation zu entwickeln, die es erlaubt anhand von Kriterien den idealen Wohnort zu finden. Dies wird durch intelligente Algorithmen erreicht, welche die Abstände und Wichtigkeit von Points-of-Interests (POIs) bewerten und auf einer interaktiven Karte visualisieren.

Im Unterschied dazu stehen die verschiedenen Immobilienportale, welche Wohnungen anhand von harten Fakten bezüglich eines Objekts ermitteln und dem Kunden eine Auswahl von Wohnungen präsentieren.

Die Zielsetzung sah vor, mit wenigen Klicks den für den Besucher idealen Wohnort im Zürcher Oberland zu finden. Ebenfalls sollte es ursprünglich möglich sein, mittels Eingabe seines Standortes herauszufinden, in welcher Entfernung die nächsten Einkaufsmöglichkeiten, Bankomaten, öffentliche Gebäude, etc. zu finden sind. Die Datenbank sollte einfach zu pflegen und für alle Bereiche anwendbar sein.

Es wurden zwei Suchmöglichkeiten vorgegeben. Zum Einen die Wohnortsuche, welche vom Webseite-Besucher verlangt, dass er die einzelnen Leistungsträger definiert und eine Distanz und Wichtigkeitsangabe dazu angibt. Weiter sollten direkt auf der Karte verfügbare Wohnungen dargestellt werden.

Die zweite Anforderung war eine Wohnortbewertung. Der Webseitenbesucher sollte dazu einen zukünftigen oder aktuellen Wohnort angeben und das Programm soll aus diesen Angaben bestehende Leistungsträger erkennen und ähnliche Standorte anzeigen.

4.2 Zielsetzung

Die Ziele für die Studienarbeit setzten sich wie folgt zusammen: Auf der einen Seite galt es, einen Wohnortfinder von Grund auf zu erstellen. Auf der anderen Seite wurden während des Verlaufes Ziele definiert, welche dem Dienst zusätzliche Funktionalitäten verleihen sollte.

Das wichtigste Ziel war es, eine Grundlage zu erstellen, auf der man den Wohnortfinder weiter ausbauen kann.

Die genaue Zielsetzung wurde mittels iterativem Vorgehen und "Scrum-Like" Vorgehen vor zu definiert um vom Kunden dementsprechend gewichtet. Durch dieses Vorgehen hat sich die ursprüngliche Zielsetzung zu Gunsten anderer Features leicht verändert.

4.3 Vorgehen, Technologie

Beim Erstellen des Wohnortfinders sind wir iterativ vorgegangen. In der ersten Phase haben wir die Ziele des Projektes abgesteckt sowie dazu Tasks definiert. Anschliessend hat diese der Kunde gewichtet. In Jira haben wir dann jeweils im Voraus 2-wöchige Sprints definiert und diese abgearbeitet.

Es standen mehrere Technologien zur Auswahl. Nach der Auseinandersetzung mit den verschiedenen Technologien haben wir uns für Python entschieden. Dies, weil wir angenommen haben, dass es viele GIS-Libraries für Python gibt und uns da einiges an Programmieraufwand abgenommen werden kann.

Schlussendlich haben wir vollständig auf Pythons Flask Microframework gesetzt. Dieses erlaubt es, einfach und schnell einen Webservice zu erstellen. Es kamen viele weitere Python Module hinzu; mehr dazu im Anhang.

4.4 Ergebnisse



Abbildung 4-1: Stand Wohnortfinder am Projektende

Der Wohnortfinder ist funktionsfähig. Wir konnten nicht alle Features implementieren und hätten noch viele Ideen gehabt. Leider hatten wir dafür zu wenig Zeit. Im Gegenzug haben wir andere Funktionen implementieren können, an die Zürich Oberland Wohnen (ZOW) überhaupt nicht gedacht hatte.

So konnten wir newhome.ch davon überzeugen, uns ihre Datenbank zur Verfügung zu stellen. Dadurch haben wir stets aktuelle Wohnungsdaten auf der Webseite präsent, ohne uns auf einen Crawling Algorithmus verlassen zu müssen. In einer frühen Version hatten wir einen Crawler für die Comparis Plattform implementiert. Als jedoch nur einige Tage später der Aufbau der Webseite verändert wurde und somit der Crawler nicht mehr funktioniert hat, wurde uns schnell klar, dass dies keine anzustrebende Lösung sein kann. Bis zum Abschluss des Projekts hat uns newhome.ch Testdaten zur Verfügung gestellt. Live Daten werden in einer späteren Phase des Projektes aufgeschaltet, da dies einen Programieraufwand von Seiten von newhome.ch verlangt.

Wir konnten auch einbringen, dass die bestehende Adressliste komplett abgelöst wird und so einen weiteren Mehrwert generieren.

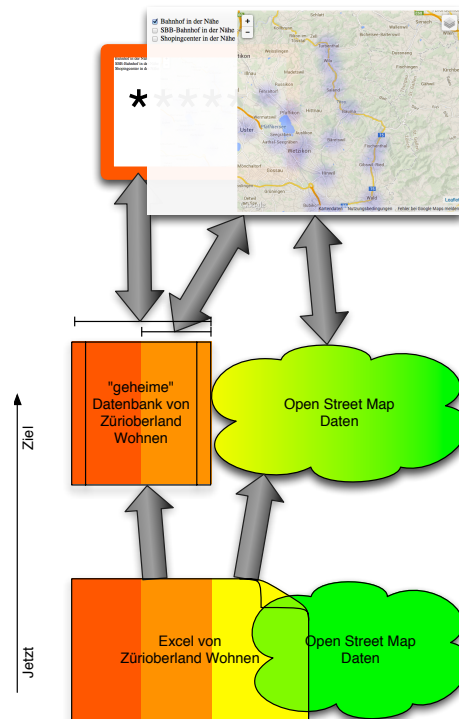


Abbildung 4-2: Übersicht Ablöseprozess der Adressliste (Details siehe 6.4.1.3)

4.5 Ausblick

Der gut strukturierte Code sollte es einfach ermöglichen, den Wohnortfinder zu erweitern. Die modular aufgebaute Architektur lässt sich erweitern und wiederverwenden. Wir konnten uns im Rahmen dieser Arbeit sowohl in neue Programmiersprachen einarbeiten und sie kennenlernen, als auch unser Wissen in bereits bekannten Sprachen vertiefen. Leider bremste dieses Lernen teilweise den Fortschritt des Projekts etwas ab. Wir konnten uns vertiefte Grundlagen in Python, PostGIS, AngularJS und weiteren Sprachen aneignen.

Es gibt teilweise noch Stabilitätsprobleme im Backend, welche noch nicht eruiert werden konnten. Wir werden uns nach dem Abschluss der Studienarbeit weiterhin um das Projekt kümmern, um die letzten wichtigen Features zu implementieren und Bugfixing zu betreiben.

Es wurden in der Studienarbeit keine Untersuchungen über das Verhalten des Dienstes bei grosser Auslastung gemacht. Es kann auch nicht prognostiziert werden, wann eine solche Situation erreicht würde. Bevor eine Integration auf einer vielbesuchten Webseite durchgeführt würde, müssten diese Untersuchungen durchgeführt werden.

Weitere Entwicklungen könnten mit der Integration zusätzlichen Datenquellen realisiert werden. Steuerfüsse, Lärmintensität oder Sonnenscheindauer wären Optionen um die Visualisierung der Wohnortregion noch anschaulicher zu machen. Eine zusätzliche Performancesteigerung könnte durch Caching der Abfragen erreicht werden.

5 Technischer Bericht der Arbeit

5.1 Einführung

Die Wohnungssuche in der Schweiz kann sich als sehr umständlich gestalten und jeder Wohnungssuchende hat andere Ansprüche an seinen perfekten Wohnort. Das zugrundeliegende Problem wird in der Informatik „Allokations-Problem“ genannt. Im Falle des Wohnortfinders (WF) heisst das, dass der (potentielle) Wohnort möglichst gut an knappe „Güter“ zugeordnet wird, wie beispielsweise der Distanz zur nächsten Schule, zum nächsten Spielplatz oder auch zum nächsten Wald. Raphael Gfeller und Pascal Hauser sind dieses Problem in ihrer Masterarbeit auch angegangen [4].

Das Ziel dieser Studienarbeit (SA) ist es, diese Lücke für das Zürcher Oberland zu schliessen und eine Webapplikation zu entwickeln, die es erlaubt, anhand von Kriterien den idealen Wohnort zu finden. Dies wird durch intelligente Algorithmen erreicht, welche die Abstände und Wichtigkeit von POIs bewerten und auf einer interaktiven Karte visualisieren.

Im Unterschied dazu stehen die verschiedenen Immobilienportale, welche Wohnungen anhand von harten Fakten bezüglich eines Objekts ermitteln und dem Kunden eine Auswahl von Wohnungen präsentieren.

5.1.1 Vision

Wie bereits beschrieben, sollte es dem Webseitenbesucher möglich sein, die ideale Wohnlage im Zürcher Oberland visuell zu präsentieren. Dieser Dienst soll in die bestehende ZOW Webseite integriert werden können.

5.1.2 Ziele

Die Zielsetzung sah vor, mit wenigen Klicks den für den Besucher idealen Wohnort im Zürcher Oberland zu finden. Ebenfalls sollte es möglich sein, mittels Eingabe seines Standortes herauszufinden, in welcher Entfernung die nächsten Einkaufsmöglichkeiten, Bankomaten, öffentliche Gebäude, etc. zu finden sind. Die Datenbank sollte einfach zu pflegen sein und für alle Bereiche anwendbar sein.

Es wurden zwei Suchmöglichkeiten vorgegeben. Zum einen die Wohnortsuche, welche vom Webseite-Besucher verlangt, dass er die einzelnen Leistungsträger definiert und eine Distanz und Wichtigkeitsangabe dazu angibt. Weiter sollten direkt auf der Karte verfügbare Wohnungen dargestellt werden.

Die zweite Anforderung war eine Wohnortbewertung. Der Webseitenbesucher sollte dazu einen zukünftigen oder aktuellen Wohnort angeben und das Programm soll aus diesen Angaben bestehende Leistungsträger erkennen und ähnliche Standorte anzeigen.

Weiter sollte die Software ebenfalls auf Tablets und Smartphones lauffähig sein und die Aktualisierung der hinterlegten Daten sollte sich möglichst einfach und im besten Falle automatisch gestalten.

5.1.3 Abgrenzungen

Die Zieldefinition dient als Vorlage und wird im Laufe der SA durch das iterative Vorgehen laufend angepasst. Es werden Schwerpunkte definiert und die Umsetzung der Features demensprechend priorisiert.

5.1.3.1 Korrektheit des Algorithmus

Die Garantie für eine absolute Realitätstreue der angezeigten Daten kann unter anderem auf Grund offensichtlicher Tatsachen nicht gegeben werden. Beispielsweise sind die Daten von

OpenStreetMap (OSM) v.a. in ländlichen Gebieten sehr lückenhaft und müssen aufgearbeitet werden.

5.1.3.2 *Bezug auf das Zürcher Oberland*

Der WF bezieht sich in der vorliegenden Version nur auf das Zürcher Oberland. Die Software kann deshalb nicht ohne Anpassungen für andere Regionen genutzt werden. Die Funktion an und für sich wäre allerdings ausweitbar, nur muss die Vollständigkeit der Daten gewährleistet werden, da sonst ein unvollständiges Bild generiert wird.

5.1.3.3 *Performance Prognose*

Von einem produktiven Einsatz der Software für eine gut besuchte Internetseite muss zurzeit noch abgeraten werden. Das Verhalten der Software und der Datenbank bei einer Überlastung des Webserver wurde nicht getestet und es können keine Vorhersagen gemacht werden, wann eine solche Überlastsituation eintreffen würde. Auch sind die Algorithmen für komplizierte Polygonberechnungen teilweise noch sehr langsam und es kann bis zu 90 Sekunden dauern, bis dem Besucher ein Bild präsentiert wird.

5.2 Ausgangslage

Die Ausgangslage für die SA, ist der Ist-Zustand von ZOW. Es besteht eine Excel-Tabelle, in welcher sich etwa 6000 Adressen von POI im Zürcher Oberland befinden.

5.2.1 Excel Tabelle

Eine der Datenquellen ist die Tabelle von ZOW. Diese Tabelle ist schwierig zu warten und hat sich als sehr unübersichtlich erwiesen.

5.2.2 OSM-Daten

Zu Beginn der Arbeit, sind die Daten fürs Zürcher Oberland sehr lückenhaft gewesen. Es bestanden einige POIs, aber es ist kein systematisches Schema zu erkennen, nach welchen Kriterien Punkte erfasst wurden.

5.3 Evaluation

Das folgende Kapitel dient der systematischen Evaluation von unterschiedlichen Möglichkeiten, um die gesteckten Ziele (5.1.2) der SA zu erreichen. Das Kapitel 5.4 (Realisierung) erläutert anschliessend, welche der Möglichkeiten tatsächlich umgesetzt wurden.

5.3.1 Framework

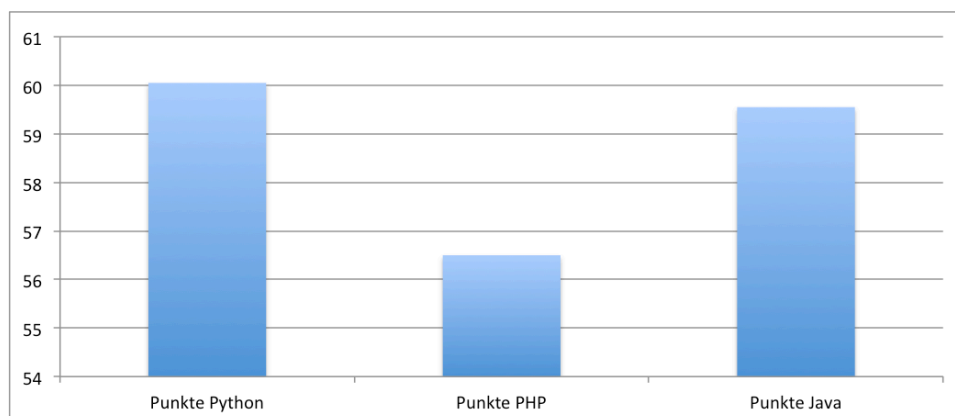


Abbildung 5-1: Ergebnis des Vergleichs der Technologien

5.3.1.1 *Symfony2 (PHP)*

Symfony ist ein in PHP5 geschriebenes Web Application Framework, das dem Model-View-Controller-Schema (MVC) folgt. Das Framework besteht aus Modulen, sogenannten Bundles, welche voneinander vollkommen unabhängig lauffähig sind, aber nahtlos in den Framework-Prozess integriert werden können. Durch die Verwendung eines Dependency Injection Containers ist die gesamte Anwendung modular aufgebaut. Dies bewirkt eine einfache Testbarkeit und Erweiterbarkeit. Ausserdem werden Namespaces unterstützt. Somit erfordert Symfony2 PHP 5.3 oder höher. Symfony2 ist unter der MIT Lizenz kostenlos verfügbar. Wir haben bereits beide Erfahrungen mit PHP sammeln können. [3]

5.3.1.2 *Play (Java)*

Play ist ein Open Source Web Application Framework, basierend auf Scala und Java, welches dem MVC Schema folgt. Es zielt darauf, die Entwicklungsproduktivität zu steigern, indem es "convention over configuration" verwendet.

5.3.1.3 *Flask (Python)*

Flask ist ein in Python geschriebenes Micro-Framework zur Entwicklung von Webanwendungen. Der Fokus von Flask liegt auf Erweiterbarkeit und guter Dokumentation. Die einzigen Abhängigkeiten sind Jinja2, eine Template Engine und Werkzeug, eine Bibliothek zum Erstellen von WSGI-Anwendungen.

Flask kommuniziert über die WSGI-Schnittstelle, derzeit die aktuelle Entwicklung für die Kommunikation zwischen Webserver und Webanwendungen im Python-Umfeld. Für Testzwecke und während der Entwicklung kann der von Flask mitgelieferte Webserver verwendet werden. [2]

5.3.2 **Heatmap**

5.3.2.1 *Leaflet Heatmap Plugin*

Das Leaflet Heatmap Plugin basiert auf Punkten und kann einfach, schön und schnell eine Heatmap visualisieren. Der Nachteil des Plugins ist, dass Seltenes in Häufigem untergeht.

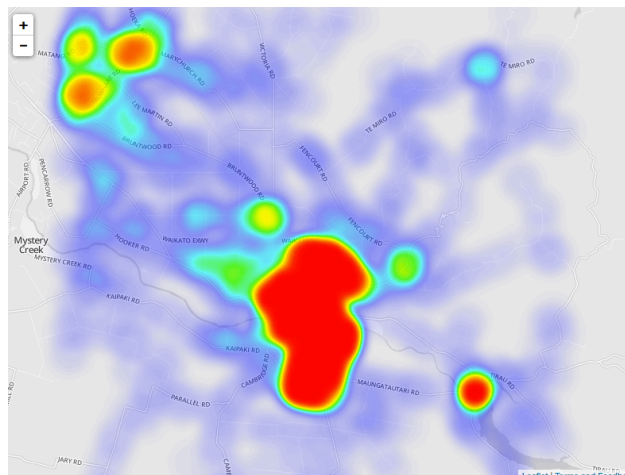


Abbildung 5-2: Beispielansicht des Heatmap Plugin

5.3.2.2 *Leaflet Data Visualization Framework*

Das Leaflet Data Visualization Framework arbeitet mit Isolinien. Somit ist eine korrekte Darstellung der Werte möglich. Es ist schwieriger zu implementieren, benötigt einen höheren Rechenaufwand und ist somit auch weniger schnell. Der visualisierte Output gegenüber dem Leaflet Heatmap Plugin sieht optisch weniger ansprechend aus.

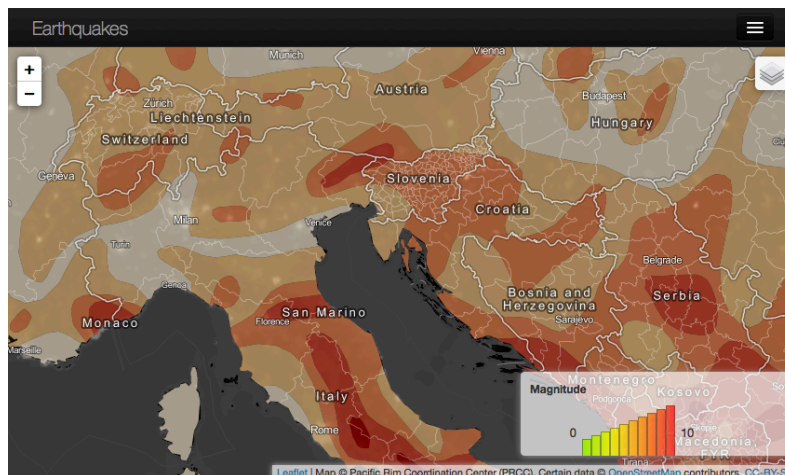


Abbildung 5-3: Beispielsicht des Data Visualization Framework

5.3.3 Wohnungsanzeige

5.3.3.1 Comparis

Comparis bietet den Service an, Wohnungsanzeigen von verschiedenen Immobilienportalen aufzuarbeiten und ein komplettes Bild des aktuellen Wohnungsmarktes darzustellen. Es wird keine API angeboten. Die Daten müssten somit per Scraper beschafft werden.

5.3.3.2 Newhome.ch

Newhome ist ein Immobilienportal, welches von den Kantonalbanken gestützt wird. Die Firma arbeitet mit anderen Immobilienportalen zusammen und hat somit ein nicht ganz komplettes Bild des aktuellen Wohnungsmarktes. Newhome bietet uns eine API an.

5.3.4 Datengrundlage

Die Webapplikation soll die erstellten geographischen Daten sinnvoll präsentieren. Daher ist es unabdingbar, dass die Applikation eine Landkarte enthält. Die Landkarte soll das Zentrum der Applikation darstellen. Es war schnell klar, dass wir viele Daten verarbeiten und aufbereiten müssen, somit war eine Zwischenspeicherung unabdingbar.

5.3.4.1 Google Maps API

Dies ist ein weit verbreitetes API, welches für kleinere Projekte kostenlos verwendet werden darf. Dieses steht jedoch nicht unter einer freien Lizenz und kann jederzeit von Google deaktiviert werden.

5.3.4.2 OpenStreetMap

Mit OpenStreetMap (OSM) steht eine opensource Alternative zur Verfügung, welche einen ähnlichen Funktionsumfang bietet. Falls gewünscht, können mit Leaflet auch die Google Karten Daten eingebunden werden. Beim Einsatz von OSM werden jedoch meistens diese Kartendaten genutzt. Da durch den Einsatz von OSM keine Abhängigkeit zu einem kommerziellen Anbieter entsteht, wird in der SA OSM eingesetzt.

5.3.5 Java-Script

Für die Erstellung einer interaktiven Webanwendung wird heutzutage meistens auf den Einsatz von Java-Script (JS) zurückgegriffen. Vorstellbar wäre es auch, eine interaktive Webseite mit Adobe Flash oder Microsoft Silverlight zu realisieren. JS hat aber den klaren Vorteil, dass es ohne die Installation von zusätzlichen Plug-Ins für den Webbrowser auskommt und auf praktisch allen Plattformen läuft. Für umfangreiche Programmfunktionen ist es empfehlens-

wert, auf JS-Bibliotheken zurückzugreifen. Es gibt eine grosse Anzahl solcher Bibliotheken, doch eine der populärsten ist sicherlich AngularJS. Angular basiert auf dem MVC Prinzip und bietet viele Vorteile zum interaktiven Bedienen einer Webseite. Somit haben wir in der SA auf Angular JS gesetzt.

Im Backend wurde auf AngularJS verzichtet und wir haben auf das etwas schlankere Framework jQuery in Kombination mit Bootstrap gesetzt.

5.4 Realisierung

Im vorhergehenden Kapitel wurden verschiedene Lösungsansätze und Implementationsvorschläge diskutiert. Das vorliegende Kapitel geht nun auf die konkret ausgewählten Varianten ein und beschreibt diese.

5.4.1 Datenhaltung

Die Evaluation hat gezeigt, dass die Datenflut für einen Bereich wie das ganze Zürcher Oberland enorm ist. Wir können nicht auf den OSM Servern beruhen, da die Anzahl der Requests an diese Server früher oder später zu einem Bann geführt hätten. Der Prozess zum Aktualisieren der Daten läuft automatisch in einem festgelegten Rhythmus ab und kann auch manuell gestartet werden. Sobald dieser gestartet wurde, laufen folgende Vorgänge automatisch ab:

- Herunterladen der OSM-Daten der gesamten Schweiz von Geofabrik
- Umgehen der View-Abhängigkeiten
- Alte Punkte und Polygone löschen
- Neue Punkte und Polygone in die Postgres Datenbank abfüllen
- Views mit Daten des Zürcher Oberlandes aktualisieren
- Materialized View aktualisieren
- Änderungsdaten aktualisieren
- Aufräumen

5.4.2 Heatmap Darstellung

Die Heatmap wird mittels PostGIS Buffern erstellt. Dabei wird so vorgegangen:

- relevante Punkte erfassen
- um die Punkte und Polygone in 5 Iterationen Buffer mit unterschiedlicher Intensität ausgeben

Dieser Algorithmus hat den Nachteil, dass bei komplizierten Polygonen (z.B. Wäldern), die Berechnungszeit sehr lange dauert.

5.4.3 Services und Schnittstellen

Die Schnittstelle zwischen Client und Server stellt Flask zur Verfügung. Es wurden Routen definiert, welche die gewünschten Daten im gewünschten Format zurückgeben.

Ressource	Beschreibung
GET branche.json	Antwortet mit den Branchen Informationen in den für den mitgelieferten Bereich
GET flats.json	liefert sämtliche Wohnungen für den Projektperimeter zurück

5.4.3.1 Branchenfinder API

Zur Integration der Daten in die ZOW-Webseite wurde eine API gewünscht, welche die POI in einer Region zurückgeben kann. Wir haben unter Absprache mit der Bitfrucht AG eine API entwickelt, auf welche die Webseite zugreifen kann.

Ressourcen URL: <http://wohnortfinder.zuerioberland-wohnen.ch/api/branche.json>

Parameter	
cities	Spezifiziert eine durch Komma getrennte Liste der abzufragenden Orte. Beispiel Werte: Wetzikon,Uster
tags	Spezifiziert die abzufragenden Tags durch Key Value Pairs, die durch Kommata getrennt werden. Beispiel Werte: key%3Dvalue shop%3Dmall,building%3Dyes

5.4.3.2 newhome.ch API

Zum Erlangen der Wohnungsdaten in unserem Projektperimeter, wurde uns von newhome.ch eine API zugesichert. Diese liefert sämtliche Wohnungen im Projektperimeter aus.

5.4.4 Datenarten

Sämtlicher Austausch der Daten zwischen verschiedenen Teilen der Software findet mittels JSON oder JSONp statt.

5.5 Ergebnisse

Der WF ist zum Schluss der Arbeit funktionsfähig. Es wurden alle definierten Ziele bis zum Schluss der Arbeit erreicht.

Folgende Funktionen konnten erfolgreich umgesetzt werden:

- Anzeige einer Heatmap auf einer Karte mittels Leaflet Plugins und PostGIS SQL abfragen
- definieren von harten Kriterien welche die Heatmap beeinflussen mittels Dropdown Menu und Slider Komponenten
- anzeigen von Wohnungen auf einer Karte. Diese Wohnungen stammen von einer API, welche von newhome.ch zur Verfügung gestellt wird, und werden in unserer Datenbank zwischengespeichert
- filtern von Wohnungen, welche auf der Karte dargestellt werden sollen, mittels AngularJS Funktionen
- Anzeige des Projektperimeters auf der Karte aufgrund einer PostGIS Abfrage
- Einbezug von Elementen, die ausserhalb des Projektperimeters liegen, mittels unsichtbarem Buffer
- Admin-Panel mit Login-Funktion als Backend, basierend auf Flask
- Benutzer und Rechteverwaltung für Admin-Bereich
- Integration von verschiedenen Datenquellen (OSM, Eigene POI, Erweiterungen für OSM-POI)

- CRUD-Operationen für OSM-POI-Erweiterungen und POIs
- definieren von Koordinaten für POI
- diverse Autovervollständigungen
- Automatisches Einlesen von aktuellen OSM-Daten
- Export-Funktion für sämtliche Datenarten
- dynamisches Definieren von Kriterien
- automatisches Abrufen der Wohnungen und Statistik über erfasste Wohnungen
- CRUD-Operationen für OSM- und ZO-Tags

Es wurden nicht alle Ziele der ursprünglichen Zielsetzung erreicht. In den diversen Sitzungen mit ZOW haben wir anhand von Aufwandschätzungen aufzeigen können, welche Features jetzt umzusetzen sind und welche besser auf einen späteren Zeitpunkt verschoben werden sollten. Umgesetzt wurde die erste Suchfunktion. Die Software ist auf Tablets funktionsfähig und kann auch vom Smartphone aus bedient werden. Ein spezielles Augenmerk wurde auch auf die Wartbarkeit der Datenbank gelegt. Nicht umgesetzt werden konnte die Standortbewertung. Dieses Feature wurde von ZOW im Nachhinein mit einer niedrigen Priorität eingestuft.

Die Priorität wurde auf andere Funktionen gelegt. So konnten wir zum Beispiel die Arbeitsabläufe bei ZOW drastisch verbessern. Bis anhin wurde eine riesige Excel-Tabelle genutzt, die als Adresskartei diente. Neu wird diese vollständig in Postgres und OSM abgebildet. Dadurch kann der Branchenfinder, der sich bereits auf der Webseite von ZOW befindet, direkt auf diese Daten zugreifen. So konnten wir eventuelle Zweispurigkeiten im Voraus eliminieren.

Ein weiteres Ziel der Arbeit war das Onlinestellen des Dienstes. Zu diesem Zweck haben wir evaluiert, welcher Server am sinnvollsten für unsere Anwendung ist. Die Entscheidung haben wir dem Kunden überlassen.

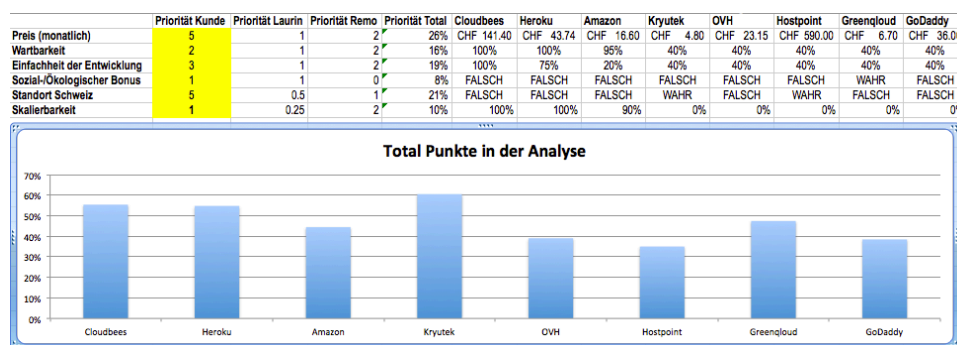


Abbildung 5-4: Vergleich der Server

ZOW hat sich für Kryutek entschieden, welche einen sehr günstigen Root-Server anbieten. Die Konfiguration und das Einrichten des Servers hat einige Zeit in Anspruch genommen.

Damit die periodischen Tasks der Datenbeschaffung den Nutzer und Administrator nicht beeinflussen, suchten wir nach einer Lösung um diese zu parallelisieren. Wir haben uns für das Celery Projekt entschieden, um diesen automatisch ablaufen zu lassen oder manuell zu starten.

Administration

Home Benutzerverwaltung ▾ POI ▾ Kriterien ▾ **FlatFinder** Tags 

Wohnungen Abfragen

 **Alle Aktualisieren**

Eingetragene Wohnungen

Stadt	Postleitzahl	Zuletzt aktualisiert	Anzahl Wohnungen
Fehraltorf	8320	25.05.2014	7
Pfäffikon ZH	8330	25.05.2014	23

Abbildung 5-5: Screenshot des Flat Finders

6 SW-Projektdokumentation

6.1 Vision

Das Ziel und die Vision der Arbeit sind in 4.2 beschrieben.

6.2 Anforderungsspezifikation

6.2.1 Anforderung an die Arbeit

Die SA sollte einen Wohnortfinder hervorbringen, welcher von ZOW verwendet werden kann, um das Zürioberland als ausgezeichnete Wohnregion zu präsentieren.

6.2.2 Use Cases

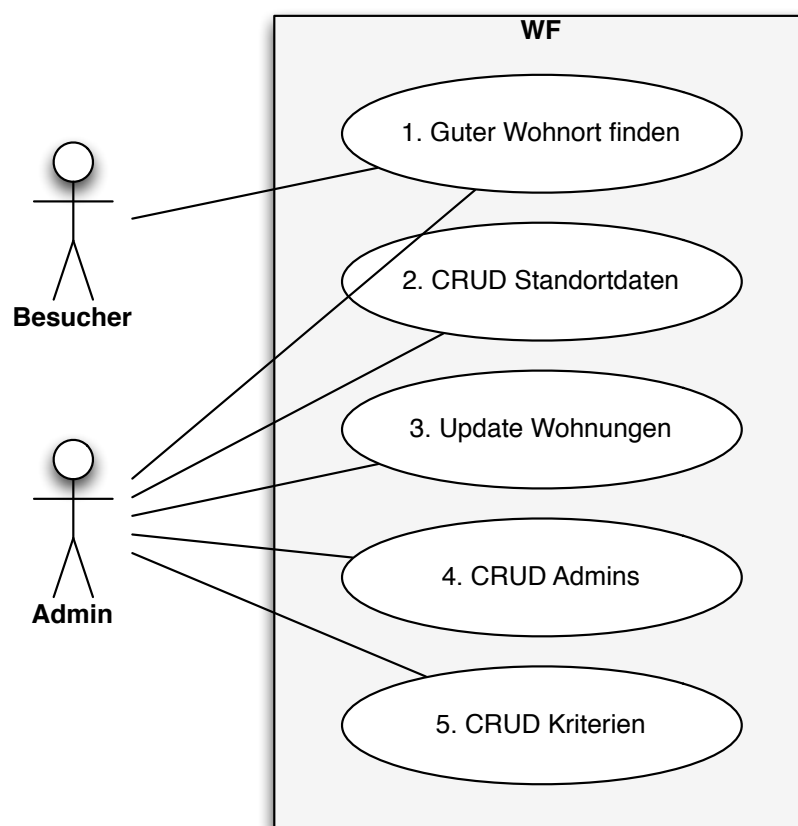


Abbildung 6-1: Haupt-Use Cases vom WF

6.2.2.1 Actors

Es ist jeweils die männliche und weibliche Form gemeint, einfachheitshalber wird jedoch nur die männliche verwendet.

6.2.2.1.1 Besucher

Der Actor Besucher repräsentiert alle Benutzer der Webseite, die keine Administratoren sind. Sie haben im Prinzip nur Interesse am Use Case 6.2.2.2.1 Guter Wohnort finden.

6.2.2.1.2 Admin

Administratoren sind Benutzer der Webseite, welche die Webseite verwalten wollen. Im Prinzip sind dies ausschliesslich Mitarbeiter von ZOW sowie deren Subunternehmen. Sie haben Interesse an allen Use Cases.

6.2.2.2 Use Cases

6.2.2.2.1 Guter Wohnort finden

Durch die Eingabe von Kriterien für seinen optimalen Wohnort wird dem Benutzer angezeigt, wo es für ihn Sinn machen würde, zu wohnen. Diese Anzeige geschieht mit einer Heatmap über der Karte.

Ausserdem werden ihm auf der Karte verschiedene freie Wohnungen angezeigt. Diese kann er nach gewissen Kriterien filtern.

6.2.2.2.2 CRUD Standortdaten

Um den optimalen Wohnort zu finden, bedarf es einige grundlegende Standortdaten. Diese können Admins verwalten. Das heisst, neue Punkte erstellen, bestehende anzeigen, bearbeiten und löschen.

6.2.2.2.3 Update Wohnungen

Die Aktualisierung der Liste von freien Wohnungen kann auch manuell angestossen werden.

6.2.2.2.4 CRUD Admins

Es können neue Admins erstellt, bestehende angesehen, bearbeitet und gelöscht werden – also die Admins verwaltet werden.

6.2.2.2.5 CRUD Kriterien

Es können Kriterien erstellt, bestehende angesehen, bearbeitet und gelöscht werden – also die Kriterien verwaltet werden.

6.2.3 System-Sequenzdiagramme

Fast alle Use Cases sind im Prinzip ganz einfache CRUDs und bedürfen keinem System Sequenzdiagramm. Der einzige spannende Use Case ist 6.2.2.2.1 Guter Wohnort finden, dieser ist deshalb auch in Abbildung 6-2 abgebildet.

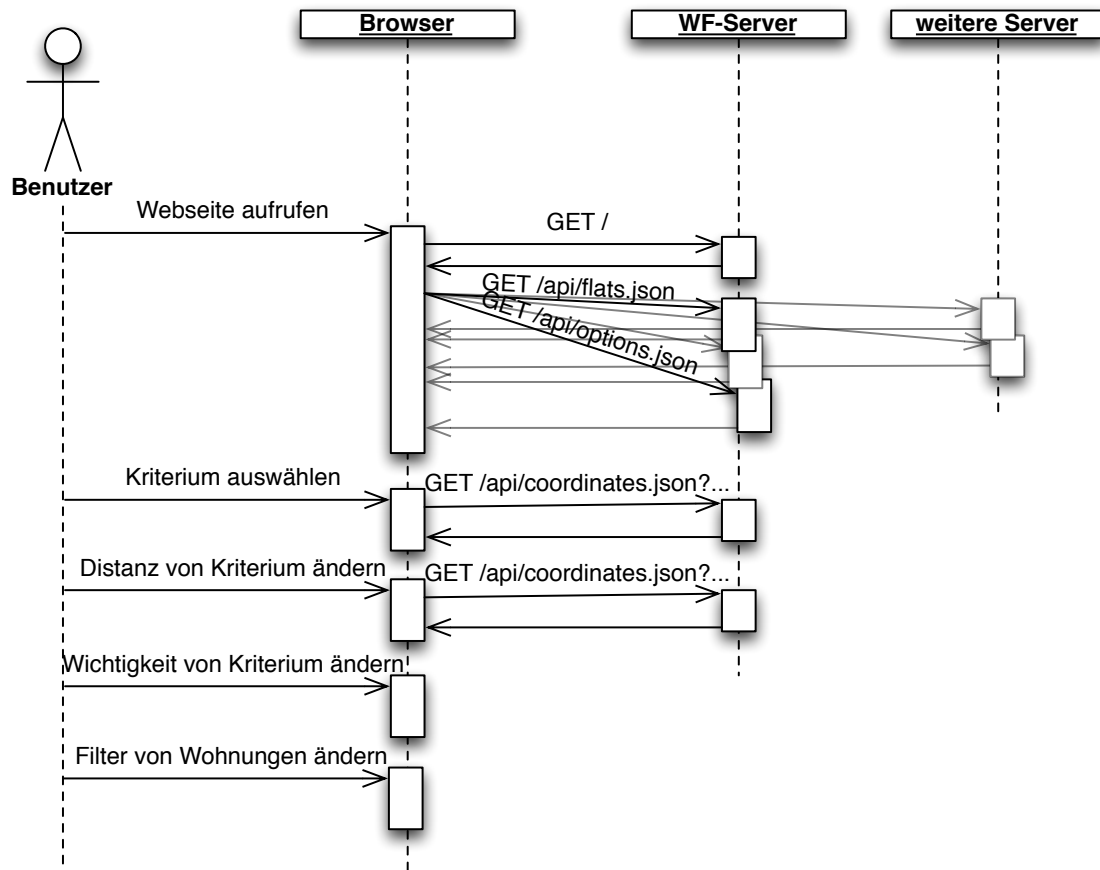


Abbildung 6-2: System Sequenz Diagramm vom Use Case 6.2.2.2.1 Guter Wohnort finden

6.2.3.1 Webseite aufrufen

Wenn der Benutzer die Webseite aufruft, wird ihm die ganze Anzeige übermittelt, dazu gehören HTML-, CSS- und JS-Daten. Zusätzlich erhält er die ganze Liste aller Wohnungen und die Liste aller möglichen Kriterien zur Erstellung der Heatmap.

6.2.3.2 Kriterium auswählen/Distanz von Kriterium ändern

Wenn der Benutzer ein neues Kriterium zur Anzeige auf der Karte auswählt oder wenn er die Distanz, in welcher das Kriterium für ihn sein muss, ändert, lädt er den Teil der Heatmap für dieses Kriterium (neu) vom Server.

6.2.3.3 Wichtigkeit von Kriterium ändern

Wenn der Benutzer die Wichtigkeit eines angezeigten Kriteriums ändert, verändert er die Transparenz der einzelnen Teile der Heatmap. Es ist dazu keine Anfrage an einen Server nötig.

6.2.3.4 Filter von Wohnungen ändern

Wenn der Benutzer neue Kriterien für seine Wohnung angibt, so filtert er die beim Aufruf der Webseite übertragene Liste aller freien Wohnungen. Es ist dazu keine neue Anfrage an einen Server nötig.

6.2.4 Weitere Funktionen

In den vorhergehenden Kapiteln wurden alle zu Beginn geforderten Funktionen aufgelistet. Der Kunde hatte zu Beginn keine weiteren zwingend geforderten Funktionen.

6.2.5 Nicht funktionale Anforderungen

Der Kunde hatte zu Beginn nicht direkt nicht funktionale Anforderungen bezogen auf das endgültige Produkt WF gestellt. Er hatte jedoch eine rund 6000 Zeilen umfassende Excel-Datei mit POIs aus der Region Zürioberland, die er gerne als Daten-Grundlage des WF verwenden wollte.

6.3 Analyse und Design Spezifikationen

6.3.1 Domain Modell

Der WF hat leider kein sonderlich spannendes Domain Modell, da keine komplexen Datenstrukturen gespeichert werden müssen, sondern es primär darum geht, geografische Punkte mit Key-/Value-Paaren an zusätzliche Information zu speichern.

Deshalb besteht das Domain Modell im Prinzip aus einem Haufen von ziemlich unzusammenhängenden (Konfigurations-)Daten, wie es in Abbildung 6-4 zu sehen ist. Zusätzlich dazu gibt es noch eine etwas spannendere und ausführlichere Ansicht der geografischen Daten mit ihren Zusatzinformationen. Dies ist in Abbildung 6-3 aufgezeigt.

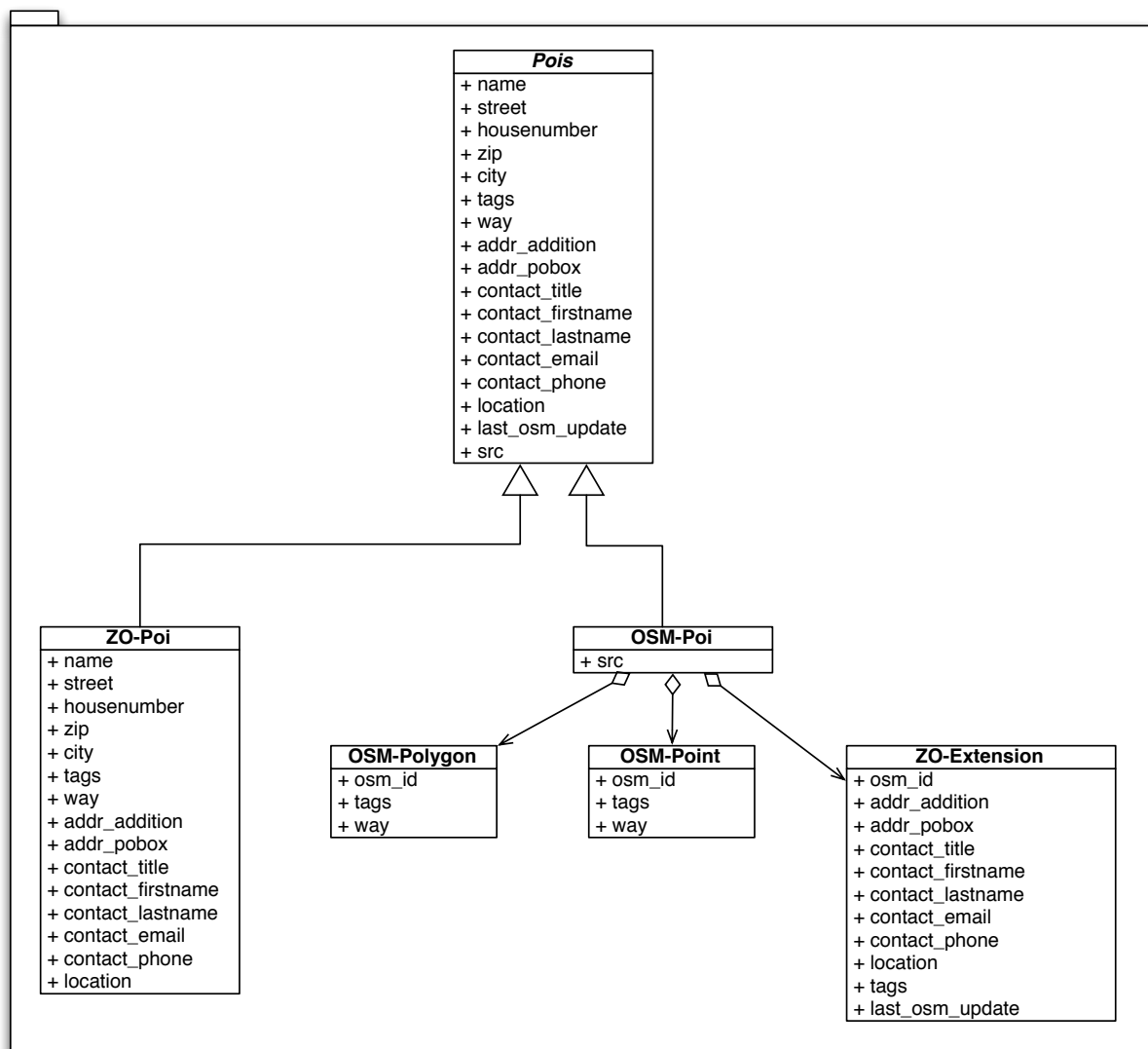


Abbildung 6-3: POI-Daten Klassen Diagramm

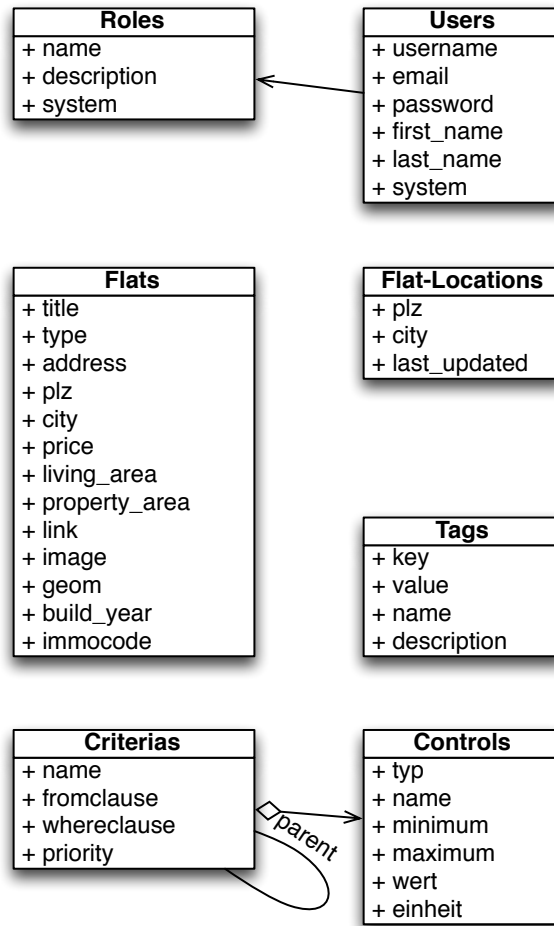


Abbildung 6-4: Überblick über die restlichen Klassen

6.3.2 Objektkatalog

Folgend ist eine Aufzählung der wichtigsten Klassen des WF zu finden.

6.3.2.1 Pois

Dies ist ein Interface, welches die Informationen auflistet, die implementierende Klassen liefern müssen. Ein Objekt dieses Interfaces repräsentiert ein Element – Punkt oder Fläche – auf der Karte. Es kann, muss aber noch nicht, geografisch zuordenbar sein.

6.3.2.2 ZO-Poi

POIs, die nicht in OSM sind und nur intern bei ZOW gespeichert werden.

6.3.2.3 OSM-Poi

POIs, die in OSM eingetragen sind und mit ZOW-internen-Daten ergänzt sind. Sie haben entweder einen OSM-Polygon oder einen OSM-Point. Und fakultativ noch eine ZO-Extension.

6.3.2.4 OSM-Polygon

Dies ist eine Fläche aus OSM mit einigen Attributen.

6.3.2.5 OSM-Point

Dies ist ein Punkt aus OSM mit einigen Attributen.

6.3.2.6 ZO-Extension

Dies sind ergänzende Infos zu Punkten in OSM, die nur ZOW-intern verwendet werden.

6.3.2.7 Tags

Dies ist eine Liste von Tags, die zu einem Punkt hinzugefügt werden können, wie beispielsweise „shop=mall“.

6.3.2.8 Roles

Dies sind Rollen, die ein Benutzer einnehmen kann und ihm gewisse Rechte geben.

6.3.2.9 User

Dies ist ein Benutzer, der sich im Backend anmelden kann.

6.3.2.10 Flats

Dies ist ein Inserat für eine Wohnung, welches auf dem GUI angezeigt wird und dort nach gewissen Kriterien gefiltert werden kann.

6.3.2.11 Flat-Locations

Dies ist die Liste aller Orte (mit PLZ), in welchen wir von einer externen Datenquelle Inserate für freie Wohnungen abrufen.

6.3.2.12 Criterias

Dies ist eine Liste von Kriterien, die der Benutzer schlussendlich auf der Webseite auswählen kann. Zusätzlich enthält sie noch Attribute, die angeben, wie für das entsprechende Kriterium Daten aus der Datenbank herausgelesen werden können.

6.3.2.13 Controls

Dies sind spezielle Kontrollfelder (z.B. Toleranzdistanz-Slider), die dem Benutzer Möglichkeiten geben ein Kriterium für sich anzupassen.

6.3.3 Architektur

Der WF (als Webanwendung) wurde in mehrere Teile unterteilt. Auf der obersten Ebene gibt es drei Ordner.

- skeleton
Darin sind alle produktiven Daten, die von uns direkt nötig für den WF sind.
- tests
Darin sind Backend-Tests, die den WF testen.
- venv
Darin sind generierte Daten, die für den WF nötig sind.

Im skeleton-Ordner liegen die wichtigsten Dateien. Diese sind wiederum aufgeteilt, nämlich in:

- helpers
Darin sind Hilfsdateien, die für die von anderen Dateien verwendet werden.
- modules
Das ist der Hauptteil, darin werden die Webseiten definiert, verwaltet und gerendert. Sie enthalten quasi die gesamte Funktionalität.
- static
Darin sind alle statischen Daten wie CSS, JS, Bilder und Ähnliches, welche direkt an den Besucher der Webseite ausgeliefert werden können.

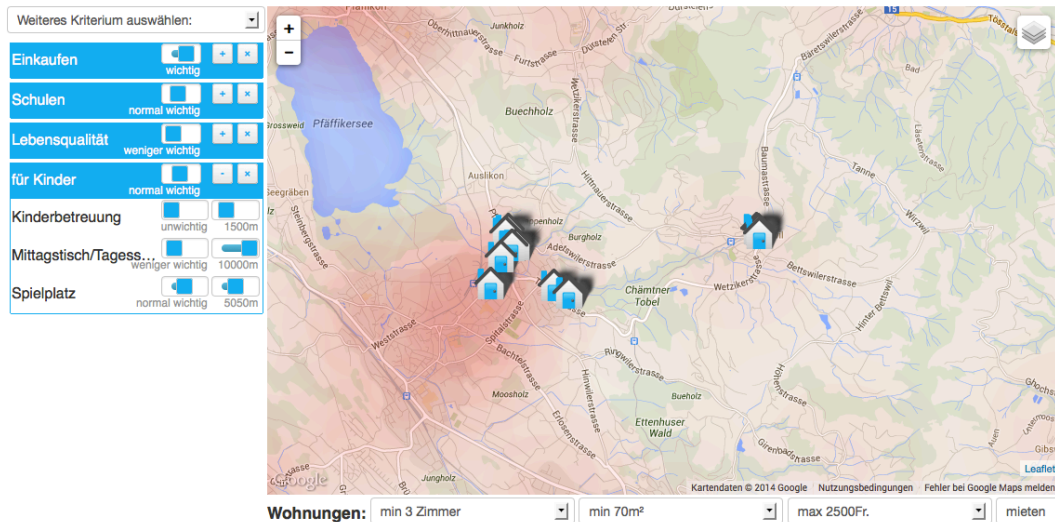


Abbildung 6-9: Printscreen vom WF am Projektende

6.3.5 Auswahl der angezeigten Kriterien

Ein weiterer spannender Punkt in unserem Projekt war es, welche Kriterien der Benutzer zur Auswahl hatte. Dazu haben wir unter mehreren Personen eine Umfrage gemacht, nach was sie ihren perfekten Wohnort aussuchen würden. Schlussendlich ergab dies die Liste in Tabelle 1.

Vorschlag 2	WHERE	Effektive Bezeichnung
Einkaufen	shop=supermarket OR shop=convenience	Supermarkt
Einkaufen	shop=bakery	Bäckerei
Einkaufen	shop=butcher	Metzgerei
Einkaufen	shop=dairy	Käserei
Einkaufen	shop=market	Markt
Einkaufen	shop=farm	Bauernhof
Einkaufen	shop=clothes OR shop=shoes	Kleider und Schuhe
Dienstleistungen	shop=chemist OR shop=pharmacy	Drogerie/Apotheke
Dienstleistungen	amenity=post_office	Poststellen
Einkaufen	shop=specialty OR shop=specialty_grocer	Regionale Spezialitäten
Schulen	amenity=kindergarten	Kindergarten
Schulen	school=primary	Primarschule
Schulen	school=secondary	Oberstufe
Schulen	school=special	Sonderschule
Schulen	school=private	Privatschule
Schulen	school:CH=berufsschule	Berufsschule
Schulen	school=gymnasium	Maturitätsschule
Gesundheit	doctors=pediatrician	Kinderarzt
Gesundheit	amenity=dentist	Zahnarzt
Gesundheit	doctors=gp	Allgemeinmedizin
Gesundheit	amenity=care_home	Spitex
Dienstleistungen	shop=hairdresser	Coiffeur
Freizeit	leisure=sports_centre	Sportzentrum
Gesundheit	amenity=hospital	Spital
Freizeit	amenity=culture_centre	Kulturzentrum
Freizeit	amenity=library	Bibliothek
Schulen	school=music	Musikschule/-unterricht
Restaurant	amenity=restaurant OR amenity=fast_food	Restaurant
Freizeit	zo_leisure=events	Events
Freizeit	tourism=museum	Museum
Freizeit	leisure=pitch	Sportplatz
Freizeit	leisure=swimming_pool	Schwimmbad
Freizeit	leisure=ice_rink	Eishalle
Freizeit	amenity=cinema	Kino
Freizeit	amenity=theatre	Theater
für Kinder	amenity=nursery	Kinderbetreuung
für Kinder	amenity=childcare	Mittagstisch/Tagesschule
Alters- und Pflegeheime	amenity=retirement_home	Alters- und Pflegeheime
Einkaufen	shop=bicycle	Veloladen
Einkaufen	shop=optician	Optiker
für Kinder	leisure=playground	Spielplatz
Lebensqualität	leisure=park	Park
Kirchen	amenity=place_of_worship	Kirchen
Lebensqualität	landuse=forest	Wald
Lebensqualität	water=lake	See
Steuern		Steuersatz
Mobilität	railway=station	Bahnhof
Mobilität		Wie lange nach ...?
Schulen	amenity=university	Universität
Lebensqualität		Aussicht
Mobilität	highway=bus_stop	Bushaltestelle

Tabelle 1: Liste von Kriterien für den Benutzer

6.4 Implementation und Test

6.4.1 Implementation

Der WF wurde gemäss der Architektur in 6.3.3 implementiert. Der WF verwendet keine ausführliche Objektstruktur, sondern verwendet prinzipiell nur die in Abbildung 6-3 und Abbildung 6-4 aufgelisteten Klassen. Interessant ist jedoch die Speicherung in der Datenbank, besonders des OSM-Teils und der Integration der ZOW-Daten.

Als Datenbank wurde PostgreSQL¹ gewählt, da diese Datenbank gute Unterstützung für geografische Daten bietet.

6.4.1.1 Datenbankstruktur aller Nicht-OSM-Daten

Wie bereits in 6.3.3 beschrieben, hat der WF sechs voneinander unabhängige „Pakete“. Diese werden auch so in der Datenbank abgebildet, ohne Referenzen zwischen einander. Die Speicherung in der Datenbank erfolgt, wie in Abbildung 6-10 aufgezeigt.

¹ <http://www.postgresql.org/>

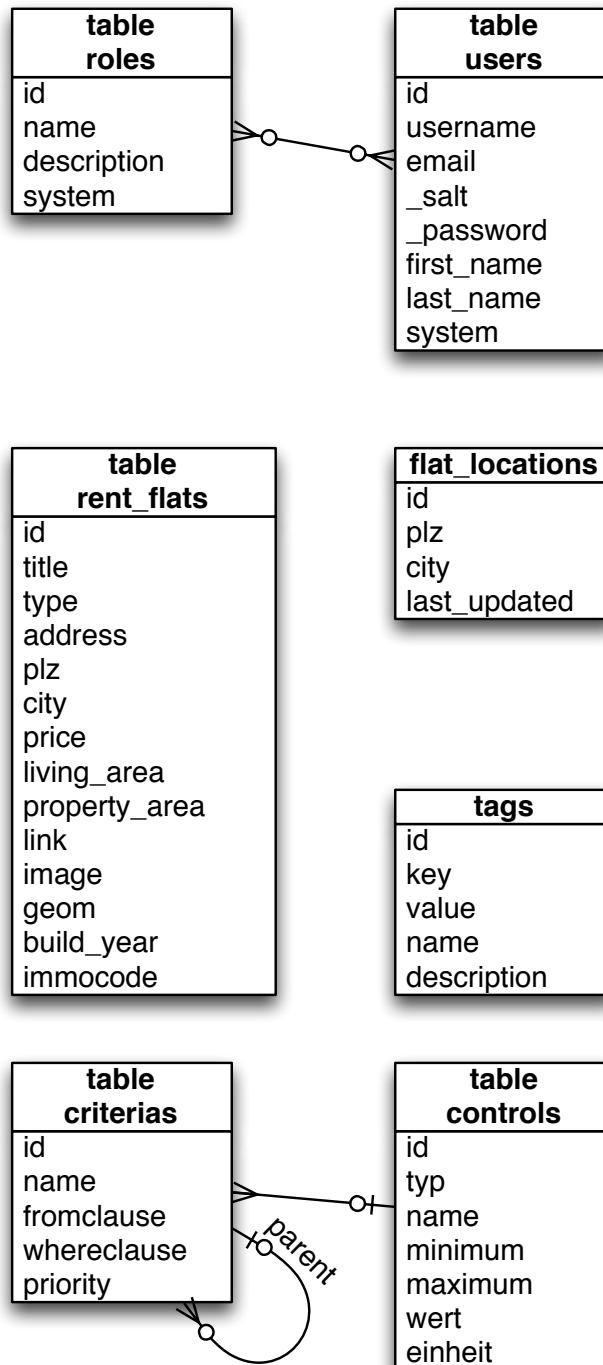


Abbildung 6-10: Datenbankschema (ausser OSM-Daten)

6.4.1.2 Datenbankstruktur aller OSM-Daten

Die Datenbankstruktur der OSM-Daten ist äusserst interessant. Ein Überblick ist in Abbildung 6-11 aufgezeichnet.

Der WF verwendet zwei Datenquellen. Auf der einen Seite verwendet er OSM Daten (osm_point und osm_polygon) und auf der anderen Seite verwendet er interne Daten von ZOW (zo_poi).

Bei den OSM-Daten ist der kleinstmögliche Teil, welcher heruntergeladen werden kann, die ganze Schweiz. Und wie in Tabelle 2 ersichtlich ist, ist das beinahe 1 GB Daten. Ein sehr grosse Datenmenge, um darauf effizient eine Heatmap zu generieren. Deshalb haben wir eine View erstellt (osm_poi_view), welche nur die Daten innerhalb des Projektperimeters

enthält. Dies alleine hätte jedoch noch nicht gereicht um die Performance markant zu steigern, sondern wir haben von dieser View auch noch eine Materialized View erstellt (osm_poi_mat). Ausserdem sind in dieser View sicher nicht benötigte POIs bereits ausgeschlossen. Ausgeschlossen sind:

- Bäume (natural=>tree)
- Strassenlaternen (highway=>street_lamp)
- Stromeinrichtungen wie Kraftwerke und Hochspannungsleitungen (Key: power)

Diese Materialized View hat dann nur noch weniger als einem Zehntel der ursprünglichen Grösse der Daten. Und auf dieser kleineren Anzahl von Daten ist es dann effizienter möglich die benötigten Berechnungen vorzunehmen.

Tabelle	Anzahl Zeilen	Speicherbedarf
osm_point	750292	141 MB
osm_polygon	1543248	752 MB
Sub total Original-OSM-Daten	2293540	893 MB
osm_poi_mat	247081	77 MB

Tabelle 2: Grösse der OSM-Tabellen

Nachdem nun die OSM-Daten etwas verkleinert wurden, werden sie nun wieder „vergrössert“. Beziehungsweise werden sie mit ZOW-Internen Informationen angereichert. Die zusätzlichen Informationen zu den bereits in OSM gespeicherten Daten sind in der Tabelle „osm_poi_zo_extension“ gespeichert. Und sie werden zusammen mit den OSM-Daten in der View „osm_poi“ zusammengeführt.

Nicht dabei sind bei dieser View noch die POIs, die ausschliesslich ZOW-intern sind. Diese werden dann in der View „pois“ auch noch dazu genommen. Fazit:

- Alle Infos lesen: „pois“-View
- POI speichern/schreiben:
 - Öffentlicher (OSM-)POI: Direkt auf openstreetmap.org
 - Zusätzliche interne (ZOW-)Infos zu einem öffentlichen (OSM-)POI: „osm_poi_zo_extension“-Tabelle
 - Interner (ZOW-)POI: „zo_poi“-Tabelle

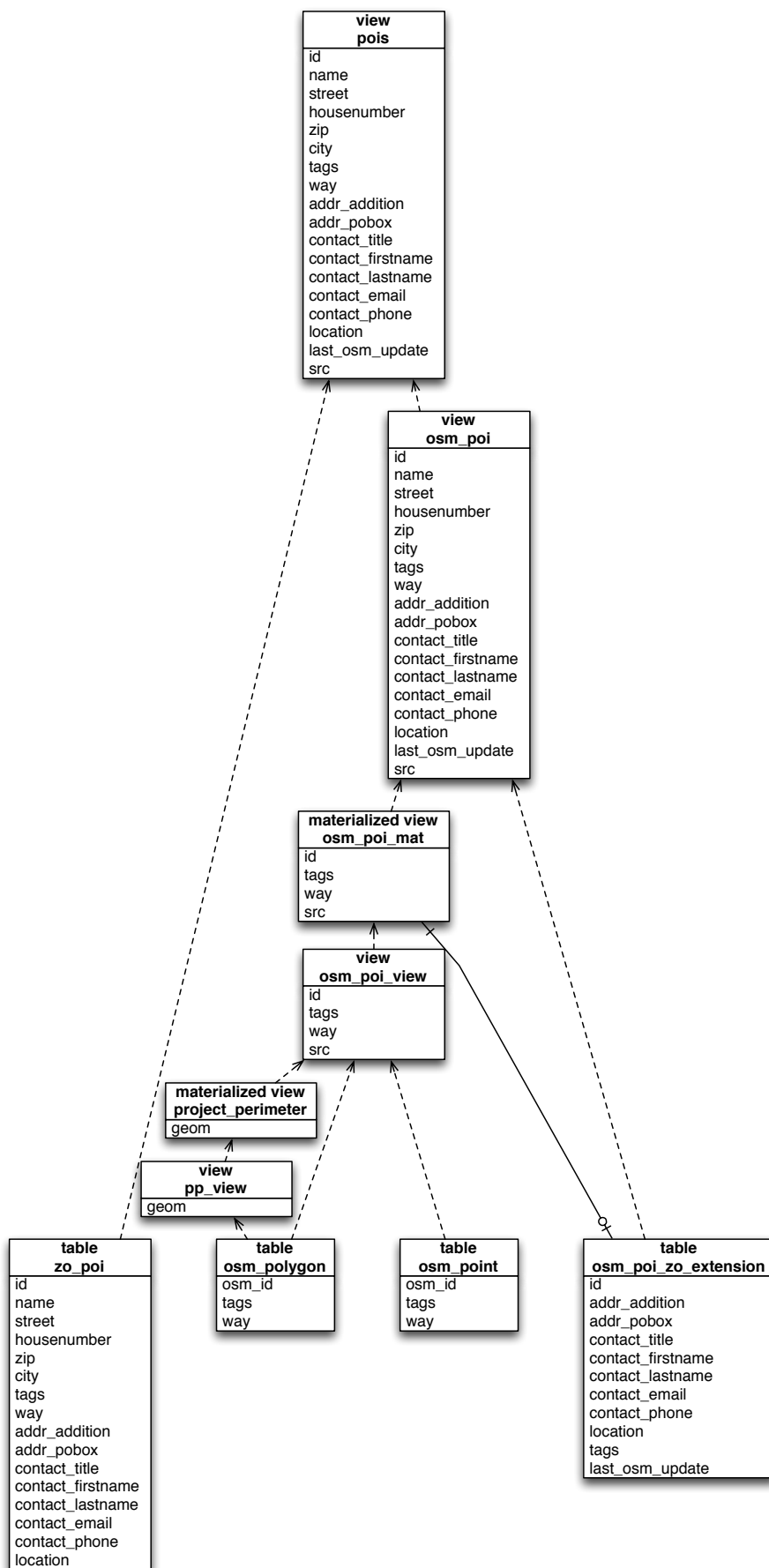


Abbildung 6-11: OSM-Datenbankschema

6.4.1.3 Integration der ZOW-Daten

ZOW verwendete ein Excel, welches sehr viele spannende POIs vom ganzen Zürioberland enthält und im Rahmen vom WF verwendet werden soll. Aus Informatik-sicht ist jedoch ein Excel kein vernünftiges Hilfsmittel für eine strukturierte Datenverwaltung. Deshalb haben wir zusammen mit ZOW beschlossen das Excel abzulösen und in eine Datenbank zu überführen.

Ausserdem enthielt das Excel sowohl Informationen, die für OSM interessant waren (wie z.B. Schulen), als auch solche Informationen, die dies nicht waren (wie z.B. Beratungsfirmen oder Wochenmärkte). Weiter gab es im Excel Informationen, welche für den WF interessant sind und aber auch solche, die dies nicht sind.

Zuerst galt es zu entscheiden, welche Kategorien zu was gehörten und wie damit umgegangen werden soll. In Abbildung 6-12 ist beschrieben, wie die Daten schlussendlich in OSM und in den WF integriert wurden.

6.4.1.3.1 Ausgangslage

Zu Beginn gab es vier verschiedene Zustände, in welcher die Informationen waren:

1. Zum einen gab es Daten, die weder für den WF noch für OSM interessant waren (in Abbildung 6-12 rot)
2. Weiter gab es Daten, die zwar für den WF, nicht aber für OSM interessant waren (in Abbildung 6-12 orange)
3. Ausserdem gab es Daten im Excel, die sowohl für den WF als auch für OSM interessant waren (in Abbildung 6-12 gelb)
4. Und schlussendlich gab es noch Daten in OSM (in Abbildung 6-12 grün)

Natürlich konnten sich die beiden letzteren überschneiden. Es konnten also Daten sowohl im Excel als auch bereits in OSM sein.

6.4.1.3.2 Endzustand

Das Ziel war es, die gelben Daten in Abbildung 6-12 in OSM zu integrieren. Davon hat nicht nur der WF profitiert, sondern dies hat für die Standortförderung Zürioberland auch weitere Vorteile:

- Mit dem Zurverfügungstellen der Daten sowohl für andere Projekte als auch allen Benutzern von OSM wird indirekt Wirtschaftsförderung betrieben. Denn damit werden die lokalen Unternehmen im Internet präsentiert.
- Es ist doch eine beträchtliche Anzahl von Punkten, die jederzeit aktuell gehalten werden sollten. Für ZOW ist dies ein erheblicher Arbeitsaufwand. Mit der Integration der Daten in OSM wird zumindest ein grosser Teil davon von der "Crowd" aktuell gehalten. Dies funktioniert ähnlich wie Wikipedia, jeder kann sein Wissen dazu beisteuern.
- Und weiter hat ZOW auch dadurch profitiert, dass sie noch weitere POIs in ihrem System haben, welche sie nicht erfassen mussten. Dies nach dem Prinzip „Nehmen und Geben“.

Schlussendlich (bzw. erst nach dem Projekt, da ZOW bis zum SA-Projektende noch nicht alle Punkte in OSM integriert hat) war eine beträchtliche Anzahl Punkte in OSM und der WF konnte darauf zugreifen. Der WF lädt automatisch jeden Morgen die aktualisierten OSM-Daten (Stand vom Vorabend) von download.geofabrik.de herunter. Und die internen ZOW-Daten können durch ein Backend im WF verwaltet werden und bei Bedarf auch wieder als CSV heruntergeladen werden.

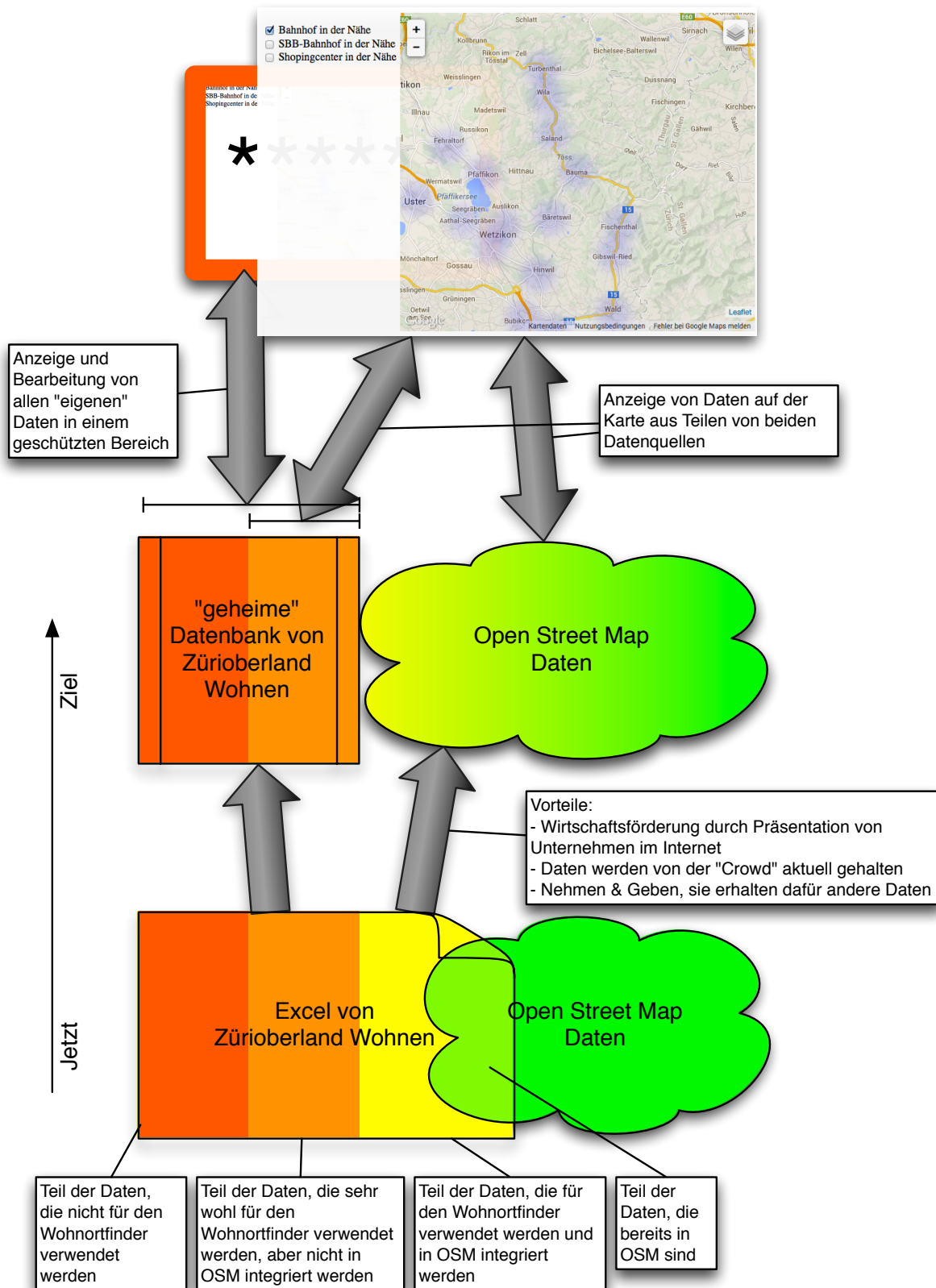


Abbildung 6-12: Handling der ZOW-Daten

6.4.2 Automatische Testverfahren,

6.4.2.1 Python Unit-Tests

Für das automatische Testen des Python-Backendes wurde auf die Python-Testumgebung Nostests gesetzt. Nostests sind einfach aufzurufen und funktionieren sehr gut mit der an-

gelegten Projektstruktur. Um die Datenbankkonnektivität und –abfragen zu testen, wird jeweils eine Testdatenbank aufgesetzt und verwendet und im Anschluss an die Tests wieder gelöscht.

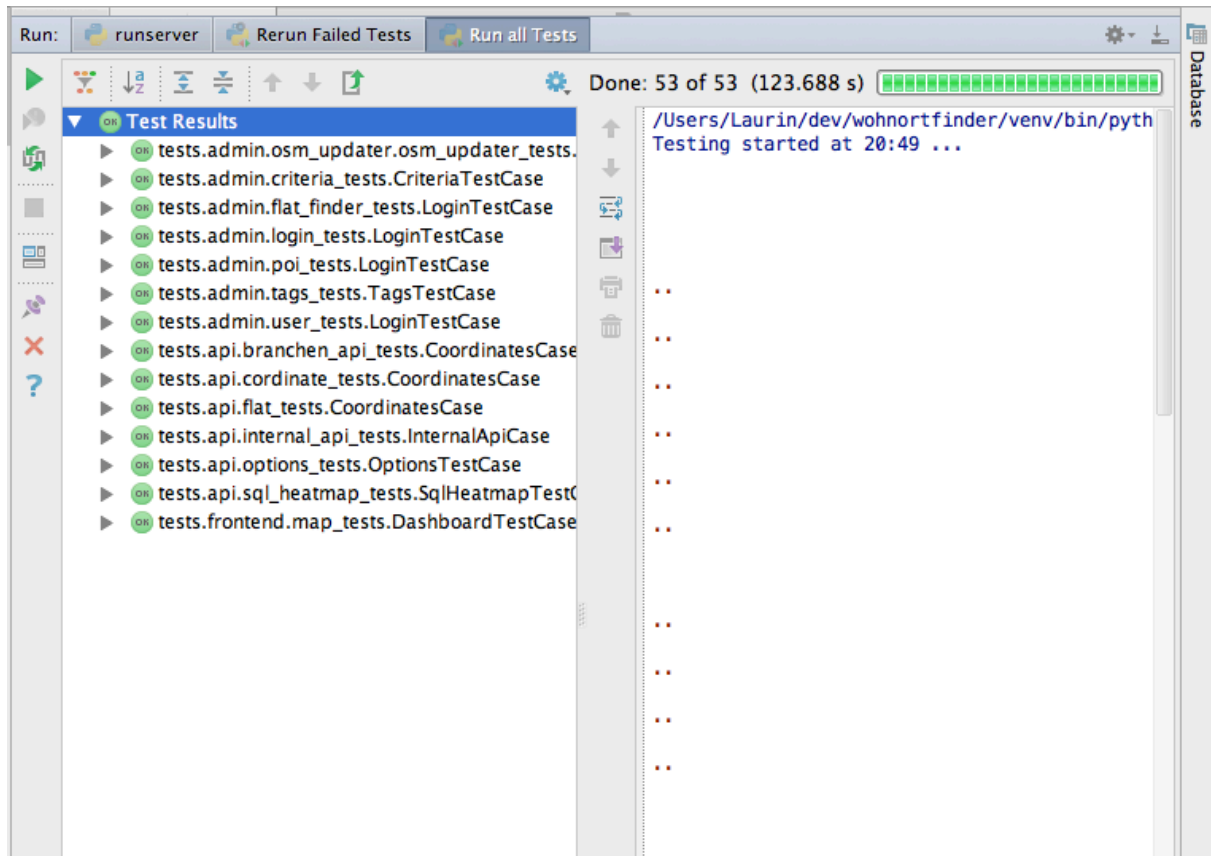


Abbildung 6-13: Erfolgreiche Nosetests

6.4.2.2 JavaScript-Tests

Für das automatische Testen des JavaScript (JS) Teil des Frontendes wurde das JS-Testing-Framework Jasmine verwendet. Es wird stark von AngularJS unterstützt und ist daher die optimale Wahl um auf AngularJS basierenden Code zu testen. Wenn der Server im Entwicklungsmodus (mit Umgebungsvariable „DEBUG=True“ gesetzt) läuft, können die Tests bequem im Webbrowser unter `<BASE_URL>2/jasmine` aufgerufen werden. Beispiele von solchen Testläufen sind in Abbildung 6-14 und Abbildung 6-15 zu finden.

² z.B. `http://localhost:5000`


```

Jasmine 1.3.1 revision 1354556913 finished in 0.332s
.....

Passing 34 specs No try/catch ☐

OptionsSelector controller
  should have no preselected option
  should have only criteria with all mandatory attributes
  should start and stop monitoring if option is (de)selected
  should correctly add and remove watchers for new selected criteria
  should add a criteria to the list of selected ones
  should be able to add criteria more than once
  has a prio-Filter
  should calculate correct human readable priorities
  has a param-Filter
  should calculate correct human readable param-values
  should calculate correct url for an option
  should have no feature displayed at start
  can handle one item
  can handle more than one item
  should calculate correct effective priorities if only one is selected
  should calculate correct effective priorities if two are selected
  should invoke the putOnMap function if criteria is added
  should not invoke putOnMap more than once for the same addition of selectedOptions
  should invoke cleanUpMap if an element is removed
  should invoke updateOpacity if priority of option is changed
  should invoke updateOpacity if priority of subcriterion is changed
  displays the new data after the buffer is changed
  should not send requests for all changes if the last one is still ongoing

FlatController
  should have 3 flats to display at start
  should calculate a marker for the two of them, that have a "geom"
  has a minmax-Filter
  has a minmax-Filter that filters based on <
  has a minmax-Filter that filters based on <=
  has a minmax-Filter that filters based on >
  has a minmax-Filter that filters based on >=
  has a minmax-Filter that filters based on =
  has a minmax-Filter that filters based on ==
  has a minmax-Filter that ignores elements with missing attributes silently
  has a minmax-Filter that ignores elements with missing attributes silently, unless the filter is null
  
```

Abbildung 6-14: Erfolgreichen Jasmine-Testlauf

```

Jasmine 1.3.1 revision 1354556913 finished in 0.283s
● X .....

Failing 1 spec No try/catch ☐
34 specs | 1 failing

OptionsSelector controller should have only criteria with all mandatory attributes.

Expected undefined to be defined.
jasmine.ExpectationResult@http://localhost:5000/_jasmine/testrunner/media/jasmine/jasmine.js:114
jasmine.Matchers.matcherFn_<@http://localhost:5000/_jasmine/testrunner/media/jasmine/jasmine.js:1240
@http://localhost:5000/static/tests/angular_controller_test.js:163
@http://localhost:5000/static/tests/angular_controller_test.js:157
@http://localhost:5000/static/tests/angular_controller_test.js:150
@http://localhost:5000/static/tests/angular_controller_test.js:145
jasmine.Block.prototype.execute@http://localhost:5000/_jasmine/testrunner/media/jasmine/jasmine.js:1064
jasmine.Queue.prototype.next_@http://localhost:5000/_jasmine/testrunner/media/jasmine/jasmine.js:2096
jasmine.Queue.prototype.start@http://localhost:5000/_jasmine/testrunner/media/jasmine/jasmine.js:2049
jasmine.Spec.prototype.execute@http://localhost:5000/_jasmine/testrunner/media/jasmine/jasmine.js:2376
jasmine.Queue.prototype.next_@http://localhost:5000/_jasmine/testrunner/media/jasmine/jasmine.js:2096
jasmine.Queue.prototype.next_/onComplete@http://localhost:5000/_jasmine/testrunner/media/jasmine/jasmine.js:2096
jasmine.Spec.prototype.finish@http://localhost:5000/_jasmine/testrunner/media/jasmine/jasmine.js:2350
  
```

Abbildung 6-15: Jasmine-Testlauf mit einem fehlgeschlagenen Test

6.4.3 Manuelle Testverfahren

Für den WF sind keine manuellen Tests nötig, es wird alles mit Unit-Tests abgehandelt.

6.5 Resultate und Weiterentwicklung

6.5.1 Resultate

Die Resultate der Arbeit sind bereits in 5.5 beschrieben.

6.5.2 Möglichkeiten der Weiterentwicklung

Da das Projekt mit Jira verwaltet wurde, sind auch alle Aufgaben da erfasst. Und wir haben jeweils alle unsere Ideen darin abgebildet. Demnach sind alle noch offenen Tasks Ideen, wie der WF weiterentwickelt werden könnte. Einige davon sind direkte Kundenwünsche, andere sind unsere eigenen Ideen und Vorstellungen. Zudem haben wir für alle möglichen Erweiterungen eine erste Zeitschätzung erstellt.

Die Liste der möglichen Erweiterungen des WF können entweder direkt im Jira³ eingesehen werden oder der Status am Ende des SA-Projekts ist auch in Abbildung 6-16 zu sehen.

T	Key	Summary	P	Original Estimate
	WF-37	Kriterienanalyse anhand von Bestehender Adresse		1 day, 7 hours
	WF-72	Identifikation der einzelnen Kriterien auf der Karte		6 hours
	WF-73	Koordinaten auch in osm_poi_zo_extension speichern		2 hours
	WF-38	Responsive Design		6 hours
	WF-71	Statische URLs (zum Versenden von Links)		4 hours
	WF-103	viele Wohnungssicons zu einem einzigen Icon zusammenfassen (damit es keine Icon-Überflut hat)		3 hours
	WF-46	Druckbare Version von Karte und Wohnungen		4 hours
	WF-101	Steigern der Serverleistung durch Caching von populären Abfragen		6 hours
	WF-98	Neues Kriterium "Aussicht"		1 day, 4 hours
	WF-105	Liste von Adressen (eigene Daten/Wohnungen/...), die nicht an einen Ort zuordenbar sind		6 hours
	WF-97	Neues Kriterium "Wie lange nach..."		2 days
	WF-107	Filter für POI View		1 day
	WF-94	Automatisches Backup der Daten einrichten		4 hours
	WF-27	Social Media Integration		6 hours
	WF-61	Abgleich der Daten mit BFS Daten zur Betriebszählung		2 days
	WF-95	Farbe der Heatmap je nach Kartendarstellung anpassen		3 hours
	WF-49	SSL Connection		2 hours
	WF-67	exclude Lakes and Forests from Heatmap		4 hours
	WF-106	Funktion der Slider überdenken		3 hours
	WF-53	Assign User to entries in POI Table on creation		3 hours

Abbildung 6-16: Liste von weiteren Funktionen aus dem Jira

3

<http://jira.lnh.ch/issues/?jql=sprint%20is%20EMPTY%20and%20status%20!=%3D%20closed%20ORDER%20BY%20priority%20DESC>

6.5.3 Weiteres Vorgehen

Das Projekt, so wie es jetzt ist, funktioniert und hat die Basisfunktionalität, nach welcher es entwickelt wurde. Für das weitere Vorgehen sehen wir im Prinzip drei Optionen, die unter Umständen auch kombinierbar sind.

6.5.3.1 Variante Kunde treibt Projekt weiter

In dieser Variante möchte der Kunde einige, der in 6.5.2 erwähnten weiteren Funktionen, implementiert haben und ist bereit uns (oder unter Umständen auch andere Informatiker) anzustellen. Es kann sein, dass dabei allgemeine oder weitere ZOW-spezifische Funktionen dazukommen.

6.5.3.2 Variante HSR hat Interesse an Weiterentwicklung

In dieser Variante möchte die HSR die Software weiterentwickeln und beispielsweise einem grösseren Publikum zur Verfügung stellen. Dabei sind insbesondere noch Probleme zu lösen für den Fall, dass der Projektperimeter eine Grösse annimmt, die nicht mehr direkt einem Browser zugemutet werden kann (zu viele Punkte um sie alle zu rendern). Ausserdem dürfen in dieser Variante – auch um der Nutzungsvereinbarung gerecht zu werden – einige ZOW-spezifische Funktionen entfernt werden. Für die Umsetzung dieser Variante gibt es wiederum die Option uns oder andere (evtl. HSR interne) Informatiker anzustellen.

6.5.3.3 Variante Projektteam entwickelt weiter

In dieser Variante würde wohl der Fokus auf eine breite Öffnung des WF unter einem anderen Namen liegen, ohne dass viele weitere Funktionen implementiert werden würden. Dabei müssten wohl die gleichen Punkte wie in 6.5.3.2 gelöst werden, namentlich Probleme mit der Grösse des Projektperimeters sowie der Entfernung von ZOW-spezifischen Funktionen. Diese Variante würde wohl durch Einblenden von Werbung finanziert werden.

6.6 Projektmanagement

Wir hatten das grosse Glück, mit unserem Kunden ein iteratives Vorgehen umsetzen zu können. Wir haben im grossen und ganzen in Zwei-Wochen-Sprints gearbeitet. Diese haben jedoch - je nach unserer Auslastung in anderen Studienmodulen - teilweise ein paar Tage länger oder kürzer gedauert. Im Durchschnitt haben die Zeiten aber gestimmt.

6.6.1 Sprint

Sprints sind im Wesentlichen nach folgenden Punkten abgelaufen:

1. Wir haben uns Gedanken gemacht, was im nächsten Sprint Sinn machen würde und haben dies dem Kunde vorgeschlagen.
2. Zusammen mit dem Kunden haben wir diese Issues im geplanten Sprint und den Backlog besprochen und teilweise noch Issues abgetauscht.
3. Wir haben zwei Wochen an den Issues im Sprint gearbeitet und sie dann abgeschlossen. Wenn es grössere Probleme mit den Issues gegeben hat, haben wir sie versucht auszuklammern, dafür einen neuen Issue erstellt, und den aktuellen Issue abgeschlossen. Die Aufteilung der Issues im Projektteam verlief nach dem Lust-Prinzip. Wer gerade Lust auf einen Issue hatte, hat ihn sich zugewiesen. Bei Issues, die niemand wollte, haben wir die Aufteilung ausgehandelt.
4. Nach dem Abschliessen eines Issues ging er in den Review und wurde da vom jeweils anderen Projektmitglied definitiv als erledigt markiert.
5. Am Ende jedes Sprints wurde das Erreichte dem Kunden (wenn möglich persönlich, sonst per Mail) präsentiert und der nächste Sprint wurde eingeleitet.

6.6.2 Prototypen, Releases, Meilensteine

Immer am Ende eines Sprints war ein Meilenstein, es gab einen Release und es gab einen funktionierenden Prototypen. Wenn wir dachten, dass die Neuerungen für den Kunden besonders interessant seien, haben wir den Prototypen für ihn im Internet zugänglich installiert.

6.6.3 Team, Rollen und Verantwortlichkeiten

Da wir nur zu zweit sind, mussten wir uns im Team nicht gross aufteilen. Es hat sich jedoch so ergeben, dass Remo Liebi sich mehr um den Backend-Bereich gekümmert hat und Laurin Murer sich mehr dem Frontend und den OSM-Daten auseinandergesetzt hat.

6.6.4 Aufwandschätzung, Zeitplan, Projektplan

Aufwandmässig war das Projekt sehr gut skalierbar. Hart gefordert war nur ein Teil, doch es hatte sehr viele spannende optionale Punkte. Somit konnte der Kunde an zweiwöchentlichen Meetings, jeweils die ihm wichtigen Punkte einbringen und musste sich nicht zu Beginn auf alle Features festlegen. Der Projektplan wurde in Form von Issues geführt, welche es abzuarbeiten galt. Die Liste der zwingenden Tasks ist in Abbildung 6-17 zu finden. Neben diesen 17 Tasks haben wir aber natürlich noch weitere Tasks erledigt. Eine Übersicht über diese ist in Abbildung 6-18 zu finden.

Immer, wenn wir einen neuen Task erstellt haben, haben wir auch gleich geschätzt, wie lange man dafür braucht. Die jeweiligen Schätzungen sind in den beiden Abbildungen auch zu erkennen.

T	Key	Summary	P	Original Estimate
	WF-1	Projektinfrastruktur aufsetzen		3 days
	WF-13	Server Hello World erstellen (inkl. ersten Test geschrieben)		4 hours
	WF-15	Datenbank erstellen und Daten laden		1 day
	WF-16	Datenbank-Abfrage des Servers an den Client ausliefern		4 hours
	WF-17	Karte mit (statischer) Heatmap integrieren		4 hours
	WF-18	Heatmap dynamisch machen mit (statischen?) Serverdaten		4 hours
	WF-19	Option zur Anzeige von 2 verschiedenen vom Server geladenen Heatmaps (inkl. ersten Test geschrieben)		1 day
	WF-26	Create Installation Routine		5 hours
	WF-28	Slider Implementieren		4 hours
	WF-29	Neue Kriterien hinzufügen		5 hours
	WF-30	Kriterien aus Datenbank erhalten		2 hours
	WF-44	Mockup der View		3 hours
	WF-56	Ausgewähltes Mockup implementieren		6 hours
	WF-57	Liste von möglichen Kriterien im Frontend dynamisch anzeigen lassen		6 hours
	WF-68	Kriterienliste dynamisch generieren		1 day
	WF-69	Sinnvolle Daten auf Heatmap anzeigen		1 day
	WF-99	Sinnvolle Standardeinstellungen beim 1. Besuch der Webseite		3 hours

Abbildung 6-17: Liste von zwingend zu erledigenden Tasks

T	Key	P	Summary	Original Estimate
☑	WF-5	↑	Genauen Projektumfang finden	2 days
☑	WF-6	↑	Framework auswählen	2 days
☑	WF-7	↑	Technologien studieren	1 hour
☑	WF-11	↓	API comparis	6 hours
☑	WF-12	↑	Servermöglichkeiten abklären	0 minutes
+	WF-20	↓	Mappingtabelle der Kategorien der ZOW-Excel-Tabelle erstellen	1 day
+	WF-21	↑	Comparis Crawler implementieren	1 day
☑	WF-22	↑	document commit strategy	3 hours
+	WF-25	↑	Set up worker thread for screenscraper	4 hours
+	WF-31	↑	Wohnungsangebote als JSON ausgeben	6 hours
+	WF-32	↑	Wohnungsanzeigen auf Karte anzeigen	3 hours
+	WF-33	↑	Filter für Wohnungsanzeigen	4 hours
+	WF-34	↑	AdminPanel mit Login Funktion	5 hours
+	WF-39	↑	AdminPanel: Update Mechanismus der OSM Daten	1 day, 1 hour
☑	WF-45	⊘	Serverabbild Skizze aufarbeiten	1 hour
☑	WF-47	↑	Server bestellen	3 hours
☑	WF-50	↑	Kryutek Server in Betrieb nehmen	4 hours
🔗	WF-52	⊘	reorganize package	1 day, 2 hours
🔗	WF-54	↑	POIs-Bearbeitung besser verknüpfen	1 day
+	WF-55	⊘	Weitere Mock-Ups entwerfen	3 hours
+	WF-58	↑	einmaliger Import aus dem ZOW-Excel planen	4 hours
☑	WF-60	⊘	Ersatz für HSR-Server	3 hours
+	WF-62	↑	API für branchenfinder	1 day
+	WF-63	↑	Tag Tabelle erstellen	5 hours
+	WF-64	↑	Design an ZO anpassen	1 day
🔗	WF-70	↑	Dokumentation vorantreiben	0 minutes
☑	WF-74	↑	Präsentation 19. Mai vorbereiten	2 hours
+	WF-75	↑	Export Data as CSV	6 hours
+	WF-77	↑	newhome.ch-API einbinden	6 hours
+	WF-94	↑	Automatisches Backup der Daten einrichten	4 hours
🔗	WF-96	↑	Kriterien nur 1x auswählbar machen	1 hour
☑	WF-100	↑	Integration der bestehenden ZO-Excel-Daten in die endgültige Installation	3 hours
+	WF-102	↑	Aktualisierungen der Daten laufen automatisch ab	1 hour

Abbildung 6-18: Liste von weiteren, fakultativen Tasks die auch im Projekt erledigt wurden

6.6.5 Risiken

Für die SA haben wir auch eine Risikoanalyse erstellt und wir haben im Ganzen nur ein kritisches Risiko, für welches es keine Prävention gibt. Dies ist der Ausfall eines Projektmitglieds. Da wir nur zwei Leute sind, wäre ein solcher Ausfall nicht zu kompensieren. Falls ein Mitglied ausfallen würde, gäbe es folgende Optionen

- Projektumfang wird (stark) verkleinert
- Projektdauer wird (stark) verlängert
- Projekt wird abgebrochen

Die gesamte Risikoanalyse ist in Tabelle 3 zu sehen.

Risikoanalyse Wohnortfinder

Risikotyp	Nr.	Wahr- sch.	Aus- wirk.	Ampel	Beschreibung	Behandlung und Kontrolle	Hinweise Status	Termin / Nächster Schritt
Standardrisiken								
Entwicklungsumgebung	1	1	2	2	Ausfall Server	Wichtige Daten vom Server werden regelmässig gebackupt	-	Projektende, 25. Mai 2014
Entwicklungsumgebung	2	1	1	1	Ausfall Jenkins	Wichtige Daten vom Server werden regelmässig gebackupt	-	Projektende, 25. Mai 2014
Entwicklungsumgebung	3	1	3	3	Ausfall Jira	Wichtige Daten vom Server werden regelmässig gebackupt	-	Projektende, 25. Mai 2014
Entwicklungsumgebung	4	0	0	0	Ausfall GIT-Repository	-	-	Projektende, 25. Mai 2014
Entwicklungsumgebung	5	1	3	3	Ausfall persönliches Arbeitsgerät	Wichtige Daten werden regelmässig gebackupt	-	Projektende, 25. Mai 2014
Personal	6	1	3	3	Ausfall Projektmitglied	kritisches Risiko	-	Projektende, 25. Mai 2014
Personal	7	2	3	6	Fehlende Zulassung Projektmitglied	Zulassung möglichst schnell abklären	erledigt	-
Personal	8	0	3	0	Teaminterne Konflikte	-	-	Projektende, 25. Mai 2014
Kunden	9	0	2	0	Schwierigkeiten mit Kunde	-	-	Projektende, 25. Mai 2014
Laufzeit und Budget	10	1	3	3	Überschreiten des Zeitbudgets	Funktionsumfang iterativ abstecken (Sprints)	-	Projektende, 25. Mai 2014
Projektbezogene Risiken								
Entwurf und Implementierung	101	2	2	4	Umsetzungsprobleme mit Technologie	Es wird nichts unmöglich sein, es kann einfach länger/lange dauern bis herausgefunden -> Notfalls Betreuer um Hilfe bitten	-	Projektende, 25. Mai 2014
Entwurf und Implementierung	102	2	2	4	Umsetzungsprobleme mit Framework	"	-	Projektende, 25. Mai 2014
Entwurf und Implementierung	103	1	2	2	Umsetzungsprobleme mit Frontendtechnologien	"	-	Projektende, 25. Mai 2014
Entwurf und Implementierung	104	2	2	4	Umsetzungsprobleme mit Kartendaten	"	-	Projektende, 25. Mai 2014
Zulieferer	105	0	2	0	Schwierigkeiten mit Betreuer	-	-	Projektende, 25. Mai 2014
Entwurf und Implementierung	106	3	3	9	Umsetzungsprobleme mit Bezug Wohnungsliste	Anzeige von Wohnungen gleich zu Beginn aus Projekt ausgeschlossen	-	Projektende, 25. Mai 2014
Zulieferer	107	2	2	4	Probleme mit Integration ZOW-Excel in Datenbank	Integration als eines der ersten Punkte definiert, und aktiv mit dem Kunde besprochen. Viel Aufwand seitens Kunde nötig, läuft auch nach Projekt weiter	-	Projektende, 25. Mai 2014
Produkt	108	1	1	1	WF enthält zu viele Funktionen	Funktionsumfang iterativ abstecken (Sprints)	-	Projektende, 25. Mai 2014
Produkt	109	1	2	2	WF funktioniert nicht zuverlässig	Unittests	-	Projektende, 25. Mai 2014
Produkt	110	1	2	2	WF ist schlecht wartbar	Review des Codes durch Projektpartner vor jedem Commit in master-Branch	-	Projektende, 25. Mai 2014
Produkt	111	1	1	1	WF wird vergoldet	Funktionsumfang iterativ abstecken (Sprints), Kunde kann Fokus legen	-	Projektende, 25. Mai 2014

Tabelle 3: Risikoanalyse WF

6.7 Projekt Monitoring

6.7.1 Soll-Ist-Zeit-Vergleich

Die Zeitschätzung war mal genauer, mal weniger genau. Im Durchschnitt hätten wir für die Tasks noch einen Drittel mehr schätzen sollen. Dies sieht man in Abbildung 6-19. Im Verlauf des Projekts haben wir jedoch unsere Schätzungen stetig verfeinert. So sieht man an der grünen Linie, dass sie im letzten Drittel um den Nullpunkt pendelt und es keine übergrossen Ausschläge mehr gibt.

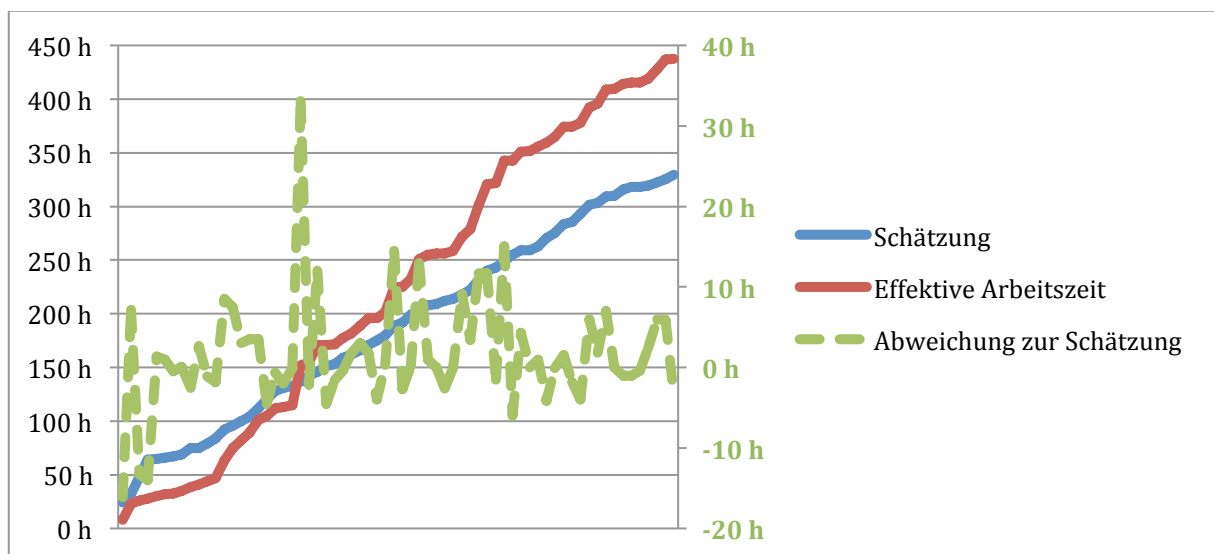


Abbildung 6-19: Schätzgenauigkeit

6.7.2 Codestatistik

Relevante Codestatistiken zu erstellen, ist bei diesem Projekt nicht ganz einfach. Wie man in Tabelle 4 sieht, scheint es so, als ob Remo Liebi alles entwickelt hätte und Laurin Murer nur destruktiv gewirkt hätte.

	Remo Liebi	Laurin Murer
Hinzugefügt	717251	8555
Gelöscht	574258	8889
Total	142993	-334

Tabelle 4: Übersicht über Total geänderte Zeilen

Dies ist aber nur scheinbar so, denn Remo Liebi hat einige Frameworks integriert und damit diesen grossen Vorsprung in der Statistik. Wenn man jedoch die Abbildung 6-20 mit der Aufteilung der Tasks anschaut, sieht man, dass am Projekt sehr ausgeglichen gearbeitet wurde. Remo Liebi hat 35 Issues erledigt und Laurin Murer fast gleich viele (39 Stück). Gefühlsmäßig dürfte dies auch im Programmierteil so sein, auch wenn die Statistik etwas anderes zeigt.

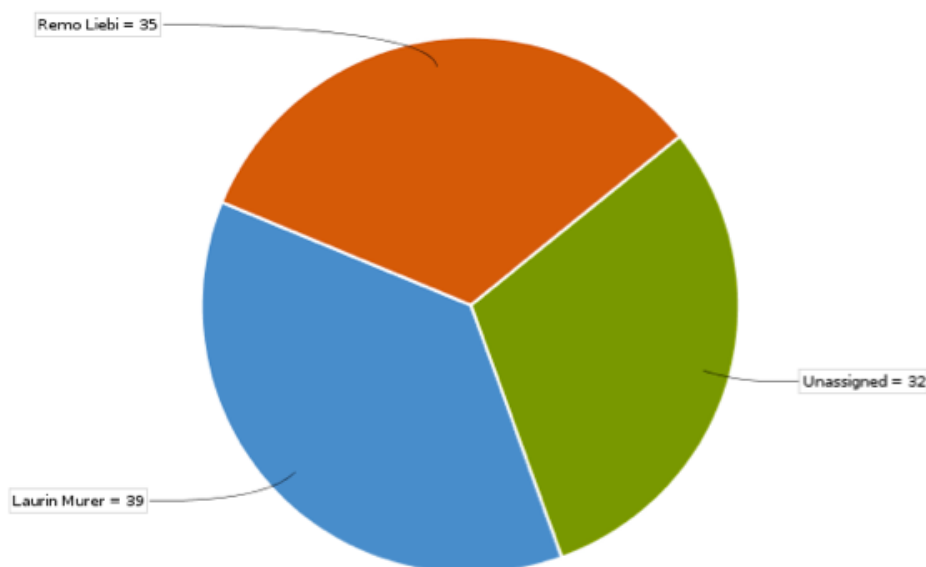


Abbildung 6-20: Aufteilung der Tasks

6.8 Softwaredokumentation

6.8.1 Installation

Die Installation des WFs ist im Entwicklungscod in der Datei „install_wf.txt“ stichwortartig beschrieben. Diese Beschreibung sollte für Informatiker mit Unix-, Postgres- und Python-Kenntnissen genügend ausreichend sein.

6.8.2 Tutorial / Benutzerhandbuch

6.8.2.1 Bedienung des Frondendes

Der WF sollte im Frontend sehr intuitiv und ohne Anleitung zu bedienen sein. Die Bedienungsoberfläche am Projektende ist in Abbildung 6-9 aufgezeigt. Links sind die Optionen aufgezeigt, die Besucher für die Anzeige der Heatmap wählen können. Am unteren Rand

können Besucher Einstellungen zu den angezeigten Wohnungen vornehmen und im Hauptteil (rechts) wird die Karte angezeigt.

6.8.2.2 Administrationsbereich des WF

Um am WF Änderungen und Einstellungen vorzunehmen muss man sich mit Benutzername und Passwort anmelden. Der Administrationsbereich ist unter <http://wohnortfinder.zuerioberland-wohnen.ch/admin/> zu finden.

6.8.2.3 Aktualisierung der Daten

Wenn Daten für den WF aktualisiert werden sollen, muss zuerst abgeklärt werden, um welchen Typ von Daten es sich handelt. Konkret muss man sich die Frage stellen: „Sind ähnliche Daten in OSM?“

6.8.2.3.1 OSM-Daten aktualisieren

Die OSM-Daten können direkt auf openstreetmap.org editiert werden oder zumindest jener Teil davon, der öffentlich ist. Die restlichen Daten zu einem POI können innerhalb des Administrationsbereichs des WF unter dem Punkt „POI“ → „Alle POI“ bearbeitet werden. Auf OSM neu erstellte POIs sind jeweils am darauffolgenden Morgen im WF sichtbar.

6.8.2.3.2 ZO-interne Daten aktualisieren

Die ZO-internen Daten sind im Administrationsbereich des WF unter dem Punkt „POI“ → „Alle POI“ zu bearbeiten. Es können dort auch neue POIs erstellt werden.

6.8.2.4 Backup / Export der Daten

Um aus den Daten des WF wieder ein „Excel-Dokument“ zu machen (also genau genommen ein CSV-Dokument), geht man in den Administrationsbereich des WF und dort auf den Punkt „POI“ → „Exportieren“. Dort kann man nun verschiedene Dokumente herunterladen:

- Alle POI herunterladen: Dies lädt eine „Excel-Liste“ aller POIs (sowohl intern als auch von OSM) herunter. Dieses Dokument wird wohl am häufigsten verwendet.
- eigene POIs herunterladen: Dies ist wie beim ersten Punkt, lädt jedoch nur die eigenen Daten herunter (nichts von OSM)
- eigene Extensions herunterladen: Auch dies ist wie beim ersten Punkt, lädt jedoch nur die eigenen Erweiterungen von OSM herunter (aber weder die OSM-Daten selber, noch die eigenen Daten)
- OSM Daten herunterladen: Und auch dies ist wie beim ersten Punkt, es lädt jedoch nur die (öffentlichen) OSM-Daten herunter (und keine von ZOW)
- Download Backup: Die vorhergehenden Punkte laden zwar alle Daten herunter, doch können sie im Problemfall des WF nicht wieder hochgeladen werden. Genau dafür ist dieser Punkt. Mit der „all_tables.sql“-Datei kann ein Informatiker die aktuellen Einstellungen des WF sowie alle von ZOW erstellten Informationen darauf wiederherstellen. (Tipp: Die Datei kann sehr gut ZIP-komprimiert werden um Speicherplatz zu sparen!)

6.8.2.5 Erstellen neuer Tags

Jeder POI im WF hat sogenannte Tags. Die Tags sind Eigenschaften eines Punktes, bzw. die Beschreibung des Punktes. Sie bestehen immer aus einem Schlüssel (Key) und einem Wert (Value). Nachfolgend einige Beispiele:

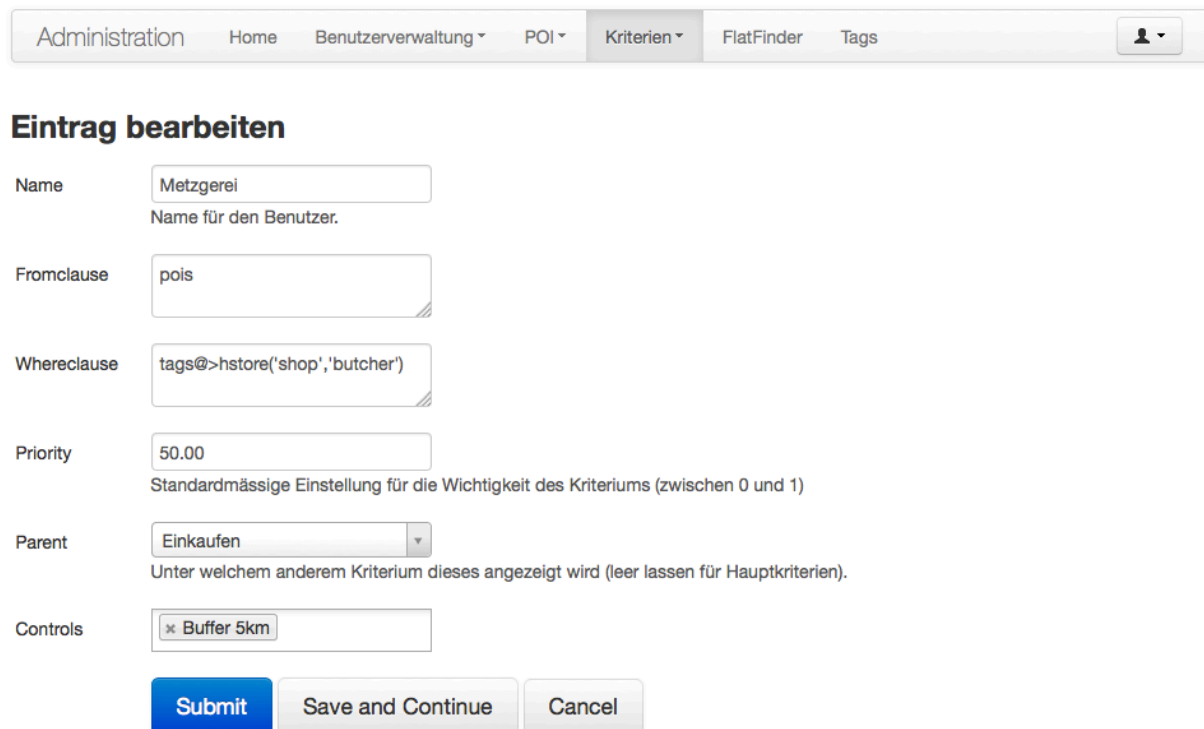
- | | | |
|-------------------|--------------------|--------------|
| • shop=bakery | repräsentiert eine | Bäckerei |
| • shop=butcher | repräsentiert eine | Metzgerei |
| • school=primary | repräsentiert eine | Primarschule |
| • amenity=cinema | repräsentiert ein | Kino |
| • amenity=theatre | repräsentiert ein | Theater |

Jeder Tag kann jeden Schlüssel höchstens 1x besitzen. Wenn also ein Laden Fleisch und Gipfeli verkaufen würde, müsste man entscheiden, ob es jetzt eher eine Bäckerei oder eine Metzgerei ist.

Die Liste aller aktuell im WF bekannten Tags ist im Administrationsbereich des WF unter „Tags“ zu finden.

6.8.2.6 Erstellen neuer Kriterien für den Benutzer

Sollte einmal der Benutzer neue Kriterien im Frontend auswählen können, so kann man dies im Administrationsbereich des WF konfigurieren. Und zwar unter „Kriterien“. Um die Felder richtig auszufüllen, wird am besten angeschaut, wie die bestehenden Kriterien konfiguriert sind, und dann das neue Kriterium entsprechend erstellt. Ein Beispiel einer Konfiguration eines Kriteriums ist auch in Abbildung 6-21 zu finden.



Administration Home Benutzerverwaltung POI Kriterien FlatFinder Tags

Eintrag bearbeiten

Name
Name für den Benutzer.

Fromclause

Whereclause

Priority
Standardmässige Einstellung für die Wichtigkeit des Kriteriums (zwischen 0 und 1)

Parent
Unter welchem anderem Kriterium dieses angezeigt wird (leer lassen für Hauptkriterien).

Controls

Abbildung 6-21: Backend des WF während dem Bearbeiten eines Kriteriums

6.8.2.7 Liste der Wohnungen aktualisieren

Diese Liste sollte automatisch aktualisiert werden und kein manuelles Eingreifen benötigen.

7 Projekt

In diesem Kapitel ist alles Projektspezifisches zusammengefasst, welches nur im Rahmen der SA relevant sind. Das Kapitel dokumentiert unsere Arbeit und zeigt unsere Leistungen auf.

7.1 Sitzungen

Während dem Projekt wurden regelmässig Sitzungen abgehalten. Und zwar wöchentlich mit dem Betreuer, und ca. jede zweite Woche mit dem Kunden. Die Diskussionspunkte mit dem Kunden wurden jeweils mit einer PowerPoint-Präsentation diskutiert. Diese Präsentationen sind auf der CD zum Projekt im Ordner „2 Präsentationen/mit Kunde/“ zu finden.

7.2 Planung des Projekts

7.2.1 Sprints

Wie in 6.6 beschrieben wurde während dem Projekt mit Sprint gearbeitet. Details dazu sind im erwähnten Kapitel zu finden.

7.2.2 Zeitabrechnung

Wie in Abbildung 7-1 zu erkennen ist, haben wir uns während des Projektablaufs ziemlich genau an den Projektplan gehalten. Schlussendlich haben wir für das Projekt total rund 550 Stunden aufgewendet, also rund 70 Stunden mehr als gemäss Studienfach vorgegeben⁴. Am Ende des Projekts hat es noch einen Ausschlag. Dies weil es wie bei jedem Projekt am Ende noch einen geplanten Sonder-Effort gegeben hat und wir das Projekt ein paar Tage vor Abgabetermin fertig haben wollten.

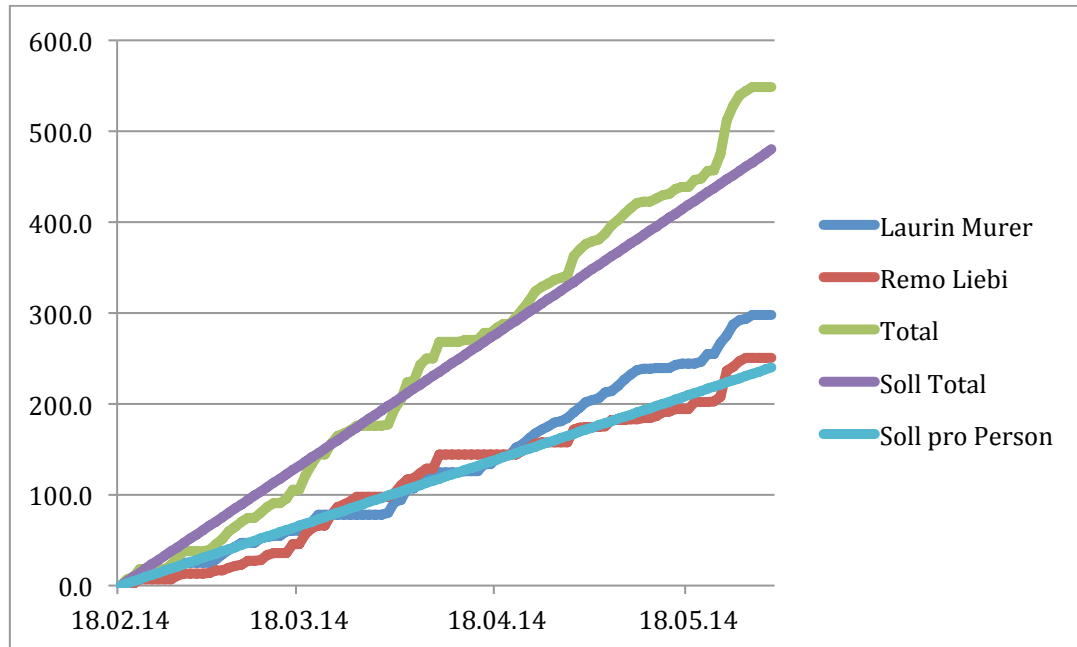


Abbildung 7-1: Zeitaufwand während dem Projektablauf

⁴ bei einem Total von 480 Stunden Aufwand (2 Personen * 8 ECTS-Credits pro Person * 30 Stunden Aufwand pro ECTS-Credit)

7.3 Persönliche Berichte

7.3.1 Remo Liebi

Der WF war mein erstes grösseres Projekt, welches auf Python aufbaut. Ich konnte viele Erfahrungen sammeln und habe es ein spannendes Thema gefunden. Ich bin zum ersten Mal in Kontakt mit einer produktiven Geo-Datenbank gekommen und konnte aus dieser sehr spannenden Materie sehr viel mitnehmen. Die strikte Trennung des Frontendes und Backendes hat es mir ermöglicht, das an der HSR Gelernte anzuwenden und den Mehrwert daraus zu erkennen. Die Zusammenarbeit mit ZOW war sehr angenehm und wir konnten unserer Kreativität freien Lauf lassen. Durch die lockere Atmosphäre mit dem Kunden konnten wir unsere Ideen einbringen und das Projekt auf den Bereich Branchenfinder ausweiten. So haben wir gleich die Arbeitsabläufe bei ZO verbessern können und mit der Verwendung von OSM auch Dienst an der Allgemeinheit bieten können. Gerne werde ich nach dem Abschluss der SA einige Issues, welche wir aus Zeitmangel nicht anpacken konnten, zusammen mit Laurin Murer und ZO weiter implementieren und den WF verbessern.

Heute würden wir das Projekt vermutlich anders angehen. Wir waren mit den Schätzungen des Aufwandes meistens etwas zu knapp. So haben wir für viele Tasks mehr Zeit aufgewendet, als wir eigentlich investieren wollten. Mit einer realistischeren Schätzung und dem Wissen, welches wir uns über die Best Practices für diverse Probleme aneignen konnten, würden wir heute effizienter und zielgerichteter an die Arbeit herantreten.

7.3.2 Laurin Murer

Das Projekt hat mir mehrheitlich grossen Spass bereitet und dies aus verschiedenen Gründen. Der erste Grund ist, dass ich mit meinen verschiedenen Projektpartnern grosses Glück hatte, obwohl ich bei diesem Projekt das erste Mal mit ihnen zusammengearbeitet hatte. Zudem war das Thema in einem Bereich, mit dem ich mich persönlich identifizieren konnte. Es ging um eine Webseite, es ging um geografische Daten und es ging um etwas Innovatives, welches schlussendlich einen effektiven Output bot: den Wohnortfinder.

Leider gab es auch einige negative Punkte, so wurde ich bis zum Schluss nicht mit Python warm. Auch wenn ich mehrheitlich mehr für die Clientseite und die OSM-Daten verantwortlich war, musste ich mich mehrmals mit Umlaut- und Encoding-Problemen herumschlagen. Und auch das verwendete Framework habe ich bis zum Schluss nicht wirklich verstanden und hat mir immer wieder ein Bein gestellt.

Trotzdem, ich würde es wieder mit Python machen. Denn ich lernte wieder eine weitere Sprache und verschiedene neue Konzepte kennen. Allenfalls würde ich jedoch nicht wieder eine neues Framework und eine neue Sprache wählen – sondern nur eines der beiden.

7.4 Unvorhergesehenes

7.4.1 Ausfall HSR Server

Am 22.4.2014 wurden wir darauf aufmerksam gemacht, dass unser HSR Server vom Netz genommen wurde. Dies vermutlich aufgrund eines Hackerangriffs, welcher den Server dann für eine DDOS Attacke gebraucht hat. Wir haben anschliessend den Server neu aufgesetzt und die Konfigurationsroutine getestet.

7.5 Inhalt der CD

- | ____1 Dokumentation
- | | ____Grafiken
- | | | ____Commit Strategy.graffle
- | | | ____Commit Strategy.png

- | | |___database_schema.graffle
- | | |___database_schema.png
- | | |___Domain Modell.graffle
- | | |___Domain Modell_Pois.png
- | | |___Domain Modell_Rest.png
- | | |___Handling der Daten.graffle
- | | |___Handling der Daten.png
- | | |___Mappingtabelle Datenattribute ZOW.xlsx
- | | |___Schätzgenauigkeit.xlsx
- | | |___System-Sequenzdiagramm Guter Wohnort finden.graffle
- | | |___System-Sequenzdiagramm Guter Wohnort finden.png
- | | |___UseCase Diagrams.graffle
- | | |___UseCase Diagrams.png
- | |___SA Wohnortfinder Zürioberland Liebi Murer.docx
- | |___SA Wohnortfinder Zürioberland Liebi Murer.pdf
- | |___Tabellen
- | | |___Risikoanalyse.xls
- | | |___Statistik automatisch zugeordneter Adressen.xlsx
- |___2 Präsentationen
- | |___Poster SA Wohnortfinder Zürioberland.pdf
- | |___Poster SA Wohnortfinder Zürioberland.ppt
- | |___Präsentation Montag.key
- | |___Übernahme der Daten.pptx
- |___3 Decisions
- | |___Auswahl des Frameworks.xlsx
- | |___Auswahl des Servers.xlsx
- | |___Filterkriterien
- | | |___statisik.csv
- | | |___statisik.xlsx
- |___4 Input
- | |___Aufgabenstellung SA Wohnortfinder.pdf
- | |___new home
- | | |___wohnortfinder.csv
- | | |___wohnortfinder_prepared_for_return.csv
- | | |___wohnortfinder_prepared_for_return.json
- | |___Rechte-Vereinbarung.pdf
- | |___Rechte-Vereinbarung_Muster.doc
- | |___vom Betreuer
- | | |___AGIT 2014 - Standortanalysen für Klein und Gross.pdf
- | | |___Computing Isochrones in Multi-Model, Schedule-Based Public Transport Networks.zip
- | |___vom Kunden
- | | |___13_5020_13_Adressliste_Wohnen.xlsx
- | | |___13_5020_17_Adressliste_Wohnen.xlsx
- | | |___13_5061_2_Beschreibung Software Wohnortfinder.docx
- | | |___14_3112_Auftragsliste_Adressdatenbank_Wohnen.docx
- | | |___Adressliste_Wohnen_Aufbereitung.xlsx
- | | |___Logos
- | | | |___wohnen_mit_web_cmyk.jpg
- | | | |___wohnen_mit_web_cmyk.tif
- | | | |___wohnen_mit_web_RGB.jpg
- |___5 Sitzungen
- | |___2014-02-25-Betreuer



```
| | |___Notizen zum Mittagessen 25.2.2014.pdf
| |___2014-04-30-Bitfrucht
| | |___Sitzungsprotokoll Kurzfassung.txt
| |___2014-05-23-Betreuer
| | |___Notizen zum Meeting.txt
| |___mit Kunde
| | |___1. KickOffMeetingPräsentation.pptx
| | |___2. Sitzung Kunde.pptx
| | |___3. Sitzung Kunde.pptx
| | |___4. Sitzung Kunde.pptx
| | |___5. Sitzung Kunde.pptx
| | |___6. Sitzung Kunde.pptx
|___6 Zeiterfassung
| |___JIRA-Getter.command
| |___JIRA-Getter.xlsx
| |___Zeitschätzung.xlsx
|___7 Mockups
| |___Mockup_2.graffle
| |___Wireframe_Example.graffle
| |___Wireframe_Example.oo3
| |___Wireframes.eps
|___8 POI-Daten
| |___Auswahl der Kriterien.xlsx
| |___Liste aller Tags.xlsx
| |___Listen von POIs in OSM im Gebiet
| | |___Liste_aller_OSM_Tags.sh
| | |___Liste_aller_OSM_Tags.xlsx
| | |___alles.csv
| | |___amenity-audiologist.csv
| | |___amenity-cinema.csv
| | |___amenity-dentist.csv
...
| | |___speciality-yes.csv
| | |___tourism-hotel.csv
| | |___tourism-museum.csv
| |___Skript zum Installieren und Laden eines OSM-Servers.rb
|___9 Quellcode
```

8 Glossar und Abkürzungsverzeichnis

Backlog	Liste, in der alle noch offenen und ungeplanten Issues erfasst werden
CRUD	Create, Read, Update, Delete (Erstellen, Lesen, Ändern, Löschen)
CSS	Formatierungsdaten für Webseiten (Cascaded Style Sheet)
ECTS	European Credit Transfer System
Heatmap	Darstellungsform von Informationen auf einer Karte, je ausgeprägter das abgebildete Kriterium an einem Ort ist, desto intensiver ist die Farbe an diesem Punkt auf der Karte
HTML	Format von Webseitentext (Hypertext Markup Language)
HSR	Hochschule Rapperswil
Issue	Aufgabe
Jira	Verwendetes Issue Management Tool
JS	JavaScript; Code, welcher (auf Clientseite im Browser) interaktive Webseiten ermöglicht
POI	Point of Interest
OSM	OpenStreetMap
SA	Studienarbeit
Sprint	Phase eines Projekts
WF	Wohnortfinder
ZOW	Zürioberland Wohnen

9 Literatur- und Quellenverzeichnis

- Jasmine Dokumentation: <http://jasmine.github.io/1.3/introduction.html>
2. Seite „Flask“. In: Wikipedia, Die freie Enzyklopädie. Bearbeitungsstand: 15. Februar 2014, 17:33 UTC.
URL:<http://de.wikipedia.org/w/index.php?title=Flask&oldid=127604071> (Abgerufen: 25. Mai 2014, 13:46 UTC)
 3. Seite „Symfony“. In: Wikipedia, Die freie Enzyklopädie. Bearbeitungsstand: 30. April 2014, 16:38 UTC.
URL:<http://de.wikipedia.org/w/index.php?title=Symfony&oldid=129980157> (Abgerufen: 25. Mai 2014, 13:46 UTC)
 4. Masterarbeit „Weighted Multi-Point Accessibility, demonstrated on Flat Hunting“, Raphael Gfeller und Pascal Hauser, 2010

10 Liste der verwendeten Libraries

Diese liste zeigt die technisch relevanten Libraries die verwendet wurden. Nicht im Dokument oben weiter beschriebene Bibliotheken sind Abhängigkeiten von anderen.

10.1 Python

Flask	0.10.1
Flask-Admin	1.0.8
Flask-Assets	0.9
Flask-DebugToolbar	0.9.0
Flask-GoogleLogin	0.3.1
Flask-Jasmine	1.4
Flask-Login	0.2.10
Flask-SQLAlchemy	1.0
Flask-Script	0.6.7
Flask-Testing	0.4.1
Flask-WTF	0.9.5
GeoAlchemy2	0.2.3
Jinja2	2.7.2
MarkupSafe	0.19
PPyGIS	0.2
SQLAlchemy	0.9.3
Shapely	1.3.0
Tempita	0.5.2
WTForms	1.0.5
Werkzeug	0.9.4
anyjson	0.3.3
argparse	1.2.1
backports.pbkdf2	0.1
beautifulsoup4	4.3.2
billiard	3.3.0.16
blinker	1.3
celery	3.1.10
coverage	3.7.1
decorator	3.4.0
ecdsa	0.11
factory-boy	2.3.1
geojson	1.0.6
gevent	1.0
greenlet	0.4.2
gunicorn	18.0
itsdangerous	0.23

kombu	3.0.14
mock	1.0.1
nose	1.3.1
paramiko	1.13.0
pbr	0.7.0
psycpg2	2.5.2
pycrypto	2.6.1
pytz	2014.2
redis	2.9.1
requests	0.14.2
selenium	2.40.0
simplejson	3.4.0
six	1.6.1
sqlalchemy-migrate	0.9
testing.postgresql	1.0.2
webassets	0.9
wsgiref	0.1.2

10.2 Java Script

Angular		JS
Angular		RangeSlider
Angular	Leaflet	Directive
jQuery		
Bootstrap		
Google	Maps	API
Google		Overlay
Bootstrap		
Leaflet		

10.3 Postgres

hstore
postgis
plpgsql

10.4 Weitere

osm2pgsql

11 Rechtliches

11.1 Nutzungsrechte

Vereinbarung

1. Gegenstand der Vereinbarung

Mit dieser Vereinbarung werden die Rechte über die Verwendung und die Weiterentwicklung der Ergebnisse der Studienarbeit „Wohnortfinder für das Zürcher Oberland“ von Laurin Murer und Remo Liebe unter der Betreuung von Prof. Stefan Keller geregelt.

2. Urheberrecht

Die Urheberrechte stehen den Studenten zu.

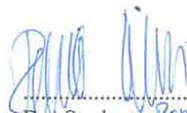
3. Verwendung

Die Ergebnisse der Arbeit dürfen von „natürli Zürioberland Wohnen“ (Projektpartner) verwendet und weiterentwickelt werden. Ergebnisse der Arbeit dürfen ebenfalls sowohl von den Studenten wie auch von der HSR (vertreten durch Prof. Stefan Keller) verwendet und weiterentwickelt werden, jedoch nicht als Ganzes und nicht als „Wohnortfinder“ ohne vorher schriftlich eine Einwilligung des Projektpartner eingeholt zu haben.

4. Beilagen:

Keine.

Rapperswil, den 21.3.14


 Der Student Remo Liebe

Rapperswil, den 21.3.14


 Der Student Laurin Murer

Rapperswil, den 21.3.2014


 Der Betreuer der Studienarbeit

Rapperswil, den 15.5.14


 Der Projektpartner Anice Tracusel

11.2 Eigenständigkeitserklärung

Wir erklären hiermit,

- dass wir die vorliegende Arbeit selber und ohne fremde Hilfe durchgeführt haben, ausser derjenigen, welche explizit in der Aufgabenstellung erwähnt ist oder mit dem Betreuer schriftlich vereinbart wurde,
- dass wir sämtliche verwendete Quellen erwähnt und gemäss gängigen wissenschaftlichen Zitierregeln korrekt angegeben haben.
- dass wir keine durch Copyright geschützten Materialien (z.B. Bilder) in dieser Arbeit in unerlaubter Weise genutzt haben.

Ort, Datum: Rapperswil, 28. Mai 2014

Name, Unterschrift:



Laurin Murer



Remo Liebi