HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL

FHO Fachhochschule Ostschweiz

OpenStreetMap
My Business

Bachelorarbeit

Abteilung Informatik

Hochschule für Technik Rapperswil

Herbstsemester 2017/2018

Autor(en):	Max Lüthi & Simon Heller
Betreuer:	Prof. Stefan Keller
Experte:	Claude Eisenhut
Gegenleser:	Prof. Olaf Zimmermann

1 Abstract

Alle Menschen kennen Google bzw. Google Maps. Es gibt aber auch Alternativen wie OpenStreetMap (OSM), das zum Teil mehr Details bietet und keine Datenschnüffelei betreibt. Im Web präsent zu sein, ist für Unternehmen wichtig und die Erfassung der Unternehmensdaten (Adresse etc.) passiert nicht von alleine.

Das Ziel der Arbeit war es, eine Webapplikation zu entwickeln, die es einfach macht, Unternehmen in OSM einzutragen und die Einträge aktuell zu halten. Dabei lag ein grosser Fokus auf der Benutzerfreundlichkeit.

Zu Beginn der Arbeit wurde festgelegt, dass man die Unternehmen als anonyme Notiz an die OpenStreetMap sendet. Zur Halbzeit des Projekts wurde einigen erfahrenen Mappern ein Zwischenstand vorgestellt. Dabei kam heraus, dass es mehr Sinn macht, dass die Webapplikation direkt Knoten in OSM erstellt. Dadurch hat der Benutzer ein direktes Feedback, da Knoten innert kurzer Zeit auf der Karte erscheinen.

Die Webapplikation basiert auf Javascript (Frontend, VueJS) und Python (Backend, Flask).

In der ersten Ansicht wird die Karte dargestellt. Durch Rechtsklick bzw. Linksklick auf ein Gebäude kann ein neues Unternehmen erfasst bzw. bearbeitet werden. Bei beiden Varianten wird man zu einem Formular weitergeleitet, mit dem Unterschied, dass dieses beim Bearbeiten bereits mit Daten gefüllt ist.

Zusätzlich wurden noch weitere Funktionalitäten eingebaut, welche die Benutzerfreundlichkeit steigern. Dazu gehören Übersetzungen Deutsch/Englisch, Fehlermeldungen, Informationen über Änderungen an Unternehmens-Daten, die man selber „beobachtet" (d.h. editiert hat), usw. Ebenfalls bietet OpenStreetMap My Business (OSMyBiz) die Möglichkeit direkt zum erstellten bzw. bearbeiteten Knoten, durch einen Link, auf die OpenStreetMap zu springen.

Webseite: http://osmybiz.osm.ch

2 Management Summary

2.1 Ausgangslage

Alle Menschen kennen Google bzw. Google Maps. Es gibt aber auch Alternativen, wie OpenStreetMap (OSM), das zum Teil mehr Details bietet und keine Datenschnüffelei betreibt. Im Web präsent zu sein, ist für Unternehmen wichtig und die Erfassung der Unternehmensdaten (Adresse etc.) passiert nicht von alleine.

Das Hauptziel dieser Arbeit war es eine Webapplikation zu bauen, die dem Benutzer ermöglicht die Daten über sein Unternehmen zu erfassen. Zusätzlich soll es möglich sein, dass der Benutzer die Daten zu seinem Unternehmen verwalten kann. Dabei liegt ein grosser Fokus auf der Benutzerfreundlichkeit.

Für diese Problemstellung gibt es bereits eine OpenSource-Lösung ”On OpenStreetMap”, kurz ”OnOSM” (1). Sie bietet ebenfalls den Unternehmen an, sich kostenlos und einfach auf die Karte zu bringen. Der Kunde kann ohne Login seine Daten erfassen und OnOSM erstellt dann eine Notiz, die auf der Karte gespeichert wird.

Alternativ ist es möglich direkt über einen Editor von OSM eine Notiz zu erstellen. Dies erfordert aber einen OSM-Login und einige Kenntnisse des jeweiligen Interfaces.

2.2 Vorgehen/Technologien

Zu Beginn der Arbeit wurde festgelegt, dass man die Unternehmen als anonyme Notiz an die OpenStreetMap sendet. OSM-Notizen werden dann von einem Mitglied der Community (sog. Mapper) abgearbeitet und definitiv eingetragen. Zur Halbzeit des Projekts wurde einigen erfahrenen Mappern ein Zwischenstand vorgestellt. Dabei kam heraus, dass es mehr Sinn macht, dass die Webapplikation direkt Knoten in OSM erstellt. Dadurch hat der Benutzer ein direktes Feedback, da das Abarbeiten der Notizen seine Zeit dauert und die Knoten innerhalb von Sekunden bzw. Minuten auf der Karte erscheinen.

Es wurden zuerst die beiden Hauptziele umgesetzt, Unternehmen erfassen und Unternehmen bearbeiten. Danach war das Ziel, den Benutzer so einfach wie möglich durch den Prozess zu leiten und möglichst viel Benutzerfreundlichkeit zu bieten.

Jede Woche gab es eine Besprechung mit dem Betreuer, um die Arbeit der letzten Woche zu analysieren und Änderungen anzubringen. Zusätzlich wurde jeweils das Vorgehen für die nächste Woche besprochen.

Die Webapplikation basiert auf Javascript (Frontend) und Python (Backend), wobei der grössere Teil das Frontend ist. Als Frontend-Framework wurde VueJS eingesetzt und im Backend das Microframework Flask.

2.3 Ergebnisse

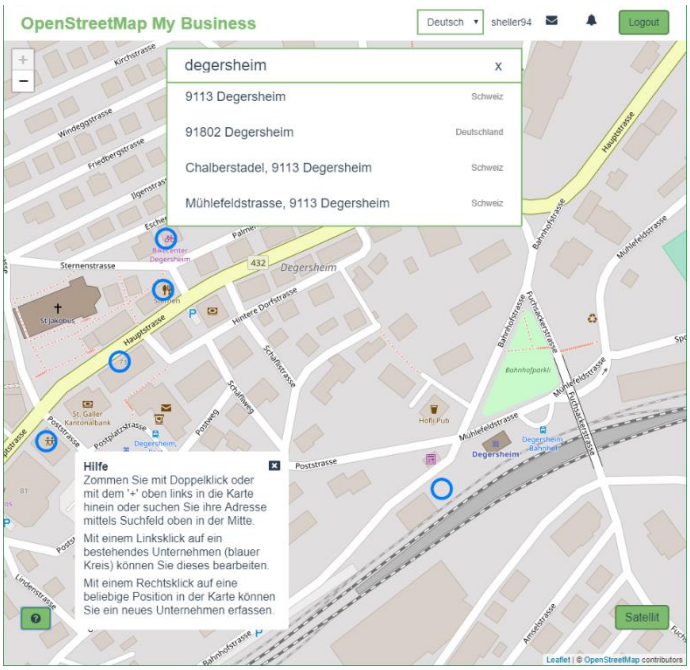


Abbildung 1 - Kartenansicht

Auf der Karte werden bereits existierende Unternehmen blau umkreist. Man kann mit einem Linksklick auf einen Kreis dieses Unternehmen bearbeiten.

Ein neues Unternehmen kann durch Rechtsklick auf ein Gebäude erstellt werden.

Die Karte kann auch auf Satellitenbilder gewechselt werden, um die Ansicht zu verbessern.
Es werden Hilfetexte (Abbildung 1, unten links) eingeblendet, die erklären was zu tun ist, wenn etwas unklar ist.

OSMyBiz bietet viele Eingabefelder, um ein Unternehmen zu beschreiben. Dies kann die Übersicht etwas beeinträchtigen, aber bietet dem Benutzer viel mehr Möglichkeiten. Er kann zum Beispiel die Adresse anpassen, wenn diese nicht genau stimmt. Um die Übersicht zu verbessern werden die Felder gruppiert (Kategorie-, Adress-, Detail- und Zusatzinformationen).

Durch die gezielte Abfrage erfasst der Benutzer eher Informationen, die er sonst weggelassen oder vergessen hätte. Da es viele Felder hat, wurde auch eine Zurücksetzen-Funktion eingebaut, die das ganze Formular wieder auf den Anfangszustand zurücksetzt. Damit aber nicht alle Daten verloren gehen, wenn man aus Versehen klickt, muss der Benutzer erst bestätigen.

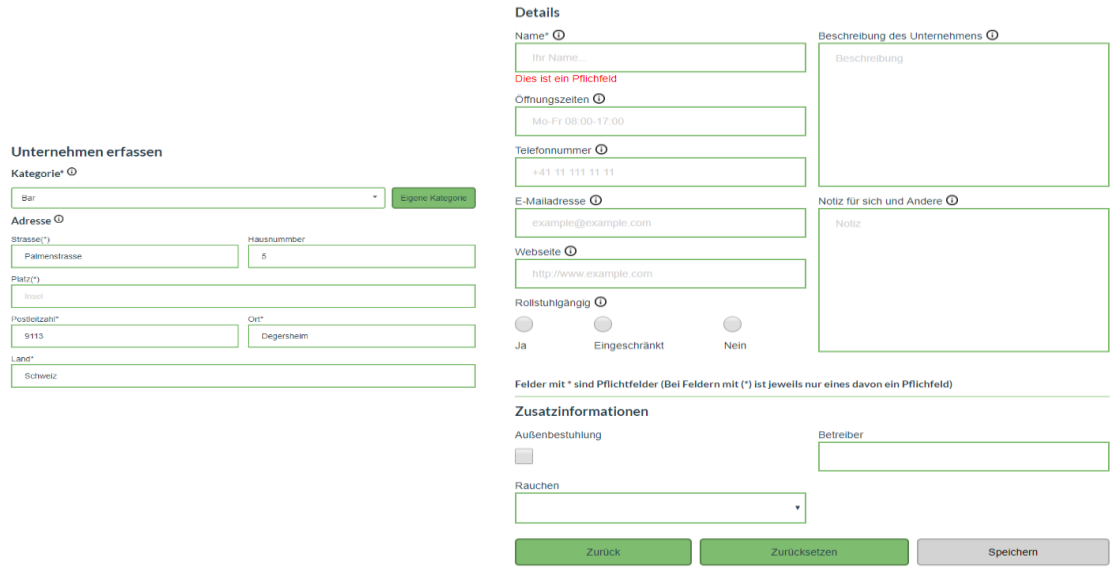


Abbildung 2 - Formular mit Kategorie-, Adress-, Detail- und Zusatzinformationen

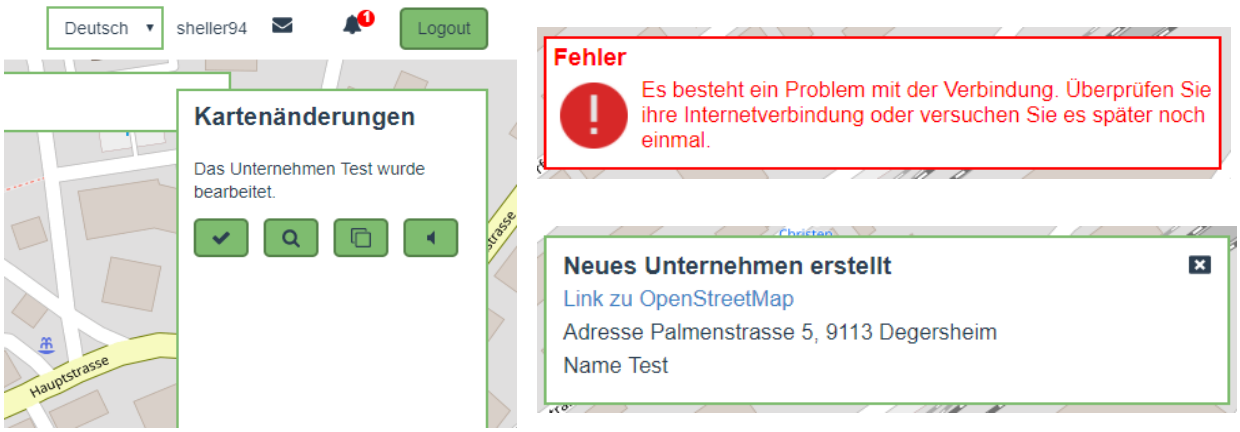


Abbildung 3 – Zusätzliche Funktionen

Zusätzliche Funktionen:

- Anzeigen von Änderungen an Unternehmen, die der Benutzer selber erstellt oder editiert hat (Abbildung 3, links)
- Sprache individuell wählen, jedoch sind vorerst nur Deutsch und Englisch verfügbar
- Fehlermeldungen, falls eine API nicht korrekt funktioniert (Abbildung 3, rechts oben)
- Erfolgsmeldungen, wenn das Erstellen oder die Änderung des Unternehmens erfolgreich war (Abbildung 3, rechts unten)

2.4 Ausblick

Für die weiterführende Entwicklung gibt es für OSMyBiz noch diverses Optimierungspotenzial.

Formular

- Beim Feld mit den Öffnungszeiten muss man selber einen Text eingeben. Besser wäre es über eine Auswahl, die dann den String automatisch erstellt.

Erweiterungen

- Die Notifikationen sind fast beliebig ausbaubar, man könnte zum Beispiel den Benutzer informieren, wenn neue Unternehmen in seiner Umgebung erstellt oder bearbeitet wurden.
- Da bisher erst Deutsch und Englisch verfügbar sind, wäre eine Optimierung zusätzliche Sprachen zu implementieren.

Zusätzliche Funktionen

- Es gibt noch offene Punkte, die aus Zeitgründen weggelassen wurden, wie den Spam-Filter oder die History (Liste mit erstellten und editierten Unternehmen). Diese einzubauen wäre sehr zu empfehlen. Vor allem die History würde noch einen klaren Vorteil für den Benutzer bringen, da er eine bessere Übersicht über seine Unternehmen hätte.
- Die Adresse könnte auch als Pluscodes (eine von Google entwickelte Art einen Ort eindeutig einem Text zuzuweisen) erfasst werden, wenn keine klare Adresse vorhanden ist.

Inhaltsverzeichnis

OpenStreetMap My Business (OSMyBiz)

Bachelorarbeit Herbstsemester 2017/18, Studiengang Informatik

Hintergrund

OpenStreetMap (OSM) ist ein Crowdsourcing-Projekt, das freie geografische Daten bereitstellt (Open Data). Aus diesen Geodaten können Karten, Routenplaner und Apps realisiert werden, die teils keine bezahlte Werbung enthalten und keine Datenschnüffelei betreiben. Nach Wikipedia ist OSM wohl das grösste gemeinschaftliche offene Werk, welches das Netz hervorgebracht hat. OSM bietet zu Google Maps eine Alternative, besonders auch, um Geschäfte zu finden. Wer selber Daten in OSM beitragen will, kann dies mit verschiedenen universellen Editoren tun. Dies stellt für Gelegenheitsnutzer jedoch eine Hürde dar, da man etwas Erfahrung braucht um mit diesen Editoren zu arbeiten.

Hier setzt nun diese Arbeit an, in dem eine Applikation entwickelt wird, die es einfach macht Geschäfte, wie z.B. Shops aber auch Restaurants oder Hotels einzutragen und die Einträge aktuell zu halten. Mit onosm.org gibt es bereits eine kleine Webapp, die jedoch sehr einfach gehalten ist. Diese erstellt aber nur eine Notiz und bietet nur begrenz Benutzerfreundlichkeit.

Ziele der Arbeit

- Im Rahmen der Arbeit soll eine Webapplikation entwickelt werden, die es Geschäftsbesitzern ermöglicht ihr Geschäft selber mit allen Daten zu erfassen, damit sie in der OpenStreetMap abgebildet werden. Im Hintergrund wird ein Node mit allen Daten erstellt.
- Zusätzlich soll abgefangen werden, wenn die Adresse nicht richtig gefunden wurde (z.B. wenn der Benutzer den Marker vor dem Gebäude statt auf dem Gebäude platziert hat) und durch Overpass der richtigen Adresse zugeordnet werden.
- Der Besitzer sollte ebenfalls in der Lage sein, bestehende Informationen zu seinem Geschäft zu verwalten bzw. zu aktualisieren. Im Hintergrund wird dazu eine Notiz mit allen Daten erstellt und in der OpenStreetMap abgebildet.
- Es soll die Möglichkeit geben eine History mit den erfassten und bearbeiteten Geschäften anzuzeigen.
- Es soll auf Spam geprüft werden, damit nur seriöse Geschäfte erfasst werden und ein Schutz gegen Bots soll eingerichtet werden. (Best Effort)
- Optional (bei genügend Zeit bei der Entwicklung) soll dem Erfasser mitgeteilt werden, sobald sein Geschäft in der OpenStreetMap verfügbar ist. Dafür ist eine E-Mail-Adresse vom Benutzer notwendig.

Deliverables

- Funktionierende Webapplikation (Frontend & Backend)
- Dokumentation: Gedruckter Bericht (gebunden) mit USB-Stick für Betreuer (mit sämtlichen Dokumenten, Quellen und Daten von Software und Bericht). Dazu kommt die Abgabe für den Studiengang gemäss sep. Instruktionen.

Vorgaben/Rahmenbedingungen

- Frontend-Framework evaluieren: React oder VueJS
- REST-/Microservices-Library evaluieren: Django REST Framework oder Flask/Connexion (Swagger integriert)

- Backend: Python, PostgreSQL/PostGIS, OSM API (bitte ggf. verbessern und contributen), Linux
- Sonstiges: Docker, Gitlab, Continuous Testing & Integration, IDE ist den SW-Entwicklern überlassen, wie auch ob sie auf Windows, Mac oder Linux entwickeln
- Nichtfunktionale Anforderungen: Responsive GUI (nicht zwingend Mobile-First), Performance (Best Effort), Docker-Container.
- Die Software soll Open Source sein, die Softwarelizenz ist „MIT“.
- Die SW-Engineering-Methode und Meilensteine werden mit dem Betreuer vereinbart.
- Software-Dokumentation sind Englisch (inkl. Installation, keine Benutzerdokumentation, höchstens eine Online-Kurzhilfe).
- Der Source Code, die Code-Kommentare und die Versionsverwaltung sind ebenfalls in Englisch.
- Die Software-Benutzerschnittstelle ist Deutsch.
- Die Projekt-Dokumentation und -Präsentation sind auf Deutsch.
- Die Nutzungsrechte an der Arbeit bleiben bei den Autoren, gehen aber auch an die HSR und den Betreuer über.
- Kein Video da es eine offene Webapplikation ist.
- Ansonsten gelten die Rahmenbedingungen, Vorgaben und Termine des Studiengangs Informatik bzw. der HSR.

Inhalt der Dokumentation

- Die fertige Arbeit muss folgende Inhalte haben:
  1. Abstract, Management Summary, Aufgabenstellung
  2. Technischer Bericht
  3. Projektdokumentation
  4. Anhänge (Literaturverzeichnis, Glossar, Dateispeicher-Inhalt)
- Die Abgabe ist so zu gliedern, dass die obigen Inhalte klar erkenntlich und auffindbar sind.
- Zitate sind zu kennzeichnen und die Quelle ist anzugeben.
- Verwendete Dokumente und Literatur sind in einem Literaturverzeichnis aufzuführen.
- Dokumentation des Projektverlaufes, Planung etc.
- Weitere Dokumente (z.B. Kurzbeschreibung, Poster) gemäss www.hsr.ch und Absprache mit dem Betreuer.

Bewertungsschema

Es gelten die üblichen Regelungen zum Ablauf und zur Bewertung der Arbeit (6 Aspekte) des Studiengangs Informatik der HSR jedoch mit besonderem Gewicht auf moderne Softwareentwicklung (Tests, Continuous Integration, einfach installierbar, funktionsfähig).

Beteiligte

Diplomanden

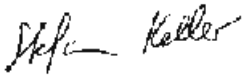
Max Lüthi und Simon Heller

Industriepartner

-

Betreuung

Verantwortlicher Dozent an der HSR ist Prof. Stefan Keller (sfkeller@hsr.ch), Geometa Lab am IFS der HSR, Nicola Jordan (Mitarbeiter am IFS der HSR).



4 Teil 1 - Technischer Bericht

4.1 Einführung

4.1.1 Problemstellung

Alle Menschen kennen Google bzw. Google Maps. Es gibt aber auch Alternativen, wie OpenStreetMap (OSM), das zum Teil mehr Details bietet und keine Datenschnüffelei betreibt. Im Web präsent zu sein, ist für Unternehmen wichtig und die Erfassung der Unternehmensdaten (Adresse etc.) passiert nicht von alleine.

Deshalb gibt es Tools, die den Unternehmensbesitzer dabei unterstützen sein Unternehmen auf der OpenStreetMap zu erfassen. Wie im Kapitel 4.3 Stand der Technik beschrieben, gibt es mehrere Möglichkeiten dies zu tun.

4.1.2 Ziele

Das Hauptziel dieser Arbeit war es eine Webapplikation zu bauen, die dem Benutzer ermöglicht die Daten über sein Unternehmen zu erfassen. Im Gegensatz zu OnOSM soll es möglich sein, dass der Benutzer die Daten zu seinem Unternehmen verwalten kann.

Um dem Benutzer ein positives Erlebnis zu ermöglichen, wurden zusätzliche Features eingebaut. Ein Fokus liegt daher auch auf der Benutzerfreundlichkeit.

4.1.3 Aufbau

Im Rahmen dieser Arbeit wurde eine Webapplikation entwickelt, die die obengenannten Ziele umsetzt.

Dieses Dokument besteht aus zwei Hauptteilen. Dem technischen Bericht und der Projektdokumentation.

Der technische Bericht behandelt die Einführung in den Themenbereich und die Arbeit, eine Grobübersicht, den Stand der Technik, benutzte Technologien und Werkzeuge und am Schluss die Resultate der Arbeit und eine Schlussfolgerung mit Ausblick auf Weiterentwicklungen.

Die Projektdokumentation beinhaltet die Spezifikation der Anforderungen (sowohl funktional als auch nichtfunktional), Architekturentscheide, Dokumentation des Source Codes, eine ausführlichere Beschreibung der Weiterentwicklungen, Qualitätsmanagement, Projektmanagement (Organisation, Zeitplan, Risiken, etc.) und zum Schluss noch das Projekt Monitoring mit Soll-Ist-Zeitvergleich und Aufgabenteilung.

4.2 Big Picture

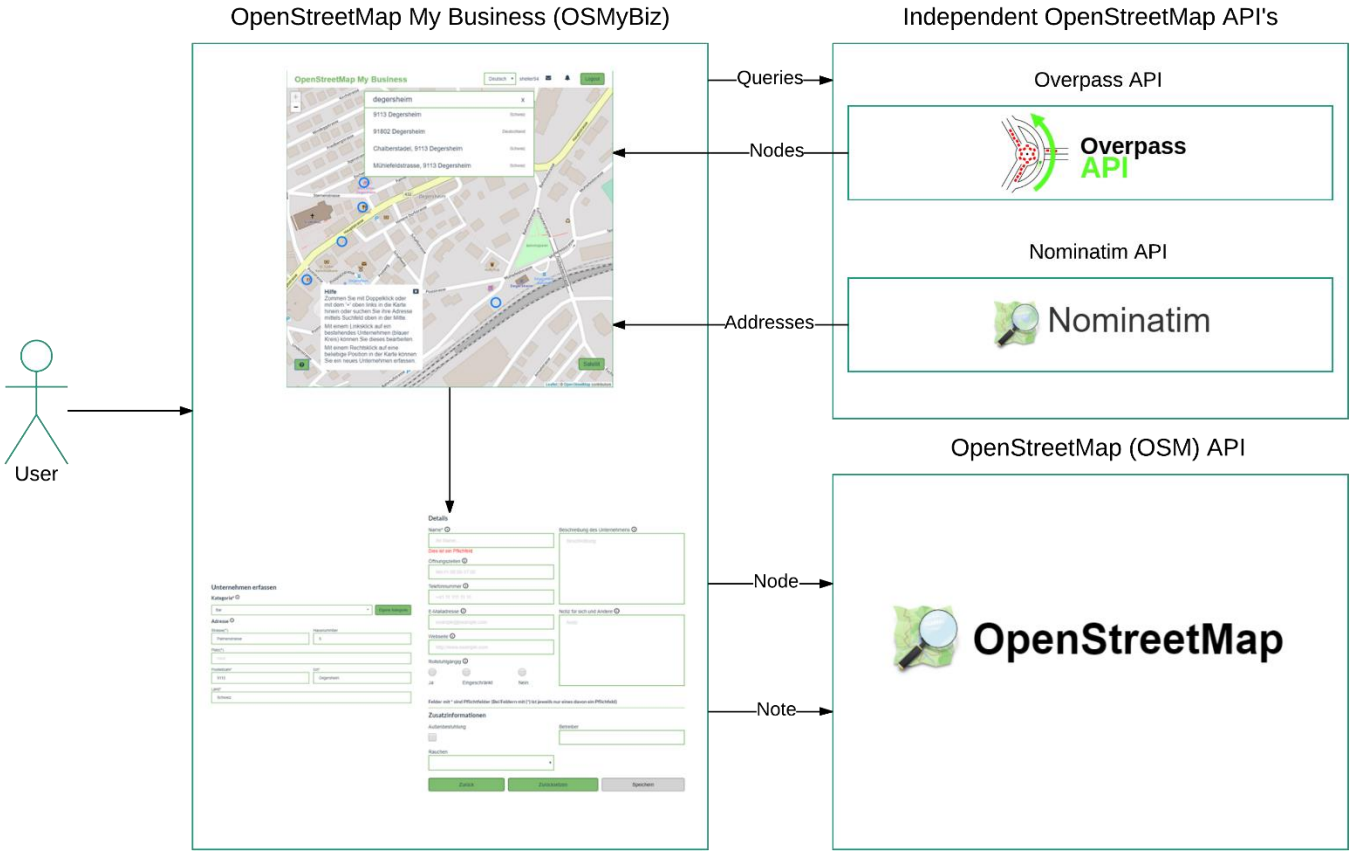


Abbildung 4 - Big Picture

Der Benutzer kommt auf die Webseite und sieht eine Karte. Über ein Suchfeld kann er eine Adresse eingeben oder mit der Zoomfunktion selber den gewünschten Ort suchen. Sucht er mittels Adresse, wird diese über Nominatim API (siehe 5.4.1.4 Nominatim API) in Koordinaten umgewandelt und der Kartenabschnitt für den Benutzer wird angepasst.

Auf der untersten Zoomstufe werden alle Unternehmen, die in einer Liste mit Kategorien (siehe 5.4.5 OSM-Tags / Kategorien) sind, mit blauen Kreisen hervorgehoben. Diese Unternehmen werden über die Overpass API (siehe 5.4.1.5 Overpass API) und einer Abfrage, die das Raster wiederspiegelt, in die Anwendung gebracht.

Der Benutzer hat zwei Möglichkeiten:

1. Ein Unternehmen bearbeiten (mit Linksklick auf einen blauen Kreis)

2. Ein neues Unternehmen erfassen (mit Rechtsklick auf ein Gebäude)

Egal welche Option er wählt, wird er zum Formular weitergeleitet. Beim Bearbeiten eines Unternehmens werden jedoch die Felder bereits mit den vorhandenen Informationen gefüllt.

Beim Bearbeiten wird nach dem Speichern der Daten eine Notiz erstellt und an OSM übermittelt. Diese Notiz muss von einem Mapper abgearbeitet werden.

Beim Erfassen wird jedoch direkt ein Knoten auf der OpenStreetMap erstellt, damit der Benutzer ein direktes Feedback erhält.

4.3 Stand der Technik

Für einen Teil der Problemstellung gibt es bereits eine OpenSource-Lösung ”On OpenStreetMap”, kurz ”OnOSM” (1). Sie ist über onosm.org erreichbar und bietet ebenfalls Unternehmen an, sich kostenlos und einfach auf der OpenStreetMap zu erfassen. Der Kunde kann ohne Login und Tracking seine Daten erfassen und OnOSM erstellt dann eine Notiz, die auf der Karte gespeichert wird. Die Unterschiede von OnOSM und OSMyBiz werden bei den Schlussfolgerungen beschrieben (siehe 4.7.2 Unterschiede/Nutzen)

Alternativ ist es möglich direkt über einen Editor von OSM eine Notiz zu erstellen. Dies erfordert einen OSM-Login und einige Kenntnisse des jeweiligen Interfaces. Ebenfalls bieten diese Editoren viel mehr Funktionen, als die meisten Anwender wirklich benötigen.

Es werden von OSM-Mappern in erster Linie zwei Editoren eingesetzt (2).

Zum einen ist das der iD-Editor (3), der in der Webseite von OSM (osm.org (4)) integriert ist. Er ist ideal, um schnell im Browser etwas zu bearbeiten.

Der andere Editor ist eine Java Desktop Applikation namens JOSM (5). Er erlaubt das Bearbeiten von Daten im Offline-Modus.

Neben diesen zwei «Grossen» gibt es noch viele andere kleinere Editoren (2), unter anderem auch in Android, iOS und Windows Phone.

Eine eher unkonventionelle Art ist es, die Erfassung direkt über ein Offline-Navi (z.B. Maps.me (6)) durchzuführen.

4.4 Google My Business (7)

Für Google Maps gibt es bereits eine Webapplikation, die Unternehmen die Möglichkeit bietet ihre Daten zu erfassen. Man muss aber nicht nur seine Benutzerdaten angeben, sondern auch noch seine Identität mittels eines Anrufs oder eines Briefs bestätigen.

Viele Benutzer nutzen Google für ihre Suche, da sie nichts anderes kennen oder damit zufrieden sind. Durch die Erfassung in Google My Business erhalten die Benutzer erleichterten Zugang zum Unternehmen.

Es gibt einige wesentliche Unterschiede zwischen OSM und Google (8). Bei Google sind die Daten alle mit Copyright (Von verschiedenen Organisationen, wie z.B. Ordnance Survey) geschützt. Google verteilt selber nur die Lizenzen für diese Daten. Wenn man die Daten von Google Maps nutzt, erstellt man eine «abgeleitete Arbeit». Die Daten behalten die urheberrechtlichen Bedingungen des Originals bei. In der Praxis bedeutet dies, dass die Daten den Lizenzgebühren und vertraglichen Beschränkungen der Kartenanbieter unterliegen. Genau das versucht OSM zu vermeiden.

Um möglichst viele Benutzer zu erreichen muss aber ein Unternehmen auf beiden Systemen, Google und OSM, angezeigt werden.

4.5 Technologien

Im folgenden Abschnitt werden die verwendeten Technologien im Detail beschrieben und erklärt, warum genau diese gewählt wurden.

4.5.1 Backend

Für die Implementierung des Backends war durch die Aufgabenstellung gegeben, dass Python (9) verwendet werden soll, die Wahl des Web-Server Frameworks war jedoch frei. Es wurden die beiden Web-Server Frameworks Django und Flask evaluiert. Dazu wurde mit beiden Frameworks ein minimales Projekt erstellt.

Bei Django (10) handelt es sich um ein, im Vergleich mit Flask, sehr grosses und vollständiges Web-Server Framework. Für alle typischen Problemstellungen im Rahmen eines Webservers gibt es eine Lösungsstrategie und diese ist direkt im Framework integriert. Für Python Anfänger kann dies schnell zu Verwirrung führen, da das Framework sehr viele Features hat, die in vielen Fällen gar nicht gebraucht werden.

Im Gegensatz dazu ist Flask (11) ein erweiterbares Microframework, sprich der Kern des Frameworks enthält nur das absolute Minimum, um einen Webserver zu implementieren. Alle weiteren typischen Webfeatures sind als Plugins und Extensions verfügbar. Für einen Schnellstart ist Flask ideal geeignet, da es zu allen üblichen Use-Cases sehr konkrete Beispiele, entweder direkt in der Flask Dokumentation oder in der Dokumentation der jeweiligen Extension, gibt.

Da Flask einen einfacheren Einstieg bietet und man nur das einbaut, was man wirklich braucht, wurde dieses Web-Server Framework gewählt.

4.5.2 Frontend

4.5.2.1 Framework/UI-Library (12) (13) (14)

Für das Frontend gibt es mit Javascript sehr viele Frameworks. Aufgrund von Vorkenntnissen (Entwickler und Weiterentwickler) war bereits klar, dass es VueJS (15), ReactJS (16) oder Angular 2 (17) werden würde. Angular 2 kam nicht in Frage, da dieses Framework für den geplanten Umfang von OSMyBiz zu viel Zusatzaufwand bedeutet hätte. Die Basisstruktur von Angular 2 bringt viel Boilerplate Code mit sich.

Also blieben noch ReactJS und VueJS. Diese beiden haben viele Gemeinsamkeiten, da sie beide reine UI-Frameworks/Libraries sind. Sie benutzen beide einen virtuellen DOM und zusammensetzbare View-Komponenten. Routing und State Management sind bei beiden vom eigentlichen Framework getrennt. Ebenfalls bieten beide eine CLI an, um das Projekt optimal aufzubauen.

Ein grosser und relevanter Unterschied liegt in der Lernkurve des Frameworks. ReactJS hat eine eher steile Lernkurve und die Dokumentation ist etwas schwieriger zu lesen. VueJS dagegen bietet eine gute Dokumentation und ist durch die etwas niedrigere Komplexität auch einfacher zu lernen.

Das Templating funktioniert ebenfalls bei beiden etwas anders. Bei VueJS wird noch mit normalem HTML gearbeitet, wobei für ReactJS mit JSX, einer Javascript Extension mit XML-Syntax, gearbeitet wird. Optional unterstützt VueJS auch JSX, jedoch ist es nicht der Standardansatz.

Obwohl bereits ein wenig Erfahrung mit ReactJS vorhanden war, wurde aus obengenannten Gründen und weil die Weiterentwickler am meisten Erfahrung damit haben, das Framework VueJS ausgewählt. Ein zusätzlicher Grund war die Gelegenheit ein neues Framework auszuprobieren und dadurch neue Erfahrungen zu sammeln.

4.5.2.2 Typescript

Auf die Verwendung von Typescript (18) wurde bewusst verzichtet. VueJS enthält im Standard-Setup bereits EcmaScript 6 (19) und diverse Babel-Erweiterungen, die Typescript mehr zu einem Hindernis gemacht hätten, als dass es geholfen hätte. EcmaScript 6 enthält zudem bereits die meisten Vorteile von Typescript.

4.5.3 Datenbank

Die Datenbank war durch den Betreuer, Professor Stefan Keller, vorgegeben. Es wurde mit PostgreSQL (20) gearbeitet, was im Bereich der Kartenverarbeitung viele Vorteile bringt, da man mit PostGIS arbeiten kann. In diesem Projekt wurde PostGIS nicht verwendet, kann aber einfach eingebaut werden, falls es in Zukunft notwendig wird.

4.5.4 Testing

4.5.4.1 Python

Für die Tests in Flask wäre pytest (21) vorgesehen gewesen. Das Backend ist aber kleiner ausgefallen als geplant und aus Zeitgründen wurden die Tests weggelassen.

4.5.4.2 Javascript

Wir verwenden Mocha (22) für die Unittests und Karma (23) als Testrunner. Dieses Setup wird von VueJS unterstützt und mit dem Standard-Setup bereitgestellt.

Mocha ist ein funktionsreiches JavaScript-Testframework, welches das asynchrone Testen vereinfacht. Mocha-Tests werden seriell ausgeführt, wodurch ein flexibles und genaues Reporting ermöglicht wird. Karma führt die Mocha-Tests durch.

4.5.5 Werkzeuge & Tools

Werkzeug/Tool	Erklärung
Git	Versionskontrolle
GitLab	Hosten des Repositories, Continous Integration, Zeitmanagement
PyCharm	Integrierte Entwicklungsumgebung (IDE) für Python von JetBrains
WebStorm	Integrierte Entwicklungsumgebung (IDE) für Javascript von JetBrains
Google Chrome*	Browser zum Testen und Debuggen von Client Code
Firefox	Browser zum Testen von Client Code
Docker	Open-Source-Software, die dazu verwendet werden kann, Anwendungen mit Hilfe von Betriebssystem-Virtualisierung in Containern zu isolieren.
Swagger	Definitionsformat um RESTful APIs zu beschreiben
Lucidchart	Web-Tool zum Erstellen von Diagrammen
Moqups.com	Online-Tool zum Erstellen von GUI-Mockups

Tabelle 1 - Werkzeuge & Tools

* Google Chrome wurde als Hauptbrowser für die Entwicklung verwendet, da es ein Chrome-Plugin gibt für die VueJS-DevTools, welche das Debugging erleichtern.

4.5.6 Javascript Libraries

Werkzeug/Tool	Erklärung
Leaflet	Open Source Javascript Library für interaktive Karten
Axios	Promise-basierter HTTP Client für den Browser und Node.js
Lodash	Utility Javascript Library
Moment	Datum Javascript Library für das Parsen, Validieren und Formatieren von Daten
Vee-Validate	Einfaches VueJS Input-Validierungsplugin
Vue-Awesome	Font Awesome (Icon Library) für VueJS
Vue-Search-Select	Durchsuchbares Formular-Dropdown
Vue-Translate-Plugin	Übersetzungsplugin
OSM-Auth	Library für Javascript OAuth mit OSM

Tabelle 2 – Javascript Libraries

4.5.7 Python Libraries

Werkzeug/Tool	Erklärung
Flask	Python Micro Web Framework
Flask SQL Alchemy	Flask Plugin zur Einbindung von SQL Alchemy
SQL Alchemy	ORM für Python
Flask Migrate	Flask Plugin für Datenbank Migrationen mit SQL Alchemy
Flask Script	Flask Plugin für externe Scripts
psycopg2	PostgreSQL-Datenbankadapter für die Programmiersprache Python

Tabelle 3 - Python Libraries

4.6 Data Flow

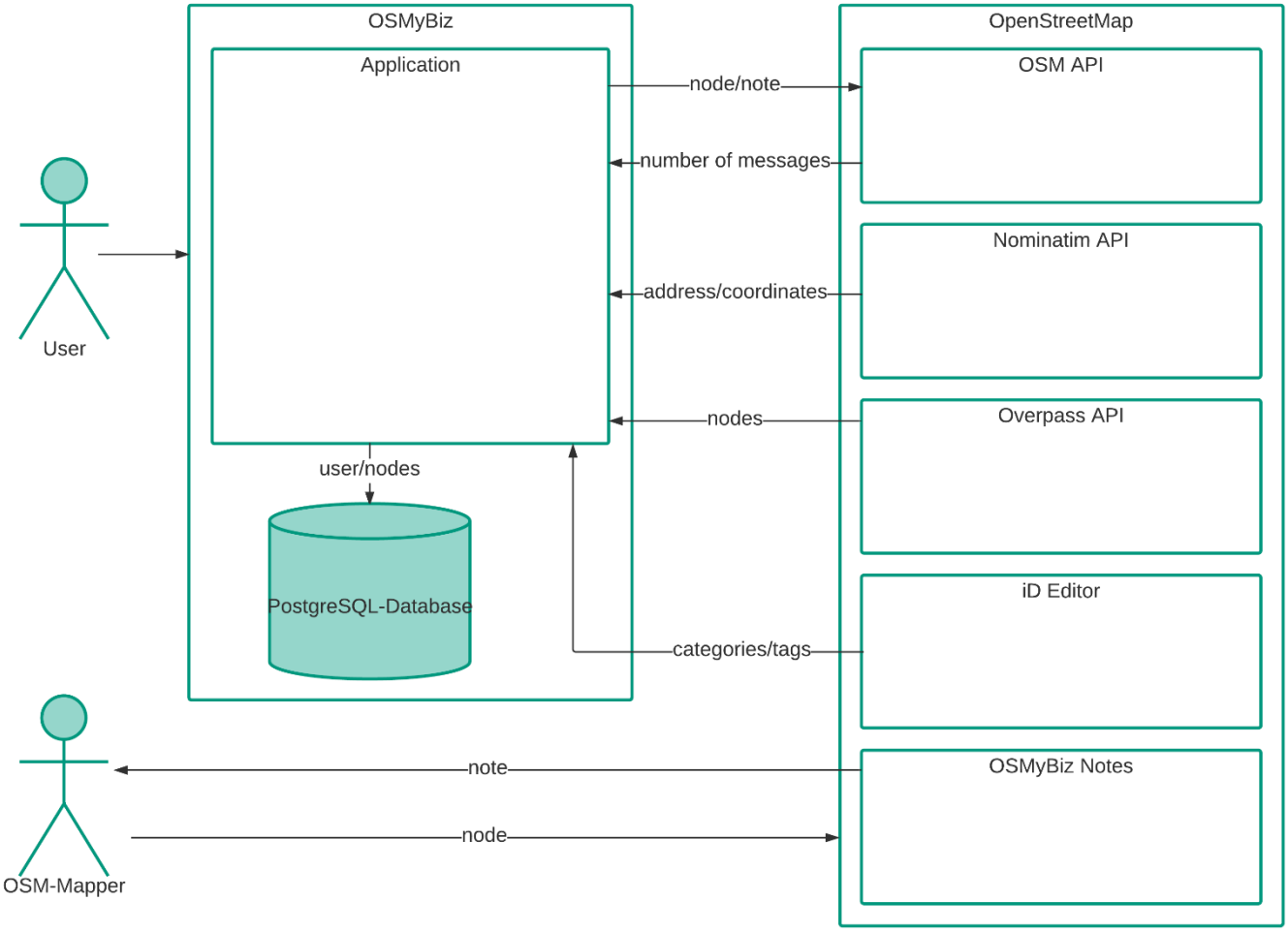


Abbildung 5 – Datenfluss-Diagramm

OSMyBiz arbeitet mit vielen verschiedenen externen APIs/Komponenten. Die meisten davon helfen, die Benutzerfreundlichkeit zu steigern.

Die erstellten Unternehmen oder die Änderungen werden über die OSM API (siehe 5.4.1.3) auf die OpenStreetMap geladen. Wiederum über die OSM API werden Nachrichten geladen, die der Benutzer erhalten hat.

Über die Nominatim API (siehe 5.4.1.4 Nominatim API) werden die Adressen der Objekte mittels der Koordinaten geladen oder aber umgekehrt die Koordinaten mittels der Adresse. Diese wird dem Benutzer angezeigt und nachher auch mit dem Objekt auf die OpenStreetMap geladen.

Mit der Overpass API (siehe 5.4.1.5 Overpass API) werden mit einer Umkreissuche alle Objekte zurückgegeben, die gewisse Anforderungen erfüllen. Im Fall von OSMyBiz werden die bereits existierenden Unternehmen blau umkreist.

Für das Formular der Unternehmensdaten werden die Kategorien des iD-Editor (3) gefiltert und für die Zwecke von OSMyBiz angepasst (siehe 5.4.5 OSM-Tags / Kategorien).

OSMyBiz Notes (24) wurde an einem Hackathon entwickelt und sammelt alle Notizen, die von OSMyBiz erstellt wurden. Damit können OSM-Mapper diese systematischer abarbeiten und die Änderungen eintragen.

4.7 Resultate & Schlussfolgerungen

4.7.1 Zielerreichung

Zu Beginn der Arbeit wurde festgelegt, dass man die Unternehmen als anonyme Notiz an die OpenStreetMap sendet. OSM-Notizen werden dann von einem Mitglied der Community (sog. Mapper) abgearbeitet und definitiv eingetragen. Zur Halbzeit des Projekts wurde einigen erfahrenen Mappern ein Zwischenstand vorgestellt. Dabei stellte sich heraus, dass es mehr Sinn macht direkt einen Knoten in OSM zu erstellen. Dadurch hat der Benutzer ein direktes Feedback, da das Abarbeiten der Notizen seine Zeit dauert und die Knoten innerhalb von Sekunden bzw. Minuten auf der Karte erscheinen.

Als Folge davon musste der Kern der Applikation etwas umstrukturiert werden. Das Erfassen eines Unternehmens an sich war aber keine grosse Herausforderung. Auch das Bearbeiten, durch das Erstellen einer Notiz, ist mit den APIs von OSM sehr einfach. Bereits mit diesen zwei Funktionen ist ein Grossteil der Ziele bereits erreicht. Nun wurde daran gearbeitet, den Benutzer so einfach wie möglich durch den Prozess zu leiten (siehe 4.7.2 Unterschiede/Nutzen).

Es war ein Feedback geplant, damit der Benutzer eine Rückmeldung erhält, wenn seine Notiz abgearbeitet wurde. Nun wird direkt ein Knoten erstellt und der Benutzer wird durch OSM über allfällige Änderungen informiert. Diese Funktion muss deshalb nicht mehr implementiert werden.

Die optionale Anforderung, Spam- und Bot-Filter, wurde aufgrund Zeitmangels weggelassen. Der Filter ist dafür gedacht, dass nur seriöse Unternehmen sich erfassen können.

Was ebenfalls aus Zeitgründen nicht mehr eingebaut werden konnte, ist eine History, in Form einer formatierten Liste, für die erfassten und bearbeiteten Unternehmen.

4.7.2 Unterschiede/Nutzen

Die Applikation bietet im Vergleich zu OnOSM (1), mehr zusätzliche Funktionen, die die Benutzerfreundlichkeit steigern sollen. Die Funktionen der Karte und die Felder des Formulars wurden erweitert, jedoch muss man sich bei OSMyBiz mit einem Login anmelden.

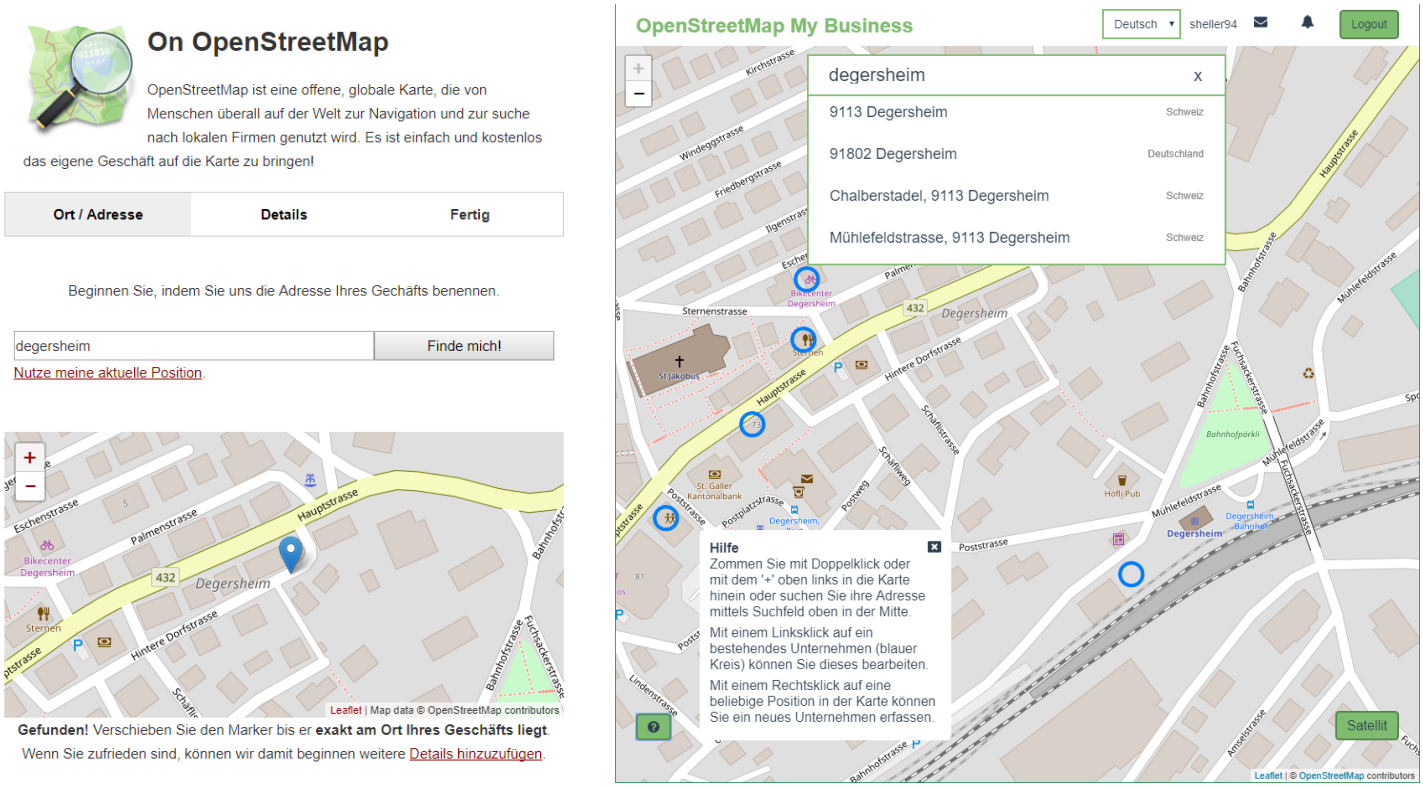


Abbildung 6 - Vergleich OnOSM-OSMyBiz Landing

OSMyBiz wirkt durch das Styling etwas moderner und übersichtlicher. Der Benutzer wird durch interaktive Hilfetexte (unten links) durch die Erfassung geführt.

Auf der Karte selber hat OSMyBiz mehr zu bieten, da bereits existierende Unternehmen blau umkreist werden. Durch Linksklick auf den Kreis kann dieses Unternehmen bearbeitet werden. Ein neues Unternehmen kann durch Rechtsklick auf das Gebäude erstellt werden. Die Karte kann auch auf Satellitenbilder gewechselt werden, um die Ansicht zu verbessern.

Beim Suchfeld werden Vorschläge gemacht, die man auswählen kann. Bei OnOSM wird man direkt zur ersten gefundenen Adresse geleitet.

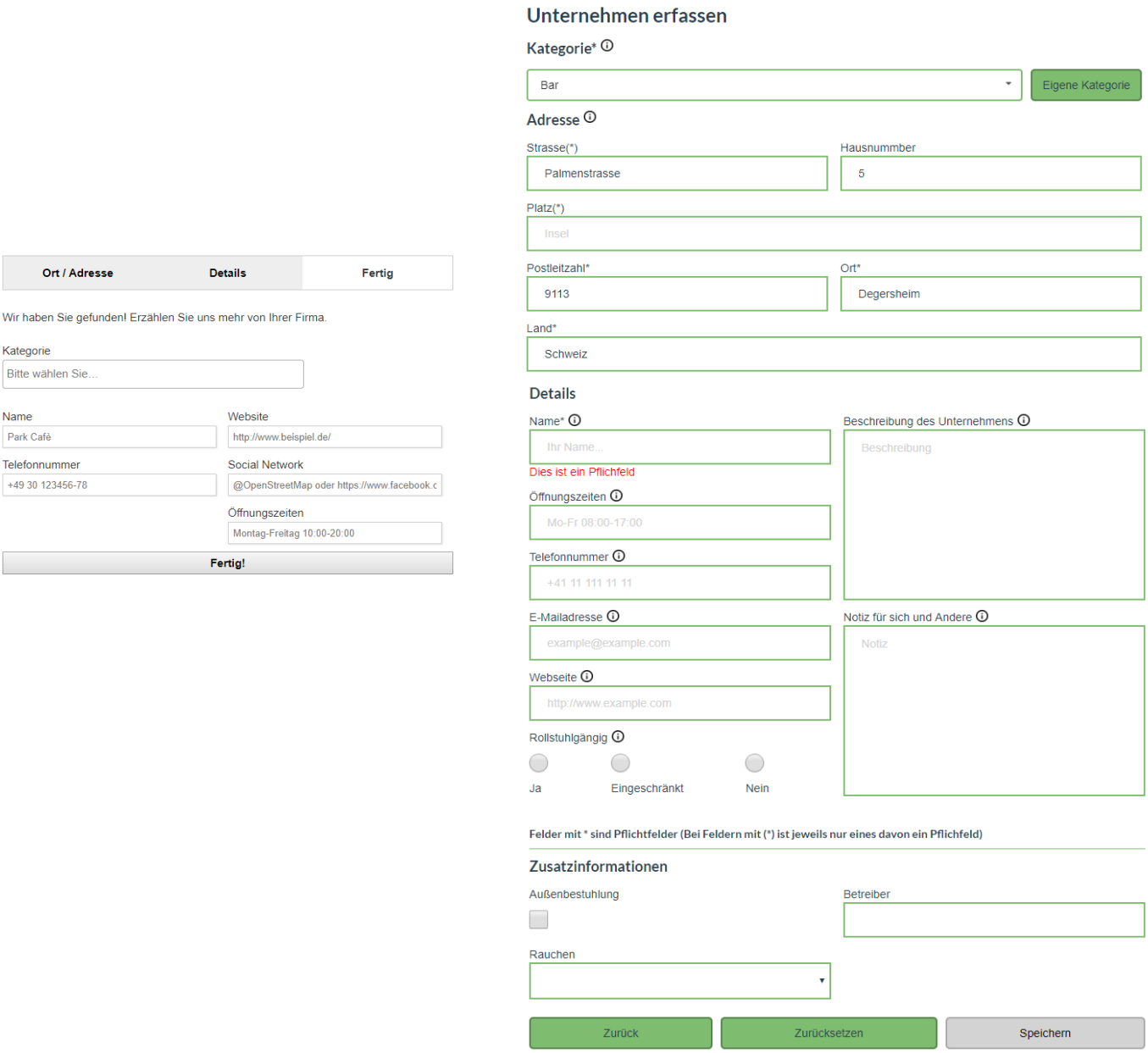


Abbildung 7 - Vergleich OnOSM-OSMyBiz Formular

Wie man auf den ersten Blick sieht, bietet OSMyBiz viel mehr Felder, um sein Unternehmen zu beschreiben. Dies kann die Übersicht etwas beeinträchtigen, aber bietet dem Benutzer viel mehr Möglichkeiten. Er kann zum Beispiel die Adresse anpassen, wenn diese nicht genau stimmt. Um die Übersicht zu verbessern werden die Felder gruppiert.

Durch die gezielte Abfrage erfasst der Benutzer eher Informationen, die er sonst weggelassen oder vergessen hätte. Bei OnOSM ist er auf die sechs Felder begrenzt, auch wenn er mehr erfassen möchte.

Ebenfalls werden Zusatzinformationen aus der Kategorie ausgelesen und angezeigt.

Da es viele Felder hat wurde auch eine Zurücksetzen-Funktion eingebaut, die das ganze Formular wieder auf den Anfangszustand zurücksetzt. Damit aber nicht alle Daten verloren gehen, wenn man aus Versehen klickt, muss der Benutzer erst bestätigen.

Es wurden aber auch zusätzliche Funktionen, die OnOSM nicht anbietet, eingebaut. Zum Beispiel das Anzeigen von Änderungen an Unternehmen, die der Benutzer selber erstellt oder editiert hat.

Abbildung 8 - Zusätzliche Funktion - Kartenänderungen

Andere zusätzliche Funktionen:

- Abrufen von OSM-Nachrichten, die der Benutzer für seinen OSM-Login erhalten hat, über das Umschlag-Icon
- Sprache individuell wählen (vorerst nur Deutsch und Englisch verfügbar)
- Fehlermeldungen, falls eine API nicht korrekt funktioniert

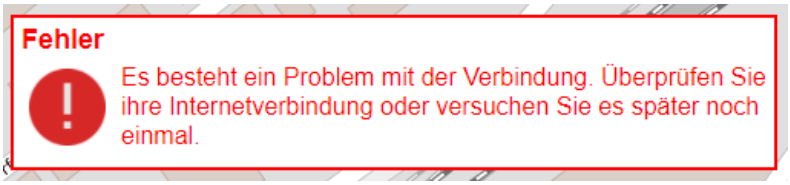


Abbildung 9 - Zusätzliche Funktion - Fehlermeldung

- Erfolgsmeldungen, wenn das Erstellen oder die Änderung des Unternehmens erfolgreich war

Abbildung 10 - Zusätzliche Funktion - Erfolgsmeldung

4.7.3 Beurteilung der Resultate

Zusammenfassend kann man sagen, dass die neue Applikation einen klaren Mehrwert gegenüber OnOSM bietet und schon vieles richtig macht. Wenn man bereit ist, sich mit einem OSM-Benutzerkonto bei OSMyBiz einzuloggen, hat man viele Möglichkeiten, die einem das Erfassen von Unternehmen erleichtern.

Die Hauptfunktionen (Unternehmen erfassen und bearbeiten) wurden beide umgesetzt und funktionieren. Auch einige zusätzliche Funktionen konnten umgesetzt werden, jedoch hat die Anwendung auch noch Optimierungspotenzial.

Es wird an vielen Orten eine Rückmeldung gegeben, falls und wieso etwas nicht funktioniert oder wenn etwas erfolgreich war. Der Benutzer weiss somit immer woran er ist und was er vielleicht anders machen kann.

4.7.4 Ausblick

Für die weiterführende Entwicklung gibt es für OSMyBiz noch diverses Optimierungspotenzial.

Formular

- Beim Feld mit den Öffnungszeiten muss man selber einen Text eingeben. Besser wäre es über eine Auswahl, die dann den String automatisch erstellt.

Erweiterungen

- Die Notifikationen sind fast beliebig ausbaubar, man könnte zum Beispiel den Benutzer informieren, wenn neue Unternehmen in seiner Umgebung erstellt oder bearbeitet wurden.
- Da bisher erst Deutsch und Englisch verfügbar sind, wäre eine Optimierungsmöglichkeit zusätzliche Sprachen zu implementieren.

Zusätzliche Funktionen

- Es gibt noch offene Punkte, die aus Zeitgründen weggelassen wurden, wie den Spam-Filter oder die History (Liste mit erstellten und editierten Unternehmen). Diese einzubauen wäre sehr zu empfehlen. Vor allem die History würde noch einen klaren Vorteil für den Benutzer bringen, da er eine bessere Übersicht über seine Unternehmen hätte.
- Die Adresse könnte auch als Pluscodes (eine von Google entwickelte Art einen Ort eindeutig einem Text zuzuweisen) erfasst werden, wenn keine klare Adresse vorhanden ist.

Im Kapitel 6.5 Weiterentwicklung werden die Optimierungsmöglichkeiten noch ausführlicher beschrieben.

5 Teil 2 – Projektdokumentation

5.1 Anforderungsspezifikation

5.1.1 Use Case Diagramm

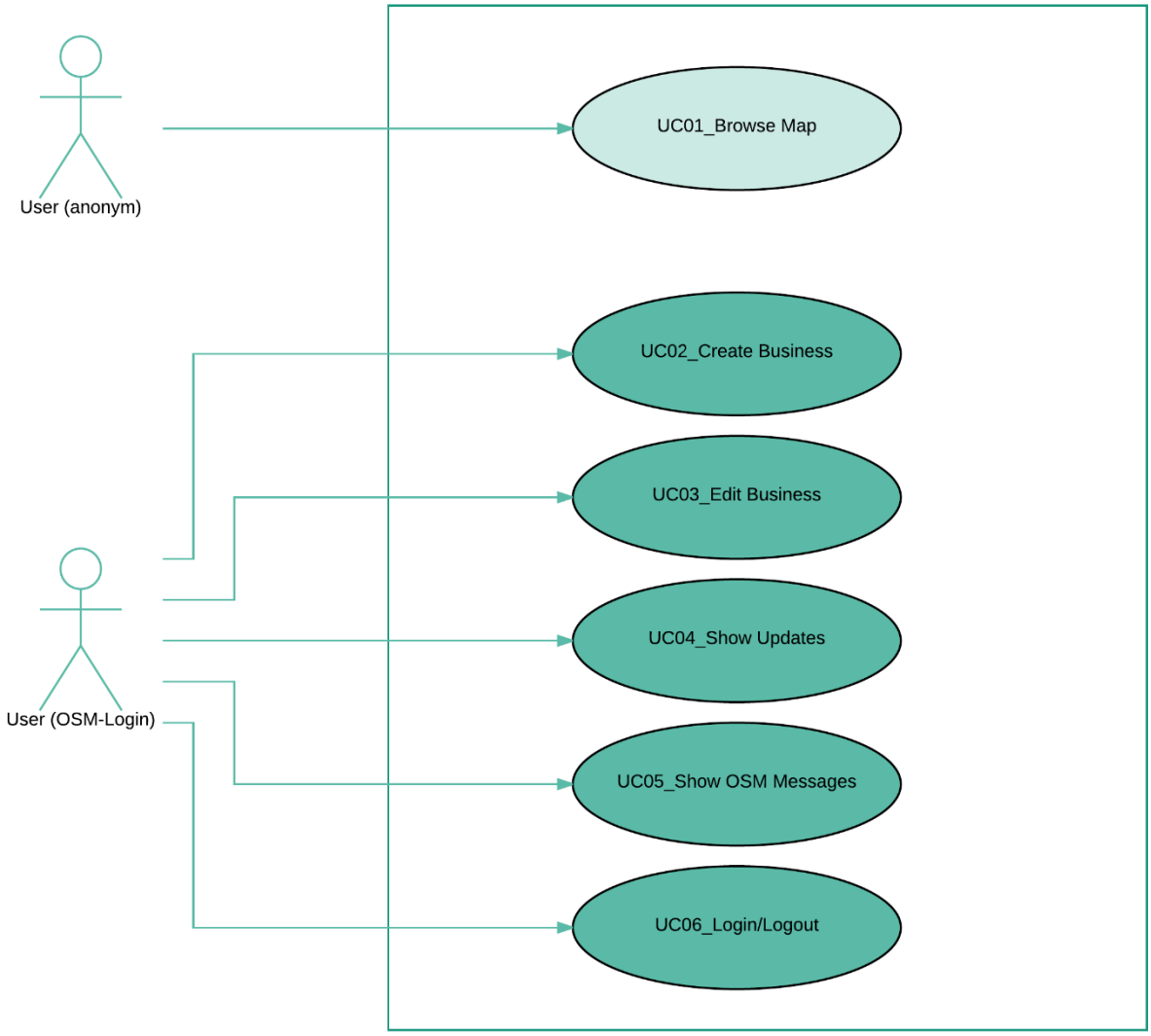


Abbildung 11 - Use Case-Diagramm

5.1.2 Use Cases

5.1.2.1 UC_01 Browse Map

Beschreibung	Der Benutzer möchte mehr über die Applikation erfahren
Actor	Benutzer (anonym)
Pre-Condition	-
Ablauf (Main Success Scenario)	1. Der Benutzer gibt die Adresse im Suchfeld ein 2. Die Karte zeigt den Ausschnitt mit der Adresse an 3. Der Benutzer klickt mit der rechten Maustaste auf das Gebäude und liest die Informationen 4. Punkte 1-3 können beliebig wiederholt werden 5. Der Benutzer schliesst die Seite
Post-Condition	-
Alternative Abläufe	1a. Der Benutzer wählt die Adresse durch zoomen selber aus 1b. Der Benutzer wählt die Adresse anhand der aktuellen Position 3a. Der Benutzer klickt mit der linken Maustaste auf das Unternehmen (blauer Kreis) 5a. Der Benutzer erstellt einen OSM-Login

Tabelle 4 - UC_01 Browse Map

5.1.2.2 UC_02 Create Business

Beschreibung	Der Benutzer möchte sein Unternehmen für die OpenStreetMap erfassen
Actor	Benutzer (OSM-Login)
Pre-Condition	Benutzer hat einen OSM-Login und ist angemeldet
Ablauf (Main Success Scenario)	1. Der Benutzer gibt die Adresse im Suchfeld ein 2. Die Karte zeigt den Ausschnitt mit der Adresse an 3. Der Benutzer klickt mit der rechten Maustaste auf das Gebäude 4. Der Benutzer kontrolliert die Adresse und drückt auf «Erstellen» 5. Der Benutzer gibt seine Daten ein 6. Der Benutzer schliesst das Erfassen ab, indem er auf «Speichern» drückt
Post-Condition	Der Knoten wurde erfasst
Alternative Abläufe	1a. Der Benutzer wählt die Adresse durch zoomen selber aus 1b. Der Benutzer wählt die Adresse anhand der aktuellen Position 6a. Der Benutzer möchte abbrechen und drückt auf «Zurück»

Tabelle 5 - UC_02 Create Business

5.1.2.3 UC_03 Edit Business

Beschreibung	Der Benutzer möchte ein bereits erfasstes Unternehmen verwalten
Actor	Benutzer (OSM-Login)
Pre-Condition	-Benutzer hat einen OSM-Login und ist angemeldet -Das Unternehmen wurde bereits erfasst
Ablauf (Main Success Scenario)	1. Der Benutzer gibt die Adresse im Suchfeld ein 2. Die Karte zeigt den Ausschnitt mit der Adresse an 3. Der Benutzer klickt mit der linken Maustaste auf das Unternehmen (blauer Kreis) 4. Der Benutzer kontrolliert die Adresse und drückt auf «Bearbeiten» 5. Der Benutzer passt seine Daten an 6. Der Benutzer schliesst die Änderung ab, indem er «Speichern» drückt
Post-Condition	Die Notiz wurde erfasst.
Alternative Abläufe	1a. Der Benutzer wählt die Adresse durch zoomen selber aus 1b. Der Benutzer wählt die Adresse anhand der aktuellen Position 6a. Der Benutzer möchte abbrechen und drückt auf «Zurück»

Tabelle 6 - UC_03 Edit Business

5.1.2.4 UC_04 Show Updates

Beschreibung	Der Benutzer möchte informiert werden, wenn Jemand Änderungen an einem bereits von ihm editierten oder erstellten Unternehmen vorgenommen hat
Actor	Benutzer (OSM-Login)
Pre-Condition	-Benutzer hat einen OSM-Login und ist angemeldet -Das Unternehmen wurde von diesem Benutzer bereits erfasst oder editiert
Ablauf (Main Success Scenario)	1. Der Benutzer sieht sich die Updates an 2. Der Benutzer bestätigt, dass er das Update gelesen hat
Post-Condition	Update wurde aus der Liste entfernt
Alternative Abläufe	2a. Der Benutzer zoomt auf das Unternehmen über OSMyBiz 2b. Der Benutzer sieht sich die Änderung auf der OpenStreetMap direkt an 2c. Der Benutzer stellt die Notifikationen für dieses Unternehmen aus

Tabelle 7 - UC_04 Show Updates

5.1.2.5 UC_05 Show OSM Messages

Beschreibung	Der Benutzer möchte informiert werden, ob er Nachrichten in seinem OSM Postfach hat
Actor	Benutzer (OSM-Login)
Pre-Condition	-Benutzer hat einen OSM-Login und ist angemeldet -Benutzer hat mindestens eine neue Nachricht
Ablauf (Main Success Scenario)	1. Der Benutzer sieht, dass er eine Nachricht hat 2. Der Benutzer drückt auf das Umschlag-Icon und wird auf die OpenStreetMap-Seite weitergeleitet 3. Der Benutzer liest die Nachricht
Post-Condition	Nachrichtenzähler wieder zurückgesetzt
Alternative Abläufe	-

Tabelle 8 - UC_05 Show OSM Messages

5.1.2.6 UC_06 Login/Logout

Beschreibung	Der Benutzer möchte sich anmelden, um ein Unternehmen zu erstellen bzw. zu bearbeiten Wenn er fertig ist, möchte er sich wieder abmelden
Actor	Benutzer (OSM-Login)
Pre-Condition	Der Benutzer hat einen OSM-Login, kennt seinen Benutzernamen und sein Passwort und ist noch nicht angemeldet
Ablauf (Main Success Scenario)	1. Der Benutzer öffnet die Applikation 2. Der Benutzer meldet sich an 3. Der Benutzer nutzt die Applikation 4. Der Benutzer meldet sich ab
Post-Condition	Der Benutzer ist abgemeldet
Alternative Abläufe	-

Tabelle 9 - UC_06 Login/Logout

5.1.3 Funktionale Anforderungen

Titel	Beschreibung
SPAM-/ und Bot-Filter	Das erfasste Unternehmen soll auf SPAM überprüft werden und nicht in der OpenStreetMap erfasst werden, wenn die Überprüfung fehlschlägt. Ebenfalls soll verhindert werden, dass Bots Unternehmen erfassen.

Tabelle 10 - Funktionale Anforderungen

5.1.4 Nicht-funktionale Anforderungen

Titel	Beschreibung
Responsive Design	Das User-Interface soll responsive sein, aber es wird nicht mit Mobile-First gearbeitet. Die Applikation soll optimiert sein für Desktop-PCs.
Wartbarkeit	Der Code für die Anwendung soll so aufgebaut werden, dass zu einem späteren Zeitpunkt noch weitere Features eingebaut werden können.
Benutzbarkeit	Die Anwendung soll so aufgebaut werden, dass sie selbsterklärend und einfach zu benutzen ist.
Performance	Schnelle Antwortzeiten sind keine Vorgaben, jedoch soll der Benutzer spätestens nach 5 Sekunden eine Rückmeldung erhalten. Nach Möglichkeit wird mit "Best Effort" eine bessere Antwortzeit ermöglicht.
Zuverlässigkeit	Es muss gewährleistet werden, dass bei Sucheingaben keine falschen Ergebnisse ausgegeben werden.

Tabelle 11 - Nicht-funktionale Anforderungen

5.1.5 Aktivitätsdiagramme

5.1.5.1 Unternehmen erfassen

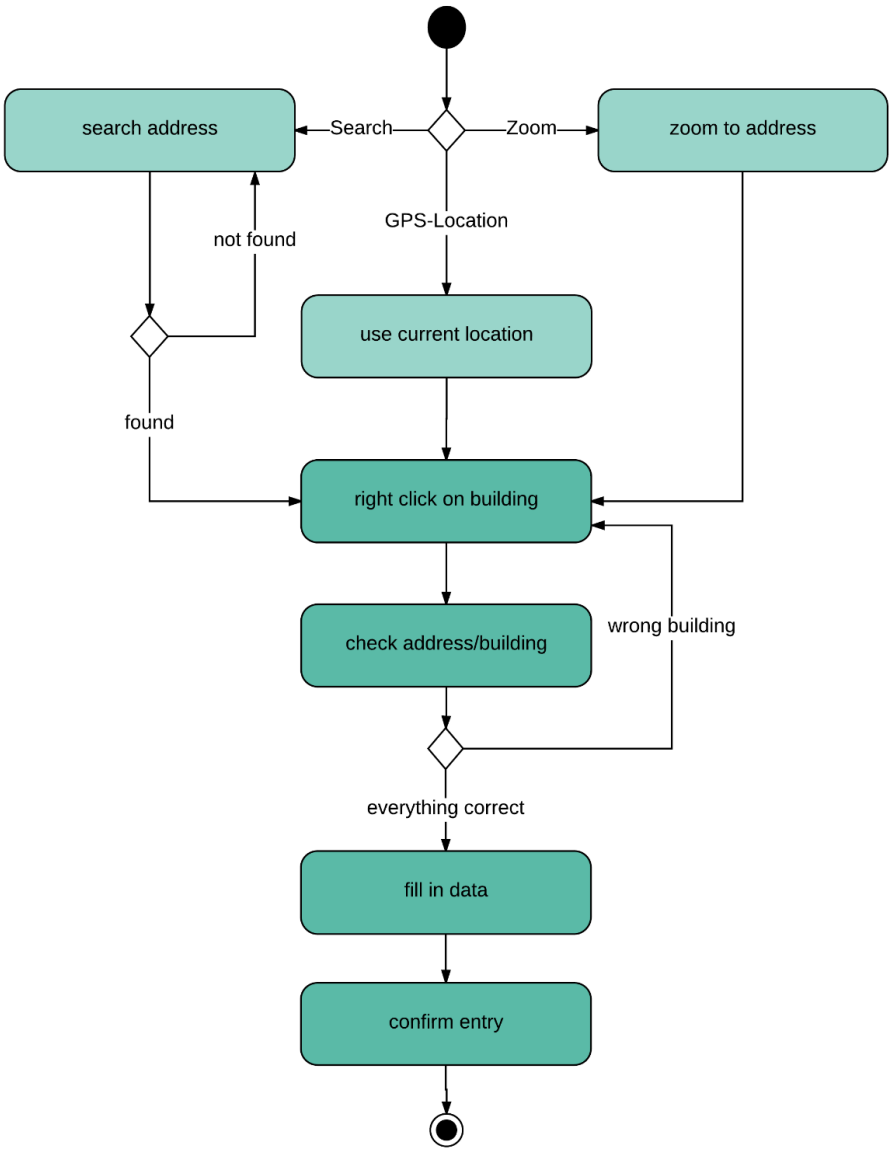


Abbildung 12 – Aktivitäts-Diagramm – Create Business

Der Benutzer hat drei Möglichkeiten, um den Kartenausschnitt mit seinem Unternehmen zu wählen. Er kann über das Suchfeld eine Adresse eingeben, bis die Adresse gefunden und der Kartenausschnitt den Koordinaten der Adresse angepasst wird. Die zweite Möglichkeit ist, über die Zoomfunktion bis zur gewünschten Adresse zu zoomen. Und als dritte Möglichkeit kann der Benutzer auch die GPS-Location von seinem Gerät verwenden und wird direkt dorthin geführt.

Mit einem Rechtsklick auf ein Gebäude kann der Benutzer ein neues Unternehmen erfassen. Es erscheint ein Popup mit der Adresse, die der Benutzer kontrolliert. Bei einer falschen Auswahl, kann der Benutzer mit neuem Rechtsklick ein anderes Gebäude auswählen.

Danach füllt er alle Daten im Formular aus, die er eingeben möchte und schickt das Formular ab.

5.1.5.2 Unternehmen bearbeiten

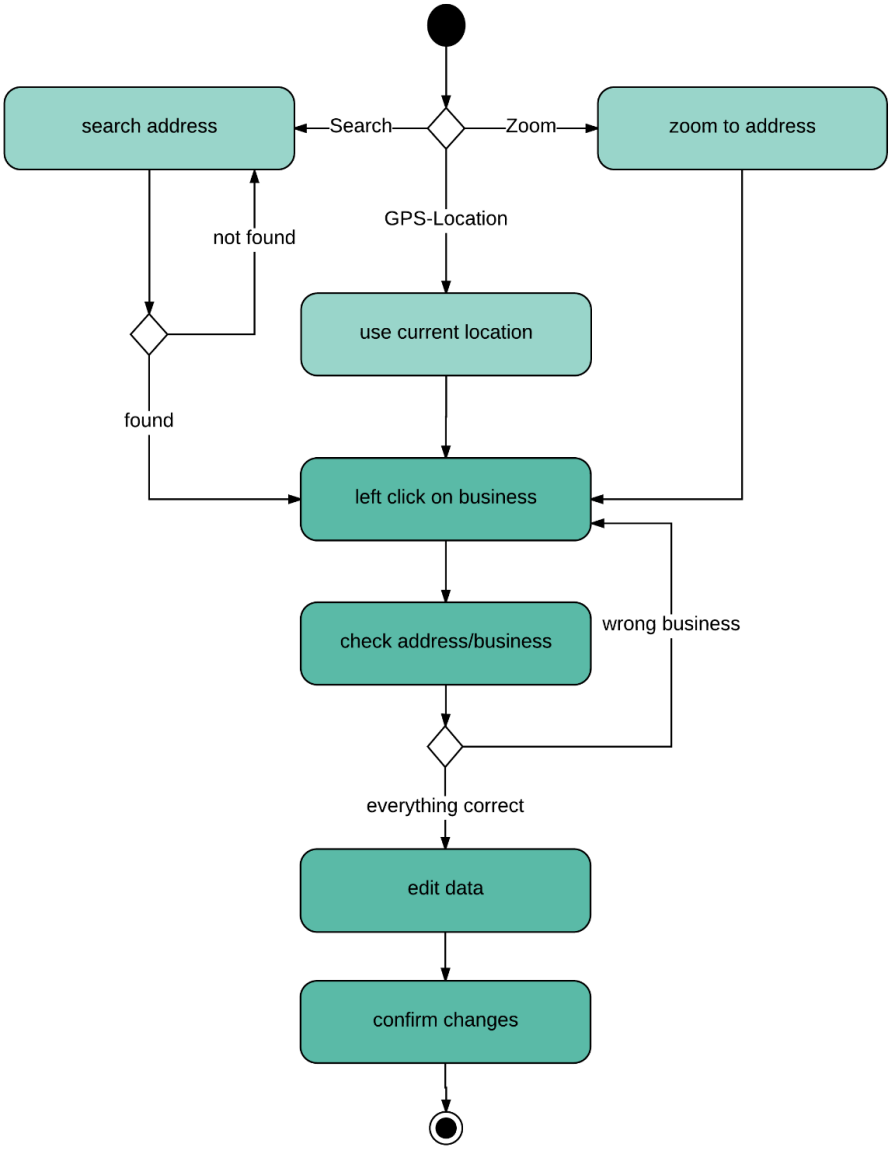


Abbildung 13 – Aktivitäts-Diagramm – Edit Business

Für das Bearbeiten eines Unternehmens hat der Benutzer die gleichen drei Möglichkeiten, um den Kartenausschnitt zu wählen.

Danach muss er aber mit einem Linksklick auf ein bestehendes Unternehmen klicken, um dieses zu bearbeiten. Auch hier erscheint ein Popup mit der Adresse und neu auch mit der Kategorie und dem Namen des Unternehmens. Der Benutzer kontrolliert diese und wählt allenfalls ein anderes Unternehmen, wenn es das Falsche war.

Die Formularfelder sind in diesem Fall bereits ausgefüllt, soweit die Daten vorhanden sind. Der Benutzer passt diese an und schickt das Formular ab.

5.2 Analyse

5.2.1 Domainmodel

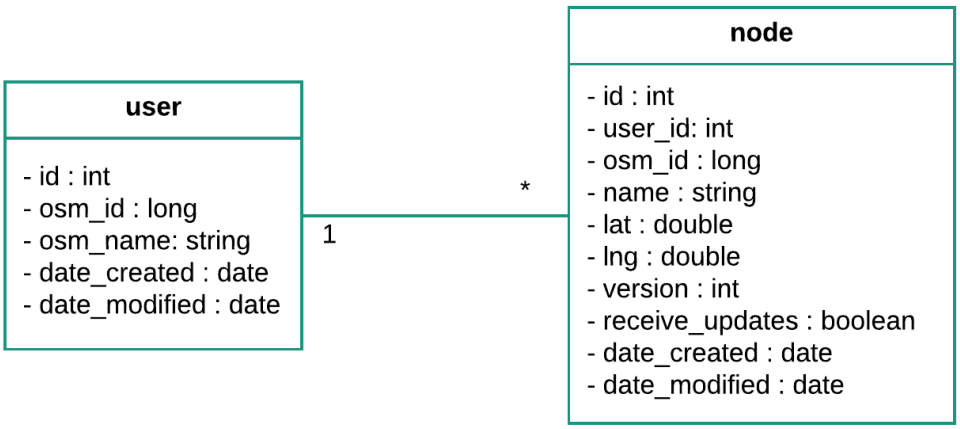


Abbildung 14 – Domainmodel

5.2.1.1 Erläuterungen

Da das OSMyBiz Backend nur verwendet wird, um die vom User bearbeiteten oder erstellten Knoten abzuspeichern, ist das dazugehörige Domainmodel entsprechend simpel. Es wird generell nur das allernötigste an Informationen abgespeichert.

In der User-Tabelle werden nur die Benutzernamen sowie die OSM-ID zur Identifikation persistiert. Da der Login über OSM OAuth (25) läuft, werden in OSMyBiz keine Emailadressen oder Passwörter benötigt. In der Node-Tabelle werden OSM Knoten, die bearbeitet oder neu erstellt wurden, abgespeichert. Als primäre Identifikation, zur Assoziierung eines Knoten mit den OSM Daten, wird die OSM-ID verwendet. Die zusätzlichen Felder, wie die Position und die Namen, werden eigentlich nur dann benötigt, wenn ein Knoten aus den OSM Stammdaten entfernt wurde und der User trotzdem noch eine sinnvolle Notifikation erhalten möchte. Die Versionsnummer und das «recieve_updates»-Feld werden benötigt um zu evaluieren ob der User eine Notifikation erhalten soll.

Es wurde bewusst darauf verzichtet das Datenbankmodell in die zweite Normalform zu bringen, da dies nur zu einer Steigerung der Komplexität bei den Abfragen führen würde. Zudem gäbe es nur eine minimale Reduktion der totalen Datenmenge.

5.2.2 Sequenzdiagramm

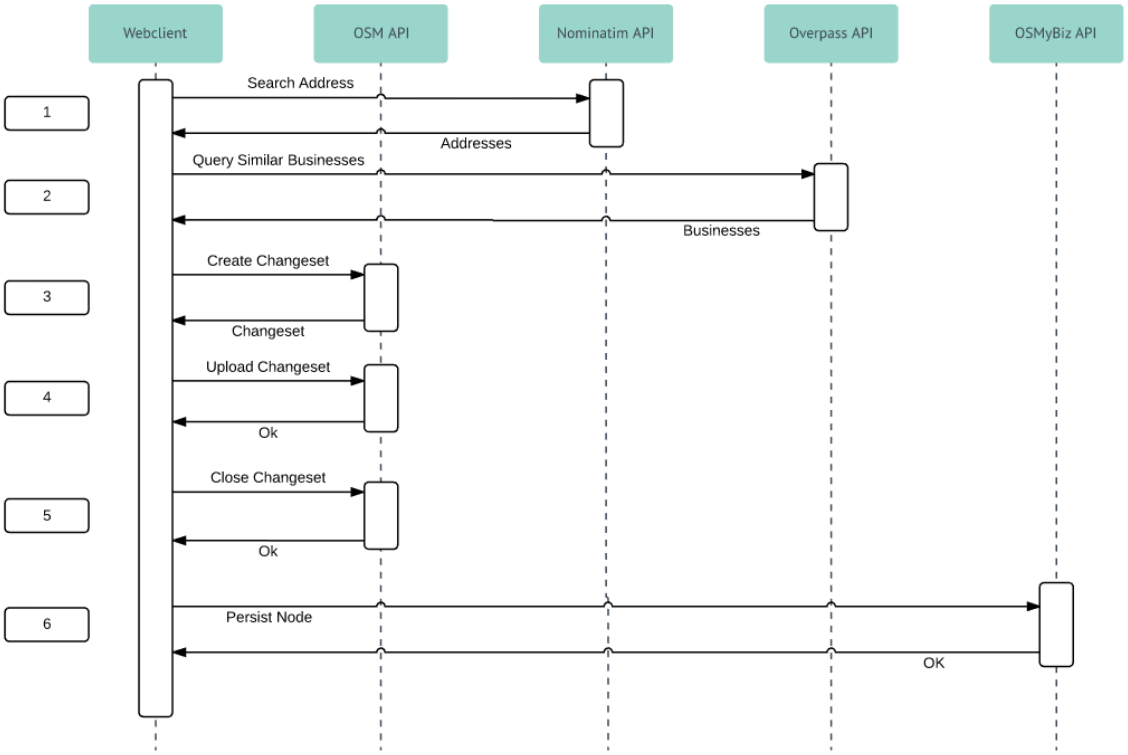


Abbildung 15 - Sequenzdiagramm Unternehmen erstellen

Der User sucht nach seiner Adresse diese wird per Anfrage auf Nominatim (siehe 5.4.1.4 Nominatim API) aufgelöst (Nr. 1).

Nachdem der User seine Unternehmensdaten erfasst hat und speichern drückt, wird eine Abfrage über Overpass (siehe 5.4.1.5 Overpass API) (Nr. 2) ausgeführt, um zu überprüfen, ob es in der nahen Umgebung bereits ein Unternehmen mit gleichem Namen und gleicher Kategorie gibt.

Ist diese Überprüfung erfolgreich, wird per OSM API ein Changeset erstellt (Nr. 3).

Im erstellten Changeset wird dann der neue Knoten hinzugefügt und es wird wiederum per OSM API Aufruf aktualisiert (Nr. 4), und gleich danach geschlossen (Nr. 5).

Im letzten Schritt wird der neu erstellte Knoten über die OSMyBiz API ins Backend gespeichert (Nr. 6).

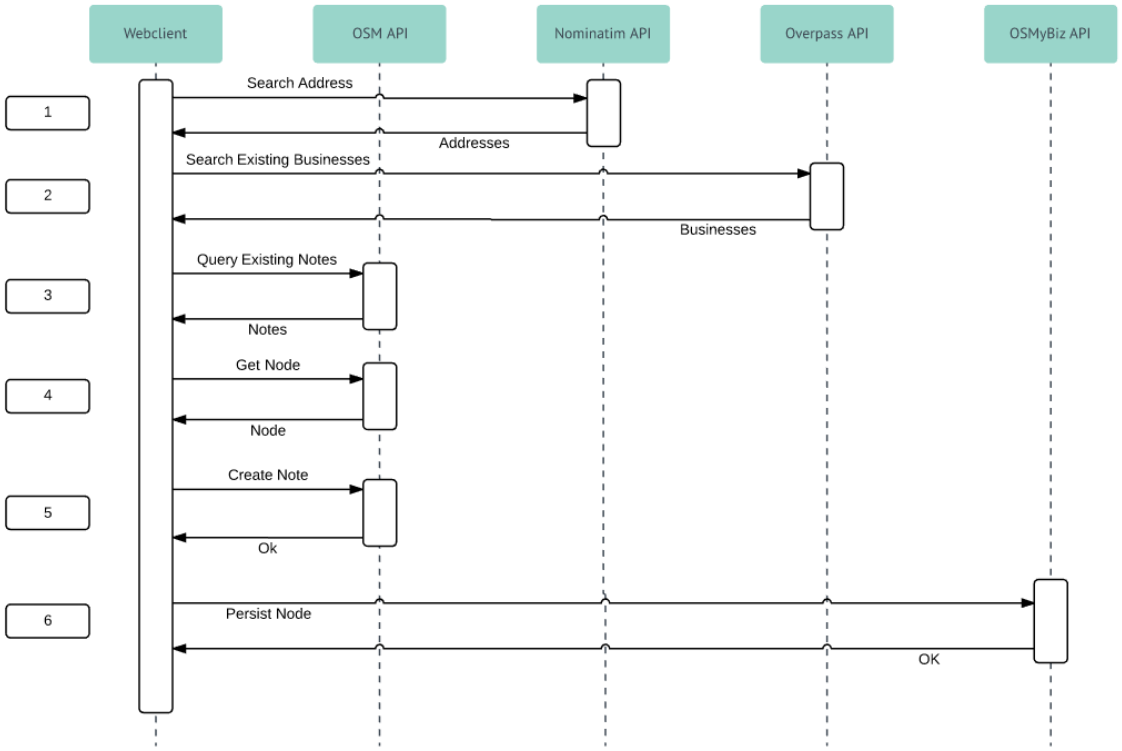


Abbildung 16 - Sequenzdiagramm Unternehmen bearbeiten

Der User sucht auch hier über Nominatim (siehe 5.4.1.4 Nominatim API) nach seiner Adresse (Nr. 1).

Danach werden über Overpass (siehe 5.4.1.5 Overpass API) die bereits eingetragenen Unternehmen geladen (Nr. 2).

Sobald der User im Popup auf bearbeiten klickt, wird eine Abfrage an die OSM API gestartet um zu überprüfen ob bereits eine Notiz zu diesem Unternehmen existiert (Nr. 3).

Ist dies nicht der Fall wird der existierende Knoten über die OSM API geladen (Nr. 4), um die bereits ausgefüllten Felder im Formular auszufüllen.

Sobald der User auf speichern klickt, wird über die OSM API eine Notiz erstellt (Nr. 5).

Der letzte Schritt ist wiederum gleich wie beim Erstellen, der betroffene OSM Knoten wird über die OSMyBiz API gespeichert (Nr. 6).

5.3 Architektur & Design

5.3.1 Client Architektur

Für die Implementation des OSMyBiz Clients wurde VueJS verwendet. Der klassische Ansatz mit VueJS entspricht dem MVC Pattern, wobei jede Komponente aus einem Controller und einem Template (View) besteht und seine eigenen Daten (Model) verwaltet.

Die Kommunikation zwischen verschiedenen Komponenten findet dabei entweder über einen globalen Eventbus oder direkt zwischen Parent und Child statt. Da dies schnell zu Problemen mit nicht synchronen Daten zwischen Komponenten führen kann, wurde anstelle von MVC das Redux-Pattern und die Hilfsbibliothek Vuex verwendet.

5.3.1.1 Redux

Um den Synchronisationsproblemen entgegen zu wirken, wird im Redux Pattern (26) sämtlicher Applikationszustand in einem zentralen Store gespeichert. Dieser Store ist für alle Views bzw. Komponenten abfragbar, kann jedoch nicht direkt mutiert werden. Um eine Änderung an den Daten im Store vorzunehmen wird eine Action an den Dispatcher geschickt. Der Dispatcher ist neben dem Store eine weitere einzigartige Komponente, die als einzige Actions an den Store schicken kann, damit dieser den Applikationszustand updaten kann. Die Actions sind fix definierte Operationen die im Store so implementiert werden, dass der Applikationszustand immer gültig bleibt.

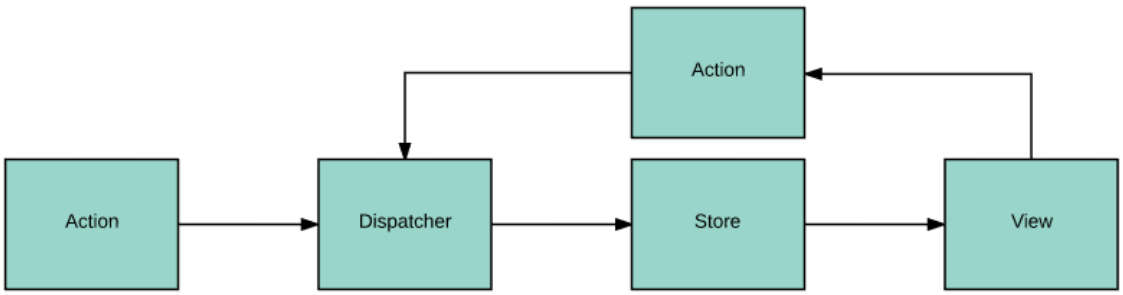


Abbildung 17 - Redux Pattern

Nach einem Update im Store werden sämtliche Views wiederum über die neuen Daten informiert. Der unidirektionale Datenfluss des Redux Patterns vereinfacht das Entwickeln der Views sehr, da diese nicht mehr direkt von anderen Views, sondern nur noch vom Store abhängig sind und keine interne Zustandsverwaltung mehr haben.

5.3.1.2 Implementation mit VueJS und Vuex

Vuex (27) ist eine Implementierung des Redux Patterns und erweitert VueJS um den zentralen Store zur Zustandsverwaltung. Der Store sieht in Vuex etwas anders aus als im klassischen Redux.

Die Actions aus Redux entsprechen bei Vuex Mutations, wobei die Actions in Vuex ein zusätzliches Feature sind um asynchrone Abläufe (z. B. Http Requests) abzubilden. Das Grundkonzept bleibt jedoch dasselbe, Mutations sind die einzigen erlaubten Änderungen am State. Der Vuex Store bietet zusätzlich noch Getters und Actions an, wobei Getters einfache Funktionen sind um Werte aus dem State oder Aggregationen von Werten aus dem State abzufragen. Actions dürfen im Vergleich zu Mutations asynchron sein und können Mutations aufrufen.

Für die Interaktion zwischen VueJS Komponenten und dem Vuex Store werden Hilfsmethoden wie „mapMutations" angeboten, um die Mutationen als Funktionen direkt in die Komponente einzubinden. Der Umweg über den Dispatcher findet zwar immer noch statt, muss jedoch nicht manuell ausgeführt werden.

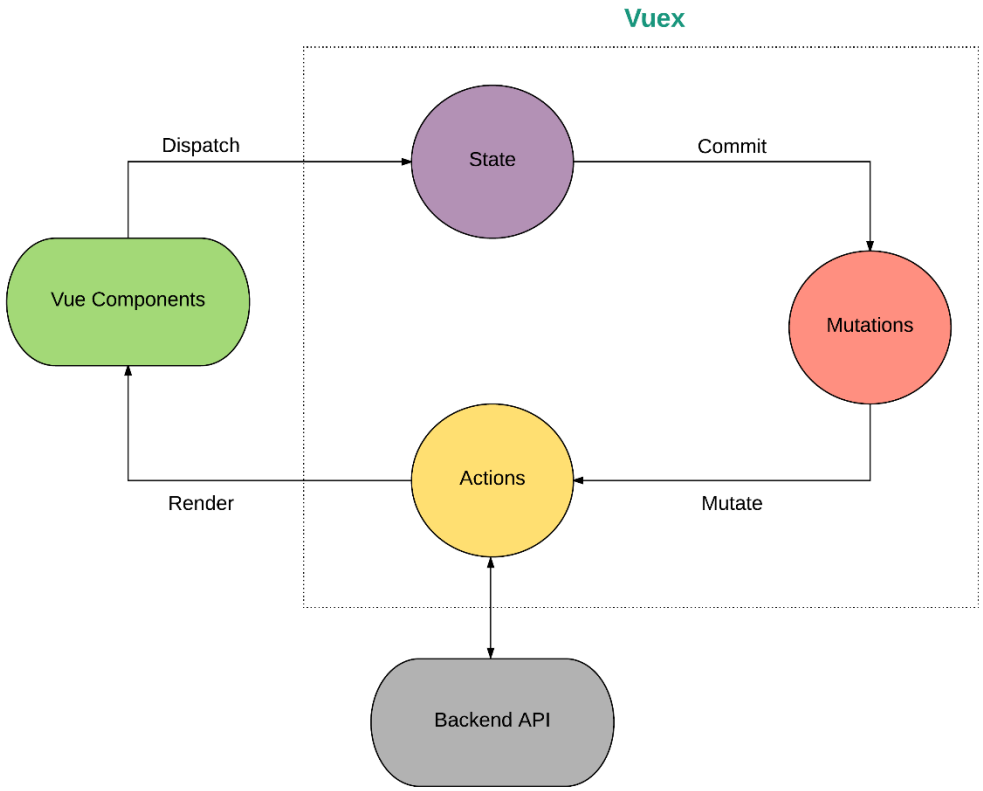


Abbildung 18 - Vuex Datenfluss

5.3.2 Server Architektur

Für die Implementation der OSMyBiz API wurden primär Flask (11) und Flask SQL Alchemy als ORM (28)verwendet. Da sich der grösste Teil Applikation im Client befindet, ist der Server sehr simpel aufgebaut. Für den Datenbankzugriff werden pro Tabelle eine Klasse mit den entsprechenden Feldern, sowie Mappings auf die unterliegenden Datenbanktypen erstellt. Diese Klassen implementieren zusätzlich Methoden, um ihren Inhalt in die Datenbank zu speichern. Da alle schreibenden Datenbankzugriffe nur aus einzelnen Updates bestehen und keine komplexen Transaktionen stattfinden, wird jede Datenbankoperation direkt ausgeführt. Für die lesenden Datenbankzugriffe werden Query Funktionen direkt auf den Modelklassen von SQL Alchemy zur Verfügung gestellt.

Die von der OSMyBiz API zur Verfügung gestellten Funktionen werden bei Flask über Attribute als Routen registriert. Diese werden dann automatisch durch Flask aufgerufen sobald ein Request auf der entsprechenden Route eintrifft. Die Routen werden so aufgebaut, dass sie einer REST API der Stufe 2 (gemäss Richardson Maturity Model (29)) entsprechen. Es wird auf verschachtelte Ressourcen zugegriffen und dies spiegelt sich in der URL wieder und es werden entsprechend der Art des Requests die Verben POST und GET verwendet. Auf die Implementierung der dritten REST Stufe wurde mangels ersichtlichem Nutzen verzichtet.

Der Rest der Serverlogik befasst sich hauptsächlich mit dem Validieren, Serialisieren und Deserialisieren von eintreffenden Daten. Hier ist zu beachten, dass alle Daten die vom Client kommen zwingend auf korrekte Struktur und Typen überprüft werden müssen, bevor sie an die Datenhaltung weitergegeben werden.

5.3.3 Deployment

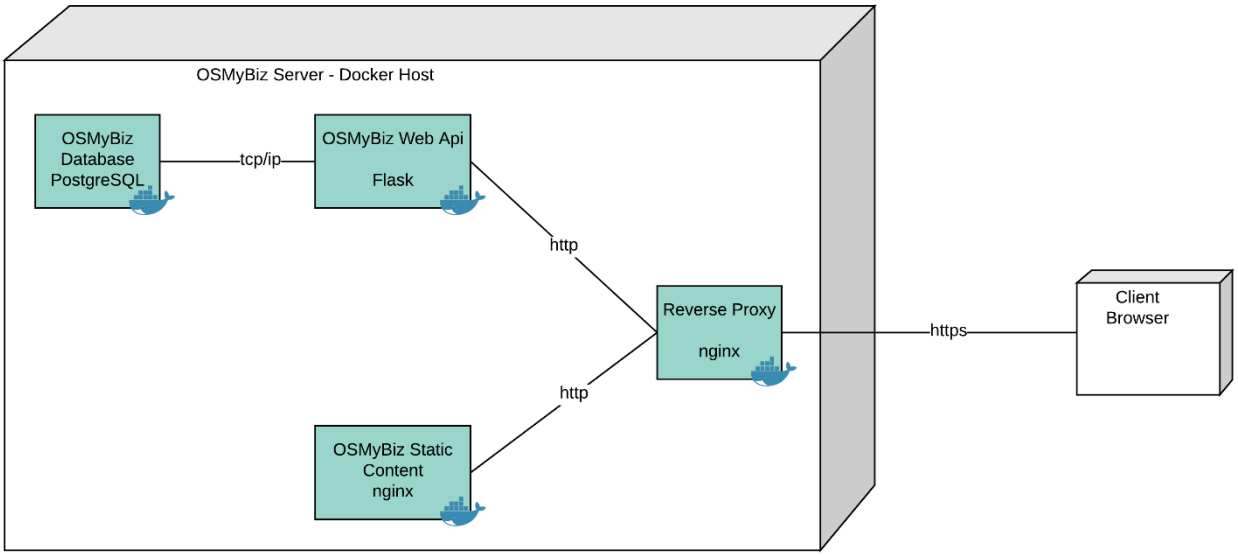


Abbildung 19 – Deploymentdiagramm

Sämtliche Komponenten im Projekt wurden als Docker (30) Container realisiert.

Um die komplette Applikation unter einer Domäne laufen zu lassen, wurde ein nginx (31) Server als Reverse Proxy aufgesetzt. Dieser nimmt alle Clientrequests entgegen und leitet sie, abhängig von der aufgerufenen URL, an den richtigen Server weiter.

Die Client Applikations-Dateien werden von einem weiteren nginx Server als statische Dateien zum Download zur Verfügung gestellt.

Die Rest API läuft in einem Python Container und ist über den Pfad "/api/" für den Client erreichbar.

Der letzte Container enthält eine PostgreSQL Datenbank und wird vom API Server für die Datenhaltung genutzt.

Im einfachsten Fall werden alle Docker Container auf demselben Server aufgesetzt, es wäre jedoch denkbar, dass die verschiedenen Container auf verschiedene physikalische Server aufgeteilt werden.

5.3.4 UI-Design

Es wurden in der Planungsphase erste UI-Entwürfe ausgearbeitet. Die Ansichten sind so dargestellt, dass sie die Situation eines angemeldeten Benutzers anzeigen.

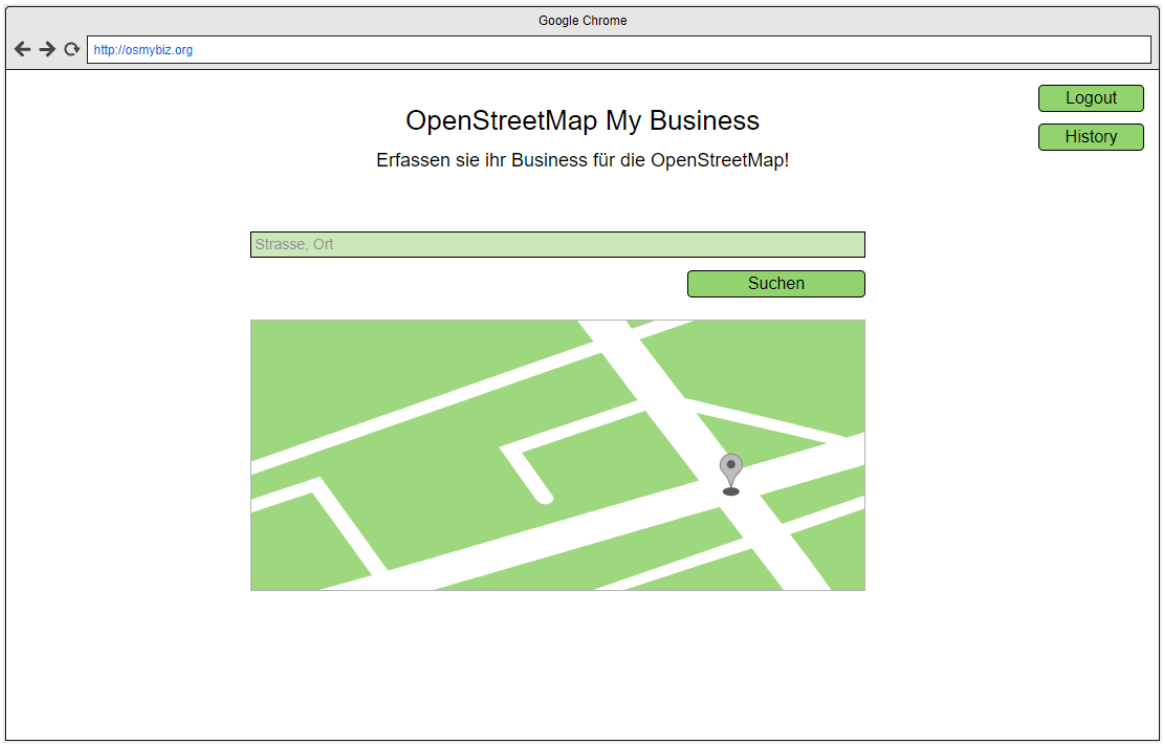


Abbildung 20 - Mockup - Landing Page

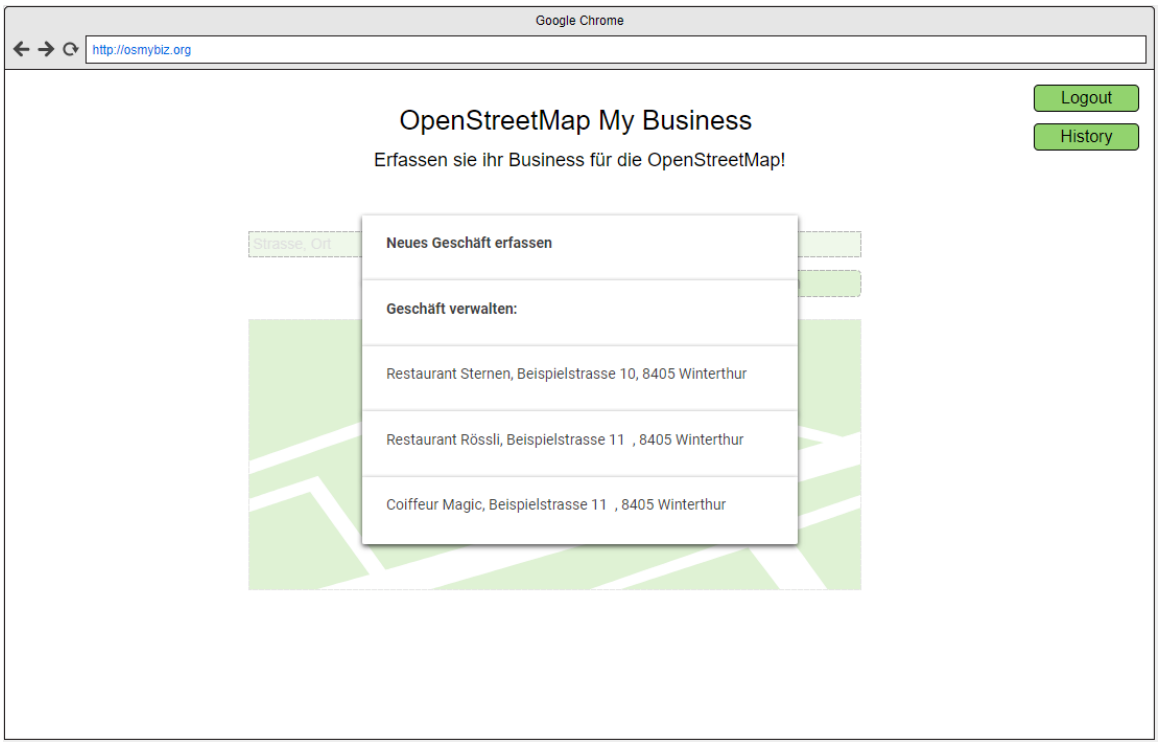


Abbildung 21 - Mockup - Manage Business Page

Zu folgender Ansicht gibt es noch eine ähnliche Seite, die für das Verwalten der Daten zuständig ist. Dabei sind die Felder bereits ganz oder teilweise ausgefüllt, damit der Benutzer nur noch die Daten anpassen muss.

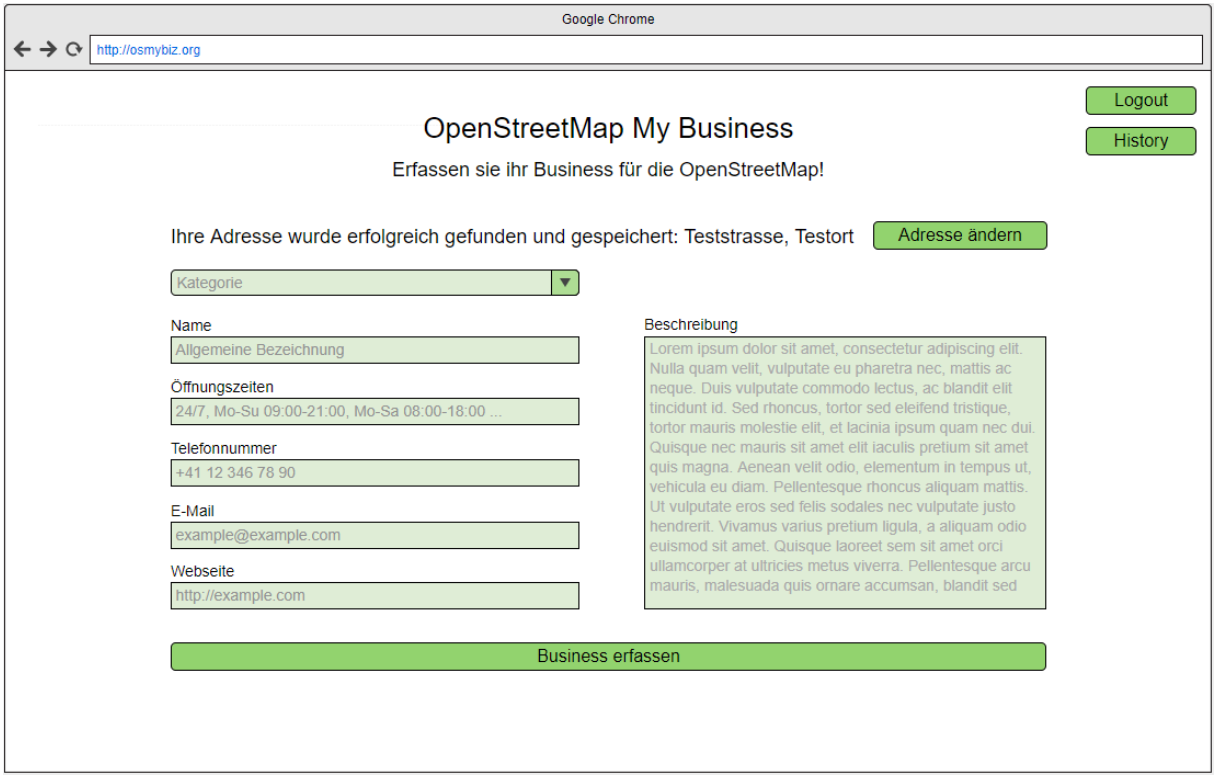


Abbildung 22 - Mockup - Create Business Detail Page

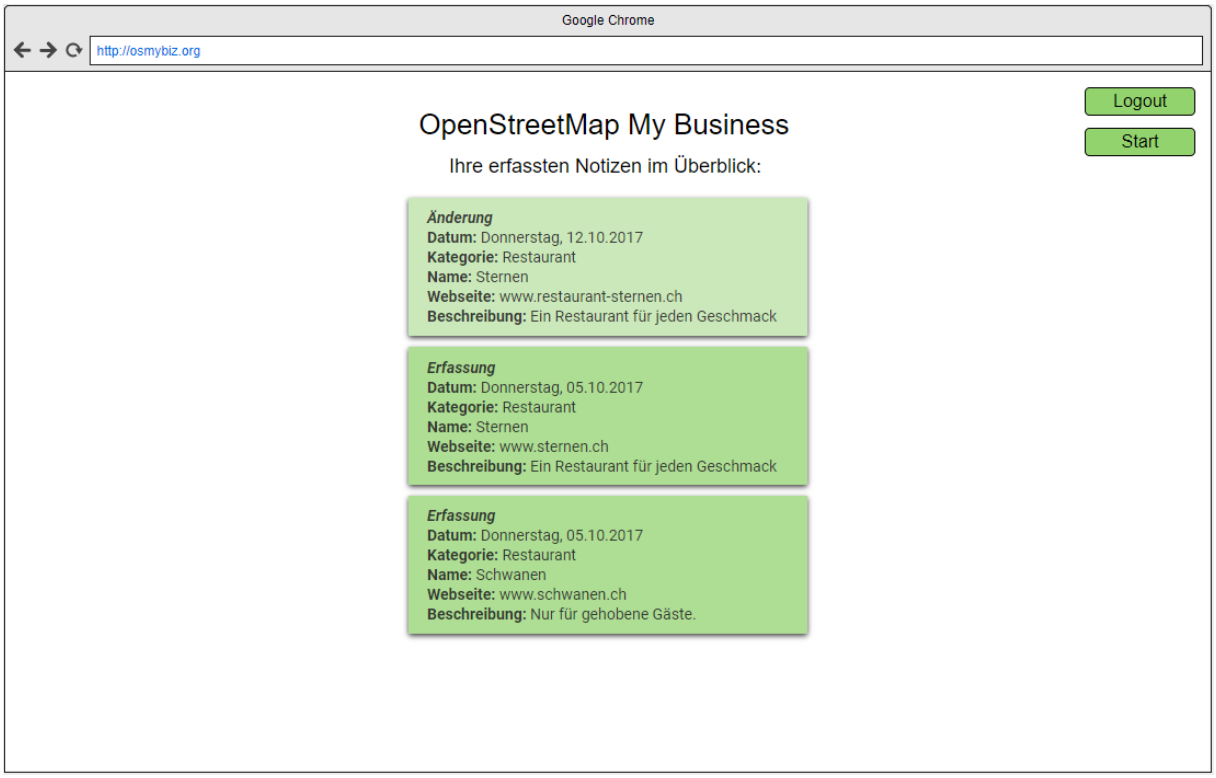


Abbildung 23 - Mockup - History Page

5.4 Implementation

5.4.1 Angebundene APIs und Datenquellen

Für die Umsetzung von OSMyBiz wurden diverse APIs und Datenquellen aus dem OSM-Ökosystem angebunden.

5.4.1.1 OSM Tile Server

Der OSM Tile Server stellt vorgerenderte Kartenausschnitte zur Verfügung. Diese werden anhand der Zoomstufe, sowie der Koordinaten, dynamisch von leaflet.js geladen und im Client zur Karte zusammengesetzt. Abhängig von der Zoomstufe werden mehr oder weniger Details in den Maptiles dargestellt. Die Tiles werden bei Änderungen asynchron in einem Hintergrundprozess neu gerendert, diese führt dazu, dass neu erstellte Knoten erst nach einer gewissen Zeit in der Karte zu sehen sind.

5.4.1.2 Mapbox Satelite Tiles

In der Satellitenansicht werden die Maptiles von Mapbox (32) bezogen. Mapbox bereitet dazu Satelliten- und Luftbilder von diversen Quellen wie NASA oder USGS auf und stellt diese in einer mit leaflet.js kompatiblen API zur Verfügung. Zur Nutzung muss ein Mapbox Account erstellt werden. In den Account Einstellungen auf Mapbox kann man dann ein Token erstellen lassen um die Kartendaten zu laden.

5.4.1.3 OSM API

Die offizielle OSM API (33) bietet Schnittstellen zur direkten Abfrage von diversen Daten, sowie zur Erstellung von neuen Objekten. Requests, die zwingend einen Benutzer benötigen, müssen zuerst per OAuth (34) und den nötigen Genehmigungen autorisiert werden. Für OSMyBiz wurden folgende Funktionen benutzt:

Pfad	Methode	Beschreibung
/user/details	GET	Lädt Benutzerdaten für den eingeloggten User
/changeset/create	POST	Erstellt ein neues Changeset
/changeset/#id/upload	POST	Speichert Änderungen in das Changeset mit der entsprechenden ID
/changeset/#id/close	POST	Schliesst das Changeset mit der entsprechenden ID
/node/#id	GET	Lädt den Knoten mit der entsprechenden Id.
/note	POST	Erstellt eine Notiz.
/note/#left.#bottom .#right.#top	GET	Lädt alle Notizen in einer Box die durch zwei Kartenpositionen (Linksunten, Rechtsoben) aufgespannt wird.

Tabelle 12 - OSM API Pfade

5.4.1.4 Nominatim API

Nominatim (35) ist ein Service, der die Kartendaten aus OSM speziell nach Adressdaten indexiert und verschiedene adressbezogene Abfragen effizient beantworten kann.

Für OSMyBiz wird die Adresssuche für die Suchleiste in der Kartenansicht benutzt. Für einen beliebigen Text werden jeweils die besten Treffer in absteigender Reihenfolge zurückgegeben. Dies wird als Grundlage für das Type-Ahead in der Kartenansicht verwendet. Die zweite Funktion von Nominatim die verwendet wurde ist das sogenannte Reverse Geocoding. Dabei handelt es sich um eine Funktion, die für beliebige Koordinaten die beste zutreffende Adresse ermittelt. Dafür wird jeweils auf Knoten und Wegen in der nahen Umgebung nach Adressdaten gesucht. Diese Funktion wird im Popup auf der Kartenansicht benutzt um die Adresse darzustellen.

5.4.1.5 Overpass API

Bei Overpass (36) handelt es sich um einen Service, der sich auf Suchen von Knoten und Wegen innerhalb von Flächen spezialisiert.

Overpass wird einerseits benutzt, um auf der Kartenansicht die Unternehmen in der höchsten Zoomstufe im aktuellen Kartenausschnitt zu laden. Andererseits wird es verwendet, um während dem Erstellen eines neuen Knotens, zu überprüfen ob bereits ein ähnlicher Knoten in der nahen Umgebung existiert.

5.4.2 Datenübermittlung

Neues Unternehmen: In diesem Fall müssen verschiedene Requests an die OSM API gesendet werden. Diese Requests müssen im XML-Format gesendet werden.

Als erstes wird ein sogenanntes Changeset eröffnet, damit man Änderungen an der Karte hochladen kann. Dieses enthält nur den Ersteller (OSMyBiz) und einen Kommentar. Als Response erhält man nur die Changeset-ID.

Danach können die Daten über die Changeset-ID in dieses Changeset gespeichert werden. Aber davor müssen diese noch richtig zusammengesetzt werden. Das XML wird als Change hochgeladen und enthält den Node (Knoten), welcher die Koordinaten und die Changeset-ID mitliefert. Innerhalb des Knoten-Tags werden die Daten mitgesendet. Diese müssen als Key-Value-Paare gesendet werden (z.B. k=”email” v=”example@example.com”). Als Response erhält man den erstellten Node jedoch als XML. Dieses muss erst noch in ein JSON umgewandelt werden und wird dann zurückgegeben.

Zum Schluss wird das Changeset wieder geschlossen und damit auch der Prozess des Erstellens im Hintergrund beendet.

Unternehmen bearbeiten: Dafür ist nur ein Request nötig, der die Notiz als String an die OSM API sendet.

Dieser String muss aber auch noch konstruiert werden. Am Anfang des Strings kommt das Hashtag für OpenStreetMap My Business (“#OSMyBiz“). Dieses Hashtag dient dazu, dass derjenige der die Notiz verarbeitet sieht, woher diese stammt.

Die restlichen Daten werden mit Zeilenumbrüchen, jeweils mit Titel und Inhalt (z.B. Kategorie: Bar) an den String angehängt.

Der String wird dann als Notiz, zusammen mit den Koordinaten, an OSM übermittelt. Die API sendet, bei Erfolg, als Antwort die erstellte Notiz zurück. Man kann beim Request mitgeben, dass die Notiz bereits als JSON zurückkommt.

5.4.3 Kartenansicht

Die Karte wurde mit Hilfe von leaflet.js (37) umgesetzt. Dabei handelt es sich um eine JavaScript Bibliothek, die Funktionen für die Darstellung und Bedienung einer Karte anbietet.

Bei der Kartenansicht wurde bewusst auf Abstände und Rahmen verzichtet, sodass die Karte einen möglichst grossen Teil des Bildschirms einnehmen kann. Die einzige Ausnahme ist der Header, dieser soll über die ganze Applikation hinweg gleichbleiben und allgemeine Funktionen wie zum Beispiel den Login anbieten.

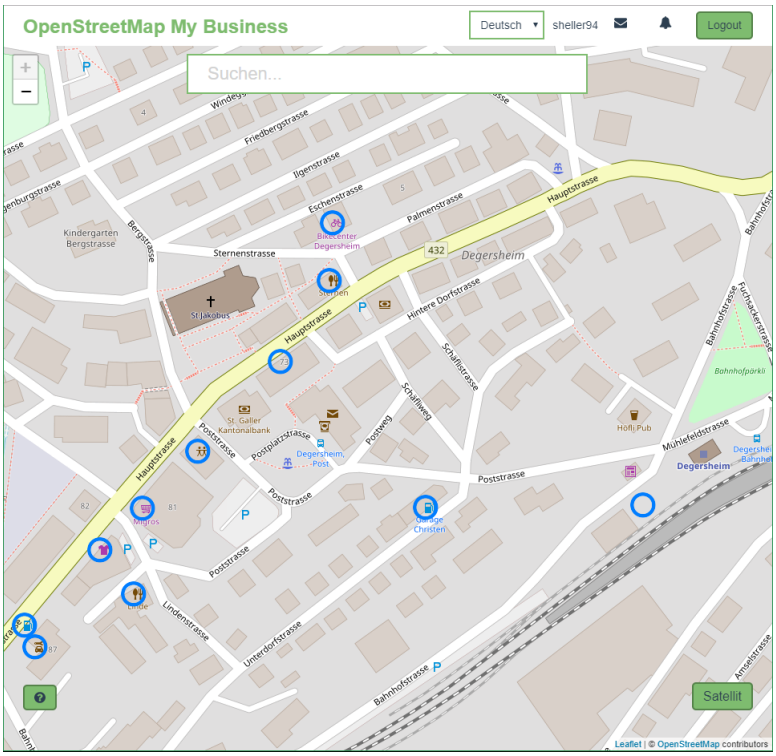


Abbildung 24 – Kartenansicht (ohne Texte)

5.4.3.1 Bedienung der Karte

Zusätzlich zu den Standardfunktionen wurde noch ein Button unten rechts eingebaut, der es ermöglicht zwischen der normalen Kartenansicht und einer Satellitenansicht zu wechseln. Die Satellitenbilder werden von Mapbox geladen. Für den Zugriff auf die Daten von Mapbox wird ein Mapbox Account benötigt. Über diesen Account kann man sich ein Token generieren lassen, welches jeweils bei den Tile-Requests angehängt werden muss, um Daten zu erhalten. Die Implementierung des Moduswechsels ist ebenfalls bereits über leaflet.js abstrahiert, es muss nur die Tile-URL angepasst werden, Mapbox lädt den aktuellen Ausschnitt dann automatisch neu.

Die initiale Kartenposition wird in einem mehrstufigen Prozess ermittelt. Höchste Priorität haben die Koordinatendaten die per URL-Hash angegeben werden können. Falls diese nicht gesetzt sind wird als nächstes im lokalen Speicher des Browsers geschaut, ob der User bereits einmal auf OSMyBiz gewesen ist. Ist dies der Fall, wird die letzte Position wiederhergestellt. Falls der User OSMyBiz zum ersten Mal besucht, wird er vom Browser nach der Berechtigung zur Verwendung der Browser Location API gefragt. Sofern der User zustimmt, wird seine aktuelle Position verwendet. Lehnt der User den

Berechtigungs-Request ab, zeigt die Karte die ganze Schweiz in einer entsprechenden Zoomstufe an.

Um bereits erfasste Unternehmen besser hervorzuheben, werden auf der höchsten Zoomstufe zusätzliche Icons an der jeweiligen Position dargestellt. Die Positionsdaten für die Unternehmen werden über Overpass (siehe 5.4.1.5 Overpass API) geladen. Dafür wird jedes Mal, wenn der User die aktuelle Position ändert, eine Query abgesetzt die für den aktuellen Kartenausschnitt alle Nodes mit den entsprechenden Tags (siehe 5.4.5 OSM-Tags / Kategorien) lädt. Um zu vermeiden, dass diese Query zu oft ausgeführt wird, wird diese um 400 Millisekunden debounced. Das heisst, es wird bei jeder Positionsänderung erst 400 Millisekunden gewartet, ob es noch eine weitere Änderung gibt. Erst wenn die Karte 400 Millisekunden an der gleichen Position geblieben ist wird geladen. Das Intervall von 400 Millisekunden wurde so gewählt, dass es sich trotzdem noch flüssig anfühlt, aber den Server nicht überlastet. Aus den geladenen Daten werden die Positionen sowie die bestehenden OSM-Tags ausgelesen, daraus wird dann ein interaktives Kartenelement in der Form eines blauen Kreises um die Position erstellt. Bei einem Klick auf ein solches Element wird der Namen sowie die Adresse in einem Popup dargestellt. Die Adresse wird, erst wenn das Element angeklickt wird, per Nominatim Reverse Geocoding (siehe 5.4.1.4 Nominatim API) ermittelt. Das Popup bietet die Option, das Unternehmen zu bearbeiten, sowie Links zur OSM Karte und zur OSM Ansicht um Kartenfehler zu melden.

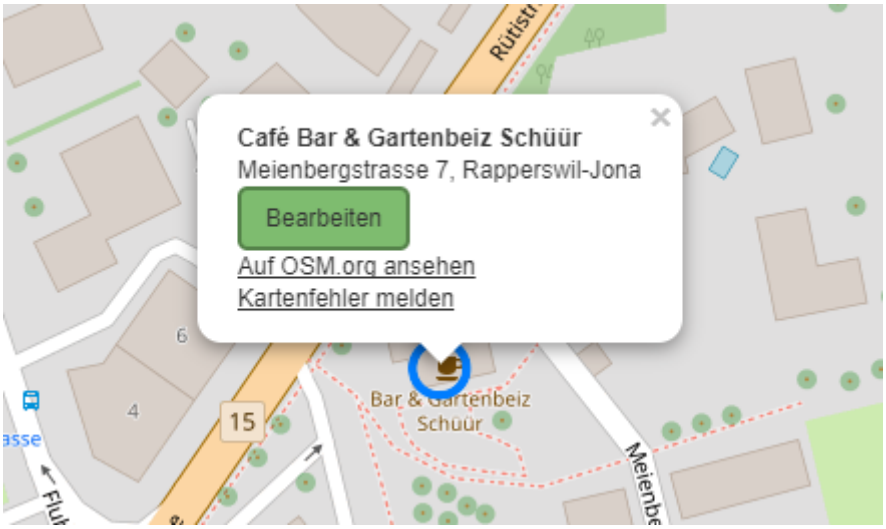


Abbildung 25 - Interaktion mit bestehendem Unternehmen

5.4.3.2 Adresssuche

Um direkt an eine Position auf der Karte zu springen, gibt es eine Suchleiste mit der nach Adressen gesucht werden kann. Die Adresssuche ist über Nominatim (siehe 5.4.1.4 Nominatim API) implementiert. Für einen beliebigen Suchstring werden die besten fünf Treffer geladen. Die geladenen Daten beinhalten dann die Koordinaten, sowie die vollständigen Adressen für alle Treffer. Um die Suche besser für den User nutzbar zu machen, werden während der Eingabe die besten Treffer bereits als Auswahl zur Verfügung gestellt. Dafür wird bei jeder Änderung im Suchbegriff eine neue Anfrage an Nominatim gestellt. Ähnlich wie der Overpass (siehe 5.4.1.5 Overpass API) Query wird auch hier ein Debounce von 400 Millisekunden verwendet. Zusätzlich wird sichergesellt, dass maximal ein Request pro Sekunde gesendet wird, um den Fair-Use Kriterien von Nominatim zu entsprechen. Wenn der User eine Option anwählt, springt die Karte zu der gewählten Position und wechselt auf die höchste Zoomstufe.

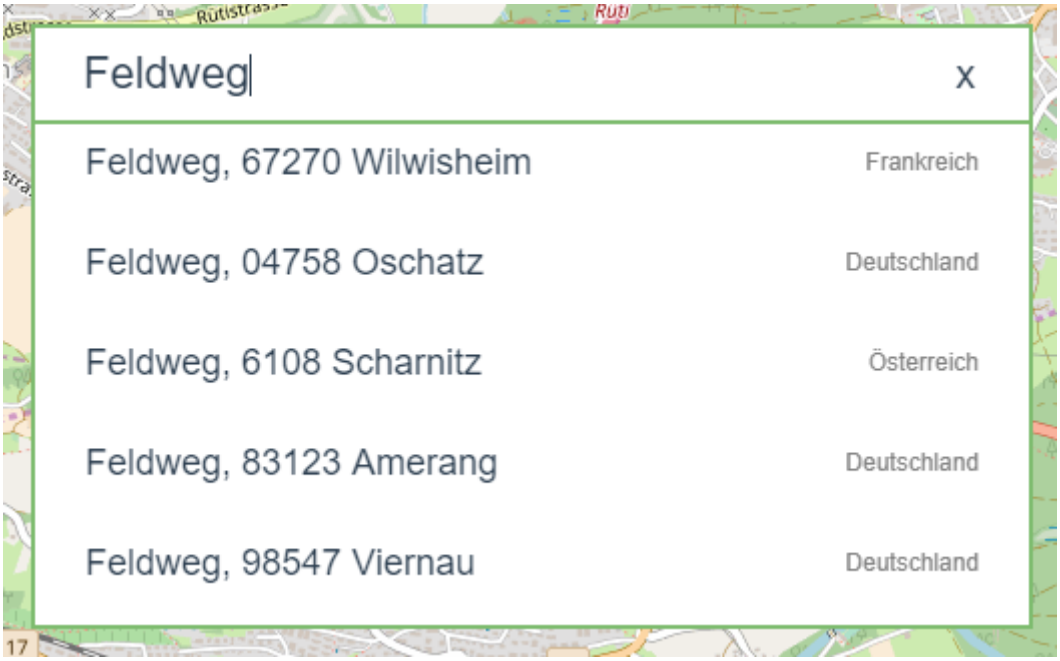


Abbildung 26 - Adresssuche mit Vorschlägen

5.4.3.3 Ein Unternehmen bearbeiten oder erstellen

Um ein bestehendes Unternehmen zu bearbeiten, muss der User das Unternehmen anwählen und danach im Popup auf bearbeiten klicken. Falls für das gewählte Unternehmen noch ein Update ausstehend ist, wird dem User eine Warnung angezeigt (dazu mehr im Kapitel 5.4.7 Duplikate).

Um ein neues Unternehmen zu erfassen, kann man auf der Karte Rechtsklicken. Ähnlich wie beim Bearbeiten wird wieder ein Popup dargestellt. Sobald man im Popup den Button zum Erstellen drückt, gelangt man in die Detailansicht.

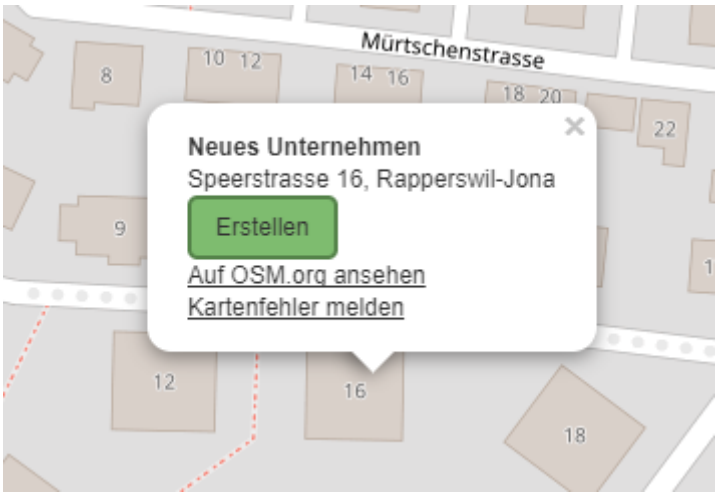


Abbildung 27 - Neues Unternehmen erstellen

5.4.3.4 Hilfestellung

Um unerfahrenen Usern mit dem Kartenumgang zu helfen, werden beim ersten Seitenbesuch kurze Hilfstexte angezeigt, sobald diese einmal geschlossen wurden, werden diese für zukünftige Besuche versteckt. Man kann sie jedoch durch einen Klick auf den Fragezeichen Button, unten links, jederzeit wieder anzeigen lassen.

Abbildung 28 - Hilfetext zur Benutzerführung

5.4.3.5 Update Notifikationen

Um den User über Änderungen an bereits bearbeiteten oder neu erstellten Unternehmen auf dem Laufenden zu halten, werden alle vom User bearbeiteten Knoten mit ihrer aktuellen Versionsnummer abgespeichert. Alle diese Knoten werden im Interface auf jeder Zoomstufe durch einen orangen Kreis hervorgehoben.

Bei erneuten Besuchen von OSMyBiz werden die Knoten über die OSM API geladen und mit der gespeicherten Version verglichen. Falls der Knoten nicht mehr gefunden wird, wird dem User in der Updateliste eine Notifikation angezeigt. Falls der Knoten noch existiert und die Versionsnummer erhöht wurde, was bedeutet, dass sich etwas am Knoten geändert, wird dem User ebenfalls eine Notifikation dargestellt.

Die Notifikationen können vom User als gelesen markiert werden, dies führt dazu, dass der Knoten in der OSMyBiz Datenbank mit der neuen Versionsnummer überschrieben wird. Wen dieser entfernt wurde, wird der Knoten aus unserer Datenbank gelöscht. Zudem kann über die Notifikation direkt an die Position des Unternehmens gezoomt werden. Im Fall einer Updatenotifikation gibt es zusätzlich noch einen Link auf das OSM Changeset, sowie die Option die Notifikationen über diesen Knoten abzuschalten.

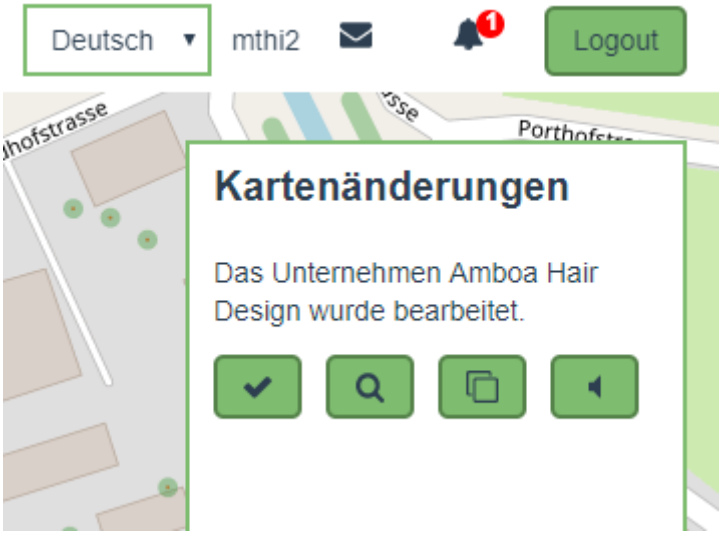


Abbildung 29 - Update Notifikation

5.4.4 Detail / Formular

Die Detailansicht besteht aus mehreren Feldern für die Daten des Unternehmens.

5.4.4.1 Kategoriefeld

Abbildung 30 - Detail - Kategorie

Als erstes wählt man die Kategorie mittels eines Dropdowns. Das Dropdown wird mit OSM-Tags gefüllt (siehe 5.4.5 OSM-Tags / Kategorien). Findet man seine Kategorie in der Liste nicht, hat man die Möglichkeit selber eine Kategorie zu definieren. Im Hintergrund wird dann aber eine Notiz, anstelle eines Knoten, erstellt, da die Kategorien im Key=Value-Format gesendet werden müssen. Dies verlangt vom Benutzer einiges an Erfahrung und die Gefahr das Fehler passieren ist damit höher.

5.4.4.2 Adressfelder

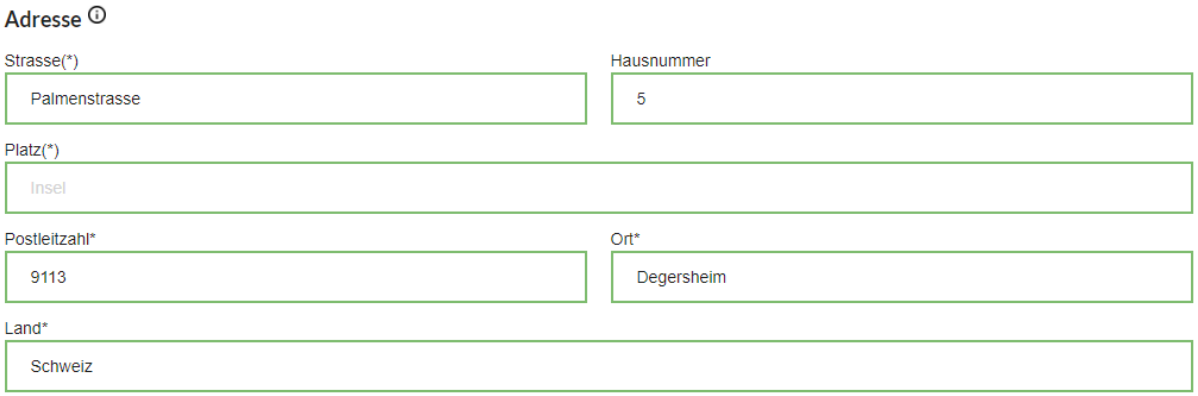


Abbildung 31 - Detail - Adresse

Nach der Kategorie kommen die Adressfelder. Diese werden mit den Daten aus Nominatim (siehe 5.4.1.4 Nominatim API) gefüllt. Der Benutzer kann diese noch anpassen, falls etwas nicht stimmt. Zusätzlich zur Strasse gibt es noch ein Feld für den sogenannten Platz. Es gibt Adressen die keine genaue Strassenangabe haben, da kommt dieses Feld zum Zug. Es kommt zum Beispiel vor, dass das Dorf so klein ist, dass man die Adresse mit Dorfname + Nummer angibt.

5.4.4.3 Detailinformationsfelder

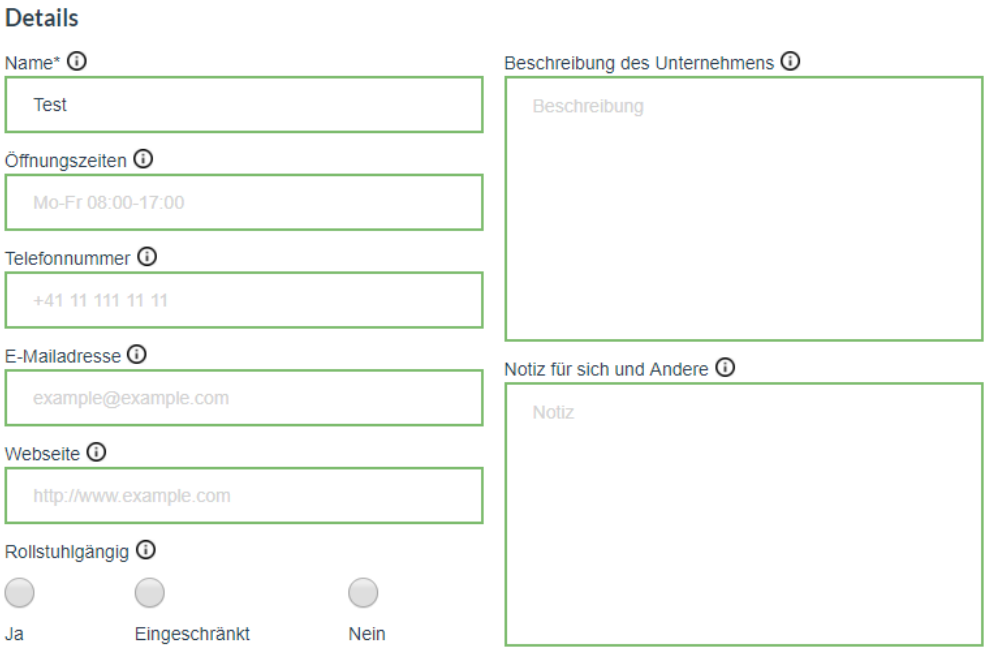


Abbildung 32 - Detail - Details

Danach hat es mehrere Inputfelder und eine Checkbox, ob das Unternehmen rollstuhlgängig ist oder nicht. Zwingend erforderlich ist dabei nur der Name, der Rest ist optional. Wenn notwendig, wird das Feld mittels Vee-Validate (38), einem Plugin für VueJS, auf Syntaxfehler geprüft.

Inputfeld	**Beschreibung**
Name	Name des Unternehmens z.B. Migros
Öffnungszeiten	Öffnungszeiten des Unternehmens z.B. Mo-Fr 07:00 - 12:00, 13:00 - 17:00
Telefonnummer	Telefonnummer unter der man das Unternehmen erreichen kann z.B. +41 71 111 11 11
E-Mail	E-Mailadresse unter der man das Unternehmen erreichen kann z.B. example@example.com
Webseite	Webseite des Unternehmens z.B. www.example.com
Beschreibung	Zusätzliche Informationen über das Unternehmen z.B. Bäckerei ist auch eine Poststelle
Notiz	Eine Notiz dient dazu, Informationen zum Unternehmen festzuhalten, die nicht offensichtlich sind, die die Intention des Autors genauer beschreiben oder die Hinweise zur weiteren Bearbeitung geben

Tabelle 13 – Statische Inputfelder

5.4.4.4 Zusatzinformationsfelder

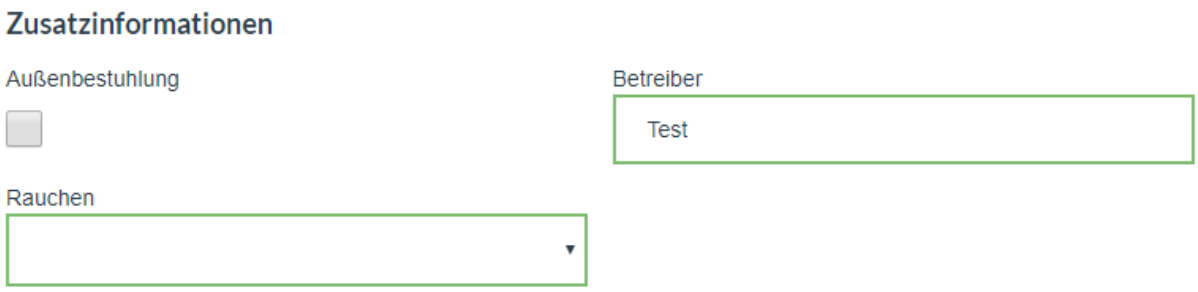


Abbildung 33 - Detail - Zusatzinformationen

Um der Anforderung von individuellen Zusatzinformationen, wie zum Beispiel Sterne bei einem Hotel, gerecht zu werden, werden diese nach der Auswahl der Kategorie dynamisch eingeblendet. Bei einer Bar wären das zum Beispiel: Aussenbestuhlung, Betreiber und Rauchen.

5.4.4.5 Informations-Popup

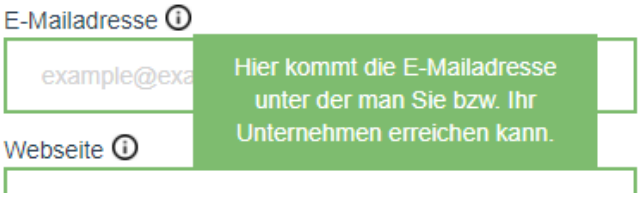


Abbildung 34 - Detail - InfoPopup

Um die Benutzbarkeit zu fördern werden Informationen für die verschiedenen Felder zur Verfügung gestellt. Der Benutzer kann diese ansehen indem er mit der Maus über das Icon mit dem «i» fährt.

5.4.5 OSM-Tags / Kategorien

Um dem Benutzer möglichst viel Benutzerfreundlichkeit zu bieten, war es sinnvoll die Kategorien für das Unternehmen als Optionen anzuzeigen. Somit kann der Benutzer aus der Liste auswählen und eine einheitliche Bezeichnung ist gewährleistet.

Da OSM keine API für diese Aufgabe (Auslesen der Namen der Tags) anbietet, wurden verschiedene Ansätze probiert. Angefangen mit dem Auslesen von HTML-Tabellen auf der Map-Features-Seite (39) von OSM, die alle vordefinierten Tags enthält. Die Struktur der Seite ist aber nicht für alle Tags gleich und alle auszulesen somit sehr schwierig. Dieser Ansatz wurde relativ schnell verworfen, da er einfach zu aufwendig und zu ungenau ist. Auch wäre es schwierig oder unmöglich weitere Informationen zu den Kategorien, wie die Fields, auszulesen.

Bei den anderen beiden Ansätzen ging es jeweils um sogenannte Presets. Ein Preset beschreibt wie eine Art Objekt aussieht, das heisst zum Beispiel wie es heisst, welche Felder es hat und welche Tags. Es gibt zwei bekannte Editoren, die mit solchen Presets arbeiten, der iD- und der JOSM-Editor. Es wurde für je ein Ansatz mit den Presets dieser Editoren gearbeitet.

Angefangen beim JOSM-Editor. Dieser arbeitet mit dem Dateiformat XML. Es gibt dabei viele verschiedene Anbieter dieser XMLs. Simon Poole stellt seine überarbeitete Version der Presets auf GitHub (40) bereit. Das XML musste noch bearbeitet und in ein JSON umgewandelt werden. Dabei mussten erst alle unnötigen Tags gelöscht werden, da in der Liste ja nur Unternehmen enthalten sein mussten. Die Probleme bei diesem Ansatz waren, dass die Tags in einem XML gespeichert sind und man erst alles umwandeln musste und die Daten auch nur auf Englisch verfügbar sind.

```
<item name="Hotel"
      icon="https://raw.githubusercontent.com/simonpoole/beautified-JOSM-preset/master/icons/png/accommodation_hotel2.png"
      type="node,closedway,multipolygon" preset_name_label="true">
  <link href="http://wiki.openstreetmap.org/wiki/Tag:tourism=hotel"
        cs.href="http://wiki.openstreetmap.org/wiki/Cs:Tag:tourism=hotel"
        de.href="http://wiki.openstreetmap.org/wiki/DE:Tag:tourism=hotel"
        es.href="http://wiki.openstreetmap.org/wiki/ES:Tag:tourism=hotel"
        fi.href="http://wiki.openstreetmap.org/wiki/Fi:Tag:tourism=hotel"
        fr.href="http://wiki.openstreetmap.org/wiki/FR:Tag:tourism=hotel"
        ja.href="http://wiki.openstreetmap.org/wiki/JA:Tag:tourism=hotel"
        pl.href="http://wiki.openstreetmap.org/wiki/Pl:Tag:tourism=hotel"
        pt.href="http://wiki.openstreetmap.org/wiki/Pt:Tag:tourism=hotel"
        ru.href="http://wiki.openstreetmap.org/wiki/RU:Tag:tourism=hotel"/>
  <space/>
  <key key="tourism" value="hotel"/>
  <reference ref="name_operator"/>
  <reference ref="stars"/>
  <text key="rooms" text="Rooms"/>
  <text key="beds" text="Beds"/>
  <reference ref="wheelchair"/>
  <space/>
  <reference ref="internet_smoking"/>
  <space/>
  <reference ref="link_contact_address_payment"/>
</item> <!-- Hotel -->
```

Abbildung 35 - Item als XML (JOSM, Simon Pooles Version)

Der iD-Editor hingegen arbeitet bereits mit JSON, was die Arbeit um vieles einfacher macht. Man kann das JSON mit allen Presets direkt über die GitHub-Seite (3) vom iD-Editor herunterladen. Auch hier müssen die unnötigen Tags gelöscht werden, aber die Struktur ist viel einfacher zu durchsuchen und man kann mit den verschiedenen relevanten Tag-Gruppen (Amenity, Shop, Leisure, Office und Tourism) die Tags besser filtern.

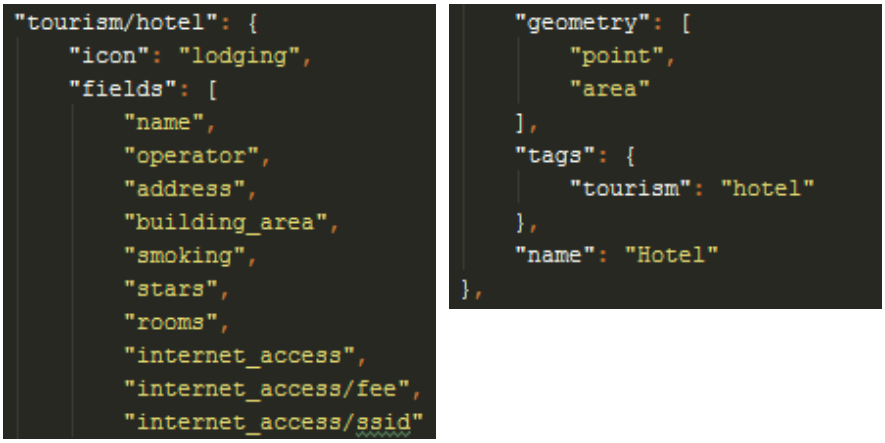


Abbildung 36 – Item als JSON (iD Editor)

Das Problem mit den Tag-Gruppen besteht darin, dass viele unnötige Kategorien angezeigt werden (z.B. Sitzbank). Die Offensichtlichsten werden rausgefiltert, trotzdem kann die eine oder andere noch angezeigt werden. Aber durch die Textsuche findet der Benutzer die richtige Kategorie bzw. kann selber eine Kategorie definieren.

Die Datei mit den Presets wird als JSON eingelesen und dann anhand der Gruppen gefiltert. Im gleichen Schritt werden auch gleich die unnötigen Kategorien rausgefiltert. Danach hat man jeweils Name und Felder der Kategorie.

Bei den Feldern hat es aber noch einige, die man schon als Inputfelder definiert hat oder die einfach nicht sehr sinnvoll sind. Diese werden ebenfalls gefiltert. Nur die Felder alleine sind aber zum Teil noch etwas verwirrend für den Benutzer. Der iD-Editor hat jedoch auch eine Datei mit den Optionen für die Felder, falls es welche hat.

Beim Ausführen des Skripts wird über die Kommandozeile die Sprache mitgegeben (z.B. de für Deutsch). Dabei werden alle Strings von den Namen der Felder bis zu den Optionen übersetzt. Dabei wird die JSON-Übersetzungsdatei vom iD-Editor verwendet und mit den gefilterten Presets abgeglichen. Dabei werden auch nur Kategorien und Felder übernommen, die in der jeweiligen Sprache existieren.

5.4.6 Benutzerfreundlichkeit

5.4.6.1 Fehlermeldungen

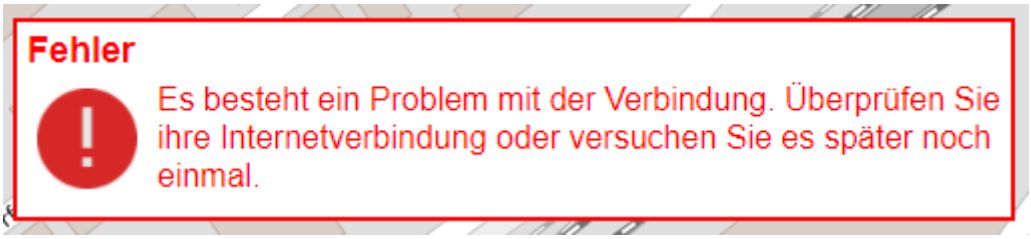


Abbildung 37 - Fehlermeldung

Die Chance ist relativ hoch, dass von Zeit zu Zeit eine der APIs oder sonst etwas nicht so funktioniert wie es sollte. Das Problem kann bei der API selber sein, an der Applikation oder auch an der Internetverbindung liegen.

Damit der Benutzer nicht klickt und klickt, ohne eine Rückmeldung, wurden dafür zum Teil spezifische und zum Teil generische Fehlermeldungen eingebaut.

5.4.6.2 Zusätzliche Formularknöpfe

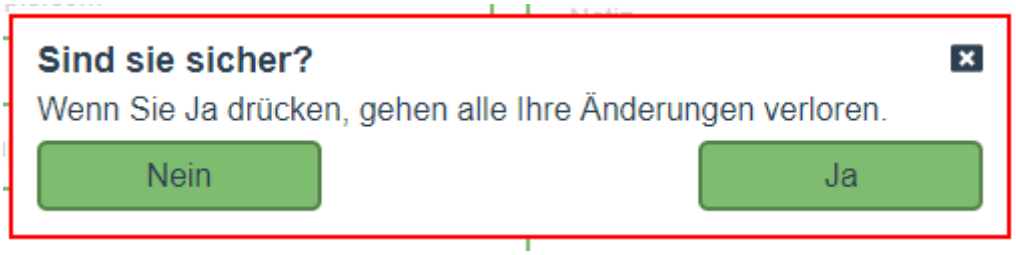


Abbildung 38 – Datenverlustwarnung

Zusätzlich zum Speichern gibt es noch zwei andere Knöpfe: Zurück und Zurücksetzen. Bei Zurück kann man abbrechen und zur Karte zurückkehren, bevor dies passiert wird der Benutzer jedoch nochmal darüber informiert, dass in diesem Fall seine Daten verloren gehen. Beim Zurücksetzen werden die Felder wieder in den Zustand gebracht indem sie zu Anfang waren. Auch hier wird der Benutzer über den Verlust der Daten informiert und muss dies bestätigen.

5.4.6.3 Erfolg

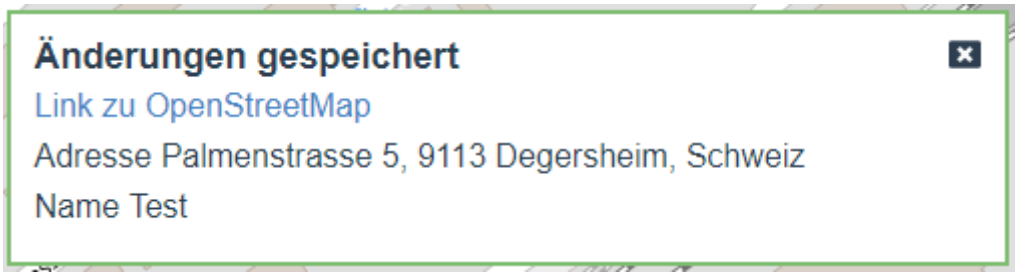


Abbildung 39 – Erfolgsmeldung

Nachdem das Unternehmen erstellt oder bearbeitet wurde, wird dem Benutzer noch eine Meldung angezeigt. Diese enthält die Adresse und den Namen als Kontrolle und einen Link der den Pfad zum Unternehmen auf OSM enthält.

Im Hintergrund werden die Daten anders ausgeliefert, je nachdem ob ein Unternehmen erstellt oder bearbeitet wurde. Der Knoten (erstelltes Unternehmen) muss von XML nach JSON umgewandelt und der Struktur von der View angepasst werden. Die Notiz (bearbeitetes Unternehmen) kommt zwar als JSON zurück, dieses enthält jedoch auch einfach wieder den String, welcher noch geparst werden muss, damit man Adresse und Name auslesen kann.

5.4.7 Duplikate

Um zu verhindern, dass an denselben Koordinaten zwei Mal das gleiche Unternehmen erstellt wird und nicht mehr als eine Notiz (Bearbeitung) gleichzeitig offen ist, werden solche Duplikate abgefangen.

Dabei wird unterschieden, ob es sich um ein neues Unternehmen (Knoten) oder um eine Änderung (Notiz) handelt.

Es wird aber bei beiden überprüft, ob sich um Umkreis von 10 Metern kein Unternehmen bzw. keine Notiz, mit derselben Kategorie und demselben Namen, befindet.

5.4.7.1 Notiz

Diese Überprüfung findet statt, sobald man in der Kartenansicht im Bearbeitungsmodus im Popup auf «Bearbeiten» drückt.

Über die OSM API wird ein Request abgesetzt, um alle Notizen zu holen, die sich im Suchradius befinden. Der Suchradius wird über eine BoundingBox festgelegt und ist somit eigentlich rechteckig. Die gefundenen Notizen werden dann alle durchsucht. Da jedoch die Notizen als String zurückkommen, müssen erst mit Javascript die Felder mit der Kategorie und dem Namen rausgefiltert werden.

Stimmen diese überein wird eine Warnung ausgelöst. Es wird noch der Link zusammengebaut, damit der Benutzer direkt das Unternehmen bzw. die Notiz ansehen kann. Er hat dann auch gleich die Möglichkeit einen Kommentar zu der Notiz zu hinterlassen, falls er noch weitere Änderungen hat.

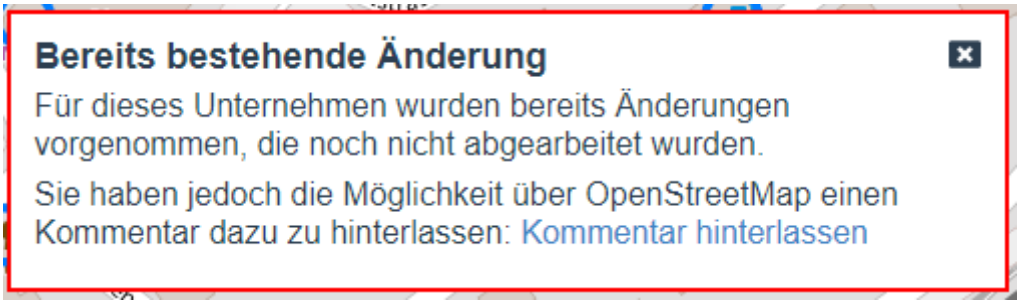


Abbildung 40 - Warnung Duplikat - Notiz

5.4.7.2 Knoten

Anders als bei der Notiz kann hier erst geprüft werden, wenn der Benutzer bereits Kategorie und Name des Unternehmens angegeben hat.

Beim Knoten läuft es dafür etwas einfacher ab. Die Overpass API (siehe 5.4.1.5 Overpass API) bietet eine Funktion an, mit der man in einem Umkreis nach Knoten suchen kann. Es wird eine Query mit den Koordinaten, der Kategorie und dem Namen zusammengebaut und mit dem Suchradius an Overpass übermittelt. Wenn mindestens ein Element zurückkommt, dann ist es ein Duplikat.

In der Meldung, dass dieses Unternehmen bereits besteht, hat es noch einen Hinweis, dass man über OSMyBiz auch das Unternehmen über eine Notiz bearbeiten kann.

Abbildung 41 - Warnung Duplikat - Knoten

5.4.8 Übersetzungen

Da OSM auf der ganzen Welt verfügbar ist und es auch auf der ganzen Welt Unternehmen gibt, die sich auf die Karte eintragen möchten, wurde eine Übersetzungsfunktionalität eingebaut. Diese ist momentan auf Deutsch und Englisch beschränkt, aber theoretisch beliebig erweiterbar.

Es wurde das Plugin VueTranslate (41) eingesetzt. VueTranslate bietet eine gute Basisfunktion, um Übersetzungen im HTML einzubauen. Man hat für jede Sprache ein eigenes JSON mit key=value, wobei der key in jeder Datei gleich sein muss, damit man diesen referenzieren kann.

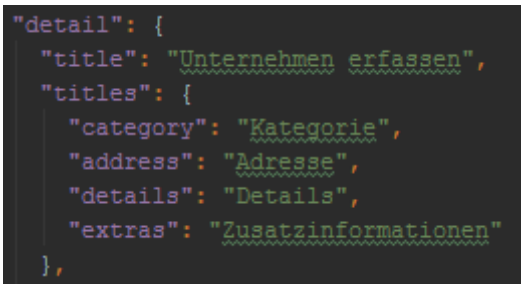


Abbildung 42 – Übersetzungsdatei

Die Übersetzungen in OSMyBiz beschränken sich aber nicht auf das HTML, es hatte auch einige Felder, die im Javascript-Teil erstellt wurden. Nur mit dem Plugin konnten diese also nicht übersetzt werden.

Es gab zwei Möglichkeiten:

1. Ein neues und besser geeignetes Plugin suchen und alles umschreiben
2. Das JSON mit den Übersetzungen abfangen und selber global zur Verfügung stellen

Vor allem aus Zeitgründen, da eine neue Suche und Evaluation von einem Plugin sehr aufwendig ist, wurde die zweite Variante gewählt.

Die Sprache kann in allen Ansichten im Header, über ein Dropdown, neu gewählt werden. Es wird jeweils direkt übersetzt ohne die Seite neu zu laden, damit keine Informationen verloren gehen.

Eine weitere Hürde bildeten die Kategorien (siehe 5.4.5 OSM-Tags / Kategorien). Diese werden ausserhalb des Frontends erstellt und hineingeladen. Man braucht also für jede Sprache eine eigene Datei. Diese wird ausgetauscht, sobald die Sprache neu gewählt wird.

5.4.9 OSM-Login

Damit ein User ein Unternehmen bearbeiten oder erfassen kann, muss er sich erst per OSM Konto auf OSMyBiz einloggen. Dafür wird die JavaScript Bibliothek OSM-OAuth (25) verwendet. Diese implementiert diverse Wrapper und Hilfsfunktionen um autorisierte Zugriffe auf die OSM API durchzuführen. Zur Konfiguration des OAuth Logins muss zuerst auf openstreetmap.org, über ein Benutzerkonto, eine neue Applikation erstellt werden. Für die Applikation muss die Applikations-URL, sowie die Callback-URL auf die URL von OSMyBiz gesetzt werden. Zusätzlich müssen folgende Genehmigungen erteilt werden:

1. Benutzerdetails auslesen
2. Karte ändern
3. Hinweise geben / Fehler melden

Nach erfolgreichem Erstellen der Applikation werden der Key und das Secret generiert. Diese müssen in der Konfigurationsdatei von OSMyBiz hinterlegt werden.

Nach einem Klick auf Login wird der User nun auf die OSM Login-Seite weitergeleitet, wo er bestätigen muss, dass er OSMyBiz die oben genannten Genehmigungen erteilen will. Der User wird dann anhand der Callback-URL zu OSMyBiz zurückgeschickt. Dort wird nun das Autorisierungstoken aus der URL gelesen und im OSM-OAuth Objekt für zukünftige autorisierte Requests hinterlegt.

Nachdem sich ein User eingeloggt hat, wird anstelle des Login-Buttons, der Logout-Button angezeigt. Zusätzlich wird dem User über das Brief Icon die Anzahl an ungelesenen Nachrichten in seinem OSM-Konto angezeigt. Ein Klick darauf leitet ihn direkt zu seiner OSM Inbox weiter.

5.4.10 Backend

Das Backend ist eine sehr simple Web API, die es ermöglicht eingeloggte User und Unternehmen, die erstellt oder bearbeitet wurden, abzuspeichern. Jedes Mal, wenn sich ein User einloggt, wird er per OSM-ID und OSM Benutzernamen im Backend abgespeichert. Identifiziert wird der User anhand seiner OSM-ID. Nachdem ein neues Unternehmen erfasst oder ein bestehendes modifiziert wurde, wird der betroffene OSM Knoten ebenfalls zusammen mit dem aktuell eingeloggten User persistiert. Für den Knoten werden lediglich die OSM-ID, der Name, die Position, die aktuelle Versionsnummer und ein Flag, ob der User über Änderungen am Knoten benachrichtigt werden will, gespeichert. Bei einem erneuten Besuch von OSMyBiz werden für den existierenden User die eingetragenen Knoten geladen.

Das Backend bietet folgende Endpunkte an:

Pfad	Methode	Beschreibung
/user	POST	Speichert einen User
/user/#id/node	POST	Speichert einen Node für den User mit der entsprechenden ID
/user/#id/node	GET	Lädt alle Nodes für den entsprechenden User
/user/#id/node/ #nodeId/delete	POST	Löscht den entsprechenden Node für den User
/user/#id/node/ #nodeId/unsubscribe	POST	Stellt zukünftige Notifikationen für den User über den entsprechenden Node ab

Tabelle 14 - OSMyBiz API Pfade

Die genaue Spezifikation ist im Anhang als Swaggerdatei hinterlegt.

5.4.11 Konfigurationen

Damit man die Tokens und URLs besser austauschen kann, wurden diese in eine eigene Datei ausgelagert.

Konfiguration	Beschreibung
MapBoxToken	Token für Satellitenbilder
InitialPosition/InitialZoom	Anfangsposition und -zoom auf der Karte
OSM API Level	OSM API Version (aktuell 0.6)
OSM API URL	URL der OSM API (Eine für die Entwicklung und eine für die Produktion)
OAuth Secret & Key	Secret und Key für das Anmelden bei OSM (Einmal für die Entwicklung und einmal für die Produktion)
Nominatim API URL & Reverse URL	URL für Nominatim API und Reverse URL für Reverse Geosearch
Overpass API URL	URL für Overpass API
Search-Radius	Suchradius für Duplikate
OSMyBiz Backend URL	URL für das Backend

Tabelle 15 - Konfigurationen

5.4.12 Security & Datenschutz

Die einzigen sicherheitskritischen Daten, die an den Client geliefert werden, sind die OSM OAuth Keys, sowie das Mapbox Token. Diese werden beide in den Client hineinkompiliert und anschliessend minified, könnten aber nach wie vor über die Developer Tools ausgelesen werden.

Mit diesen Tokens könnte sich eine andere Applikation als OSMyBiz ausgeben und Benutzer im Namen von OSMyBiz zum einloggen auffordern. Da der Login selbst jedoch wieder über OSM abgewickelt wird und kritische Daten wie Passwörter und Email-Adressen nie an OSMyBiz weitergeleitet werden, stellt dies nur eine minimale Bedrohung dar. Zudem ist das Erstellen von autorisierten Applikationen auf OSM frei für jeden zugänglich.

Die Nutzerdaten die im OSMyBiz Backend persistiert werden, sind Teil des OSM Datenstamms und somit öffentlich verfügbar. Damit besteht in Bezug auf Datenschutz keine Gefahr.

5.5 Weiterentwicklung

Im Verlauf der Arbeit sind viele Ideen für weitere Features entstanden. Da OSMyBiz nach dem Abschluss dieser Arbeit als Open Source Projekt weiterentwickelt wird, werden diese Ideen in die Weiterentwicklung mit einfliessen.

5.5.1 Spam-Filter

Da OSMyBiz neue Unternehmen direkt in die Stammdaten von OSM einfügt, müssten noch zusätzliche Checks implementiert werden um Spameinträge zu vermeiden. Die wichtigsten Checks für duplizierte Daten mit denselben Koordinaten sind zwar bereits implementiert, aber es ist nach wie vor möglich das gleiche Unternehmen an verschiedenen Orten einzutragen.

5.5.2 Eingabe von Öffnungszeiten

Aktuell können Öffnungszeiten wie auf OSM selbst nur als Freitext erfasst werden. Besser wäre es, wenn es dafür ein dediziertes Interface gäbe, welches ermöglicht, die Öffnungszeiten in einem klar definierten Format zu bearbeiten.

Die Empfehlung wäre dabei auf den Code von YoHours (42) zurückzugreifen. Dieser bietet bereits die Funktionalität an, über eine Benutzerschnittstelle, die Zeiten auszuwählen und erstellt daraus einen strukturierten Text.

5.5.3 History

Momentan erhalten die eingeloggten User zwar Benachrichtigung, wenn sich etwas an ihren Unternehmen geändert hat, jedoch gibt es keine Möglichkeit für User eine Liste aller dieser Unternehmen einzusehen. Dieses Feature war bereits von Anfang an eingeplant, wurde jedoch aus Zeitgründen weggelassen. Für die Implementierung müsste man eine neue Seite erstellen, die nur für eingeloggte Benutzer zugänglich ist. Auf dieser Seite würde eine Liste mit allen Unternehmen dargestellt werden.

5.5.4 Mehr Sprachen

OSMyBiz ist so aufgebaut, dass neue Sprachen mit sehr wenig Aufwand hinzugefügt werden können. Es muss lediglich eine JSON-Datei für die neue Sprache hinterlegt werden und die Kategorie Daten aus den Presets für die neue Sprache generiert werden.

5.5.5 Plus Codes

Plus Codes (43) sind eine von Google entwickelte Art einen Ort eindeutig einem String zuzuweisen. Ein global einzigartiger Plus Code ist elf Zeichen lang, alternativ kann auch ein Ortsname und die letzten sieben Zeichen zur Identifikation benutzt werden. Für OSMyBiz wäre es gut, zusätzlich zur Suche per Adresse auch die Suche nach Plus Codes zuzulassen.

5.5.6 Notifikationen

Die Notifikationen für den Benutzer sind fast beliebig ausbaubar. Jedoch sollte man immer noch den Überblick behalten können und viele Informationen sind für den Benutzer schlicht nicht relevant. Auch haben zu viele Notifikationen einen negativen Einfluss auf die Performance.

Zwei Beispiele die man noch einbauen könnte:

Benutzer informieren, wenn

- Unternehmen in seiner näheren Umgebung erstellt, bearbeitet oder gelöscht wurden
- neue Unternehmen mit gleicher Kategorie im gleichen Ort erstellt, bearbeitet oder gelöscht wurden

5.6 Qualitätsmanagement

5.6.1 Besprechungen

Jede Woche, jeweils donnerstags oder freitags, wurde mit Professor Keller eine Besprechung durchgeführt. Zum einen um ihn über den aktuellen Stand zu informieren, aber auch um offene Fragen zu klären und die nächsten Schritte zu diskutieren.

5.6.2 Pair Programming

Die Entwicklung wurde durch Pair Programming vorangetrieben. Da beide Entwickler an zwei Tagen in der Woche zusammen am selben Ort waren, konnte das auch gut umgesetzt werden. So konnte man alles direkt diskutieren und einander bei Problemen helfen. Für die restliche Zeit wurden die Arbeiten aufgeteilt und wenn notwendig über Kommunikationskanäle ausgetauscht.

5.6.3 Review

Beim Mergen eines Feature Branches hat jeweils der andere Entwickler den Pull Request geprüft. Nach einem Test des Features wurde der Merge durchgeführt.

5.6.4 Testing

Für unsere Applikation werden nach Fertigstellung der Funktionen Unittests geschrieben.

Um die Tests in regelmässigen Abständen laufen zu lassen, verwenden wir Pipelines von GitLab CI. Diese lassen die Tests für jeden Commit durchlaufen. Der Feature Branch lässt sich auch nur mergen, wenn die Pipeline keine Fehler hat.

5.6.5 Coding Guidelines

Sprache	Guideline
Javascript / VueJS	ESLint (44) mit Defaulteinstellungen von VueJS
Python / Flask	Python Style Guide PEP-8 (45)
Docker	Best Practices von Docker (46)

Tabelle 16 - Coding Guidelines

5.7 Projektmanagement

5.7.1 Projektorganisation

Rolle	Verantwortlichkeit
Professor Stefan Keller	Projektbetreuung, Beratung
Nicola Jordan	Support, Beratung
Max Lüthi	Entwickler
Simon Heller	Entwickler

Tabelle 17 - Projektorganisation

5.7.2 Risiken

Risiko	Vorbeugung	Verhalten beim Eintreten
Fehlendes Knowhow	Lernzeit für Technologien fest einplanen	Frühzeitig Hilfe suchen beim Teammitglied oder Betreuer
Technologie erfüllt die Anforderungen nicht	Technologien evaluieren und mit Anforderungen prüfen	Mögliche Integration zusätzlicher Technologie oder Funktionsstreichung in Betracht ziehen
OSM/APIs/ Development-Umgebung nicht erreichbar	-	Mit OSM kommunizieren, Features soweit wie möglich implementieren
Feature nicht umsetzbar, da nicht mit OSM kompatibel	Vor der Implementation APIs studieren und sich gründlich informieren	Mit OSM-Community kommunizieren und eventuell Feature weglassen

Tabelle 18 – Risiken

5.7.3 Issues

Hier werden eingetretene Risiken als Issues beschrieben.

Datum	Issue	Massnahmen
01.12.2017	Development Umgebung von OSM teilweise nicht funktionsfähig (Nodes können nicht mit GET gefunden werden)	GET auf Produktions-Umgebung umgeleitet

Tabelle 19 - Issues

5.7.4 Zeitplan

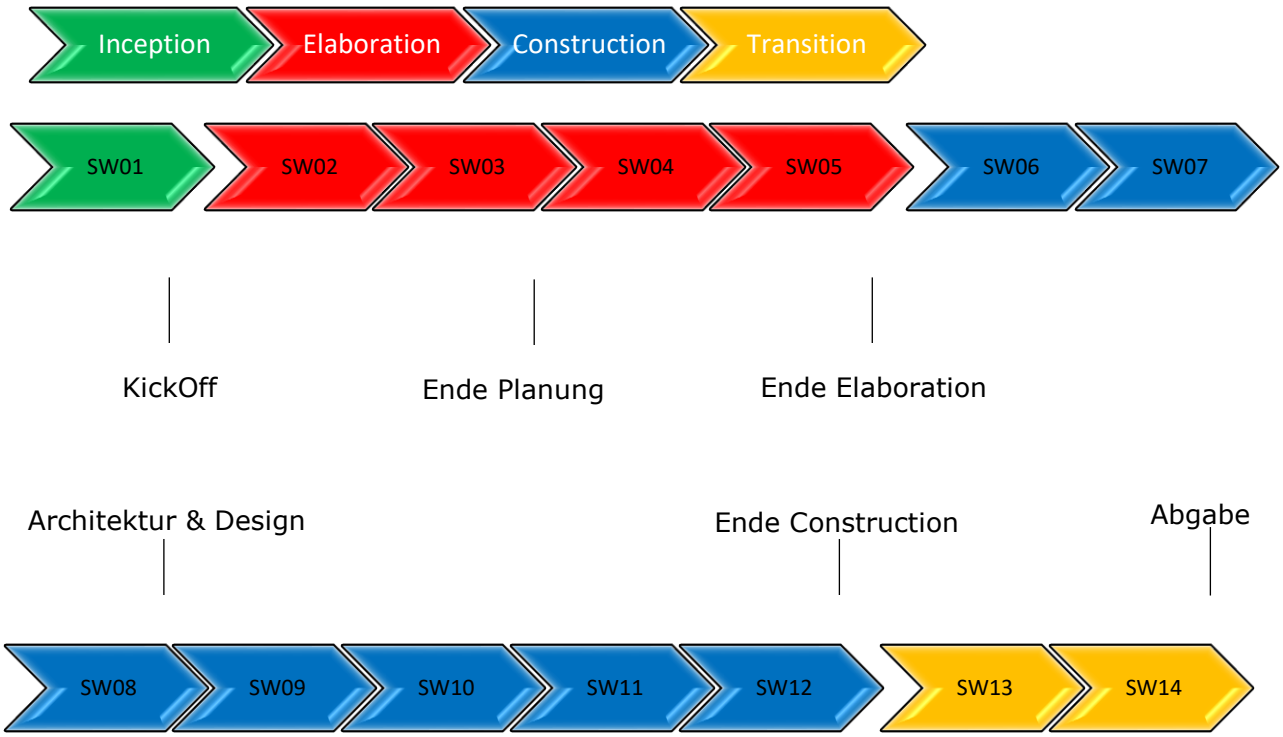


Abbildung 43 - Zeitplan mit Phasen

Nach der Inception-Phase wurde mit zweiwöchigen Iterationen gearbeitet. Bei Beginn der Iteration wurden die Tasks aus dem Backlog ausgewählt und der Iteration zugewiesen.

5.7.5 Meilensteine

Meilenstein	Datum	Beschreibung
M_01 – Kick-off	22.09.2017	Erstes Meeting und Einarbeitung in das Thema
M_02 – Ende Planung	6.10.2017	Aufgabenstellung finalisiert, Anforderungen definiert, Use Cases brief, Domainmodell, Sequenz-/ Aktivitätsdiagramm, Projekt-Setup
M_03 – Ende Elaboration	20.10.2017	Prototyp fertig (Minimal Use Case Unternehmen erfassen fertig), Anforderungen / Use Cases finalisiert, UI Mockups, Werkzeuge und Technologien gelernt, Architektur-Entwurf erstellt
M_04 – Architektur & Design	10.11.2017	Prototyp weiterentwickelt, Architektur dokumentiert
M_05 – Ende Construction	8.12.2017	Code freeze, alle Features eingebaut
M_06 - Abgabe	22.12.2017	Alle Bugs gefixt, Bericht vollständig fertig, Poster und Abstract erstellt

Tabelle 20 - Meilensteine

5.8 Projekt-Monitoring

5.8.1 Soll-Ist-Zeitvergleich

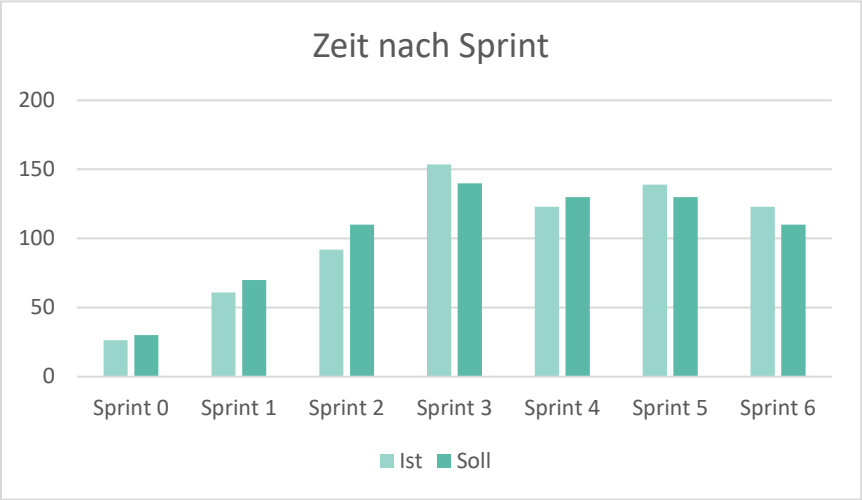


Abbildung 44 - Zeit nach Sprint

In den ersten drei Sprints war man etwas unter dem geplanten Soll, da aufgrund der offenen Aufgabenstellung etwas zu hoch geschätzt wurde. Im Sprint 3 sind dann die Änderungen gekommen, dass man Unternehmen direkt als Knoten abspeichert. Da mussten einige Änderungen vorgenommen werden, was Zeit gekostet hat.

Am Schluss wurden noch so viele Funktionen wie möglich eingebaut und das Erstellen des Berichts hat auch einige Zeit beansprucht.

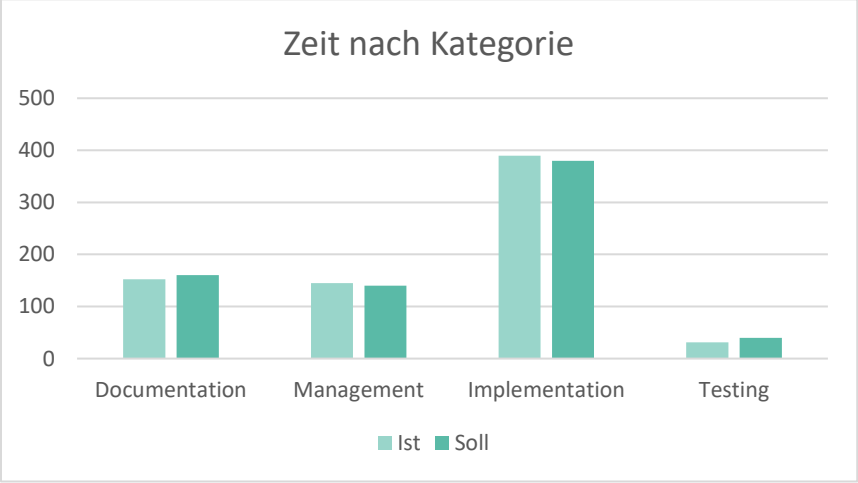


Abbildung 45 - Zeit nach Kategorie

Die Zeiterfassung wurde in vier Kategorien unterteilt. Die Verteilung zwischen Soll und Ist ist dabei ziemlich gleich. Management, also Meetings, etc., haben etwas mehr Zeit gebraucht, was auch den Änderungen in der Mitte des Projekts zuzuschreiben ist. Auch die Implementation hat etwas mehr Zeit benötigt. Das Testing dafür weniger, da das Backend sehr klein geblieben ist.

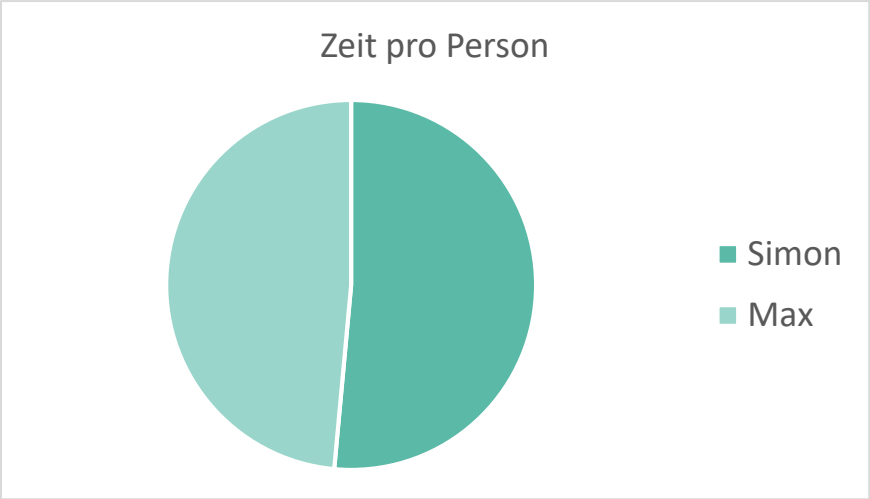


Abbildung 46 - Zeit pro Person

Der Zeitaufwand pro Person ist sehr ähnlich. Die Entwickler haben auch oft zusammen in der Schule an dem Projekt gearbeitet. Simon hat etwas mehr Zeit benötigt, da er auch etwas weniger praktische Erfahrung aufweist als Max.

5.8.2 Aufgabenteilung

Person	**Bereiche**
Max Lüthi	Kartenansicht, OSM-Login, Update Notifikationen, Backend
Simon Heller	Formular, OSM-Tags / Kategorien, Dialoge, Übersetzungen, Überprüfung Duplikate

6 Glossar

Begriff	Erklärung
API	Application Programming Interface – Programmierschnittstelle
CI	Continious Integration – Fortlaufendes Zusammenfügen von Komponenten einer Anwendung
CLI	Command-Line Interface – Interaktion mit einem Programm über die Kommandozeile
DOM	Document Object Model – Programmierschnittstelle, welche HTML- oder XML-Dokumente als Baumstruktur darstellt
JSON	JavaScript Object Notation – Dateiformat für Datenaustausch
JSX	JavaScript Extension Syntax
MVC	Model – View – Controller – Muster zur Trennung von Software in drei Komponenten
OSM	OpenStreetMap
OSMyBiz	OpenStreetMap My Business
UI	User Interface – Benutzeroberfläche, die Interaktion zwischen Benutzer und System ermöglicht
XML	Extensible Markup Language – Auszeichnungssprache zur Darstellung hierarchisch strukturierter Daten

7 Verzeichnisse

7.1 Abbildungsverzeichnis

7.2 Tabellenverzeichnis

7.3 Literaturverzeichnis

1. **On OpenStreetMap.** ***onosm.org.*** **[Online]**

2. OSM Editoren. ***http://wiki.openstreetmap.org/wiki/Editors.*** **[Online]**

3. iD-Editor. ***https://github.com/openstreetmap/iD.*** **[Online]**

4. OpenStreetMap (Official Website). ***openstreetmap.org.*** **[Online]**

5. JOSM. ***https://josm.openstreetmap.de/.*** **[Online]**

6. Maps.me. ***https://mapsme.de/.*** **[Online]**

7. Google My Business. ***https://www.google.ch/intl/de/business/.*** **[Online]**

8. Why OpenStreetMap.
http://wiki.openstreetmap.org/wiki/Why_OpenStreetMap%3F. **[Online]**

9. Python. ***https://www.python.org/.*** **[Online]**

10. Django Rest Framework. ***http://www.django-rest-framework.org/.*** **[Online]**

11. Flask. ***http://flask.pocoo.org/.*** **[Online]**

12. Hackernoon. ***https://hackernoon.com/debate-vuejs-vs-reactjs-9d2a8867b69b.*** **[Online]**

13. Vuejs Comparison. ***https://vuejs.org/v2/guide/comparison.html.*** **[Online]**

14. Deliciousbrains. ***https://deliciousbrains.com/vue-vs-react-battle-javascript/.*** **[Online]**

15. VueJS. ***https://vuejs.org/.*** **[Online]**

16. ReactJS. ***https://reactjs.org/.*** **[Online]**

17. Angular 2. ***https://angular.io/.*** **[Online]**

18. Typescript. ***http://www.typescriptlang.org/.*** **[Online]**

19. EcmaScript 6. ***http://www.ecma-international.org/ecma-262/6.0/.*** **[Online]**

20. PostgreSQL. ***https://www.postgresql.org/.*** **[Online]**

21. Pytest. ***https://docs.pytest.org/en/latest/.*** **[Online]**

22. Mocha. ***http://mochajs.org/.*** **[Online]**

23. Karma. ***http://karma-runner.github.io/1.0/index.html.*** **[Online]**

24. OSMyBiz Notes. ***https://github.com/mthi/osmybiz-notes.*** **[Online]**

25. GitHub OSM-Auth. ***https://github.com/osmlab/osm-auth.*** **[Online]**

26. Redux Pattern. ***https://redux.js.org/.*** **[Online]**

27. Vuex. ***https://vuex.vuejs.org/en/intro.html.*** **[Online]**

28. Flask SQL Alchemy. ***http://flask-sqlalchemy.pocoo.org/2.3/.*** **[Online]**

29. Richardson Maturity Model. ***https://www.martinfowler.com/articles/richardsonMaturityModel.html.*** **[Online]**

30. Docker. ***https://www.docker.com.*** **[Online]**

31. Nginx. ***https://nginx.org/.*** **[Online]**

32. Mapbox. ***https://www.mapbox.com/.*** **[Online]**

33. OSM API. ***http://wiki.openstreetmap.org/wiki/API.*** **[Online]**

34. OAuth. ***https://oauth.net/.*** **[Online]**

35. Nominatim. ***http://www.nominatim.org/.*** **[Online]**

36. Overpass API. ***http://wiki.openstreetmap.org/wiki/Overpass_API.*** **[Online]**

37. Leaflet. ***http://leafletjs.com/.*** **[Online]**

38. NPMJS "VeeValidate". ***https://www.npmjs.com/package/vee-validate.*** **[Online]**

39. OSM Map Features. ***http://wiki.openstreetmap.org/wiki/Map_Features.*** **[Online]**

40. Github.com "Beautified JOSM Preset. ***https://github.com/simonpoole/beautified-JOSM-preset.*** **[Online]**

41. GitHub VueTranslate. ***https://github.com/javisperez/vuetranslate.*** **[Online]**

42. YoHours. ***http://github.pavie.info/yohours/.*** **[Online]**

43. Plus Codes. ***https://plus.codes/.*** **[Online]**

44. ESLint. ***http://eslint.org/.*** **[Online]**

45. Python Style Guide. ***https://www.python.org/dev/peps/pep-0008/.*** **[Online]**

46. Docker Best Practices. ***https://docs.docker.com/engine/articles/dockerfile_best-practices/.*** **[Online]**

8 Anhang

8.1 Anleitungen

8.1.1 Zusätzliche Sprache

Anleitung wie eine zusätzliche Sprache zu OSMyBiz hinzugefügt werden soll.

Beispiel mit Italienisch:

1. Neue Sprachdatei hinzufügen
    a. Im Ordner «locales» die Datei (it.json) mit Copy & Paste einfügen
2. App.vue anpassen

```
a. import it from './locales/it.json'

b. Vue.locales({
     de: de,
     en: en,
     it: it
   })
```

3. Sprache im Select hinzufügen (HeaderBar.vue)

```
a. <select v-on:change="onSelect">
     <option value="de">Deutsch</option>
     <option value="en">English</option>
     <option value="it">Italiano</option>
   </select>
```

4. Neue Tags-Sprachdatei über Python-Tool hinzufügen
    a. https://github.com/openstreetmap/iD/tree/master/dist/locales aufrufen und it.json kopieren
    b. Im Ordner «locales» die Datei (it.json) vom iD-Editor mit Copy & Paste einfügen
    c. Auf der Kommandozeile diesen Befehl ausführen:

```
...\osmybiz\tools\src>filter_presets it
```

5. Tags im Locale-Store hinzufügen

```
a. import tagsIt from '../assets/tags/it.json'

b. setTags (state, lng) {
     switch (lng) {
       case 'de': state.languageTags = tagsDe
         break
       case 'en': state.languageTags = tagsEn
         break
       case 'it': state.languageTags = tagsIt
         break
    default:
      state.languageTags = tagsDe
  }
}
```

8.1.2 Presets aktualisieren

Anleitung wie man die Kategorien aktualisiert, wenn der iD-Editor neue Presets veröffentlicht. Alle Änderungen werden im Ordner «tools» vorgenommen.

1. presets.json aktualisieren
    a. https://github.com/openstreetmap/iD/tree/master/data/presets aufrufen
    b. presets.json kopieren und im Ordner «assets» einfügen
2. fields.json aktualisieren
    a. https://github.com/openstreetmap/iD/tree/master/data/presets aufrufen
    b. fields.json kopieren und im Ordner «assets» einfügen
3. locales aktualisieren
    a. https://github.com/openstreetmap/iD/tree/master/dist/locales aufrufen
    b. alle Sprachdateien kopieren und im Order «assets/locales» einfügen
4. Frontend-Dateien aktualisieren
    a. Für jede Sprache auf der Kommandozeile diesen Befehl ausführen:

```
...\osmybiz\tools\src>filter_presets [Sprachabkürzung z.B. de]
```

8.2 OSMyBiz Swagger


```yaml
swagger: "2.0"
info:
  version: "1.0.0"
  title: "OSMyBiz"
host: "localhost:8080"
basePath: "/api"
schemes:
- "http"
paths:
  /user:
    post:
      summary: "Register or update a User"
      operationId: "addOrUpdateUser"
      produces:
      - "application/json"
      parameters:
      - in: "body"
        name: "body"
        required: true
        schema:
          $ref: "#/definitions/User"
      responses:
        200:
          description: "Successful operation"
        400:
          description: "Bad Request"
  /user/{userId}/node:
    post:
      summary: "Create or update a node for the user"
      operationId: "addOrUpdateNode"
      produces:
      - "application/json"
      parameters:
      - in: "path"
        name: "userId"
        required: true
        type: "string"
      - in: "body"
        name: "body"
        required: true
        schema:
          $ref: "#/definitions/Node"
      responses:
        200:
          description: "Successful operation"
        400:
          description: "Bad Request"
        404:
          description: "User not found"
    get:
      summary: "Get nodes for user"
      operationId: "fetchnodes"
      produces:
      - "application/json"
      parameters:
      - in: "path"
        name: "userId"
        required: true
        type: "string"
      responses:
        200:
          description: "Successful operation"
          schema:
            type: "array"
            items:
              $ref: "#/definitions/Node"
  /user/{userId}/node/{osmNodeId}/delete:
    post:
```

```
      summary: "delete the node for the given user"
      operationId: "deleteNode"
      produces:
      - "application/json"
      parameters:
      - in: "path"
        name: "userId"
        required: true
        type: "string"
      - in: "path"
        name: "osmNodeId"
        required: true
        type: "string"
      responses:
        200:
          description: "Successful operation"
        404:
          description: "User or Node not found"
  /user/{userId}/node/{osmNodeId}/unsusbcribe:
    post:
      summary: "unsubscribe the user for updates to this node"
      operationId: "unsubscribe"
      produces:
      - "application/json"
      parameters:
      - in: "path"
        name: "userId"
        required: true
        type: "string"
      - in: "path"
        name: "osmNodeId"
        required: true
        type: "string"
      responses:
        200:
          description: "Successful operation"
        400:
          description: "Bad Request"
        404:
          description: "User or Node not found"
definitions:
  User:
    type: "object"
    properties:
      osmId:
        type: "integer"
        format: "int64"
      username:
        type: "string"
  Node:
    type: "object"
    properties:
      osmId:
        type: "integer"
        format: "int64"
      name:
        type: "string"
      lat:
        type: "number"
        format: "double"
      lng:
        type: "number"
        format: "double"
      version:
        type: "integer"
        format: "int32"
      recieveUpdates:
        type: "boolean"
```