EntwicklerInnen-Dokumentation YellowOSM

YellowOSM (https://www.yellowosm.com) hat es sich zum Ziel gesetzt, freie und offene Gewerbedaten einfach zugänglich zu machen. Was sich in einem Satz erklären lässt bedarf im Hintergrund allerdings einiges an technischem Aufwand. Vereinfacht lässt sich die Systemarchitektur wie folgt zusammenfassen:

(Crawler =>) OpenStreetMap => Datenbank => Suchserver => Website

Der Pfeil => weist dabei auf einen Datenfluss von links nach rechts hin. Die Basis für **YellowOSM** bildet OpenStreetMap. Die Daten von OpenStreetmap sind in einer Vielzahl an Formaten frei verfügbar - für den Anfang beschränken wir uns auf die Daten für Österreich und importieren diese in unsere Datenbank. Dankenswerterweise stellt die Geofabrik aus Karlsruhe dafür einen passenden Datenauszug zur Verfügung: **https://download.geofabrik.de/europe/austria.html**. Diese Daten importieren wir mit dem von OpenStreetMap entwickelten Tool **osm2pgsql** in unsere lokale PostgreSQL-Datenbank. Der komprimierte Dump der Geofabrik ist etwa einen halben Gigabyte groß - importiert in PostgreSQL ergibt dies mehrere Gigabyte an Daten, die binnen einer guten Stunde importiert werden.

Nun da die Daten bei uns im System sind können wir sie für die Website zugänglich machen. Prinzipiell könnte die Website direkt (bzw. mit wenig Programmcode dazwischen) auf die PostgreSQL-Datenbank zugreifen - wesentlich performanter ist der Zugriff allerdings mit einem dezidierten Suchserver. Es gibt zwei große Open-Source-Suchserver, Solr und Elasticsearch, die beide weit verbreitet sind. Beiden gemein ist auch Apache Lucene als Unterbau, einer mittlerweile seit fast 20 Jahren etablierten Bibliothek für Information Retrieval wie etwa Suche und Ranking von Suchergebnissen. Beide Suchserver bieten für Geodaten effiziente Suchmöglichkeiten für Daten innerhalb eines geographischen Bereichs (einer sogenannten Bounding-Box), etwa innerhalb des aktuell dargestellen Kartenausschnittes. Elasticsearch bietet dabei **die etwas bessere Performance** und ist dabei häufiger in Verwendung als Solr - gute Gründe um sich dafür zu entscheiden.
Es gilt nun, Elasticsearch mit Daten zu füttern - dies geschieht durch eine Hand voll Datenbankabfragen, die die relevanten Daten aus der Datenbank extrahieren und in ein JSON-Format für den Import in Elasticsearch bringen. Von OpenStreetMap verwendet werden dabei hautpsächlich die "nodes", also Knoten, exportiert, die Punkte repräsentieren (etwa ein Punkt der für ein Restaurant steht). Ergänzend dazu dienen Polygone - etwa im Falle der Grazer "Apotheke zu Maria Trost", die nicht als Punkt sondern anhand der Gebäudeumrisse (ein Polygon) in OpenStreetMap abgebildet ist. Von Polygonen und Multi-polygonen werden die Schwerpunkte in unseren Datensatz übernommen.

Den letzten Puzzlestein bildet dann das Frontend - hier kommt, wie derzeit überall üblich, JavaScript zum Einsatz. Für JavaScript bieten sich derzeit drei Frameworks an: React, Vue und Angular. Alle drei bieten solide Möglichkeiten für moderne Frontendentwicklung,

weswegen wir das wählen, womit wir die meiste Erfahrung haben: **Angular**, das wir mit **Angular-CLI** aufsetzen.
Als Mapping-Library verwenden wir OpenLayers, das im Gegensatz zu leaflet mehr Möglichkeiten für Anpassungen bietet, auch wenn die **Dokumentation** aber ausbaufähig ist.

Letzter Baustein bildet ein Crawler, der Websites nach Daten zur Ergänzung in OSM abgrast – die dann wiederum in unserem Datenfluss landen. Um nicht gegen OSM Vorgaben zu verstoßen müssen diese gecrawlten Daten über einen Umweg in OSM eingepflegt werden. Automatisierte Edits sind nicht erlaubt, daher werden gefundene Daten zuerst in Maproulette eingespielt, um dort manuell von Editoren geprüft und eingepflegt zu werden.

Im Backend setzen wir auf Python-Responder. Es bietet dem Frontend ein paar Hilfsfunktionen. Etwa den Download der Kontaktdaten als V-Card oder die Verortung der Nutzer-IP um den bestmöglichsten Kartenausschnitt anzuzeigen. Die V-Card kann so problemlos in das eigene mobile Adressbuch gespeichert werden. Weiters verwenden wir das Backend um Geo58-Kurzlinks zu erstellen, die auf Gewerbe permanente Links ermöglichen sollen. Geo58 ist als eigenständige Python-Bibliothek verfügbar.
Das Backend stellt auch eine öffentliche API zur Verfügung. Diese kann genutzt werden um OSM-Gewerbedaten in eigene Projekte zu integrieren. Die Daten stehen unter der Open Data Commons Open Database License (OdbL).

Der gesamte Code dazu findet sich unter https://github.com/YellowOSM/YellowOSM, wo auch in den jeweiligen Modulen (backend, crawler & frontend) noch eine Readme.MD die genaueren technischen Schritte beschreibt.