1 Aufbau des Projekts

Das Projekt besteht grundlegend aus einem Workflow welcher für den Download und die Verarbeitung der Adressen zuständig ist, einer Weboberfläche zum Abfragen von Adressen und einem Backend welches die Anfragen überprüft. Im Folgenden werden die einzelnen Teile genauer erläutert.

2 Workflow

Der Ablauf setzt sich aus vier Schritten zusammen. Zunächst werden die Adressen von `http://results.openaddresses.io/` heruntergeladen, entpackt und unnötige Dateien entfernt. Anschließend werden die Daten in das HDFS kopiert wo unnötige Zeilen und Spalten entfernt werden. Schließlich werden die Daten in eine End-User-Datenbank kopiert, von wo aus sie durch eine separate Anwendung verwendet werden können.

2.1 Main

Der gesamte Workflow wird in einem main-Job verwaltet. Dieser Job enthält die einzelnen Schritte vom Download der Datei bis zum Export der Daten in eine End-User-Datenbank.

2.2 Download Addresses

Im ersten Job wird zunächst das Download-Verzeichnis `openaddresses` erstellt, welches wiederum ein Verzeichnis `raw` enthält. Für diese Verzeichnisse wird überprüft ob noch Dateien enthalten sind, welche ggf. gelöscht werden. Anschließend wird die Adressdatei herunter geladen und schließlich entpackt, sodass sich im Verzeichnis `raw` die einzelnen Verzeichnisse der jeweiligen Länder befinden.

2.3 Copy Addresses to HDFS

Bevor die Dateien in das HDFS kopiert werden, werden zunächst einige Vorbereitungen getroffen. Das in der Download-Datei enthaltene `Summary` Verzeichnis, sowie alle „nicht-`.csv`-Dateien “, werden gelöscht, da diese im weiteren Verlauf nicht benötigt werden. Da die Verzeichnisse der jeweiligen Länder Unterverzeichnisse enthalten können, werden außerdem alle Dateien aus den Unterverzeichnissen in das Root-Verzeichnis des jeweiligen Landes kopiert. Schließlich wird jedes Verzeichnis in `country=`*`Länderkürzel`* umbenannt, damit später im HDFS leicht partitioniert werden kann. (Bild mit vorher nachher???)

Anschließend wird im HDFS das Root-Verzeichnis erstellt und das Download-Verzeichnis dort hin kopiert. Schließlich wird noch Download-Verzeichnis geleert, da dieses von hier an nicht mehr benötigt wird.

Anschließend wird zunächst eine externe Tabelle in Hive angelegt. Diese erhält als Location den Ort, an dem das `raw`-Verzeichnis im HDFS gespeichert wurde. Des Weiteren wird diese Tabelle nach `country` partitioniert. Da durch die Struktur des Verzeichnis die Daten schon vor partitioniert sind, muss Hive

nurnoch mitgeteilt werden, dass diese Verzeichnisse als Partitionen verwendet werden sollen. Dies kann einfach über den Befehl `MSCK REPAIR TABLE addresses` erreicht werden.

2.4 Optimize Addresses

Im nächsten Job sollen die Adressen optimiert werden. Dazu wird zunächst eine interne Hive Tabelle erstellt, welche die finalen Daten enthalten soll. Dazu werden nur die, für den weiteren Verlauf benötigten, Attribute ausgewählt. Diese sind Straße, Hausnummer, Stadt und Postleitzahl. Des Weiteren wird diese Tabelle wieder nach `country` partitioniert.

Anschließend werden die finalen Daten aus der `raw` Tabelle in die `final` Tabelle kopiert. Dazu müssen zunächst der Partitionierungsmodus und die Anzahl an möglichen Partitionen angepasst werden.

Da die `.csv`-Dateien als Header die Spaltennamen besitzen, müssen diese manuell entfernt werden. Dazu werden im letzten Schritt dieses Jobs alle Zeilen entfernt, deren Einträge city=„CITY“, postcode=„POSTCODE“, usw. besitzen.

2.5 Export Addresses To Enduser Database

Im letzten Job werden die Adressen in eine End-User-Datenbank kopiert. Dazu wird zunächst eine MySQL Tabelle erstellt, die alle nötigen Attribute besitzt. Anschließend wird selektiert eine Transformation die Daten aus der Hive Tabelle und kopiert sie in die finale MySQL Tabelle.

3 Client

Um zu überprüfen, ob eine Adresse existiert oder nicht, wird eine einfache, auf Angular 6 basierende Weboberfläche zur Verfügung gestellt. Diese stellt je ein Feld für Straße, Hausnummer, Stadt und Postleitzahl bereit. Da in manchen Fällen für bestimmte Länder oder Städte keine Postleitzahl und/oder Stadt vorhanden ist, sind Straße und Hausnummer Pflichtfelder und Stadt und Postleitzahl nur optional. Wird eine Adresse eingegeben, wird der Benutzer über eine einfache Ausgabe benachrichtigt, ob die Adresse gültig ist oder nicht.

4 Server

Die Daten die Vom Benutzer in die Felder der Weboberfläche eingegeben werden, werden an die REST-API einer auf Node.js basierenden Backend-Anwendung gesendet. Diese überprüft, ob sich die eingegeben Adresse in der Datenbank befindet oder nicht und antwortet dementsprechend mit `true` oder `false`.