Hund ist nicht Katze

Fachmodul ÖV Journey Planning

Christian Bühler Flavio Tobler

August 4, 2018

1 Abstract

Abstract halt

Inhaltsverzeichnis

1 Abstract **I**

2 Einleitung **2**

3 Augabenstellung **3**

4 Ausgangslage **4**
4.1 JourneyPlanning 4
4.2 OpenTripPlanner 4
4.3 Algorithmen 4
4.4 General Transit Feed Specification 4

5 Methode **5**

6 Produkt **6**
6.1 OTP mit Schweizer Daten implementieren 6
6.2 Kann OTP ohne .osm file ausgeführt werden? 6
6.3 Modellierung der Datenstrukur 6
6.4 Automatische generierte Klassendiagramme 6
6.5 Dummy-GTFS Daten erstellen 6
6.6 Mocking 6
6.6.1 CSAMock 6
6.6.2 TimeTableBuilderMock 6
6.6.3 JourneyToTripPlanConverterMock 6
6.7 TimeTableBuilder 6
6.8 JourneyToTripPlanConverter 6
6.8.1 Knackpunkte 7

7 CSAfuerOTP **8**
7.1 Datenstruktur 8
7.1.1 TimeTable 8
7.1.2 StopCSA 8
7.1.3 TripCSA 8
7.1.4 FootpathCSA 8
7.1.5 ConnectionCSA 8
7.1.6 Journey 8
7.1.7 JourneyPointer 8
7.1.8 LegCSA 8
7.2 Programmablauf 8
7.2.1 TimeTableBuilder 9
7.2.2 Server 9
7.2.3 Webseitenaufruf 9
7.2.4 CSA 9

7.2.5 JourneyToTripPlanConverter 9

8 Resultate **10**

9 Fazit **11**

2 Einleitung

Einleitung: Nutzen und Sinn des Projektes + Grobe Kapitelübersichtsbeschreibung der Arbeit

3 Augabenstellung

Unsere BA Aufgabenstellung

4 Ausgangslage

Um unsere Programm und unser vorgehen zu besser verstehen muessen wir zuerst das von uns verwendete Basisprogramm sowie den von uns verwendeten Basisalgorithms erlaeutern.

4.1 JourneyPlanning

Erklärung von JourneyPlanning an sich

4.2 OpenTripPlanner

Erklärung des OTP: Funktion, Entwicklungsgruppe, Code-Grob, Eingabeparameter

4.3 Algorithmen

Erklärung vom Dijkstra kurz und vom CSA lang

4.4 General Transit Feed Specification

Erklärung was GTFS ist und was dessen aufbau und regeln sind

5 Methode

Evaluationen usw.

6 Produkt

6.1 OTP mit Schweizer Daten implementieren

Routetype 1700 -¿ Miscellaneous Service + PC-Auslastung

6.2 Kann OTP ohne .osm file ausgeführt werden?

6.3 Modellierung der Datenstrukur

Java container klassen + UML Diagramm

6.4 Automatische generierte Klassendiagramme

6.5 Dummy-GTFS Daten erstellen

6.6 Mocking

Weshalb mocking?

6.6.1 CSAMock

einlesen von Timetable + Rückgabe eines Journeys

6.6.2 TimeTableBuilderMock

manuelles erstellen eines Timetableobjektes

6.6.3 JourneyToTripPlanConverterMock

auslesen und speichern eines JSON der Rückgabe einer Response der orginal Software + manuelles nachbauen erst ohne laufwege und umsteigen dann immer mehr dazu AgencyAndId + From/to erstellt aber nicht ins leg hinzugefügt + walksteps erstellen + startzeit stopzeit erstellen Gregorian Calendar

6.7 TimeTableBuilder

einstiegspunkt in GTFSMODUL + Namenskonflikt mit der onebusaway Biblothek + Problematik Objektreferenz

6.8 JourneyToTripPlanConverter

bildet TripPlan aus Journey + Schleife für jedes Journey Itinerary

6.8.1 Knackpunkte

Walkdistance aus Koordinaten berechnen (lon,lat) + Himmelsrichtung aus Koordinaten berechnen Timezoneoffset für Footpath + Start-, Stoptime für Footpath + LegGeometry aus Koordinaten berechnen Walksteps erstellen + Datetime aus Time von Timetable, Date from request zusammensetzen

7 CSAfuerOTP

Dies ist eine Erläuterung des Endprogramms

7.1 Datenstruktur

Der CSA benötigt zwei Datenstrukturen. Einen TimeTable für die Eingabe von Daten und ein Journey für die Rückgabe von Daten. Da sich einige Klassennamen mit den Bezeichnungen des Dijkstra Algorithmuses überschneiden ist deren Namen mit einem "CSAërweitert worden.

Datenstrukturen des TimeTables und des Journeys erläutern. Sagen dass CSA hinter namen. Auch sagen was und wieso diese containerklasse

7.1.1 TimeTable

Der TimeTable ist die vom ConnectionScanAlgorithmus als Eingabe benötigte Datenstruktur. Er ist ein Quadrupel aus Sets von StopCSA, TripCSA, FootpathCSA und ConnectionCSA. Das ConnectionCSA-Set ist ein LinkedHashSet, da sie für den Algorithmus anhand der Abfahrtszeit auf- oder absteigend sortiert werden muss. Die anderen Sets sind HashSets. Neben den äddünd ßhow"Funktionen für die Sets enthält die TimeTable-Klasse die Methode "getFootpathChange"welche für einen Stop die Umsteigzeit zurückgibt.

7.1.2 StopCSA

Ein Stop ist eine Haltestelle für öffentliche Verkehrsmittel. Ein Stop besitzt einen Namen, Längen- und Breitengrad sowie eine AgencyAndID-Nummer. Die Klasse besitzt neben den Gettern, Settern und Konstruktoren eine Methode um den Stop zu klonen.

7.1.3 TripCSA

Als Trip wird die Fahrt eines Öffentlichen-Verkehrsmittels von der Start-Station bis zur End-Station bezeichnet. Es ermöglicht den CSA ohne umsteigen erreichbare Orte zu erkennen. - Weiterschreiben

7.1.4 FootpathCSA

Ein Footpath kann zwei verschiedene Funktionen haben. Er besteht aus einem DepartureStop, einem ArrivalStop sowie einer Dauer. Wenn der DepartureStop und der ArrivalStop gleich sind repräsentiert der Footpath einen Umsteigeprozess. Wenn sie unterschiedlich sind repräsentiert er einen Laufweg zu einem Stop hin oder von einem Stop weg. Neben Gettern, Settern und Kosntruktoren hat der Footpath keine weiteren Methoden.

7.1.5 ConnectionCSA

7.1.6 Journey

Ein Journey ist ein vom CSA berechneter Weg vom Start- zum Zielpunkt. Er besteht aus einem StartPath, welcher den Fussweg zur ersten Station hin darstellt, sowie eine Liste aus journeyPointern welche den Weg mit allen Umsteigestationen repräsentiert. Neben den Gettern und Settern gibt es eine Klonfunktion. Für die Liste der journeyPointer gibt es Add-Funktionen zum Einfügen am Anfang, am Ende oder an einem bestimmten Punkt anhand eines Indexes.

7.1.7 JourneyPointer

Ein JourneyPointer ist eine Hilfskonstruktion welche der CSA anlegt um die berechnete Abfolge von Stationen später wieder rekonstruieren zu können.Ein JourneyPointer besteht aus einem Leg sowie einem Footpath. Dabei handelt es sich beim Leg um eine Fahrt in einem ÖV vom einsteigen bis zum Aussteigen und beim Footpath um das darauffolgende Umsteigen oder das erreichen des Ziels.

7.1.8 LegCSA

Ein Leg ist die Fahrt in einem Öffentlichen Verkehrsmittel vom einsteigen bis zum aussteigen. Dies ermöglicht es den Journey nicht von Station zu Station, sondern von Umsteigen zu Umsteigen zu rekonstruieren. Ein Leg besteht aus einer EnterConnection und einer ExitConnection. Er besitzt neben den Gettern, Settern und Konstruktoren keine weiteren Methoden.

7.2 Programmablauf

TimetableBuilder zu Serverstart zu Webseitenaufruf zu CSA zu JourneyToTripPlanConverter

7.2.1 TimeTableBuilder

Erstellte einen Timetable aus GTFS-Daten

7.2.2 Server

Startet server welchen die Webseite für afrufe zur verfügung stellt

7.2.3 Webseitenaufruf

Aufruf über webseite abgesetzt. PlannerRessource primärer einstiegspunkt

7.2.4 CSA

Eigentlicher algorithmus welcher journeys bildet

7.2.5 JourneyToTripPlanConverter

Wandelt vom algorithmus zurückgegebene Journeys in vom Server verlangten TripPlan um

8 Resultate

Unsere Tests und deren Ergebnisse darstellen

9 Fazit

Fazit der BA über den CSA und das Projekt selbst

Abbildungsverzeichnis