Modul zur Kostendarstellung:

Aufwandsschätzung

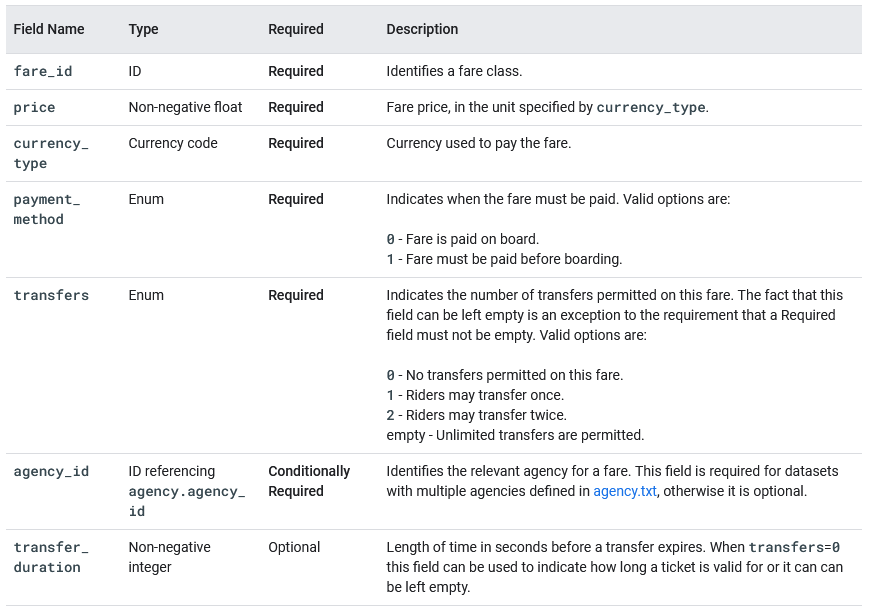
* Fares 🡪 Kosten
* Die Kosten welche in einer Gtfs Datei dargestellt werden können, werden dort in zwei verschiedenen Text (.txt) Dateien dargestellt.
  + fare\_attributes.txt
  + fare\_rules.txt
* Da Kosten eine Optionale Angabe ist sind diese bei den meisten Gtfs-Feeds nicht vorhanden. In Simulationen können kosten jedoch eine wichtige Rolle spielen umso zu beobachten was eine Kostenveränderung mit dem Verhalten gegenüber den ÖPVN geschieht.
* **INFO :** Ähnliche Methoden die hier angewendet werden, bzw. diskutiert werden eigenen sich auch für die Umsetzung der Graph Manipulation nur mit den entsprechenden Gtfs Dateien und werden gegen Ende noch einmal kurz Diskutiert

## fare\_attributes.txt

beschreibt die Tarife welche angeboten werden, d.h. ob man für umstiege zahlen muss, ob man in einer Zone mit einer Karte fahren kann, was die Kosten sind wie bezahlt werden kann usw.

Es werden somit hier keine Zuordnungen zu den Routen gemacht, sondern nur die Definition der Fare Arten geklärt

**Fare information for a transit agency's routes.**



**Beispiel: fare\_attribute.txt**

fare\_id,price,currency\_type,payment\_method,transfers,transfer\_duration  
1,0.00,USD,0,0,0  
2,0.50,USD,0,0,0  
3,1.50,USD,0,0,0  
4,2.00,USD,0,0,0  
5,2.50,USD,0,0,0

Es werden hier zwar alle einzelnen Attribute aufgelistet, jedoch sind nicht alle wirklich nötig. Daher sollte man sich zumindest bei ersten Versuchen der Implementation bzw. Darstellung darauf spezialisieren die nötigen Attribute umzusetzen. Für eine sehr genaue Simulation, könnte oder sollte jedoch am besten alles berücksichtigt werden auch wenn selbst die Gtfs Daten nicht immer alles preisgeben.

## fare\_rules.txt

Hier werden dann die Zuordnungen beschrieben. Den einzelnen Routen kann dann jeweils Kostentarife zugeordnet werden womit dann ein Graph erstellt werden kann der auch die Kosten der Reise berücksichtig.

Die meisten Fare Strukturen nutzen jedoch einen Mix aus mehreren Regeln, welche wären:

* Fare hängt von Herkunft und oder Ziel ab
* Fare hängt davon ab durch welche Zonen die Route durchfährt
* Fare hängt davon ab welche route die Reiseroute nutzt

**Als Beispiel zu den eben genannten Regeln :** <https://code.google.com/archive/p/googletransitdatafeed/wikis/FareExamples.wiki>

|  |  |
| --- | --- |
|  |  |

|  |  |
| --- | --- |
| **Rules to apply fares for itineraries.** |  |

**Beispiel : fare\_rules.txt**

fare\_id,route\_id,origin\_id,destination\_id,contains\_id  
a,TSW,1,1,  
a,TSE,1,1,  
a,GRT,1,1,  
a,GRJ,1,1,  
a,SVJ,1,1,  
a,JSV,1,1,  
a,GRT,2,4,  
a,GRJ,4,2,  
b,GRT,3,3,  
c,GRT,,,6

Hier sind ein paar Basisregeln dargestellt aber nicht alle möglichen. Für genauere Erklärung ist es nötig sich die Beispiele zu den Regeln anzuschauen. (<https://code.google.com/archive/p/googletransitdatafeed/wikis/FareExamples.wiki>)

## Fare Examples

* Beispiel 1 : Alle Trips haben denselben Preis, unbegrenzte Transfers
* Beispiel 2 : Alle Trips haben denselben Preis, keine Transfers
* Beispiel 3 : Alle Trips haben denselben Preis, Transfers sind erlaubt
* Beispiel 4 : Verschiedene Preise für lokale und express Routen
* Beispiel 5 : Kaufen eines Transfers erhöht den Preis
* Beispiel 6 : Preis hängt von Stationen paaren ab, wie man dahin kommt ist egal
* Beispiel 7 : Preis hängt von Zonen ab

Beispiel 7 wird wahrscheinlich die Variante sein, welche am ansprechendsten ist, da man große Anzahl an Daten direkt abhaken kann ohne einen großen Aufwand zu betreiben.

## IDEEN : Möglichkeiten für Umsetzung

* Definieren von Zonen und Preise dafür, womit dann im Nachhinein Kosten berechnet werden
* Schreiben von den Fare Text Dateien (eigene) und diese zu den existierenden Gtfs Feeds hinzuzufügen

### Erster Vorschlag :

Zonen könnten mithilfe von Polygonen definiert werden, für welche dann ein Preis festgelegt werden könnte der sich auf das Reisen innerhalb der Zone bezieht. Desweiteren können diese dann Problemlos mit weiteren Zonen verglichen werden und auch Strafen für das überqueren einer Zone geben was in einen höheren Preis resultieren würde. Oder simple bei befahren mehrere Zonen einfach den höchsten Preis einer Zone nehmen.

Die Schwierigkeit darin liegt bei dem implementieren in den Graph sodass Preise während der Routenfindung schon berechnet werden können. Dafür müsste jedoch der Graph bzw. Graphhopper an sich angepasst werden wodurch es zu einem recht hohem Aufwand kommen könnte da für mich nicht genau bekannt ist was dort alles geändert werden muss und ich Graphhopper immer noch nicht genau im Detail kenne. Möglich jedoch wäre es jedem Stop eine Zone zuzuteilen mit der dann gerechnet werden kann bevor die Routen bestimmt werden. (wie jedoch noch unklar)

Daher würde diese Anwendung der Kosten würde jedoch dann erst vollzogen werden sobald die Routen fertig gebildet wurden und dann geschaut wird welche befahrenen Stops sich in welcher Zone befinden, somit würden diese als Kriterium herausfallen. Vorteil davon wäre jedoch das man Preise und Zonen relativ einfach anpassen und ändern kann.

### Zweiter Vorschlag :

Die nächste und durchaus ansprechendere Möglichkeit wäre es die meistens fehlenden beiden Fares-Dateien des Gtfs-Formates zu diesem noch hinzuzufügen. Da Graphhopper diese beiden Dateien schon unterstützt und mit in den Graphen einbezieht. Schwierigkeit hierbei liegt darin, dass das man erst einmal korrekte Gtfs-Daten erstellen muss, d.h. entweder man schreibt einen Writer, um die Fares Dateien zu schreiben oder man sucht ein Modul welches dies schon tut, dies aber eventuell nach eigenen Anforderungen dann noch angepasst werden muss. Desweiteren müssen auch die bereits vorhanden Daten mit den neuen logisch verknüpft werden, was dazu führen könnte, das die Präparation Zeit für die Simulation erhöht, jedoch an der Performance der eigentlichen Funktion nicht viel ändern sollte.

Vorteil hierbei ist natürlich das im Vergleich zum ersten Vorschlag die Kosten direkt als Entscheidungskriterium mit einbezogen werden können auch das sich um die eigentliche Funktion der Kostenrechnung nicht groß gekümmert werden muss da diese mithilfe der Dateien in Graphopper schon implementiert ist.

Eine Möglichkeit diese Dateien zu schreiben, ist das OneBusAway - GTFS Modul zu Nutzen welches schreiben und lesen von GTFS Dateien ermöglicht. Dadurch müsste nur noch die eigentliche Funktion für die Fares schreiben implementieren und mit logischen Daten verknüpft werden.

### Zwischenergebnis:

Der erste Prototyp ist soweit fertig und Funktioniert auch, wie in Vorschlag zwei erwähnt wurde dieser mit dem Gtfs Modul und dem Gtfs Transformer Modul von OneBusAway gemacht.

Dabei wurden auch teilweise Klassen wie FareAttribute von OneBusAway genutzt um diese nicht noch einmal selbst schreiben zu müssen.

Hierbei ist es nun Möglich **fare\_attribute.txt** und **fare\_rule.txt** Dateien zu schreiben und diese zu definieren. Mit den geänderten Gtfs Daten kann dann auch ein Graph erzeugt werden und wie gewohnt Abfragen abgerufen werden. Wie genau lässt sich in der README.md im Package **graphEditing** genauer nachlesen.

Auch ist die Logik wie z.B. die Fare\_Rules zugeordnet werden noch vom Benutzer selbst zu definieren.

**Probleme:** Die Funktion ist noch nicht wirklich komplett fertig. Man kann sie zwar nutzen, jedoch erfordert dies noch recht viele und genaue Eingaben vom Benutzer sodass auch teilweise Daten angelegt werden können die semantisch keinen Sinn ergeben. Man sollte also wissen wie die beiden Text Dateien aufgebaut sind und wie man diese definiert. Auch ist war ich noch nicht ganz in der Lage zu testen ob diese Kosten dann im erstellen Graph richtig angegeben und benutzt werden. Dies sollte allerdings wenn die Gtfs Daten richtig erstellt worden sind kein Problem darstellen (bei weitere Entwicklung sollte dies trotzdem mal betrachtet werden). Auch muss beachtet werden das falls schon Fares Daten vorhanden sein sollten, man aufpassen muss vorhandene Zusammenhänge etc. nicht einfach zu überschreiben da es vielleicht zu Komplikationen kommen kann.

**Schätzung:** sollte der Prototyp weiter so genutzt werden sollte es nicht der größte Aufwand sein diesen zu erweitern oder auch mit weiterer Logik zu füllen. Dies sollte man in eins bis zwei Wochen meiner Schätzung nach erledigen können.

# Graph Manipulation:

Im Gegensatz zu der Kostendarstellung mit den fares Dateien welche so gut wie nie in einem gtfs Feed angegeben sind ist die Manipulation des Graphs etwas komplizierter, da die Dateien wie routes.txt oder stops.txt immer Angegeben sein muss. Da Fares oft nicht angegeben sind, gibt es dort auch nicht viel zu beachten ob schon vorhanden Daten angepasst werden müssen und man kann davon ausgehen das es keine Verbindungen dazu gibt außer denen, welche man selbst definiert.

Möchte man z.B. einen neuen Stop hinzufügen müssen die Routen angepasst werden damit alle verbundenen Stops und Trips sowie StopZeiten, oder auch die Stop Reihenfolge der Route. Hier bedarf es schon einen etwas größeren Aufwand da hier die Logik des GTFS Formates genau betrachtet und Implementiert werden muss um Logik Fehler zu vermeiden.

Um diesen validierungs-Prozess zu unterstützen, gibt es aber auch Validatoren welche Gtfs Feeds auf deren Logik überprüfen können. Google selbst bietet so etwas auch an aber auch andere Programme haben diese Funktion.

Was die Umsetzung angeht, so kann diese ähnlich der Kostendarstellung erfolgen. Es kann hier auch das **OneBusAway** Programm mit den **gtfs** und dem **gtfsTransfomer** Modul benutzt werden. Denn auch hier müssen Gtfs Daten gelesen, geschrieben oder angepasst werden. Dafür können die schon vorhandenen Klassen **ReaderHandler** und **TransformerHandler**  genutzt werden und zur beliebigen Funktion angepasst werden. Dazu könnte dann noch eine Klasse z.B. StopController welcher dafür zuständig sein soll Stops hinzuzufügen oder zu entfernen zu erstellen und diese Änderungen dann als Manipulation Strings an den Transformer weitergeben. Dieser Controller dann mit der notwendigen Logik füllen und der Facade Klassen hinzufügen um über diesen dann den Controller ansprechen zu können. Dies könnte dann für alle gewünschten Transformationsprozesse gemacht werden.

Der Vorteil hiervon ist das es ziemlich ähnlich zu der schon vorhanden Fares Implementation wäre.

Jedoch ist der Aufwand hier um einiges größer, durch die ganzen Anpassungen welche mit einzelnen Änderungen in Daten einhergehen.