Demonstration der Nutzung von Fahrplandaten im Open Data Kontext

Malte Heidenreich

2. August 2016

Inhaltsverzeichnis

Zusammenfassung

Dieser Bericht beschreibt die Nutzung von Fahrplandaten im Open Data Kontext. Zudem werden Lizenzmodelle und Formate beschrieben. Eine Beispielimplementierung zeigt die Verwendung der Daten von Geofox.

1 Einleitung

Viele Verkehrsbetriebe bieten eine Fahrplanauskunft auf ihrer Website an. Um diese Informationen auch in externen Anwendungen mit wenig Aufwand nutzen zu können, muss der Anbieter eine geeignete Schnittstelle anbieten. Eine alternative Nutzung von Fahrplandaten ist beispielsweise die Simulation des Streckennetzes, um Echtzeitdaten mit dem Soll-Fahrplan zu vergleichen.

Der Begriff Open Data beschreibt den offenen Zugang zu Informationen. Häufig werden die Daten in strukturierter Form angeboten oder sind über eine Programmierschnittstelle abrufbar. [5] Nach dieser Definition sind Fahrplandaten, die über eine geeignete Schnittstelle angeboten werden zunächst als Open Data anzusehen.

Im Folgenden wird gezeigt, welche Bedingungen an die Nutzung geknüpft sind und welche Formate zur Bereitstellung genutzt werden. Anschließend wird eine Beispielimplementierung gezeigt, die Fahrplandaten von Geofox abruft. Zum Schluss wird diskutiert, wie nützlich diese Angebote sind und ob sie in den Open Data Kontext eingeordnet werden können.

2 Lizenzen

In diesem Abschnitt werden Lizenzen beschrieben, die von den unten aufgelisteten Anbietern von Fahrplandaten genutzt werden.

2.1 Creative Commons Attribution 4.0 International (CC BY 4.0)

Creative Commons ist eine gemeinnützige Organisation, die Lizenzverträge verfasst, um der Öffentlichkeit die Nutzung von Urheberrechtlich geschütztem Material zu erleichtern. [1]

Diese Lizenz[1] erlaubt die uneingeschränkt kommerzielle und nichtkommerzielle Nutzung der Daten unter Angabe des Bereitstellers, der Lizenz und einem Link auf den Datensatz. Änderungen an den Daten müssen vermerkt werden.

2.2 Datenlizenz Deutschland – Namensnennung – Version 2.0

GovData ist eine Abteilung der Finanzbehörde der Stadt Hamburg mit dem Ziel Standards zu fördern, um "Metadaten künftig einfacher und umfassender austauschen zu können und so die Mehrwerte aller Datensysteme für ihre Nutzer zu erhöhen" [3].

Diese Lizenz[2] kann als deutsche Übersetzung der CC BY 4.0 gesehen werden. Sie erlaubt ebenfalls die kommerzielle und nicht kommerzielle Nutzung der Daten und Metadaten.

3 Formate und Schnittstellen

In diesem Abschnitt werden Formate und Schnittstellen beschrieben, die von den unten aufgelisteten Anbietern von Fahrplandaten genutzt werden.

3.1 General Transport Feed Specification

Die General Transit Feed Specification (GTFS) definiert ein Format für öffentliche Fahrpläne und die dazugehörigen geografischen Informationen. GTFS-Feeds ermöglichen Verkehrsunternehmen, ihre Fahrplandaten zu veröffentlichen, um diese Entwicklern von Anwendungen zur Verfügung zu stellen. Die Schnittstelle liefert die Daten im CSV-Format.

Das folgende fiktive Beispiel zeigt für die angefragten Reisen *AWE1* und *AWD1* (*trip_id*) die Ankunfts- und Abfahrtszeiten (*arrival_time* und *departure_time*) der Verkehrsmittel an den individuellen Haltestellen (*stop_id*). Eine Sequenznummer (*stop_sequence*) ermöglicht eine Sortierung der Haltestellen. Ob das Ein- oder Aussteigen an einer Haltestelle möglich ist, wird über die Felder *pickup_type* und *drop_off_type* angegeben.

```
trip_id,arrival_time,departure_time,stop_id,↩
    stop_sequence,pickup_type,drop_off_type
AWE1,0:06:10,0:06:10,S1,1,0,0
AWE1,,,S2,2,1,3
```

[1] https://creativecommons.org/licenses/by/4.0/

[2] https://www.govdata.de/dl-de/by-2-0

```
AWE1,0:06:20,0:06:30,S3,3,0,0
AWE1,,,S5,4,0,0
AWE1,0:06:45,0:06:45,S6,5,0,0
AWD1,0:06:10,0:06:10,S1,1,0,0
AWD1,,,S2,2,0,0
```

Für GTFS gibt es einige Bibliotheken, die ein direktes Nutzen dieser Schnittstelle in verschiedenen Programmiersprachen (Java, Python, Go, JavaScript und weitere) ermöglichen.

3.2 ÖPNV-Datenmodell 5.0

Das ÖPNV-Datenmodell 5.0 wurde vom Verband Deutscher Verkehrsunternehmen (VDV) entwickelt. Es hat zum Ziel, den Austausch von Fahrplandaten und Daten für die Verkehrsplanung zwischen unterschiedlichen Anwendungen in Verkehrsbetrieben zu ermöglichen. Die Schnittstellenbeschreibung beinhaltet nur Daten zum Liniennetz und zum Fahrplan. Bereitstellung der Daten kann über zwei Schnittstellen erfolgen. Das Exportprogramm kann zum einen Dateien erzeugen, die im Zielsystem über ein Importprogramm eingelesen werden. Ein anderer Weg, um Daten bereitzustellen ist der Export in eine Datenbank, auf die mit SQL-Anfragen zugegriffen wird. [7]

3.3 Echtzeit Kommunikations- und Auskunftsplattform

Diese Echtzeit Kommunikations- und Auskunftsplattform (EKAP) wurde ebenfalls vom Verband Deutscher Verkehrsunternehmen (VDV) entwickelt. XML wurde als Sprache für die Schnittstelle gewählt. Es können Daten zu Linien, Fahrzeugen, Haltestellen und Tarifen abgerufen werden. [8]

3.4 Proprietäre Schnittstellen

Einige Verkehrsunternehmen bieten selbst entwickelte, proprietäre Schnittstellen an. Diese haben den Nachteil, dass für jedes Verkehrsunternehmen, das eine solche Schnittstelle anbietet, ein separates Verfahren für den Import von Daten implementiert werden muss. Zudem gibt es seltener fertige Programmbibliotheken, die genutzt werden können, um den Aufwand der Implementierung zu verringern.

4 Anbieter von Fahrplandaten

In Deutschland gibt es viele Anbieter von Fahrplandaten. In diesem Abschnitt werden einige Dienste mit den jeweiligen Nutzungsbestimmungen bzw. der Lizenz zur Nutzung und dem angebotenen Format beschrieben. Die Auswahl ist keine vollständige Liste und ist auf Anbieter beschränkt, die eine einfache Nutzung der Daten erlauben und für eine Beispielimplementierung geeignet sind.

Dies wird durch wenig einschränkende Lizenzen und/oder unkompliziertes Konsumieren der Schnittstelle gefördert. Alle Anbieter fordern vom Nutzer der Schnittstelle, den Nutzungsbedingungen zuzustimmen. Dieser Vertrag regelt die Art der Nutzung, also den Verwendungszweck für die abgerufenen Fahrplandaten.

4.1 Verkehrsverbund Berlin-Brandenburg (VBB)

In den Nutzungsbestimmungen wird die CC BY Namensnennung angegeben. Es wird angenommen, dass es sich hierbei um die CC BY 3.0 DE[3] handelt. Der Verkehrsverbund Berlin-Brandenburg nutzt für die Bereitstellung der Fahrplandaten das GTFS-Format.

4.2 Verkehrsverbund Rhein-Sieg GmbH

Der Nutzer der Schnittstelle muss die Verkehrsverbund Rhein-Sieg GmbH oder deren Logo in den Abfrageergebnissen anzeigen, soweit es möglich ist. [10] Dies entspricht weitestgehend der *Creative Commons Attribution* oder der *Datenlizenz Deutschland – Namensnennung*. Der Verkehrsverbund Rhein-Sieg nutzt für die Bereitstellung der Fahrplandaten das GTFS-Format.

4.3 Deutsche Bahn

Die Daten der Schnittstelle[4] werden unter der Lizenz Creative Commons Attribution 4.0 International (CC BY 4.0) zur Verfügung gestellt. [2] Die Schnittstelle zu den Fahrplandaten ist proprietär. Es wird eine Beschreibung im Swagger-Format bereitgestellt [5].

4.4 Verkehrsverbund Rhein-Neckar

Der Verkehrsverbund Rhein-Neckar nutzt als Lizenz die Datenlizenz Deutschland – Namensnennung. Er bietet zwei Möglichkeiten zum Abrufen von Fahrplandaten. Zum einen wird das ÖPNV-Datenmodell 5.0 [9] für den Zugriff auf Fahrplandaten genutzt. Über die Echtzeit Kommunikations- und Auskunftsplattform kann der Nutzer auf Echtzeitdaten zu den Verkehrsmitteln zugreifen. [6]

4.5 Geofox

Geofox ist die Online-Fahrplanauskunft des Hamburger Verkehrsverbundes (HVV). Zudem wird auch eine Datenschnittstelle für Entwickler angeboten. Sie wurde vom Hamburger Berater Team entwickelt. Über dieses Angebot können auch

[3] `https://creativecommons.org/licenses/by/3.0/de/`

[4] `http://data.deutschebahn.com/apis/fahrplan/`

[5] `http://editor.swagger.io/#/?import=http://download-data.deutschebahn.com/static/apis/fahrplan/swagger.yaml`

die überregionalen Fahrplandaten der Deutschen Bahn abgerufen werden. Eine weitere Eigenschaft dieses Angebots ist die Möglichkeit auszuwählen, ob Echtzeitdaten oder der Soll-Fahrplan genutzt wird.

Die Nutzungsbedingungen schreiben vor, dass "auf der Ergebnisseite der Fahrplanauskunft [...] stets ein gut sichtbarer Link zur Auskunfts- und Vertriebsplattform des HVV im Internet (www.hvv.de) angeboten werden"[4] muss.

Zudem muss an einer geeigneten Stelle angegeben werden, wer diesen Dienst betreibt und daran beteiligt ist. Die Herkunft der Daten muss ebenfalls angegeben werden.

Die Schnittstelle ist proprietär und nicht mit anderen Dienstanbietern kompatibel. Angefragte Daten werden als JSON oder XML zurückgegeben.

5 Beispielimplementierung

In diesem Abschnitt wird der Fahrplandienst *ScheduleService* beschrieben. Es wird gezeigt, welche Anforderungen an ihn gestellt werden und welche Funktionalität angeboten wird.

5.1 Ziel

Das Ziel dieser Beispielimplementierung ist die Demonstration der Nutzung von Fahrplandaten im Open Data Kontext. Es soll an einem konkreten Beispiel gezeigt werden, welche Schritte nötig sind, um ein solches Vorhaben zu realisieren.

5.2 Anforderungen

Der ScheduleService soll die Schnittstellen von Fahrplandiensten kapseln. Die folgenden Anforderungen wurden für einen Dienst formuliert, der eine große Menge an Anfragen für Fahrplandaten in kurzer Zeit bearbeiten können soll. Eine Persistenzschicht vermeidet unnötige Wiederholungen von Anfragen an externe Dienste.

5.2.1 Geofox nutzen

Die Schnittstelle von Geofox soll genutzt werden, da sie alle nötigen Informationen für eine Beispielimplementierung bereithält. Es können alle Linien und Stationen des Hamburger Verkehrsverbund (HVV) und der Deutschen Bahn für eine überregionale Anwendung abgefragt werden.

5.2.2 Persistenz ermöglichen

Der Fahrplandienst des Anbieters ist nicht darauf ausgelegt, Daten aus der Vergangenheit auszuliefern. Um historische Daten zu sammeln, müssen bereits abgefragte Daten persistiert werden. Beispielsweise können diese später genutzt werden, um einen früheren Zeitpunkt zu simulieren.

Da der Anbieter auf eine hohe Verfügbarkeit seines Dienstes angewiesen ist, wird nur eine bestimmte Anzahl an Zugriffen pro Zeit zugelassen. Die lokale Datenhaltung ermöglicht die Aufhebung dieser Beschränkung für bereits abgerufene Daten.

5.3 Abgrenzung

Für diese Beispielimplementierung werden keine Mechanismen genutzt, um die eigene Web-Schnittstelle abzusichern. Da es sich um öffentlich verfügbare Daten (z.B. Zugriff über die Fahrplanauskunft) handelt, ist dies nicht nötig.

5.4 Geofox API

Geofox stellt zum Abrufen von Fahrplandaten und Informationen zum Schienennetz einen RESTful Webservice zur Verfügung. Wie die Kontaktaufnahme mit dem Schnittstellenanbieter abläuft, wird im Anhang A beschrieben.

Die Tabelle 5.4 zeigt die Funktionen, die bereits implementiert wurden.

Funktion	**Beschreibung**
init	Abfragen von Basis-Informationen vom Server
checkName	Verifizieren eines Namens und ermitteln von zugehörenden Informationen
getRoute	Ermitteln einer ÖPNV-Verbindung
departureList	Ermitteln der Liste aller Abfahrten an einer Haltestelle
listStations	Ermitteln einer Liste von Haltestellen
listLines	Ermitteln einer Liste von Linien
departureCourse	Ermitteln eines Fahrtverlaufes z.B. die vorhergehenden oder nachfolgenden Haltestellen mit Zeiten und Koordinaten

Tabelle 1: Implementierte Funktionen in der Geofox-Komponente

Die Authentifizierung gegenüber dieses Dienstes erfolgt über die HTTP-Header-Felder *geofox-auth-user* (Benutzername) und *geofox-auth-signature* (Signatur der Anfrage).

5.5 Erstellen der Signatur

Alle Anfragen an den Webserver von Geofox werden signiert, um eine Identitätsprüfung zu ermöglichen. Das Erstellen der Signatur in C# gestaltete sich schwierig, da unterschiedliche Encodings und Hashverfahren genutzt werden mussten. Beispielimplementierungen sind in der Dokumentation der Schnittstelle nur für Objective-C, Java und PHP gegeben. Der folgende Code kann

genutzt werden, um eine Signatur in C# zu erstellen:

```
private static string EncodeHmac(string key, string ↩
    message)
{
    var enc = Encoding.UTF8;
    HMACSHA1 hmac = new HMACSHA1(enc.GetBytes(key));
    byte[] stringBytes = enc.GetBytes(message);
    byte[] hashedValue = hmac.ComputeHash(stringBytes);
    return Convert.ToBase64String(hashedValue);
}
```

5.6 Komponenten

Die Abbildung 1 zeigt die Komponenten des ScheduleService, die in den folgenden Abschnitten erklärt werden.

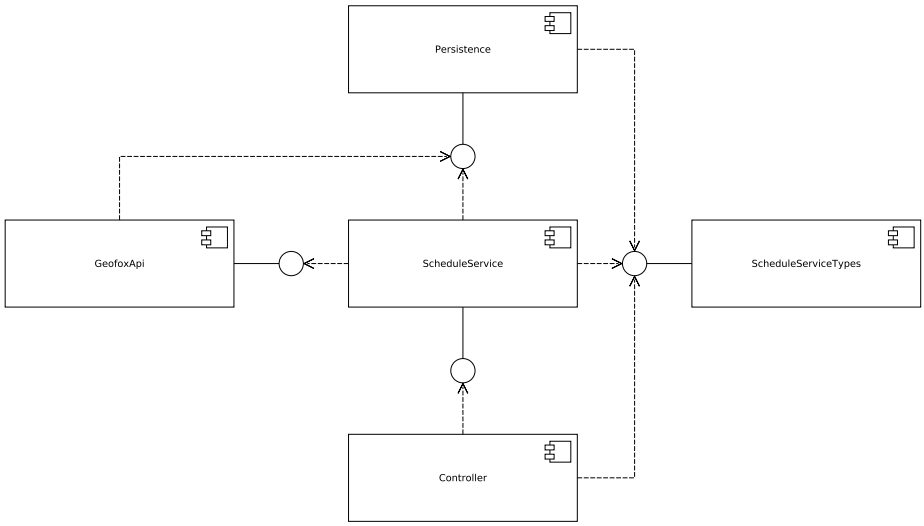

Abbildung 1: Komponenten des ScheduleService

5.6.1 GeofoxApi

Diese Komponente bietet Zugriff auf den Webservice von Geofox. Im Abschnitt 5.4 werden die Funktionen gezeigt, die bereits über diese Komponente ausführbar sind.

5.6.2 Persistence

Um unnötige Wiederholungen von Anfragen an externe Dienste zu stellen, kann die Persistenz-Komponente Daten zwischenspeichern. Alle Anfragen an den

Fahrplandienst können durch diese Komponente geleitet werden, um zunächst lokal zu prüfen, ob bereits eine Antwort vorhanden ist. Dies ist vor allem nötig, wenn die Namen oder IDs von Stationen in Objekte übersetzt werden müssen, um mit der Geofox-Schnittstelle zu kommunizieren.

Diese Komponente ist ebenfalls hilfreich, wenn nicht mehr auf die externe Schnittstelle zugegriffen werden kann. Dies kann auftreten, wenn keine Verbindung zum Anbieter hergestellt werden kann oder der Nutzungsvertrag nicht verlängert wurde. Für diese Art der Verwendung ist es selbstverständlich erforderlich, dass die zu verwendenden Daten zuvor abgerufen wurden.

Für die Persistenz wird eine MySQL-Datenbank genutzt. Die nötigen Tabellen werden von der *Persistence*-Komponente selbst verwaltet.

5.6.3 ScheduleServiceTypes

Diese Komponente enthält die Typen, die genutzt werden, um mit anderen Komponenten zu kommunizieren. Dies vermeidet außerdem eine zyklische Abhängigkeit zwischen den Komponenten *Persistence* und *ScheduleService*.

5.6.4 ScheduleService

Die wichtigste Komponente ist der *ScheduleService*. Dort werden alle vorhergehenden Komponenten zusammengeführt. Sobald eine Anfrage an diese Komponente gestellt wird, prüft die *Persistence*-Komponente, ob bereits eine Antwort vorhanden ist. Dies ist möglich, da die Hashes aller Anfragen zusammen mit ihren Antworten persistiert werden. Sollte die Antwort noch nicht vorhanden sein, wird über die *GeofoxApi*-Komponente eine Anfrage an den Webdienst gestartet. Die Antwort wird direkt persistiert.

5.6.5 Controller

Der Controller hält eine Instanz des *ScheduleService* und stellt die Web-Api zur Verfügung. Dadurch ergibt sich der Vorteil, dass in den Anwendungen, die diesen Dienst nutzen, keine Zugangsdaten zu den verwendeten Fahrplandiensten gespeichert werden müssen.

5.7 Externe Schnittstelle

Die externe Schnittstelle ist als Webservice gestaltet. Es werden Ressourcen angeboten, die mit der HTTP-Methode POST abgerufen werden.

5.8 Schwierigkeiten

Um die Geofox-Schnittstelle nutzen zu können, müssen sehr viele Klassen für die verschiedenen Anfrage- und Antworttypen implementiert werden. Diese Typen sind teilweise sehr stark verschachtelt oder werden für ähnliche Zwecke doppelt implementiert. So wird beispielsweise der Typ *StationLight* in einer Liste *stationSequence* im Typ *SublineListEntry* verwendet. Der Typ *SDName* besitzt alle

Felder von *StationLight* und zusätzlich andere optionale Felder. So ist eine Konvertierung nötig, sollten die nächsten Abfahrtzeiten an einer Station aus dieser Liste über die Funktion *departureList* abgefragt werden.

6 Testbetrieb der Beispielimplementierung

Für den Testbetrieb wird Docker[6] verwendet. Docker erlaubt es in diesem Fall, den ScheduleService plattformunabhängig in Betrieb zu nehmen.

6.1 Bauen der Anwendung

Die Anwendung wird mit Hilfe von Mono[7] in einem Docker-Container gebaut:

```
$ docker run --rm --name scheduleservicebuilder -v $(↩
    pwd):/source -w /source -ti mono:latest xbuild /p:↩
    Configuration=Release ScheduleService.sln
```

Der ScheduleService wird in einen eigenen Container kopiert:

```
$ docker build -t scheduleservice .
```

Das zugehörige *Dockerfile* zeigt den Kopiervorgang (COPY), die Portweiterleitung (EXPOSE) und das Kommando, das ausgeführt wird, wenn der Container gestartet wird (ENTRYPOINT):

```
FROM mono:latest
ENV bin /service
EXPOSE 80
WORKDIR ${bin}
COPY Starter/bin/Release/ $bin
ENTRYPOINT ["mono", "Starter.exe"]
```

6.2 Starten der Anwendung

Zunächst wird der Datenbank-Container gestartet:

```
$ docker run -d --name devdb -e MYSQL_ROOT_PASSWORD=↩
    root -p 3306:3306 mariadb
```

Nach etwa 20 Sekunden kann über das folgende Kommando die Datenbank *ScheduleService* angelegt werden:

```
$ mysql -h 127.0.0.1 -uroot -proot -e "create␣database␣↩
    ScheduleService";
```

Der ScheduleService wird ebenfalls in einem Docker-Container gestartet:

[6] https://www.docker.com/
[7] http://www.mono-project.com/

```
$ docker run -d --name scheduleservice -p 80:80 \
          -v $(pwd)/ScheduleService.config:/service/↩
              ScheduleService.config \
          -e "GEOFOX_PASSWORD=$GEOFOX_PASSWORD" ↩
              scheduleservice
```

Der ScheduleService ist anschließend über den Port 80 auf dem lokalen Rechner erreichbar. Beispielsweise kann anschließend die *status*-Methode aufgerufen werden:

```
$ curl -d '' http://localhost/api/status
{
    "Data": {
        "ReturnCode": "OK",
        "ErrorText": null,
        "ErrorDevInfo": null
    },
    "Error": null
}
```

Das Ergebnis dieser Anfrage zeigt, dass der Dienst gestartet wurde.

6.3 Nutzen der Implementierung

In diesem Abschnitt soll der Nutzen der Implementierung geprüft werden.

6.3.1 Latenz

Zunächst muss die *warmup*-Methode aufgerufen werden, um vorab die Liste der Stationen abzurufen:

```
$ curl -d '' http://localhost/api/warmup
{
    "Data": [
        {
            "Name": "",
            "Duration": "00:00:07.8194540",
            "ReturnCode": "Imported Stations = 7857",
            "ErrorText": null,
            "ErrorDevInfo": null
        }
    ],
    "Error": null
}
```

Das Ergebnis dieser Anfrage zeigt, dass 7857 Stationen innerhalb von etwa 8 Sekunden importiert wurden und keine Fehler aufgetreten sind.

Anschließend können die Abfahrtszeiten der U-Bahnen für einen bestimmten Zeitraum abgerufen werden:

```
$ time curl --data 'ServiceTypes=UBAHN' --data '↩
    DateFrom=2016-08-04 9:00:00' --data 'DateTo↩
    =2016-08-04 10:00:00' http://localhost/api/↩
    departures > /dev/null
```

Die folgende Tabelle und die Abbildung 2 zeigen, dass die Dauer einer Anfrage beim ersten Versuch vergleichsweise lange dauert. Sollten die Daten erneut abgerufen werden müssen, ermöglicht es die Persistenz die Bearbeitungszeit deutlich zu reduzieren. In diesem Beispiel wird ein maximaler Zeitraum von 24 Stunden verglichen, da dieser durch die API von Geofox vorgeben ist.

Zeitraum	1. Anfrage	2. Anfrage	Abfahrten
1 Minute	57,327 s	0,659 s	72
60 Minuten	62,195 s	1,447 s	2065
8 Stunden	83,600 s s	5,653 s	16294
24 Stunden	101,838 s	12,289 s	36288

Tabelle 2: Gegenüberstellung der 1. und 2. Anfrage

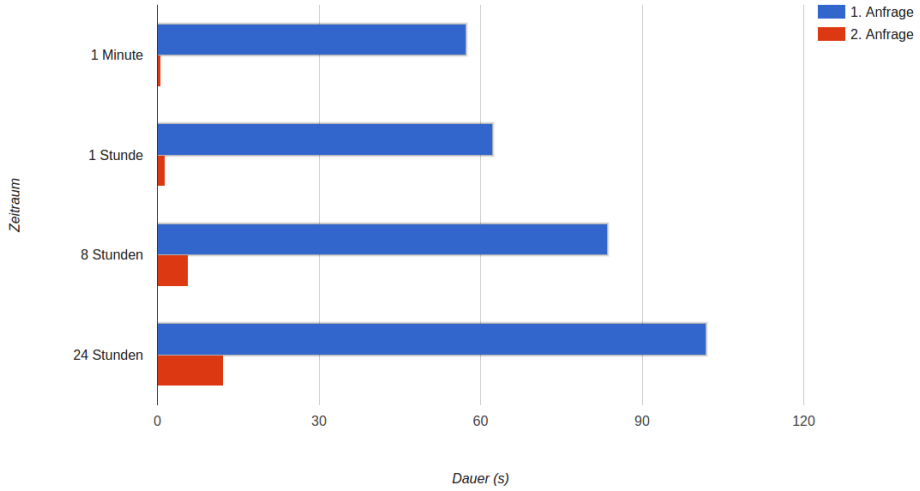

Abbildung 2: Gegenüberstellung der 1. und 2. Anfrage

Diese Gegenüberstellung zeigt, dass durch die Persistenz eine Zeiteinsparung von bis zu 99% für kleine Zeiträume (1 Minute) und bis zu 88% für größere Zeiträume (24 Stunden) erreicht werden können.

Die Erklärung für das langsame Ansteigen der Zeiten für die erste Anfrage ist, dass für jede Station die Abfahrtszeiten einzeln abgefragt werden müssen. Eine nebenläufige Abfrage könnte dazu führen, dass die maximale Anzahl an API-Zugriffen pro Zeit zu schnell erreicht wird.

6.3.2 Speicherbedarf

Um den Speicherbedarf der Abfahrten analysieren zu können, muss zunächst der Speicherbedarf pro Datensatz festgestellt werden. Dieser lässt sich der folgenden Tabelle entnehmen:

Feld	**Typ**	**Ungefährer Speicherbedarf (Bytes)**
LineDirection	TEXT	20
Station	TEXT	20
StationId	TEXT	12
Time	INT	4
Line	TEXT	3
ServiceType	TEXT	6
Summe		**65**

Tabelle 3: Berechnung des Speicherbedarfs pro Datensatz

Für ein Rechenbeispiel werden die Abfahrten der Hamburger U-Bahnen herangezogen. Die Anzahl der Abfahrten pro Tag sind der Tabelle aus dem Abschnitt 6.3.1 entnommen.

```
Speicherbedarf pro Tag:  36288 * 65 Bytes = 2,25 MB
Speicherbedarf pro Jahr:   365 *  2,25 MB =  821 MB
```

Der Speicherbedarf wurde an dieser Stelle anhand einer nicht normalisierten Tabelle berechnet. Sollte der Speicherbedarf wichtig für die Anwendung sein, könnten die Textfelder durch *INT*-Fremdschlüssel ersetzt werden, was den Speicherbedarf pro Datensatz auf 24 Bytes (6 * 4 Bytes) reduzieren würde. Dies könnte jedoch zu Performanceverlusten führen.

7 Fazit

Die öffentlich verfügbaren Fahrplandaten verschiedener Anbieter sind nützlich, um kleine Dienste zu implementieren oder eine Aggregation mehrerer Angebote durchzuführen. Sollte es nötig sein massenhaft darauf zuzugreifen, beispielsweise für Simulationen, wird es schwierig, da alle Anbieter eine solche Art der Nutzung ausschließen. Zudem sind die Live-APIs darauf ausgelegt, Daten für aktuelle und zukünftige Ereignisse auszuliefern. Historische Daten werden nicht angeboten oder müssen über andere Dienste abgerufen werden.

Die Nutzungsvereinbarungen verbieten teilweise die Erzeugung von Statistiken[10], was eine wissenschaftliche Verwertung erschweren könnte. Da es zu jedem Zugang zu Fahrplandaten einen jeweiligen Nutzungsvertrag gibt, in dem der Ver-

wendungszweck angegeben werden kann, sind die so gewonnenen Daten dennoch als Open Data anzusehen.

Literatur

[1] Creative Commons: What we do (2016), https://creativecommons.org/about/

[2] DB Vertrieb GmbH: Fahrplan API (2016), http://data.deutschebahn.com/dataset/api-fahrplan

[3] Finanzbehörde - Geschäfts- und Koordinierungsstelle GovData: Deutschland auf dem Weg zum neuen Metadatenstandard (2016), https://www.govdata.de/web/guest/standardisierung

[4] Hamburger Hochbahn Aktiengesellschaft: Vereinbarung über die Nutzung von Schnittstellen für Zwecke der Fahrplanauskunft (2015)

[5] Matzat, L.: Definitionen: OpenData, OpenGovernment, Gov2.0 und Co. (2010), http://datenjournalist.de/definitionen-opendata-opengovernment-gov2-0-und-co/

[6] Rhein-Neckar, V.: Open Service (2016), http://www.vrn.de/vrn/einfach-ankommen/fahrplanauskunft/openservice/index.html

[7] Verband Deutscher Verkehrsunternehmen: VDV-Standardschnittstelle Liniennetz/Fahrplan (2013), https://www.vdv.de/service/downloads.aspx?id=1671&forced=true

[8] Verband Deutscher Verkehrsunternehmen: Echtzeit Kommunikations- und Auskunftsplattform EKAP (2014), https://www.vdv.de/vdv-431-2-ekap-schnittstellenbeschreibung.pdfx?forced=true

[9] Verkehrsverbund Rhein-Neckar: Open Data (2016), http://www.vrn.de/vrn/einfach-ankommen/fahrplaene/opendata/index.html

[10] Verkehrsverbund Rhein-Sieg GmbH: Nutzungsvereinbarung (2016), https://www.vrsinfo.de/fileadmin/Dateien/api/NutzervereinbarungODOS.pdf

Alle Links wurden zuletzt am 2. August 2016 besucht.

A Zugang zur Geofox-Api

A.1 Kontakt mit der Hochbahn

Den Zugang zur Geofox-Api bekommt man über die Hamburger Hochbahn AG. Der zuständige Kontakt ist Herr Michael Wittke. Ihm teilt man folgende Punkte mit:

- die Art der Verwendung der Schnittstelle
- die Adresse des Unternehmens oder der Person, die die Schnittstelle nutzt
- den Nutzungszeitraum

Die vollständigen Kontaktdaten:

> Michael Wittke
> Hamburger Hochbahn AG
> Sachgebiet Auskunftsmedien
> Steinstr. 7 D-20095 Hamburg
>
> Phone: +49 40 3288 4717
> Fax: +49 40 3288 2866
> Mail: Michael.Wittke@hochbahn.de

Die Hochbahn verschickt nach der Kontaktaufnahme eine Nutzungsvereinbarung, die in zweifacher Ausführung unterschrieben zurück gesendet werden muss. Nach wenigen Tagen kommt ein von der Hochbahn unterschriebenes Exemplar zurück und zeitgleich die Aufforderung, sich mit dem Hamburger-Berater-Team in Verbindung zu setzen.

A.2 Kontakt mit dem Hamburger-Berater-Team (Geofox)

Um den folgenden Schritt etwas abzukürzen, sollten in der Anfrage zur Freischaltung der Testphase an api@geofox.de folgende Fragen[8] beantwortet sein:

1. Wir wüssten gerne, für welche Firma / welches Produkt der API-Zugang verwendet wird. Da wir in die Vorgänge der HOCHBAHN nicht eingebunden sind, haben wir darüber in Ihrem Fall bisher keine Informationen und bitten Sie, uns kurz mitzuteilen, im Rahmen welcher Anwendung die Daten verwendet werden sollen.
2. Welche Funktionen benötigen Sie voraussichtlich? (Siehe Tabelle 5.4)
   (a) init
   (b) checkName
   (c) getRoute

[8] Die Fragen stammen aus dem Mailkontakt mit dem HBT

(d) departureList

(e) listStations

(f) listLines

(g) departureCourse

(h) getAnnouncements (zur Zeit nicht benötigt)

(i) getIndividualRoute (zur Zeit nicht benötigt)

3. Wie werden Sie auf den Test-Server (und später auch auf den produktiven Server) zugreifen?

   Greifen Sie über einen eigenen Server zu, der eine feste IP-Adresse besitzt oder von unterschiedlichen Systemen mit vielen unterschiedlichen IP-Adressen?

   Die Authentifizierung ist über eine von Ihnen erzeugte Signatur oder über die Freigabe einer festen IP-Adresse möglich, wir bevorzugen die Signatur, weil dadurch die Konfiguration bei Ihnen und bei uns flexibler ist. Bei Zugriff von wechselnden IP-Adressen ist die Signatur unbedingt notwendig. Die Erstellung ist in der beiliegenden Dokumentation beschrieben.

4. Mit wie vielen Anfragen pro Minute / pro Tag müssen wir in der Test-Phase rechnen, mit wie vielen später in der produktiven Phase?

5. Wie lange dauert die mit der Hochbahn vereinbarte Testphase?

Nach wenigen Tagen versendet das Hamburger-Berater-Team eine Bestätigung, dass der Zugang eingerichtet wurde. Diese enthält folgende Informationen:

- Den Benutzernamen für die Authentifizierung
- Das Passwort für die Signatur-Erzeugung zur Authentifizierung
- Das Ablaufdatum des Zugangs
- Den DNS-Namen des Servers