Logistics Business Information Web Service

Inhaltsverzeichnis

Einleitung

Erstellt wird ein Webservice, der die Unternehmensdaten und -logik eines Fahrrad-Logistik-Unternehmens beherbergt bzw. über dritte Services transparent bereitstellt.

Ziel ist es dabei, eine einheitliche Schnittstelle zu definieren und zu realisieren, die im täglichen Betrieb von vielen Orten und Systemen aus genutzt werden kann und gleichzeitig jederzeit Anpassungen der jeweiligen Implementationen ermöglicht. So ist es später zum Beispiel kurzfristig möglich, den Routing-Anbieter zu wechseln oder sogar auf ein eigenes System umzustellen, ohne dass sich an der allgemeinen Schnittstelle, die von den verwendeten Systemen erwartet wird, etwas ändern muss.

Offene Fragen und aktuelle Bemerkungen sind gelb hinterlegt.

Beschaffenheit des Webservices

Mögliche Basis-URL: https://api.velogista-hh.de/

REST-Service, CRUD mittels HTTP-Methoden GET, POST, PUT, DELETE

Kommunikation mittels JSON

Implementierung: Jersey, Jackson

Es fehlt noch eine Datenhaltung: Oracle, MySQL, MongoDB oder was?

- GET: Ermittelt einzelne Datensätze oder Listen von Datensätzen
  z.B: https://api.velogista-hh.de/customer/id/5
  oder https://api.velogista-hh.de/route/list
  Listen können über Anfrage-Parameter eingeschränkt werden, z.B:
  https://api.velogista-hh.de/route/list?customer_id=5
  für eine Liste aller Routen, an denen Kund_In mit der ID 5 beteiligt ist.

- POST: Erstellen neuer Datensätze
  Die Daten können entweder als JSON (application/json) oder als HTML-Formular(application/x-www-form-urlencoded) im Body der Anfrage gesendet werden. XML wird vorerst nicht unterstützt.
  Der Upload von Dateien geschieht HTTP-konform über (multipart/form-data).
  [Noch keine Ahnung wie das außerhalb eines Webbrowsers funktioniert]
  Eventuell gesetzte IDs werden ignoriert, da der Webservice diese selbst generiert. Die neue ID ist – zusammen mit der URL des neuen Datensatzes – in der Antwort enthalten.

- PUT: Aktualisieren von Datensätzen
  Für die übersendeten Daten gelten die gleichen Regeln wie für die POST-Methode.
  Der einzige Unterschied ist, dass PUT-Daten eine ID gesetzt haben müssen, die den Datensatz festlegen, der aktualisiert werden soll.
  [Optional könnte die ID auch über die URL mit übergeben werden]
  [Später können auch Arrays übergeben werden, um viele Datensätze gleichzeitig zu aktualisieren]

- DELETE: Löschen von Datensätzen
  Die ID des zu löschenden Datensatzes wird in der URL übergeben, z.B:
  HTTP-DELETE an https://api.velogista-hh.de/location/id/5

Um Missbrauch zu verhindern, müssen Anfragen als Parameter *key* grundsätzlich einen gültigen API-Schlüssel enthalten. Dieser kann in der URL oder in einem HTML-Formular übertragen werden. Der Übersichtlichkeit halber wird dieser Schlüssel in Beispielen der Dokumentation weggelassen.

Antworten enthalten grundsätzlich eine Eigenschaft *info,* die den Erfolg oder Fehlschlag der Anfrage zurück gibt und beschreibt. Mögliche Statuscodes sind in den einzelnen Kapiteln beschrieben

Verwaltete Daten (a.k.a Resourcen)

Der Webservice soll vorerst alle zum Routing notwendigen Daten verwalten.

Kundendaten

Die Kundendaten umfassen die Stammdaten von Auftraggebenden.

Eigenschaft	Datentyp	Beschreibung
id	Integer	Eindeutige ID der Kund_In
name	String	Name der Privatperson oder Firma
nameExt	String	Namenserweitung (z.B: Niederlassung, c/o, …)
street	String	Straße und Hausanschrift
postcode	String	Postleitzahl
city	String	Stadt
country	String	Land
phone	String	Telefonnummer
fax	String	Faxnummer
email	String	E-Mail-Adresse
website	String	Website

Abrufen (*GET)*

Es können einzelne Kundendaten abgerufen werden, die anhand einer ID oder eines Namens eindeutig identifiziert sind. Außerdem können Listen anhand von Kriterien (Straße, Stadt etc) ermittelt werden

Einzelne Kund_Innen über ID

Anfrage:

https://api.velogista-hh.de/customer/id/{customer_id}

Pfad-Parameter:

- costumer_id: ID der zu ermittelnden Kund_In

```
GET https://api.velogista-hh.de/customer/id/5
```

Antwort:

```
{
      'id' : '5',
      'name' : 'Musterfirma GmbH',
      'street' : 'Waidmannstraße 12b',
      …,
      'info': {
                        'status': …,
                        'statusMessage': …
                }
}
```

Mögliche Statusmeldungen

Status	Bedeutung
0	Erfolg

1	Keine ID übermittelt
2	Kund_In mit übermittelter ID existiert nicht

Einzelne Kund_Innen über Name

Anfrage

https://api.velogista-hh.de/customer/name/{customer_name}

costumer_name: Name der zu ermittelnden Kund_Innen

```
GET https://api.velogista-hh.de/customer/name/Musterfirma+GmbH
```

Antwort

```
{
      'id' : '5',
      'name' : ' Musterfirma GmbH',
      'street' : 'Waidmannstraße 12b',
      …,
      'info': {
                      'status': …,
                      'statusMessage': …
                }
}
```

Mögliche Statusmeldungen

Status	Bedeutung
0	Erfolg
1	Kein Name übermittelt
2	Kund_In mit übermitteltem Namen existiert nicht

Kund_Innenlisten mit Kriterien

Anfrage:

https://api.velogista-hh.de/customer/list/?<criteria=value>&...

Parameter:

- street: Alle Kund_Innen aus dieser Straße
- city: Alle Kund_Innen aus dieser Stadt

```
GET https://api.velogista-hh.de/customer/list/?city=Hamburg
```

Antwort:

```
[{
      'id' : '5',
      'name' : ' Musterfirma GmbH',
      'street' : 'Waidmannstraße 12b',
      …
 },
 {
  …
 },
 …,
      'info': {
                      'status': …,
                      'statusMessage': …
                }
}
```

Mögliche Statusmeldungen

Status	Bedeutung
0	Erfolg
1	Kriterium wird nicht unterstützt (Kriterium)

Erstellen (POST)

Erstelle neue Kund_Innendaten durch Übergabe der Daten in JSON (application/json) oder als HTML-Formular(application/x-www-form-urlencoded).

Anfrage:

```
POST https://api.velogista-hh.de/customer
name=Telekom+AG
street=Wandsbeker+Chaussee+666
postcode=22222
city=Hamburg
```

Antwort:

```
{
     'info': {
                    'status':'0',
                    'statusMessage':''
               }
}
```

Mögliche Statusmeldungen

Status	Bedeutung
0	Erfolg
1	Kein Name übermittelt
2	Name wird bereits verwendet

Aktualisieren (PUT)

Aktualisiere bestehende Kund_Innendaten durch Übergabe der Daten in JSON (application/json) oder als HTML-Formular(application/x-www-form-urlencoded). Die Kund_Innen-ID muss gesetzt sein und existieren.

Anfrage:

```
PUT https://api.velogista-hh.de/customer
id=5
name=Musterfirma+AG
street=Mönckebergstrasse+1
postcode=21000
city=Hamburg
```

Antwort:

```
{
     'info': {
                    'status':'0',
                    'statusMessage':''
               }
}
```

Mögliche Statusmeldungen

Status	Bedeutung

0	Erfolg
1	Keine ID übermittelt
2	Kund_In mit übermittelter ID existiert nicht
3	Name wird bereits unter anderer ID verwendet

Löschen (DELETE)

Anfrage:

```
DELETE https://api.velogista-hh.de/customer/id/5
```

Antwort:

```
{
     'info': {
                    'status':'0',
                    'statusMessage':''
                }
}
```

Mögliche Statusmeldungen

Status	Bedeutung
0	Erfolg
1	Keine ID übermittelt
2	Kund_In mit übermittelter ID existiert nicht

Orte

Orte enthalten Informationen über einen Punkt in der physischen Welt und werden als Abhol- und Lieferadressen verwendet. Sie haben grundsätzlich einen Namen und können eine postalische Adresse und ein Paar geographischer Koordinaten beinhalten. Letztere können auch unbekannt sein und über den Geocoding-Dienst ermittelt werden.

Eigenschaft	Datentyp	Beschreibung		
id	Integer	Eindeutige ID des Ortes		
name	String	Name des Ortes		
postalAddress	Objekt	Postadresse des Ortes (wenn bekannt)		
		street	String	Straße und Hausnummer
		postcode	String	Postleitzahl
		city	String	Stadt
		country	String	Land
coordinates	Objekt	Geographische Koordinaten des Ortes (wenn ermittelt)		
		lat	Double	Breitengrad (-90° < lat < 90°)
		lng	Double	Längengrad (-180° < lng < 180°)

Abrufen (*GET)*

Es können einzelne Orte abgerufen werden, die anhand einer ID oder eines Namens eindeutig identifiziert sind.

Einzelne Orte über ID

https://api.velogista-hh.de/location/id/{location_id}

location_id: ID des zu ermittelnden Ortes

Anfrage:

```
GET https://api.velogista-hh.de/location/id/1
```

Antwort:

```
{
      'id' : '1',
      'name' : 'Kasten an der Ecke',
      'postalAddress' : {
             'name' : 'Tante Emmas Warenkiste',
             'street' : 'Waidmannstraße 12b',
             'postcode' : '22222',
             'city' : 'Hamburg'
      },
      'coordinates' : {
             'lat' : '50.12345',
             'lon' : '-10.00023'
      },
      'info': {
                    'status': …,
                    'statusMessage': …
      }
}
```

Mögliche Statusmeldungen

Status	Bedeutung
0	Erfolg
1	Keine ID übermittelt
2	Ort mit übermittelter ID existiert nicht

Einzelne Orte über Name

https://api.velogista-hh.de/location/name/{location_name}

location_name: Name der zu ermittelnden Ortes

Anfrage:

```
GET https://api.velogista-hh.de/location/name/Kasten+an+der+Ecke
```

Antwort:

```
{
      'id' : '1',
      'name' : 'Kasten an der Ecke',
      'postalAddress' : {
             'name' : 'Tante Emmas Warenkiste',
             'street' : 'Waidmannstraße 12b',
             'postcode' : '22222',
             'city' : 'Hamburg'
      },
      'coordinates' : {
             'lat' : '50.12345',
             'lon' : '-10.00023'
      },
      'info': {
                    'status': …,
```

```
                    'statusMessage': …
     }
}
```

Mögliche Statusmeldungen

Status	Bedeutung
0	Erfolg
1	Kein Name übermittelt
2	Ort mit übermitteltem Namen existiert nicht

Ortslisten mit Kriterien

Anfrage:

https://api.velogista-hh.de/location/list/?<criteria=value>&...

Parameter:

- street: Alle Orte aus dieser Straße
- city: Alle Straßen aus dieser Stadt

```
GET https://api.velogista-hh.de/location/list/?city=Hamburg
```

Antwort:

```
[{
     'id' : '1',
     'name' : 'Kasten an der Ecke',
     'postalAddress' : {

     }
 },
 {
  …
 },
 …,
     'info': {
                    'status': …,
                    'statusMessage': …
     }
}
```

Mögliche Statusmeldungen

Status	Bedeutung
0	Erfolg
1	Kriterium wird nicht unterstützt (Kriterium)

Erstellen (POST)

Erstelle neuen Ort durch Übergabe der Daten in JSON (application/json) oder als HTML-Formular(application/x-www-form-urlencoded).

Anfrage:

```
POST https://api.velogista-hh.de/location
name=Penny+Steilshoop
street=Schwarzer+Weg+3
postcode=22309
city=Hamburg
```

Antwort:

```
{
     'info': {
                    'status':'0',
                    'statusMessage':''
                }
}
```

Mögliche Statusmeldungen

Status	Bedeutung
0	Erfolg
1	Kein Name übermittelt
2	Name wird bereits verwendet

Aktualisieren (PUT)

Aktualisiere bestehende Ortsdaten durch Übergabe der Daten in JSON (application/json) oder als HTML-Formular(application/x-www-form-urlencoded). Die Orts-ID muss gesetzt sein und existieren.

Anfrage:

```
PUT https://api.velogista-hh.de/location/
id=2
name=Penny+Steilshoop
street=Mönckebergstrasse+1
postcode=21000
city=Hamburg
```

Antwort:

```
{
     'info': {
                    'status':'0',
                    'statusMessage':''
                }
}
```

Mögliche Statusmeldungen

Status	Bedeutung
0	Erfolg
1	Keine ID übermittelt
2	Ort mit übermittelter ID nicht gefunden
3	Keine Name übermittelt
4	Name wird bereits unter anderer ID verwendet

Löschen (DELETE)

Anfrage:

```
DELETE https://api.velogista-hh.de/location/id/5
```

Antwort:

```
{
     'info': {
```

```
                    'status':'0',
                    'statusMessage':''
                }
}
```

Mögliche Statusmeldungen

Status	Bedeutung
0	Erfolg
1	Keine ID übermittelt
2	Kein Ort mit übermittelter ID gefunden

Routen

Routen beschreiben eine Lieferfahrt, die mehrere Stops zur Abholung oder Ablieferung von Waren enthalten kann.

Eigenschaft	Datentyp	Beschreibung					
id	Integer	Eindeutige ID der Route					
name	String	Name der Route					
stops	Array	Liste aller Stops einer Route, Start- und Endpunkt sind fixiert					
		Für gültige Werte pro Objekt siehe Orte					
optimized	Objekt	Optimierte Route (falls Optimierung bereits erfolgt)					
		time	Integer	Berechnete Dauer für diese Route in Sekunden Je nach verfügbaren Daten können aktuelle Verkehrsbedingungen in diese Schätzung eingehen oder eben nicht.			
		distance	Float	Entfernung in Kilometern			
		stopIndices	Array	Integer-Liste, die die optimierte Reihenfolge der Stops enthält. So kann das locations-Array entsprechend der optimierten Route sortiert werden.			
		shape	Array	Enthält geographische Koordinaten, die das Zeichnen einer Form entlang dieser Route ermöglichen			
		legs	Array	Enhält die Streckenabschnitte zwischen den einzelnen Stops in bereits optimiert sortierter Reihenfolge passend zu locationIndices			
				distance	Float	Entfernung in Kilometern	
				time	Integer	Dauer für diesen Abschnitt in Sekunden	
				destIndex	Integer	Index des Ziels dieses Abschnittes	
				maneuvers	Array	Fahranweisungen	
						signUrl	String	Passendes Icon zu diesem Maneuver
						narrative	String	Textuelle Anweisung

Abrufen (*GET*)

Es können einzelne Routen abgerufen werden, die anhand einer ID oder eines Namens eindeutig identifiziert sind.

Einzelne Routen über ID

https://api.velogista-hh.de/route/id/{route_id}

route_id: ID der zu ermittelnden Route

Anfrage:

```
GET https://api.velogista-hh.de/route/id/1
```

Antwort:

Wenn die Route gefunden wurde, enthält die Antwort mindestens 'id', 'name' und ein leeres 'locations'-Array. Sind Orte hinterlegt und auch eine optimierte Route berechnet worden, kann die Antwort wie folgt aussehen:

```
{
      'id' : '1',
      'name' : 'BIOBOBs Freitagtour',
      'locations' : [{
             'id' : '1',
             'name' : 'Tante Emmas Warenkiste',
             'street' : 'Waidmannstraße 12b',
             'postcode' : '22222',
             'city' : 'Hamburg'
                       },
                       …
                       ],
      'locationIndices' : [ 0,4,2,3,1,5 ],
      'legs' : [{
                  },
                  …
                ],
      'info': {
                    'status': …,
                    'statusMessage': …
      }
}
```

Mögliche Statusmeldungen

Status	Bedeutung
0	Erfolg
1	Keine ID übermittelt
2	Keine Route mit übermittelter ID gefunden

Erstellen (POST)

Erstelle neue Route durch Übergabe der Daten in JSON (application/json) oder als HTML-Formular(application/x-www-form-urlencoded).

Routendaten hinterlegen

Anfrage:

```
POST https://api.velogista-hh.de/route
name=Unsere+erste+Route
stops=1,4,7,8
```

Antwort:

```
{
      'info': {
```

```
                    'status':'0',
                    'statusMessage':''
                }
}
```

Mögliche Statusmeldungen

Status	Bedeutung
0	Erfolg
1	Kein Name übermittelt
2	Name wird bereits bei anderer Route verwendet
3	Keine oder zu wenig Stops übermittelt

Route optimieren

Nachdem eine Route hinterlegt wurde, kann über diese Aktion eine optimierte Route zu diesen Daten ermittelt werden.

Anfrage:

Mögliche Optimierungsoptionen:

Parameter	Beschreibung
routeType	• **fastest** - Schnellste Fahrzeit • **shortest** - Kürzeste Strecke • **pedestrian** - Keine Autostraßen, keine Abbiegebeschränkungen • **bicycle** - Nur Straßen, die Fahrräder erlauben
locale	Sprachumgebung für Fahranweisungen - ISO 639-1 Code - Eine Auswahl: Deutsch(Deutschland) = `de_DE(default)` Deutsch(Schweiz) = `de_CH` Englisch(USA) = `en_US` Englisch(Großbritannien) = `en_GB` Französisch(Kanada) = `fr_CA` Französisch(Frankreich) = `fr_FR` Spanisch(Spanien) = `es_ES` Spanisch(Mexiko) = `es_MX` Russisch(Russland) = `ru_RU`

```
PUT https://api.velogista-hh.de/route/optimize/1
routeType=fastest
locale=fr_FR
```

Antwort:

```
{
     'info': {
                    'status':'0',
                    'statusMessage':''
                }
}
```

Mögliche Statusmeldungen

Status	Bedeutung
0	Erfolg
1	Keine ID übermittelt

2	Keine Route mit übermittelter ID gefunden
3	Optimierungsoption wird nicht unterstützt (option)
4	Wert für Optimierungsoption wird nicht unterstützt (option=wert)

Aktualisieren (PUT)

Aktualisiere bestehende Route durch Übergabe der Daten in JSON (application/json) oder als HTML-Formular(application/x-www-form-urlencoded). Die Routen-ID muss gesetzt sein und existieren.

Achtung: Eine eventuell existierende, optimierte Route wird entfernt!

Anfrage:

```
PUT https://api.velogista-hh.de/route
id=2
name=Unsere+neue+Route
stops=1,4,7,1
```

Antwort:

```
{
      'info': {
                        'status':'0',
                        'statusMessage':''
                }
}
```

Mögliche Statusmeldungen

Status	Bedeutung
0	Erfolg
1	Kein Name übermittelt
2	Name wird bereits bei anderer Route verwendet
3	Keine oder zu wenig Stops übermittelt

Löschen (DELETE)

Anfrage:

```
DELETE https://api.velogista-hh.de/route/id/5
```

Antwort:

```
{
      'info': {
                        'status':'0',
                        'statusMessage':''
                }
}
```

Mögliche Statusmeldungen

Status	Bedeutung
0	Erfolg
1	Keine ID übermittelt
2	Keine Route mit übermittelter ID gefunden

Karten

Dieser Service erstellt Kartenansichten in Form von Bilddateien mit festzulegenden Abmessungen. Der zu erstellende Ausschnitt kann durch Kartenmittelpunkt und Zoomfaktor oder passend zu einer hinterlegten Route festgelegt werden. Der Streckenverlauf optimierter Routen kann dabei visualisiert werden. Erstellte Karten können vom Service im Hintergrund gecacht werden, um Antwortzeiten und die Anzahl an Anfragen an Drittdienste zu minimieren.

Der Typ der generierten Bilddatei kann abhängig sein vom verwendeten Kartenanbieter. jpg-Dateien werden bevorzugt. Je nach Dienst können aber auch png-, oder gif-Dateien generiert werden.

Da Karten (abgesehen vom Cache) immer dynamisch erstellt werden, existiert hier nur eine GET-Methode. In Zukunft könnten Karten fest (unter eigener ID) im System hinterlegt und damit weitere Methoden nötig werden.

Es kann Sinn machen, die gecachten Versionen von visualisierten Routen mit den entsprechend erstellten optimierten Routen zu verknüpfen. Wird eine optimierte Route gelöscht oder aktualisiert, sollte auch der dazugehörige Kartenausschnitt aktualisiert werden.

Abrufen (*GET*)

Basis-URL: https://api.velogista-hh.de/map

Allgemeine Parameter:

Parameter	Beschreibung	Beispiel
size	Abmessungen der Bilddatei (Angabe immer erforderlich)	size=640,480
type	Typ der Karte • **map** - Straßenkarte (Standard) • **sat** - Satellitenkarte • **hyb** - Kombination	type=map
image	Typ der Bilddatei • **jpg** (Standard) • **gif** • **png**	image=jpg

Mittels Kartenmittelpunkt und Zoomfaktor

Basis-URL: https://api.velogista-hh.de/map/static

Anfrage:

Parameter	Beschreibung	Beispiel
center	Mittelpunkt der zu erstellenden Karte	53.60609,10.05402
zoom	Zoomfaktor für Karte (1-18)	12

```
GET https://api.velogista-hh.de/map/static?size=640,480&center=53.60609,10.05402&zoom=12
```

Antwort:

Bei Erfolg:

```
[entsprechende Bilddatei, was einfügen]
```

Bei Misserfolg:

JSON mit *info*-Eigenschaft:

Status	Bedeutung
1	Abmessungen fehlerhaft (zB negative Werte oder 0, oder außerhalb des Bereichs des Drittdienstes)
2	Mittelpunkt fehlerhaft (keine korrekten, geographischen Koordinaten)
3	Zoom fehlerhaft, außerhalb des Bereichs 1-18

Mittels hinterlegter optimierter Route

Basis-URL: https://api.velogista-hh.de/map/route

Anfrage:

Parameter	Beschreibung	Beispiel
id	ID der Route zu der eine Karte erstellt werden soll	id=1
margin	Mindestrand um die visualisierte Route	margin=50

```
GET https://api.velogista-hh.de/map/route?id=1&size=640,480&margin=50
```

Antwort:

Bei Erfolg:

```
[entsprechende Bilddatei, was einfügen]
```

Bei Misserfolg:

JSON mit *info*-Eigenschaft:

Status	Bedeutung
1	Abmessungen fehlerhaft (zB Wert <= 0, oder außerhalb des Bereichs des Drittdienstes)
2	Mindestrand fehlerhaft (zB Wert <= 0, oder außerhalb des Bereichs des Drittdienstes)
3	Keine Route mit übermittelter ID gefunden