Web Services: Manual

Cedric De Schepper

November 2018

1 Libraries

Python packages necessary to run the project:

- requests
  - Used to easily make HTTP/1.1 requests
  - url: http://docs.python-requests.org/en/master
- flask & flask_restful
  - microframework to create webservices / RESTful APIs
  - url flask: http://flask.pocoo.org/
  - url flask_restful: https://flask-restful.readthedocs.io/en/latest/index.html

I have included a requirements.txt file that includes all required packages.
You can install these with: "pip install -r requirements.txt".

2 APIs used

- DeLijn
  - Url: https://data.delijn.be
  - Used for the necessary public transport information.
- HERE
  - Url: https://developer.here.com
  - Used for its Routing and Weather API.

3 web services

All of my web services require the same 3 parameters so I will discuss them here to avoid unnecessary repetition.

Parameter	Type	Description
province	string	province of the line
line number	integer	number of the line
direction	string	direction of the line

Province has to be the correct name of a Flemish province thus being an element of the set {Antwerpen, West-Vlaanderen, Oost-Vlaanderen, Limburg, Vlaams-Brabant}. The capitalization of letters does not matter but the correct position of the hyphen (-) is mandatory.

The direction is either "HEEN" or "TERUG" (Dutch words for "back and forth"). Once again, the capitalization is not important.

A request is considered correct if all arguments are correct and the line number exists for the given province/direction.

3.1 stops

Returns the information, such as location and weather description, of the stops on a given line.

Request url [GET]: *'api/stops/{province}/{line number}/{direction}'*

Output after correct request:

```
{
 "stops": [
    {
    "coordinates": {
        "lat": STOP LATITUDE,
        "lng": STOP LONGITUDE
        },
    "description": STOP DESCRIPTION/NAME,
    "stopNumber": STOP NUMBER,
    "weather": WEATHER DESCRIPTION
    },
    ...
  ]
}
```

Output after incorrect request:

```
{
 "stops": []
}
```

These stops get calculated by requesting the stop information of DeLijn. The weather for the stops gets calculated for every town the stops are in. This gives accurate weather insight without too time consuming.

3.2 buses

Returns the information, such as location and next stop, of the buses on a given line.

Request url [GET]: *'api/buses/{province}/{line number}/{direction}'*

Output after correct request:

```
{
 "buses": [
    {
    "coordinates": {
        "lat": BUS LATITUDE,
        "lng": BUS LONGITUDE
        },
    "nextStop": NEXT STOP DESCRIPTION,
    "number": BUS NUMBER
    },
    ...
  ]
}
```

Output after incorrect request:

```
{
 "buses": []
}
```

The bus locations get calculated by using the time schedule of the DeLijn and routing information by two steps provided by HERE. Firstly, the two stops a bus is in between get calculated. Then we calculate the description

of the route between those two points. By using the different direction, we can get an estimate of the location the bus is at. I don't calculate coordinates myself since this easily allows for buses to be at a coordinate that's not on the route drawn by HERE on the website.

3.3 route

Does not return the route taken by buses of a given line but returns the routing parameters necessary for drawing the route by HERE routing.

Request url [GET]: *'api/route/{province}/{line number}/{direction}'*

Output after correct request:

```
{
 "parameters": {
    "maneuverattributes": "direction,action"
    "mode": "fastest;publicTransport;traffic:disabled",
    "representation": "display",
    "routeattributes": "waypoints",
    "waypoint0": "geo!LATITUDE,LONGITUDE",
    "waypoint1": "geo!LATITUDE,LONGITUDE",
    ...
    "waypointN": "geo!LATITUDE,LONGITUDE",
    }
}
```

Output after incorrect request:

```
{
 "parameters": {}
}
```

Most importantly here are the different waypoints. These will force the route to go through these points, resulting in a fairly accurate estimate of the route taken by a bus. HERE will use the resulting parameters to calculate the route and draw it on the map.

4 simple website

The user interface for my website is very basic and contains only 2 major elements. Firstly, there is a form where a user can submit the 3 mandatory

arguments: province, line number and direction.

Then we have the map of Flanders where all the information will be shown. After submitting the right parameters, the map will zoom to the correct route. It will also display the route, stops and current buses. Pressing on a stop or bus will open an info bubble with some details about the bus/stop. Be advised, you have to close these bubbles by hand. If you submit a wrong request (line number that does not exist), the map will reset.

Page examples are on the next page.

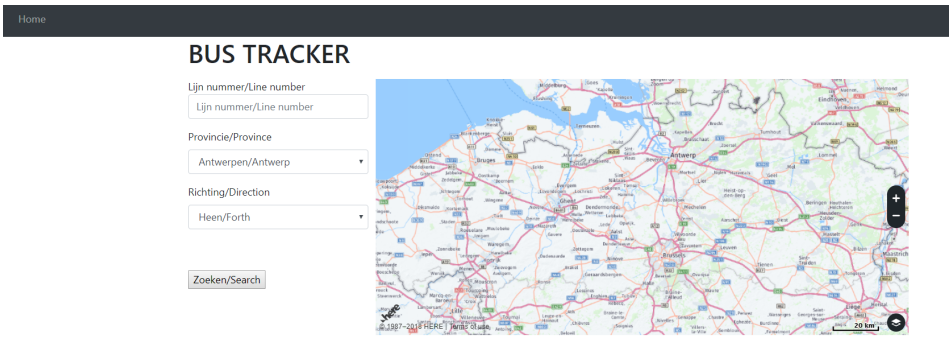

Figure 1: Start page

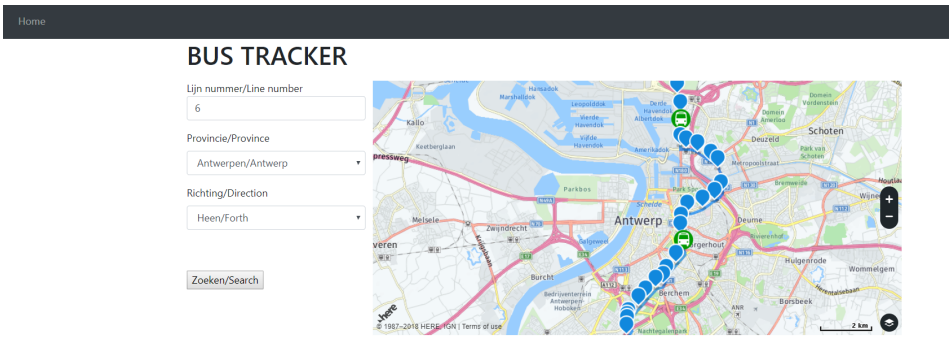

Figure 2: Correct request result

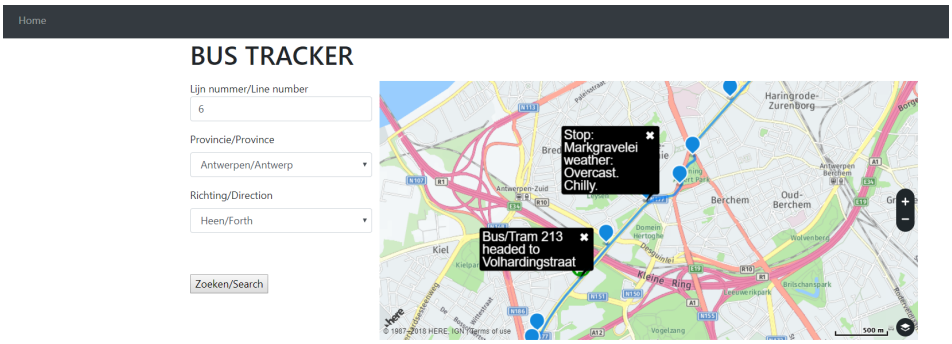

Figure 3: Info bubbles