openrouteservice - route the world dynamically

Nils Nolde

HeiGIT - University of Heidelberg

openroute service

HeiGIT | HEIDELBERG INSTITUTE FOR GEOINFORMATION TECHNOLOGY

KTS KLAUS TSCHIRA STIFTUNG GEMEINNÜTZIGE GMBH

UNIVERSITÄT HEIDELBERG ZUKUNFT SEIT 1386

Fact Sheet

- **openrouteservice** is a project of the KTS backed HeiGIT institute at the University of Heidelberg
- started in 2008, **openrouteservice** was one of the first routing API's
- entirely **open-source**, backend written in **Java**
- based on **OpenStreetMap** data
- 2500 requests per day **for free**

Available API's

Directions

Isochrones

Geocoding

Matrix

Places

openroute service

HeiGIT | HEIDELBERG INSTITUTE FOR GEOINFORMATION TECHNOLOGY

KTS KLAUS TSCHIRA STIFTUNG GEMEINNÜTZIGE GMBH

UNIVERSITÄT HEIDELBERG ZUKUNFT SEIT 1386

Local installation

License: Apache 2.0

```
git clone https://github.com/GIScience/openrouteservice
cd docker && sudo docker-compose up
```

Approx. **64-128 GB RAM** per profile for `planet.pbf`.

Geocoding and **Places** need special setup:

https://github.com/pelias/pelias

https://github.com/GIScience/openpoiservice

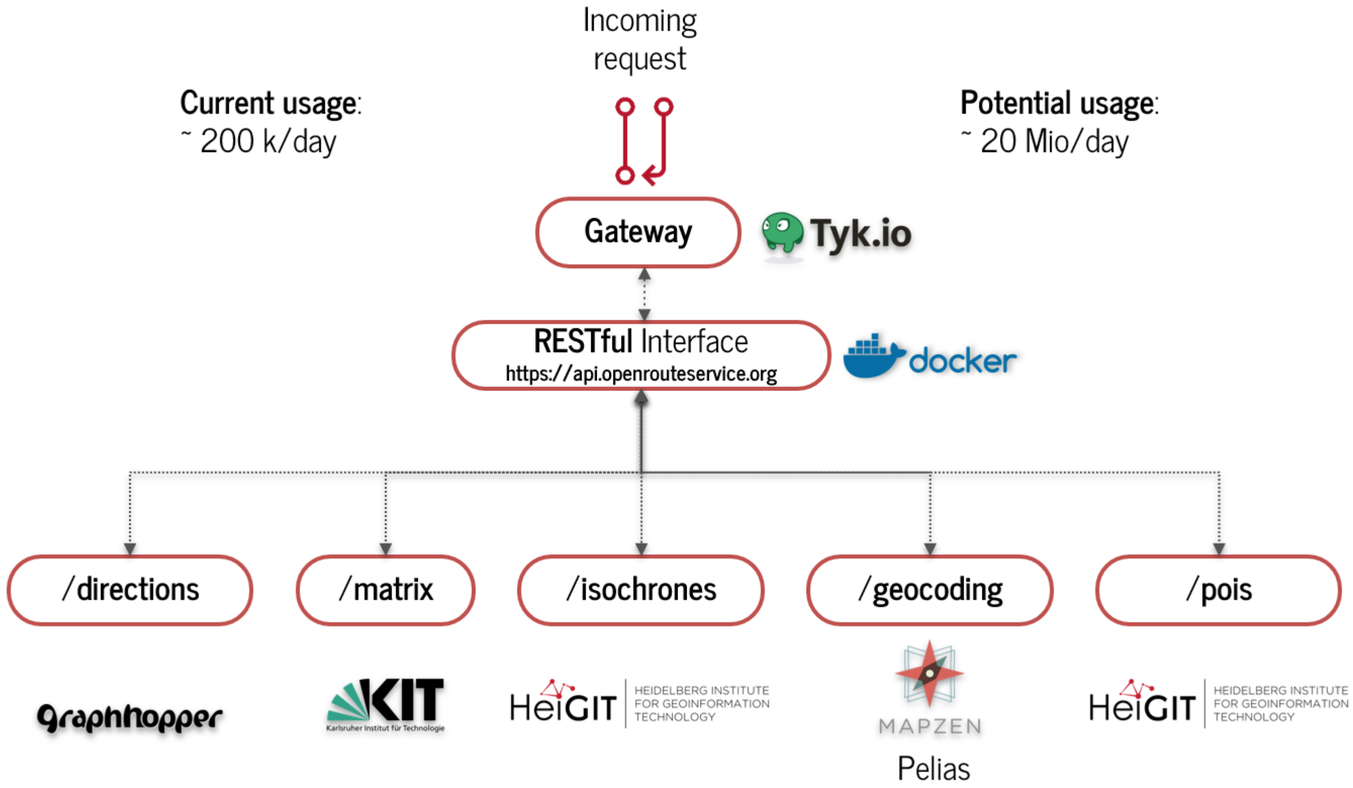
Architecture

openrouteservice

HeiGIT | HEIDELBERG INSTITUTE FOR GEOINFORMATION TECHNOLOGY

KTS
KLAUS TSCHIRA STIFTUNG GEMEINNÜTZIGE GMBH

UNIVERSITÄT HEIDELBERG
ZUKUNFT SEIT 1386



openroute service

HeiGIT HEIDELBERG INSTITUTE FOR GEOINFORMATION TECHNOLOGY

KTS KLAUS TSCHIRA STIFTUNG GEMEINNÜTZIGE GMBH

UNIVERSITÄT HEIDELBERG ZUKUNFT SEIT 1386

Infrastructure

Clients

Python client

Github PyPI RTD

R client

Github Docu

QGIS plugin

Github Plugin repo

Web app

Github

Microservices

openpoiservice

Flask application to serve OSM Places of Interest (POI's) via a Rest API.

Features

- fully self-contained microservice to create, drop and populate PostGIS tables
- `POST` endpoint, which accepts GeoJSON as geometry and OSM tags as filters
- Customizable setup with Docker container

Github

Coming Soon!!

openfuelservice

Flask application to estimate fuel consumption on > 600 car models. Takes GeoJSON and car model as input.

openroute service

HeiGIT HEIDELBERG INSTITUTE FOR GEOINFORMATION TECHNOLOGY

KTS KLAUS TSCHIRA STIFTUNG GEMEINNÜTZIGE GMBH

UNIVERSITÄT HEIDELBERG ZUKUNFT SEIT 1386

Directions API

Classical routing API, based on the excellent GraphHopper stack.

Features

- 10 routing profiles:
  - 2 car: `driving-car`, `driving-hgv`
  - 5 cycling, incl. `*-mountain`, `*-safe`
  - 2 walking: `foot-walking`, `foot-hiking`
  - **`wheelchair`**
- Output formats: `geojson`, `gpx`
- 13 languages for instructions
- Returns extra way information for route segments, e.g. steepness, surface, tollways

Functionalities

- `bearing`, `radiuses`: Control input point snapping behavior
- `continue_straight`: Avoid u-turns, even if faster
- `avoid_*`: Avoid features, e.g. ferry, **countries**, **polygons**
- vehicle dimensions: use OSM dimension tags to route compliantly
- **green/quiet**: prefer green and/or quiet routes (**only** available in Germany right now)
- **landmark** backed pedestrian navigation (**only** available in Germany)

openroute service

HeiGIT HEIDELBERG INSTITUTE FOR GEOINFORMATION TECHNOLOGY

KTS KLAUS TSCHIRA STIFTUNG GEMEINNÜTZIGE GMBH

UNIVERSITÄT HEIDELBERG ZUKUNFT SEIT 1386

Directions - Avoid features for `cycling-regular`

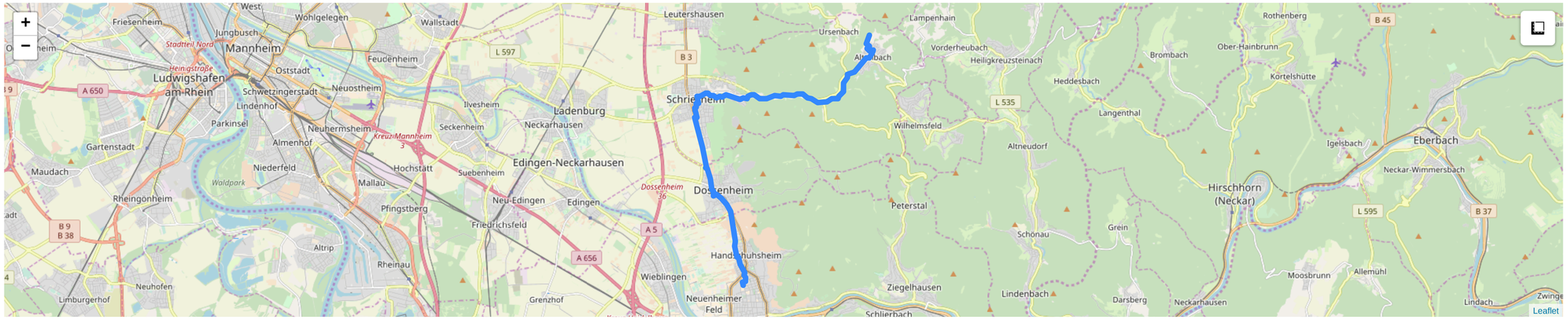
In [4]:

```
import openrouteservice as ors

ors_key = api_key["ORS"] # keys are stored in local file
clnt = ors.Client(key=ors_key) # set up client
# This function is called by the widget's event callback
def avoidHills(mode):
    params = {'coordinates': [[8.681602, 49.422141],[8.737221, 49.49333]],
              'profile': 'cycling-regular',
              'format_out': 'geojson'}
    if mode != 'none':
        params['options'] = {'avoid_features': mode}
    route = clnt.directions(**params)
    return route

widget_hills
```

Avoid: hills | unpavedroads | none



openroute service

HeiGIT HEIDELBERG INSTITUTE FOR GEOINFORMATION TECHNOLOGY

KTS KLAUS TSCHIRA STIFTUNG GEMEINNÜTZIGE GMBH

UNIVERSITÄT HEIDELBERG ZUKUNFT SEIT 1386

Directions - Avoid countries for `driving-hgv`

In [20]:

```python
import openrouteservice as ors

ors_key = api_key["ORS"] # comes from local file
clnt = ors.Client(key=ors_key) # set up client
# This function is called by the widget's event callback
def avoidCountry(country):
    country = '|'.join([str(c) for c in country])
    params = {'coordinates': [[7.987061, 47.682032],[9.733887, 47.090696]],
              'profile': 'driving-hgv',
              'format_out': 'geojson',}
    if country != 'None':
        params['options'] = {'avoid_countries': country}
    route = clnt.directions(**params)
    return route

widget_country
```

Countries
DE
CH
AT
None

openroute service

HeiGIT HEIDELBERG INSTITUTE FOR GEOINFORMATION TECHNOLOGY

KTS KLAUS TSCHIRA STIFTUNG GEMEINNÜTZIGE GMBH

UNIVERSITÄT HEIDELBERG ZUKUNFT SEIT 1386

Directions - Route green or quiet for `foot-walking`

In [22]:

```python
import openrouteservice as ors

ors_key = api_key["ORS"] # comes from local file
clnt = ors.Client(key=ors_key) # set up client
# This function is called by the widget's event callback
def routeGreen(pref):
    params = {'coordinates': [[8.690872, 49.406673],[8.692117, 49.430124]],
              'profile': 'foot-walking',
              'optimized': 'false',
              'format_out': 'geojson'}
    if pref != 'none':
        params['options'] = {'profile_params': {'weightings': {pref: {'factor': 1.0}}}}
    route = clnt.directions(**params)
    return route

widget_green
```

Preference: green | quiet | none

openroute service

HeiGIT HEIDELBERG INSTITUTE FOR GEOINFORMATION TECHNOLOGY

KTS KLAUS TSCHIRA STIFTUNG GEMEINNÜTZIGE GMBH

UNIVERSITÄT HEIDELBERG ZUKUNFT SEIT 1386

Isochrones API

Features

- 9 routing profiles:
  - 2 car: `driving-car`, `driving-hgv`
  - 5 cycling, incl. `*-mountain`, `*-safe`
  - 2 walking: `foot-walking`, `foot-hiking`
- Query up to 5 locations and 10 intervals in one request

Functionalities

- **`attributes`**: Returns e.g. total population within isochrone[1]
- plus similar functionalities as **Directions** API

[1] Global data coverage by EU GSH

openroute service

HeiGIT HEIDELBERG INSTITUTE FOR GEOINFORMATION TECHNOLOGY

KTS KLAUS TSCHIRA STIFTUNG GEMEINNÜTZIGE GMBH

UNIVERSITÄT HEIDELBERG ZUKUNFT SEIT 1386

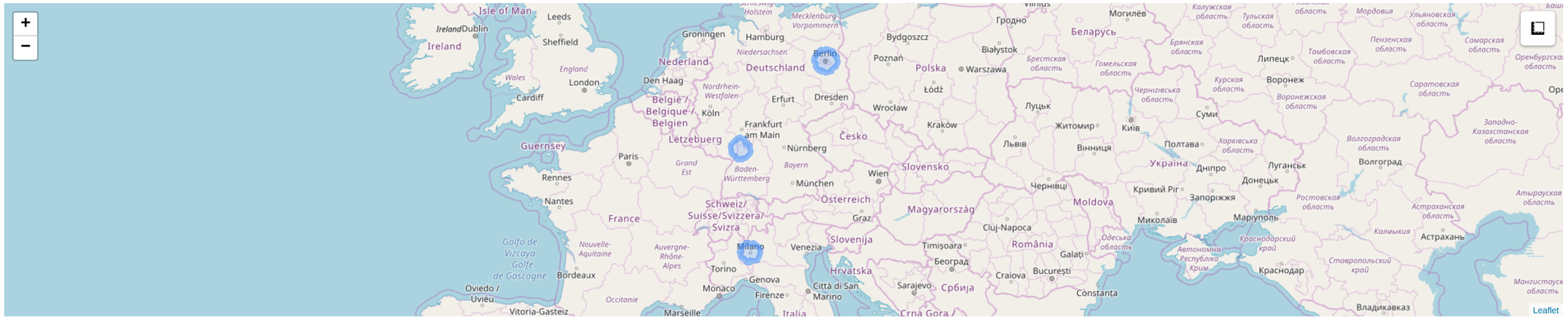
Isochrones - Retrieve population

In [24]:
```python
import openrouteservice as ors

ors_key = api_key["ORS"] # comes from local file
clnt = ors.Client(key=ors_key) # set up client
# This function is called by the widget's event callback
def calculateIso(dim):
    number = 1800 if dim == 'time' else 50000
    # Milan, Heidelberg, Berlin
    params = {'locations': [[9.170837, 45.47554],[8.684692, 49.409186], [13.41332, 52.521626]],
              'profile': 'driving-car',
              'range_type': dim,
              'intervals': [number],
              'segments': number,
              'attributes': ['total_pop']}
    iso = clnt.isochrones(**params)
    return iso

widget_iso
```

Type: time distance



openroute service

HeiGIT HEIDELBERG INSTITUTE FOR GEOINFORMATION TECHNOLOGY

KTS KLAUS TSCHIRA STIFTUNG GEMEINNÜTZIGE GMBH

UNIVERSITÄT HEIDELBERG ZUKUNFT SEIT 1386

Geocoding API

Fork of [Pelias](#) geocoding engine.

Features

- 3 endpoints:
  - `geocoding/search?`
  - `geocoding/reverse?`
  - `geocoding/autocomplete?`
- 4 data sources:
  - **WhosOnFirst**: Global administrative boundaries
  - **OSM**
  - **openaddresses**: ~ 500 Mio addresses
  - **geonames**: Additional place names and locations

Functionalities

- **libpostal**: address parser and normalizer
- **placeholder**: address parser for unstructured text
- **interpolation**: attempts interpolation for unknown house numbers

These extra services are stand-alone API's and can be used in any application.

openroute service

HeiGIT | HEIDELBERG INSTITUTE FOR GEOINFORMATION TECHNOLOGY

KTS KLAUS TSCHIRA STIFTUNG GEMEINNÜTZIGE GMBH

UNIVERSITÄT HEIDELBERG ZUKUNFT SEIT 1386

Geocoding - Running out of names?!

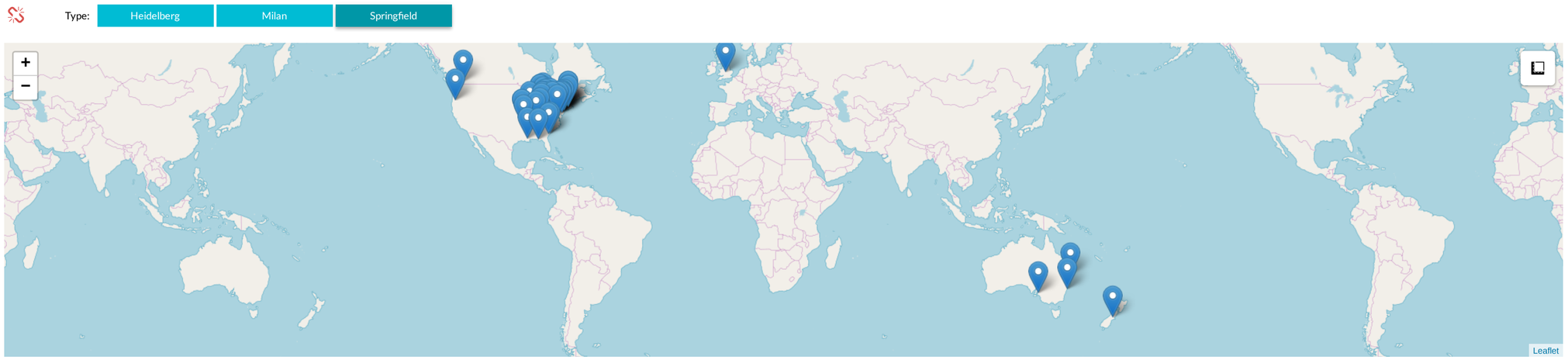
In [14]:

```
import openrouteservice as ors

ors_key = api_key["ORS"] # comes from local file
clnt = ors.Client(key=ors_key) # set up client
# This function is called by the widget's event callback
def geocode(name):
    route = clnt.pelias_search(name, size=50)
    return route

widget_geocode
```

```
Amount of cities worldwide:     40
Amount of cities in US:         34
```



openroute service

HeiGIT HEIDELBERG INSTITUTE FOR GEOINFORMATION TECHNOLOGY

KTS KLAUS TSCHIRA STIFTUNG GEMEINNÜTZIGE GMBH

UNIVERSITÄT HEIDELBERG ZUKUNFT SEIT 1386

Matrix API

Features

- Calculates OD matrices for up to **50x50 locations**
- For all 9 profiles
- Both `distance` and `duration` are returned

POI API

Features

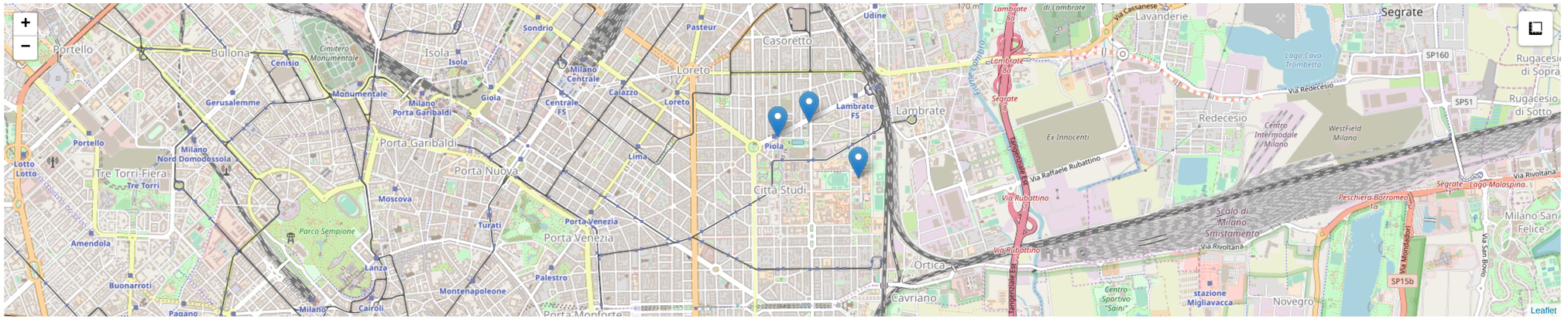
- Query OSM POI's for up to 5 categories in one request
- Use GeoJSON geometries or bbox to filter geographically
- Use OSM tags to filter POI's, e.g. `smoking: no`, `wheelchair: yes`
- Returns up to 2000 POI's per request

POIs - What's around us

In [28]:
```
import openrouteservice as ors
from geojson import Point

ors_key = api_key["ORS"] # comes from local file
clnt = ors.Client(key=ors_key) # set up client
# This function is called by the widget's event callback
def getPois(location):
    params = {'request': 'pois',
              'geojson': Point(location),
              'buffer': 1000, # 1km radius around SOTM location
#               'filter_category_ids': [570],
              'sortby': 'distance'}
    route = clnt.places(**params)
    return route
pois = getPois(milan)
options = populate_list(pois)
widget_pois = widgets.Dropdown(options=options)
widget_pois.observe(on_change_pois, names='value')
widget_pois
```

atm



openroute service

HeiGIT | HEIDELBERG INSTITUTE FOR GEOINFORMATION TECHNOLOGY

KTS KLAUS TSCHIRA STIFTUNG GEMEINNÜTZIGE GMBH

UNIVERSITÄT HEIDELBERG ZUKUNFT SEIT 1386

You know what's on for dinner?

Let's see how OSM and Yelp make the restaurant choice a little easier:

- Find restaurants within 10 min walking distance from SOTM location
- Filter restaurants with Yelp rating >= 4
- Create widget to filter by restaurant category

Set up *openrouteservice* client

1. Sign up on our homepage
2. Create API key on our developer portal
3. Review our ToS
4. `pip install openrouteservice`
5. Fire away!

Here we:

- ***Geocode*** the SOTM official address
- Create an ***isochrone*** with 10 mins walking radius

In [17]:
```
import openrouteservice as ors

clnt = ors.Client(key='5b3ce3597851110001cf624870cf2f2a58d44c718542b3088221b684')

params_geocode = {'text': 'Politecnico di Milano - Piazza Leonardo da Vinci, Milan, Lombardy, Italy'}

sotm_location = clnt.pelias_search(**params_geocode)['features'] # Geocode from official address
sotm_isochrone = clnt.isochrones(locations=sotm_location[0]['geometry']['coordinates'], # Calculate isochrones
                                 intervals=[600],
                                 segments=600,
                                 profile='foot-walking')
```

openroute service

HeiGIT HEIDELBERG INSTITUTE FOR GEOINFORMATION TECHNOLOGY

KTS KLAUS TSCHIRA STIFTUNG GEMEINNÜTZIGE GMBH

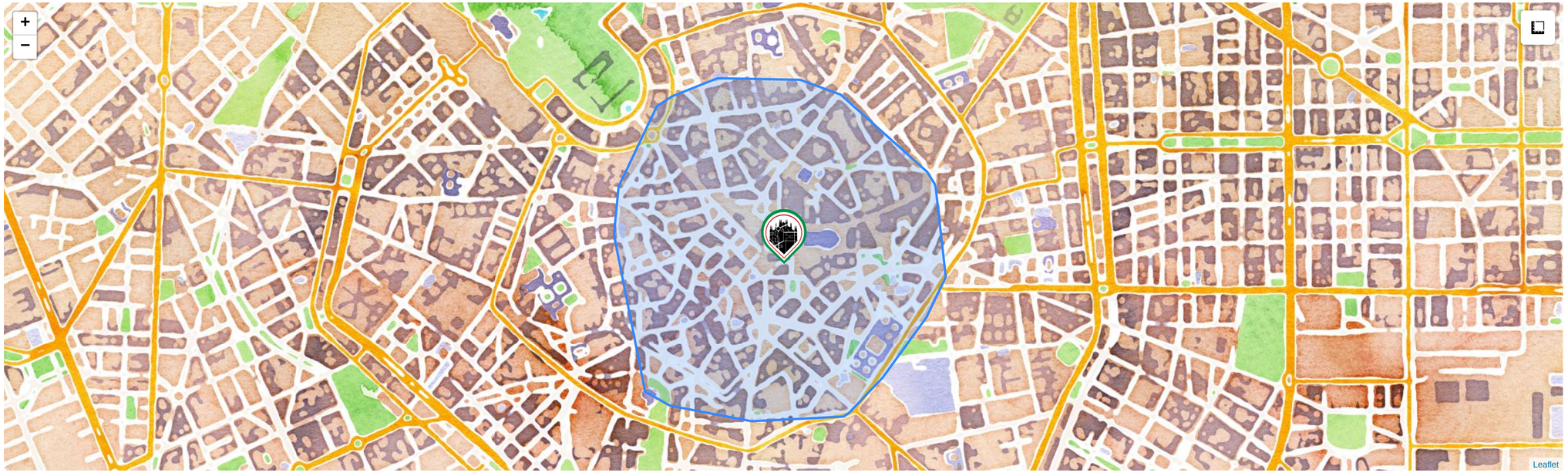
UNIVERSITÄT HEIDELBERG ZUKUNFT SEIT 1386

Set up a basic `folium` map

- The `createWidgetBaseMap` factory sets up the initial map, so we start fresh every time

In [19]:
```
fig, m = createWidgetBaseMap('100%', 600) # Call map factory with figure dimensions
fig.add_child(m)
fig
```

Out[19]:



openroute service

HeiGIT HEIDELBERG INSTITUTE FOR GEOINFORMATION TECHNOLOGY

KTS KLAUS TSCHIRA STIFTUNG GEMEINNÜTZIGE GMBH

UNIVERSITÄT HEIDELBERG ZUKUNFT SEIT 1386

Request restaurant POI's

- `/pois?` endpoint takes GeoJSON
- we can pipe the GeoJSON output of `/isochrones?` directly to `/pois?` to limit the query to the isochrones geometry

In [20]:
```
param_pois = {'request': 'pois',
              'geojson': sotm_isochrone['features'][0]['geometry'],
              'filter_category_ids': [570], # ID list: https://github.com/GIScience/openrouteservice-docs#sustenance--560
              'sortby': 'distance'}

sotm_restaurants = clnt.places(**param_pois)['features']
display(HTML('<br><br>Amount of restaurants within 10 mins walking radius: <strong>{}</strong>'.format(len(sotm_restaurants))))
```

Amount of restaurants within 10 mins walking radius: **121**

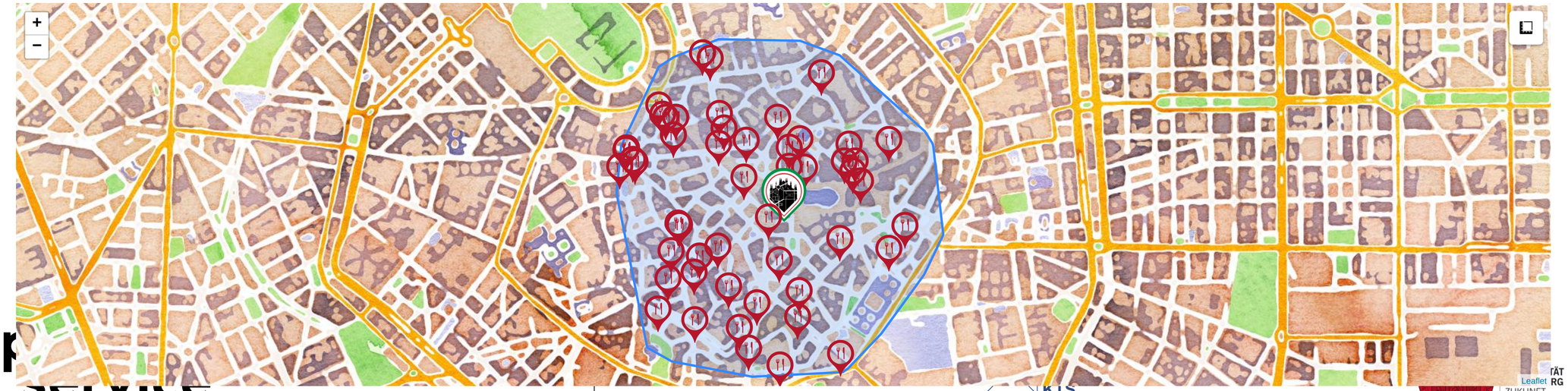
Yelp rating

- use the Yelp businesses API to query the reviews of our restaurants
- extract `name`, `rating` and `url` to feature popup

In [22]:
```
import branca

fig, m = createWidgetBaseMap('100%', '500')
for restaurant in sotm_restaurants:
    r_name = restaurant['properties']['osm_tags'].get('name', '')
    if r_name != '':
        r_coords = restaurant['geometry']['coordinates']
        yelp_return = yelp_response(r_name=r_name, # Call hidden function to retrieve Yelp results with rating >= 4 stars
                                    r_coords=r_coords)
        if yelp_return:
            restaurant['yelp'] = yelp_return
            restaurant_icon = folium.features.CustomIcon('https://openrouteservice.org/wp-content/uploads/2017/07/restaurant2.png',
                                                         icon_size=(50, 50))
            popup_html = '<h4>{0}</h4><strong><a href="{1}" target="_blank">URL</a><br>Stars: </strong>{2}'.format(r_name, yelp_return['url'], yelp_return['rating'])
            iframe = branca.element.IFrame(html=popup_html, width=300, height=100)
            popup = folium.Popup(iframe, max_width=300)
            folium.map.Marker(list(reversed(r_coords)), icon=restaurant_icon, popup=popup).add_to(m)
fig.add_child(m)
fig
```

Out[22]:



op

service

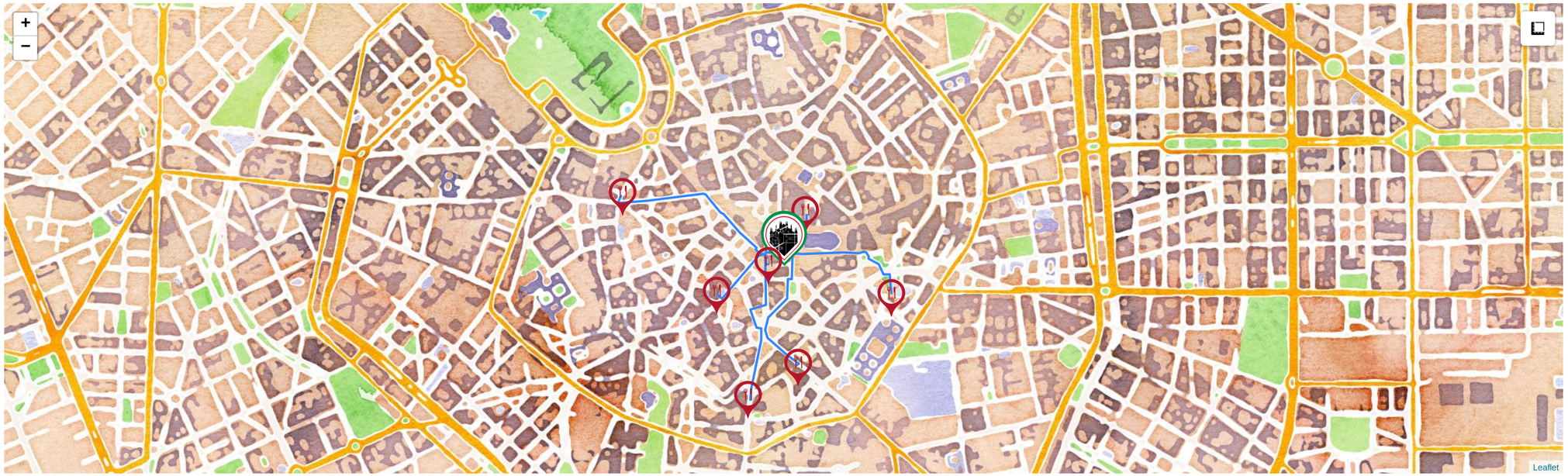
IPython widget

The final product is a drop-down widget to query the directions to nearby restaurants by restaurant category:

In [25]:
```
category_titles = []
for restaurant in sotm_restaurants:
    if restaurant.get('yelp', '') != '':
        category_titles.append(restaurant['yelp']['categories'][0]['title'])

widget_dd = widgets.Dropdown(options=set(category_titles)) # Create widget with categories in drop-down
widget_dd.observe(on_category_change, names='value') # Hidden method is called by the listener
display(widget_dd)
```

Pizza



openroute service

HeiGIT HEIDELBERG INSTITUTE FOR GEOINFORMATION TECHNOLOGY

KTS KLAUS TSCHIRA STIFTUNG GEMEINNÜTZIGE GMBH

UNIVERSITÄT HEIDELBERG ZUKUNFT SEIT 1386

Thanks to...

- **Pelias** and **GraphHopper** crews for developing cutting-edge open-source tech!
- **Damian Avila** for the `rise` development which powers this presentation
- **Jupyter** development crew
- **Yelp** for providing a free attractions API

The source code and PDF of this presentation can be found here:

https://github.com/nilsnolde/sotm2018

openroute service

HeiGIT | HEIDELBERG INSTITUTE FOR GEOINFORMATION TECHNOLOGY

KTS KLAUS TSCHIRA STIFTUNG GEMEINNÜTZIGE GMBH

UNIVERSITÄT HEIDELBERG ZUKUNFT SEIT 1386

Sign up for 2500 requests/day for free!

[Github ORS backend]
[Github ORS frontend]

Heidelberg Institute for Geoinformation Technology (HeiGIT)
Berliner Str. 45 (Mathematikon)
69120 Heidelberg

 @nilsnolde

 @realNilsNolde

Nils Nolde

Software & business development

openroute service

HeiGIT | HEIDELBERG INSTITUTE FOR GEOINFORMATION TECHNOLOGY

KTS KLAUS TSCHIRA STIFTUNG GEMEINNÜTZIGE GMBH

UNIVERSITÄT HEIDELBERG ZUKUNFT SEIT 1386