Lets look at trains with rpm values zero that are not within the proximity of a train station

In [22]:
```
import pandas as pd
from sqlalchemy import create_engine, text



dbname = 'DataMining'
user = 'postgres'
password = 'datamining'
host = 'localhost'  # localhost or the server address
port = '5433'  # default PostgreSQL port is 5432

# Establish a connection to the database
connection_str = f"postgresql://{user}:{password}@{host}:{port}/{dbname}"
engine = create_engine(connection_str)
```

In [16]:
```
query = """
select * from vehicle_data
where (rs_e_rpm_pc1 =0) and (rs_e_rpm_pc2=0)
;
"""

# Execute the query and fetch the data into a DataFrame
data = pd.read_sql_query(query, engine)
```

Let us find the locations of all train stations

In [17]:
```
import overpy
import pandas as pd
from geopy.distance import great_circle

# Initialize Overpass API
api = overpy.Overpass()

# Query for train stations in Belgium (modify the area as needed)
query = """
area["ISO3166-1"="BE"][admin_level=2];
node["railway"="station"](area);
out;
"""
result = api.query(query)

# Extract train station locations
stations = pd.DataFrame([{
    'lat': float(node.lat),
    'lon': float(node.lon)
} for node in result.nodes])
```

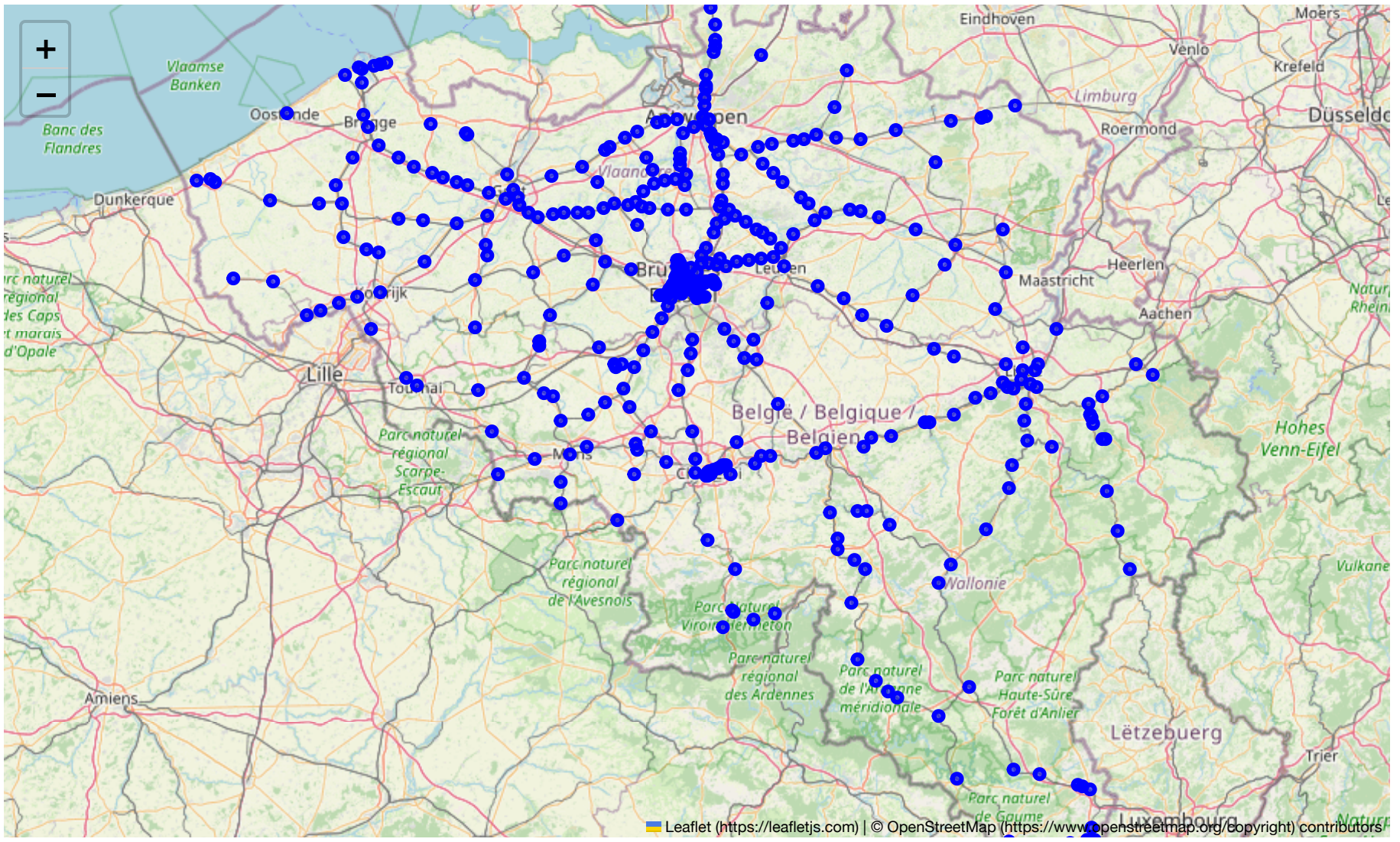
Let us check these locations quickly

In [18]:
```
import folium

# Create a map centered around the average coordinates of Belgium
belgium_center = [50.5039, 4.4699]  # Roughly the center of Belgium
map_belgium = folium.Map(location=belgium_center, zoom_start=8)

# Add the data points to the map
for idx, row in stations.iterrows():
    folium.CircleMarker(
        location=[row['lat'], row['lon']],
        radius=3,
        color='blue',
        fill=True,
        fill_color='blue',
        fill_opacity=0.6
    ).add_to(map_belgium)
map_belgium
```

Out[18]:



Let us store these locations into a new table using postgis

In [25]:
```
with engine.connect() as conn:
    conn.execute("""
        CREATE TABLE IF NOT EXISTS train_stations (
            location GEOMETRY(Point, 4326)
        );
    """)
```

In [27]:
```
# Assuming 'stations' is a DataFrame with train station data including columns 'name', 'lat', 'lon'
# Insert data into the train_stations table
for index, row in stations.iterrows():
    with engine.connect() as conn:
        conn.execute(text("""
            INSERT INTO train_stations (location)
            VALUES (ST_MakePoint(:lon, :lat)::geometry)
        """), {'lat': row['lat'], 'lon': row['lon']})
```

Okey so now we have a table with the train stations

In [28]:
```
query = """
SELECT vd.*
FROM vehicle_data vd
WHERE vd.rs_e_rpm_pc1 = 0 AND vd.rs_e_rpm_pc2 = 0
AND NOT EXISTS (
    SELECT 1
    FROM train_stations ts
    WHERE ST_DWithin(vd.pg_point::geography, ts.location::geography, 3000)
);
"""

# Execute the query and fetch the data into a DataFrame
data_dist_train_station = pd.read_sql_query(query, engine)
```

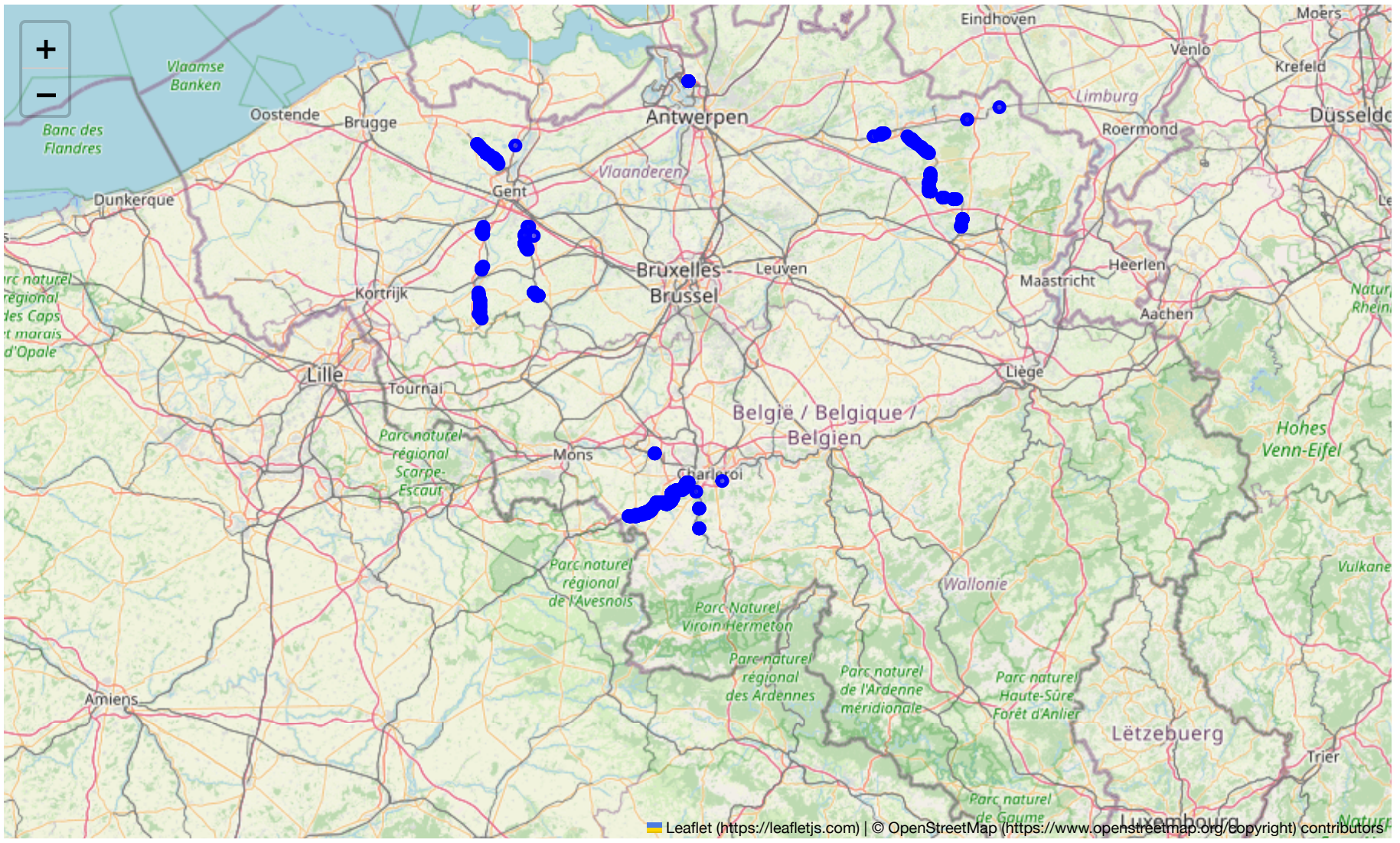
In [40]:
```
print(len(data_dist_train_station))
data_dist_train_station['anomaly_id'] = 'train_stopped'
data_dist_train_station.to_csv('r12.csv')
```

```
668
```

In [30]:
```
# Create a map centered around the average coordinates of Belgium
belgium_center = [50.5039, 4.4699]  # Roughly the center of Belgium
map_belgium = folium.Map(location=belgium_center, zoom_start=8)

# Add the data points to the map
for idx, row in data_dist_train_station.iterrows():
    folium.CircleMarker(
        location=[row['lat'], row['lon']],
        radius=3,
        color='blue',
        fill=True,
        fill_color='blue',
        fill_opacity=0.6
    ).add_to(map_belgium)
map_belgium
```

Out[30]:



So only 668 points. This could be interesting. We analyse each for certain vehicle id's. We are looking for data points together which might show a train that is malfunctioning, and not just slowing down.

In [31]:
```
# Display the unique 'mapped_veh_id's
unique_mapped_veh_ids = data_dist_train_station['mapped_veh_id'].unique()
print(unique_mapped_veh_ids)
```

```
[170 192 160 185 151 140 169 155 196 153 173 128 110 142 147 134 164 130
 122 121 133 191 171 158 183 156 178 188 126 159 136 165 172 131 186]
```

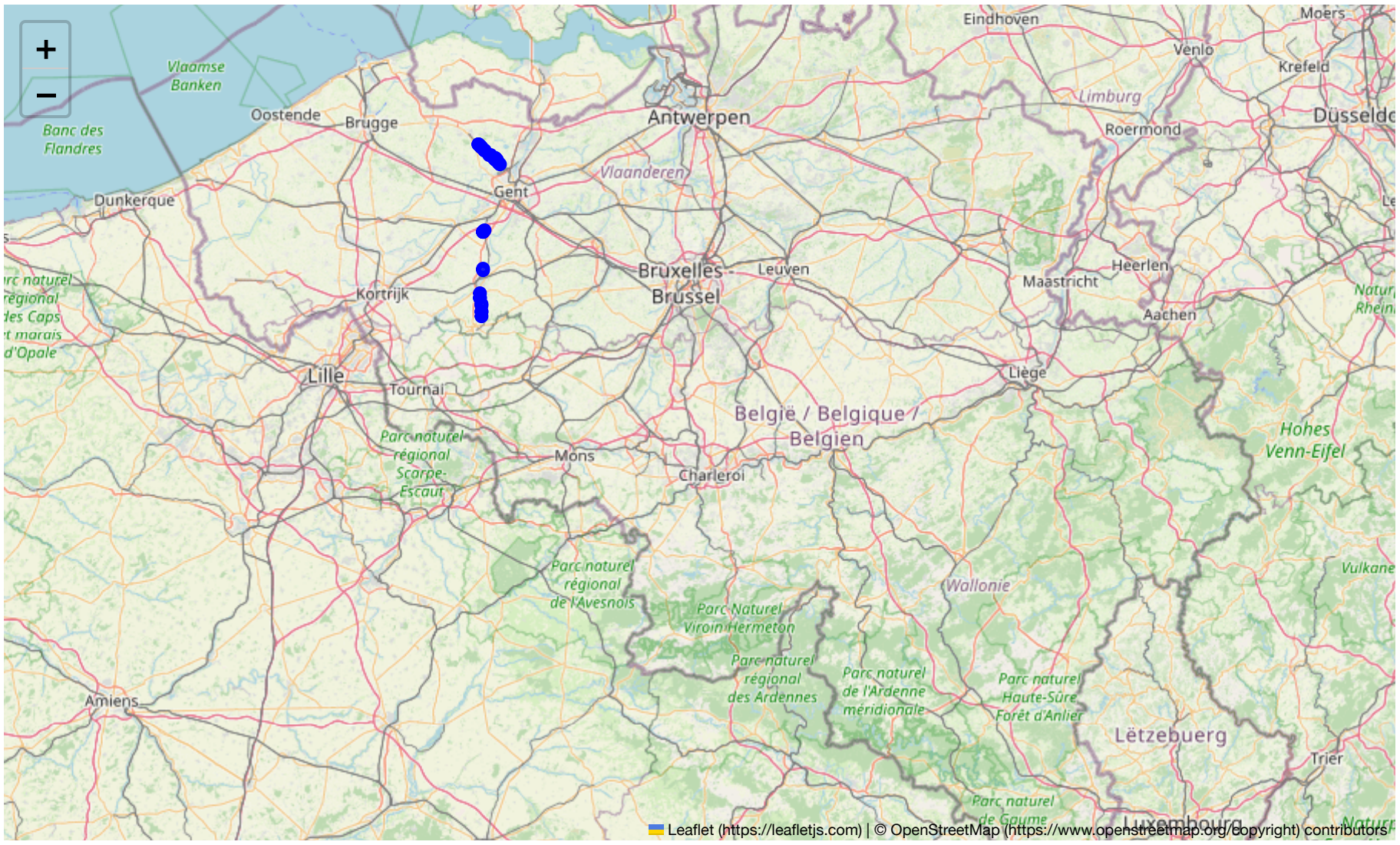
In [37]:
```
specific_veh_data = data_dist_train_station[data_dist_train_station['mapped_veh_id'] == 160]
print(len(specific_veh_data))

belgium_center = [50.5039, 4.4699]  # Roughly the center of Belgium
map_belgium = folium.Map(location=belgium_center, zoom_start=8)

# Add the data points to the map
for idx, row in specific_veh_data.iterrows():
    folium.CircleMarker(
        location=[row['lat'], row['lon']],
        radius=3,
        color='blue',
        fill=True,
        fill_color='blue',
        fill_opacity=0.6
    ).add_to(map_belgium)
map_belgium
```

34

Out[37]:


Eindhoven
Moers
Venlo
Krefeld
Limburg
Düsseldo
Roermond
Vlaamse Banken
Oostende
Brugge
Antwerpen
Banc des Flandres
Vlaanderen
Gent
Dunkerque
Bruxelles - Brussel
Leuven
Heerlen
Maastricht
Aachen
Kortrijk
Lille
Tournai
Liège
België / Belgique / Belgien
Hohes Venn-Eifel
Parc naturel régional Scarpe-Escaut
Mons
Charleroi
Parc naturel régional de l'Avesnois
Wallonie
Parc Naturel Viroin-Hermeton
Parc naturel régional des Ardennes
Parc naturel de l'Ardenne méridionale
Parc naturel Haute-Sûre Forêt d'Anlier
Amiens
Lëtzebuerg
Trier
Parc naturel de Gaume
Luxembourg
Leaflet (https://leafletjs.com) | © OpenStreetMap (https://www.openstreetmap.org/copyright) contributors