



Demand-Responsive  
Transport to ensure  
accessibility, availability  
and reliability of rural  
public transport

# MAPS WITH VISUALISATION OF COMPILED SERVICE AVAILABILITY OFFERED BY ALL PUBLIC AUTHORITIES

**Output I3.1**  
**Aalborg University**  
**Department of Computer Science**

**Authors**  
**Kristian Torp**  
**Magnus N. Hansen**

**31/12/2021**  
Project nr. #R101  
(Interreg Baltic Sea Region)



# Table of Content

	1
<b>1 OVERVIEW</b>	<b>3</b>
<b>2 ROUTING</b>	<b>4</b>
<b>3 ISOCHRONE</b>	<b>5</b>
<b>4 COST MATRIX</b>	<b>6</b>
<b>5 TRAVELING SALESPERSON</b>	<b>7</b>
<b>6 RESTFUL API APPROACH</b>	<b>8</b>
6.1 PARAMETERS	8
6.2 RESTFUL API ENDPOINTS	8
6.3 ISOCHRONE USING A RESTFUL API ENDPOINT	9
6.4 COST-MATRIX USING A RESTFUL API ENDPOINT	10
6.5 TRAVELING SALESPERSON USING A RESTFUL API ENDPOINT	11
<b>7 CONCLUSION</b>	<b>12</b>
<b>8 REFERENCES</b>	<b>13</b>

# 1 Overview

This report describes the various digital maps and visualizations that are possible with the geodata-based data warehouses and RESTful APIs built-in Activity A2.1 of the RESPONSE project [1]. The visualizations are connected to the two KPIs travel time and fuel consumption that are the focus in Activity A2.4 [2]. The maps and queries presented in this report are all available from the RESPONSE website at <https://mapapi.cs.aau.dk>. This website covers fully the three countries Denmark, Faroe Islands, and Sweden.

Two main approaches to creating digital maps are provided. In the *web browser approach*, geodata is displayed directly on maps in the browser. No installation of software is required and the data can be used as screenshots or the data can be downloaded. In the *RESTful API approach*, the geodata can be downloaded in the widely used GeoJSON format and displayed in GIS software. With the latter approach, the geodata made available via the RESPONSE project can easily be integrated with other geodata sources in transportation organizations. Note that all functionality provided by the web browser approach is also available via the RESTful API approach.

For all relevant RESTful API endpoints, it is possible the different models for travel time estimation proposed in Appendix A of Output 2.1 of the RESPONSE project [1]. Understanding these models requires a certain level of technical knowledge. To avoid confusion from non-technical users Model C [1] is always the default model as it generally is the most accurate.

For privacy reasons only the RESTful endpoints that rely on the prediction of travel times are available online for all. The other RESTful endpoints require that a user is registered and logged in.

## 2 Routing

The routing option using the web approach is shown in Figure 1 for a trip in the area of Leksand, Sweden. Please note that public addresses are used in all examples, e.g., banks and parks.

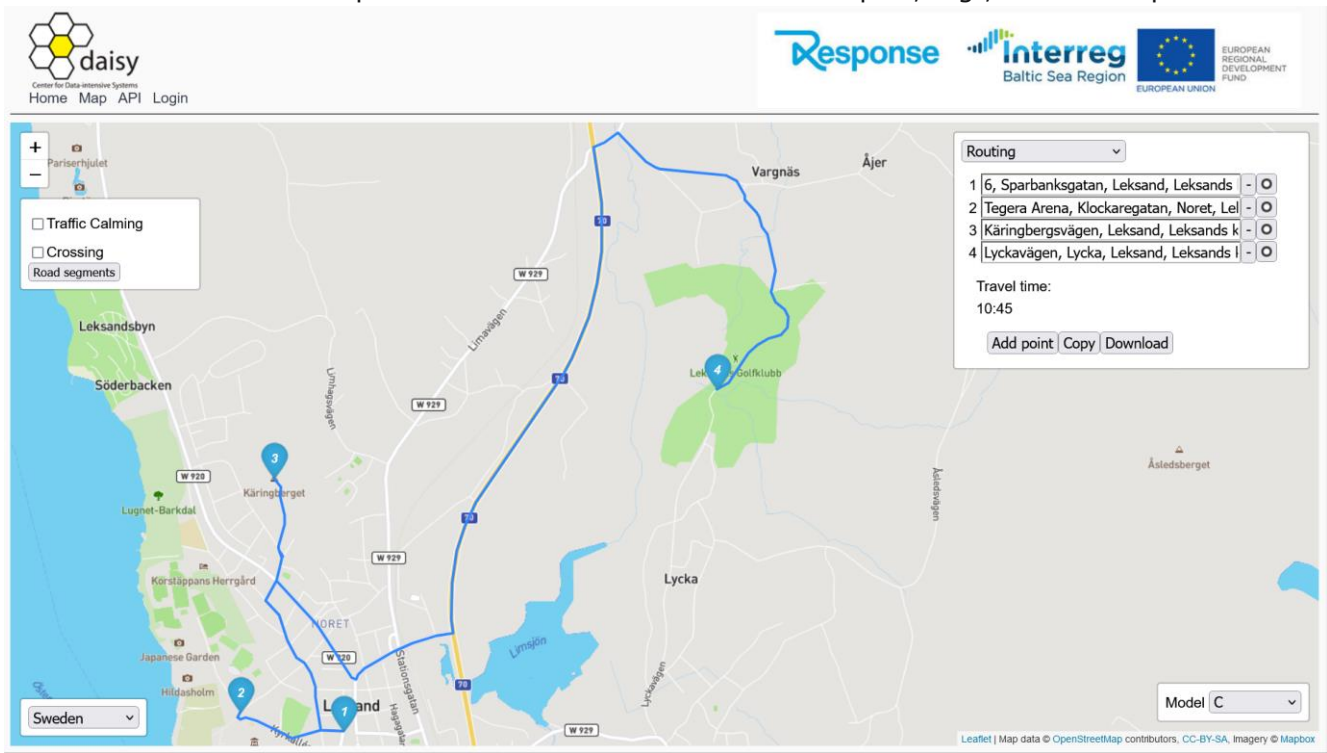


Figure 1 Routing Map for Leksand, Sweden

The functionality is the following.

- The route proposed will start at the needle label 1 and then pass through each of the needles in increasing order, i.e., 1, 2, 3, and for in the example in Figure 1.
- The routing is supported in Denmark, the Faroe Islands, and Sweden.
- The routing supports multiple via points, in the web-browser approach, it is hard to add more than via points. In the RESTful API approach, any number of via points are supported.
- The route can be downloaded in the GeoJSON format [2] for presentation in a GIS tool.
- The five different models from Appendix A in Output o2.1 of the RESPONSE project can be used as the foundation for routing. The default is Model C which generally is the most accurate [1].
- The points used for the route can be easily copied and the route tested in other online tools such as Bing Maps or Google Maps.

Note the following.

- The speed of the routing algorithm depends on the number of via points but is in general fast enough to be used interactively.
- The background map is based on OpenStreetMap (OSM) [3] that covers all of the EU.
- The GeoJSON output is rendered using the open-source Leaflet JavaScript library [4]-

### 3 Isochrone

The isochrones option using the web approach is shown in Figure 2 with a center at the bridge between the Swedish mainland and the island of Åland.

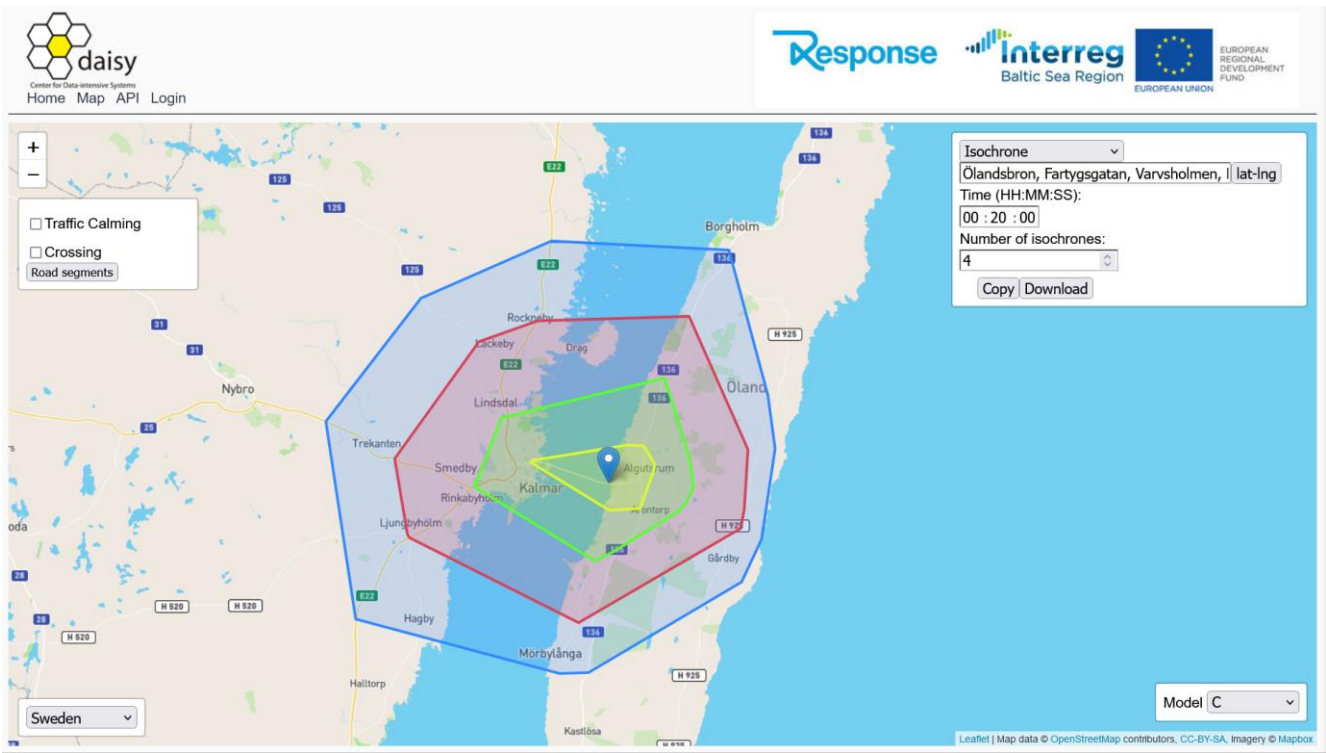


Figure 2 20 Minutes Isochrone for Åland, Sweden

The functionality is the following.

- The user picks the isochrone center by placing the needle on the map.
- The user specifies the maximum travel time in an hour:minute:second format.
- The user can specify between 1 and 4 isochrones to be computed. In the example in Figure 2, 20 minutes and four isochrones. Each isochrone will then cover 5 additional minutes, i.e., 5, 10, 15, and 20 minutes of travel time.
- The isochrone can be downloaded in the GeoJSON format [2] for presentation in a GIS tool.

Note the following.

- The isochrones functionality builds on top of the pgRouting [5] PostgreSQL extension.
- An isochrone query visits nearly all road segments within a region and is therefore inherently very computationally intensive. A three-hour, four-level isochrones takes 20-30 seconds to compute.

## 4 Cost Matrix

The cost matrix using the island of Lolland, Denmark for five points is shown in Figure 3. The cost matrix computes the travel time between all the points specified (except the point itself).

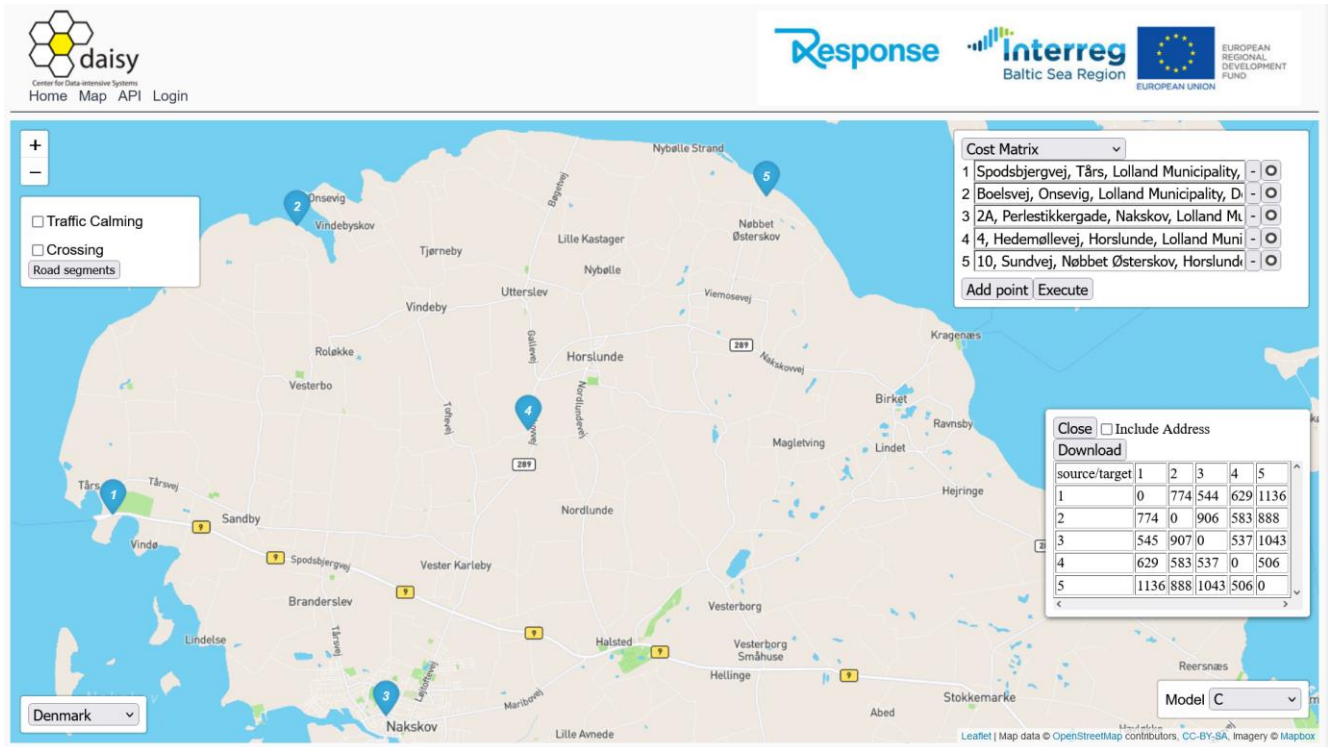


Figure 3 Cost Matrix for the Island of Lolland, Denmark

The functionality is the following.

- The user picks places needed for addresses to include in the cost matrix by using the “Add point” button.
- The user any number of points can be specified using the RESTful API. In the web interface, having more than 10-12 points is impractical.
- The cost matrix is recalculated each time the “Execute” button is pressed. The user has to explicitly press the button because a varying number of points can be used.
- The user can download the cost matrix in Excel spreadsheet format. The addresses and locations (latitude, longitude) can be included in the download to be able to repeat the query at a later time or to use a different service.

Note the following.

- The cost matrix functionality builds on top of the pgRouting [5] PostgreSQL extension.

## 5 Traveling Salesperson

The travel salesperson functionality is shown in Figure 4 for areas near the city of Faaborg, Denmark. The traveling salesperson finds an efficient way to visit all addresses specified by the user starting from the address specified by the needle label 1 and returning to this address again.

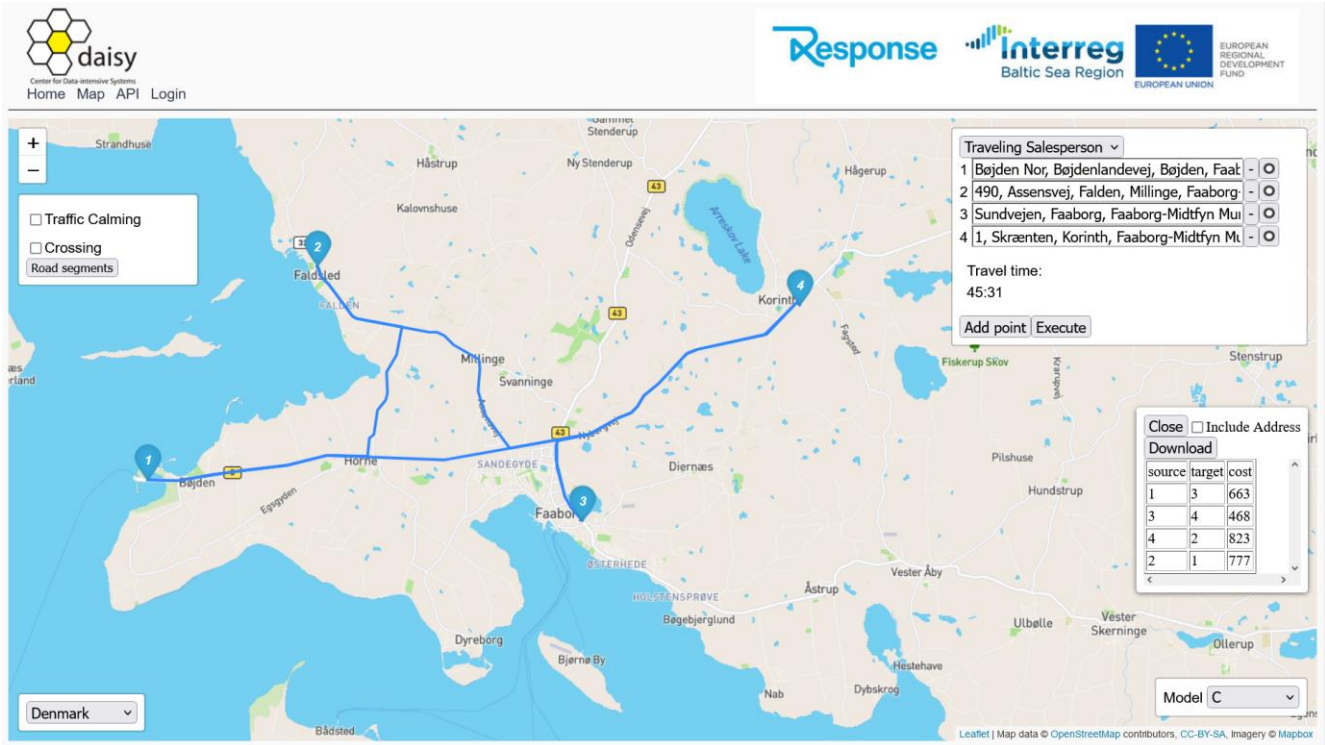


Figure 4 The Traveling Salesperson for Faaborg, Denmark

The functionality is the following.

- The user picks places needles for addresses to include in the query by using the "Add point" button.
- The user any number of points can be specified using the RESTful API. In the web interface, having more than 10-12 points is impractical.
- The traveling salesperson query is recalculated each time the "Execute" button is pressed. The user has to explicitly press the button because a varying number of points can be used.
- The user can download the result in Excel spreadsheet format. The addresses and locations (latitude, longitude) can be included in the download to be able to repeat the query at a later time or to use a different service.

Note the following.

- The traveling salesperson functionality builds on top of the pgRouting [5] PostgreSQL extension.
- The traveling salesperson problem is computational very heavy to compute.
- All known algorithms for solving the traveling salesperson problem approximates the result by using heuristics. The optimal result can generally not be found efficiently.

## 6 RESTful API Approach

In this section, we provide an overview of the maps and other results that can be produced by the RESTful API approach. The API approach provides a programmatic way to access the data stored in the data warehouse and is very widely used. Compared to the web browser approach, the API approach requires more technical knowledge. This has been lowered as much as possible by being able to specify the parameters to all RESTful API endpoints using the Swagger UI [6]. The Swagger UI is widely used and makes it possible to query the RESTful API from a web browser. The result can then be downloaded, e.g., Excel or CSV format. For a tutorial on using the RESTful API and detail on the output format please see [7].

### 6.1 Parameters

The RESTful endpoints use several parameters. In this section, we describe these parameters in detail. The good news is that there is a very high degree of reuse of parameters between the endpoints. Meaning that when a user has first learned to use one endpoint the others follow a very similar pattern. Further default values are provided in the Swagger UI for many parameters such that a user has to specify a minimal set of required parameters to get started using the RESTful API endpoints.

The list of parameters is presented in the order they are typically used. The list of the most used parameters is the following.

- *countryCode* A country code, i.e., DK, FO, or S
- *fromPoint* A point, i.e., longitude, latitude pair
- *toPoint* A point, i.e., longitude, latitude pair
- *fromDate* A date smartkey, e.g., 20140101, inclusive, has a default
- *toDate* A date smartkey, e.g., 20140131, inclusive, has a default
- *days* A day-of-week bit pattern, e.g., 1111100 for Monday to Friday, has a default
- *fromTime* A time smartkey, e.g., 601, inclusive, has a default
- *toTime* A time smartkey, e.g., 845, inclusive, has a default
- *costModel* A cost model baseline or A to D see Appendix A in [1]
- *format* An output format, e.g., CSV or excel, see [7]
- *geometry* A geometry output format, i.e., GeoJSON or WKT [8]

### 6.2 RESTful API Endpoints

The usage of the RESTful API is described in detail in O2.4 [7] of the RESPONSE project. This report also provided a complete tutorial for the more advanced users that want to access the endpoints using a programming language such as Python. The report O2.4 also contains many visualizations where geodata from the RESTful API are presented on a digital map. In the report, we extend this tutorial with additional examples and keep the overlap with O2.4 as small as possible.



### 6.3 Isochrone Using a RESTful API Endpoint

As an example of using the Isochrone RESTful API endpoint, the Swagger UI is shown in Figure 5. In the Swagger UI, there are default values for all parameters and a user can start experimenting right away with the API.

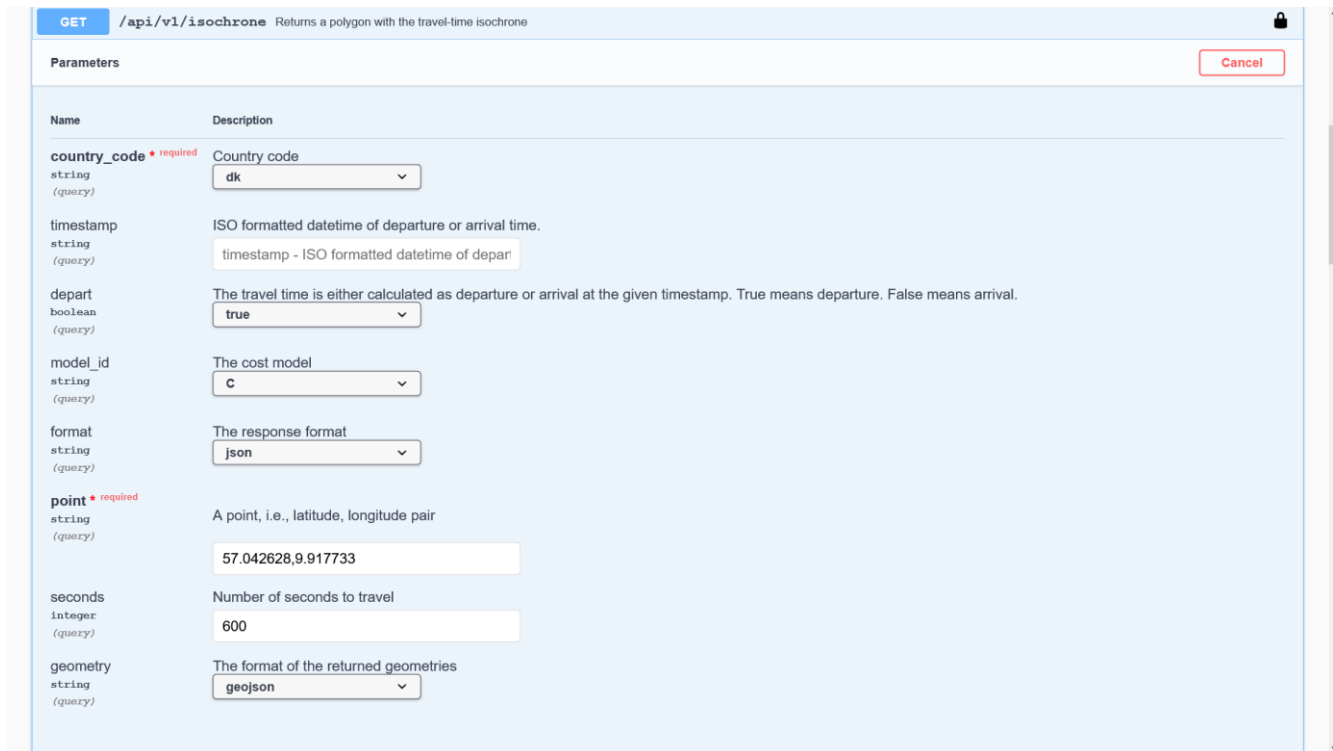


Figure 5 Specifying Parameters for the Isochrone Endpoint Using Swagger UI

The same query as shown in Figure 5 can be executed over the internet using the HTTPS protocol. This is shown in the text box below. This text box is an example of using a programmatic way to access the data provided by the RESPONSE project. Please note that the HTTPS example below is easy to create and use in most programming languages such as Python, for detail see RESPONSE report O2.4 [7].

```
https://mapapi.cs.aau.dk/api/v1/isochrone?country_code=dk&depart=true&model_id=C&format=json&point=57.042628%2C9.917733&seconds=600&geometry=geojson&key=WyIyIiwiJDUkcm91bmRzPTUzNTAwMCRtVEhBVzMzeTlWbURlZ3JKJEw3b0QvRnVTLlpPanFjMWtFUEgySTF1TzRYTTdoSDNiaU1wYj1CNzVQUjkiLCJmMTQ5YzFmZTMxMdc0ZTJlOGQ2ODcwZTgXNWwNWE5YSJd.YcL5bA.m4HtnGrd7sL2AXPR0UEVT4ipQnA
```

The HTTPS protocol-based call of the RESTful API is provided directly using the Swagger UI. A user can copy and paste the first version or a call directly to their programming code. This approach is the same for all endpoints and we therefore only show this single example.

## 6.4 Cost-Matrix Using a RESTful API Endpoint

An example of using the cost-matrix RESTful API endpoint, from the Swagger UI is shown in Figure 6. Again the Swagger UI provides default values for most parameters. Please note that the points (needles shown in Figure 3) are provided using a list of points in the parameter named *points*. This approach is also used in many commercial RESTful API providers and allows the cost matrix to support any number of points. To use the cost matrix a user only has to provide a list of points to the *points* parameter and then call the RESTful endpoint from the Swagger UI.

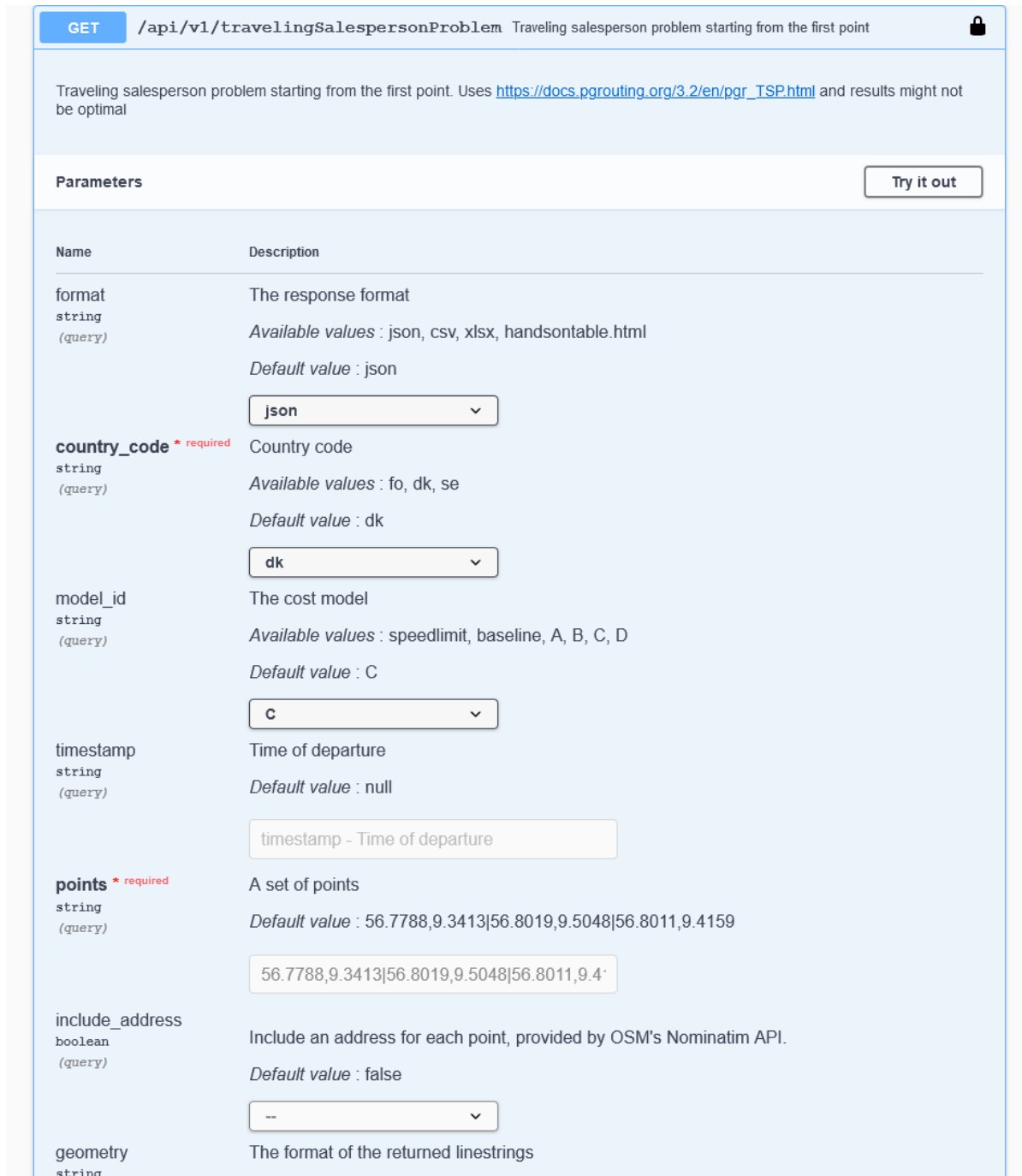
Name	Description
<b>format</b> string (query)	The response format <i>Available values</i> : json, csv, xlsx, handsontable.html <i>Default value</i> : json <input type="text" value="json"/>
<b>country_code</b> * required string (query)	Country code <i>Available values</i> : fo, dk, se <i>Default value</i> : dk <input type="text" value="dk"/>
<b>model_id</b> string (query)	The cost model <i>Available values</i> : speedlimit, baseline, A, B, C, D <i>Default value</i> : C <input type="text" value="C"/>
<b>timestamp</b> string (query)	Time of departure <i>Default value</i> : null <input type="text" value="timestamp - Time of departure"/>
<b>points</b> * required string (query)	A set of points <i>Default value</i> : 56.7788,9.3413 56.8019,9.5048 56.8011,9.4159 <input type="text" value="56.7788,9.3413 56.8019,9.5048 56.8011,9.4159"/>
<b>include_address</b> boolean (query)	Include an address for each point, provided by OSM's Nominatim API. <i>Default value</i> : false <input type="text" value="--"/>

**Responses** Response content type:

Figure 6 Using the Cost-Matrix RESTful API Endpoint

## 6.5 Traveling Salesperson using a RESTful API Endpoint

An example of using the traveling salesperson RESTful API endpoint, from the Swagger UI is shown in Figure 7. Please note that the set of parameters in Figure 7 is the same as the set of parameters shown for the cost matrix RESTful endpoint shown in Figure 6. This makes the RESTful API easier to learn for users. The idea is that if a user understands one endpoint it is easy to learn an additional endpoint.



GET /api/v1/travelingSalespersonProblem Traveling salesperson problem starting from the first point

Traveling salesperson problem starting from the first point. Uses [https://docs.pgRouting.org/3.2/en/pgRouting\\_TSP.html](https://docs.pgRouting.org/3.2/en/pgRouting_TSP.html) and results might not be optimal

Parameters Try it out

Name	Description
format string (query)	The response format Available values : json, csv, xlsx, handsontable.html Default value : json <input type="text" value="json"/>
country_code * required string (query)	Country code Available values : fo, dk, se Default value : dk <input type="text" value="dk"/>
model_id string (query)	The cost model Available values : speedlimit, baseline, A, B, C, D Default value : C <input type="text" value="C"/>
timestamp string (query)	Time of departure Default value : null <input type="text" value="timestamp - Time of departure"/>
points * required string (query)	A set of points Default value : 56.7788,9.3413 56.8019,9.5048 56.8011,9.4159 <input type="text" value="56.7788,9.3413 56.8019,9.5048 56.8011,9.4159"/>
include_address boolean (query)	Include an address for each point, provided by OSM's Nominatim API. Default value : false <input type="text" value="--"/>
geometry string	The format of the returned linestrings

Figure 7 Using the Traveling Salesperson RESTful API Endpoint

## 7 Conclusion

This report describes O3.1 that is a set of map layers. The layers produce geodata output in the widely used formats GeoJSON and WKT. Two approaches to display geodata are presented. The first approach is called the *web browser approach* and allows a user to render a digital map directly in a web browser. This approach uses the open technologies OpenStreetMap as the map foundation and the Leaflet open-source JavaScript library for visualizing the geodata. The second approach is called the *RESTful API approach* that allows more advanced users to query the geodata using the Swagger IO or call the RESTful API directly from a programming language. RESTful API calls are supported in most programming languages. The output of the RESTful API approach can very easily be integrated with other geodata in both open-source and commercial GIS systems.

The web browser approach covers four very typical queries in transport organizations, i.e., routing, isochrones, cost matrix, and the traveling salesperson problem. The RESTful API consists of 24 endpoints. All endpoints use geodata as input or produce geodata output.

Both the web browser and RESTful API approaches cover the counties Denmark, the Faroe Islands, and Sweden. The web browser approach is freely available on the internet from the address <https://mapapi.cs.aau.dk>. For part of the RESTful API approach, a user must be registered and logged in to use the API.

The focus in both approaches is on the two important KPIs travel time and fuel consumption in transport organizations. All data can be downloaded and used in existing analysis platforms. This makes it possible to use the RESPONSE project data, e.g., to compare cost-effectiveness, level of environmental sustainability, and service quality of transport in both private and public organizations.

## 8 References

- [1] K. Torp and M. N. Hansen, "Open-Source Based Data Warehouse, Report O2.1," The RESPONSE Project, 2021.
- [2] Internet Engineering Task Force (IETF), "The GeoJSON Specification (RFC 7946)," [Online]. Available: <https://geojson.org/>.
- [3] OpenStreetMap, "OpenStreetMap," [Online]. Available: <https://www.openstreetmap.org/>.
- [4] V. Agafonkin, "Leaflet," [Online]. Available: <https://leafletjs.com/>.
- [5] pgRouting, "pgRouting," [Online]. Available: <https://pgrouting.org/>.
- [6] Smartbear, "Swagger IO," [Online]. Available: <https://swagger.io>.
- [7] K. Torp and M. N. Hansen, "Regional Workshop on Open Data Development for Public Authorities, Report O2.4," The RESPONSE Project, 2021.
- [8] OGC Standard, "Well-known text representation of coordinate reference systems," [Online]. Available: <https://www.ogc.org/standards/wkt-crs>.