D3.3 Recommended Routing Corridor Service

Project Number	
Classification	Public
Deliverable No.	D3.3
Work Package(s)	WP3
Document Version	1.0
Issue Date	December 31, 2019
Document Timescale	Project Start: June/July 1st (Poland), 2018
Final version due	2019-12-31
Compiled by	PUEB
Authors	Dominik Filipiak, Krzysztof Węcel, Milena Stróżyna, Błażej Lisiecki, Witold Abramowicz
Issue Authorisation	Sebastian Feuerstack, OFF

HANSA is funded by the MarTERA partners German Federal Ministry of Economic Affairs and Energy (BMWi), Polish National Centre for Research and Development (NCBR) and Research Council of Norway (RCN) and co-funded by the European Union.

DISTRIBUTION LIST

Copy type[1]	Company and Location	Recipient
T	HANSA Consortium	all HANSA Partners

RECORD OF REVISION

Date	Status Description	Author
2019-12-31	First draft	PUEB

Table of Contents

1 Introduction

This document presents a web service that was developed in the HANSA project in order to make both the mesh and recommended corridors available to external components (e.g. systems and applications used by the project's partners). A recommended corridor is provided by the service based on a query from external components, specifying e.g. a ship and a voyage for which the route should be planned. The service uses the mesh(es), previously generated by the HANSA system (a conceptual representation of all possible routes at sea, see Deliverable D3.1 and D3.2 for more details) to determine the recommended corridor. There are two types of requests available in the service: 1) provides the mesh for a specified maritime area; 2) provides the recommended corridor as the fastest or the shortest route, based on the data provided in a query.

The developed web service is compatible with the REST style. It was written in `python` using the `Flask` framework and the `Flask-RESTPlus` extension which adds support for conveniently building REST APIs. The Neo4j graph database was used as storage for meshes (waypoints and their connecting edges), thanks to which the fastest or shortest route between given waypoints can be quickly determined.
The source code for the web service is located in Deliverable D2.2.
Source code of the module used by the web service in determining mesh and recommended corridor is available in Deliverable D2.2.

In the following sections of the report we describe how to deploy of run own instance of the service (Section 2, then the service for providing the mesh (Section 3), and the service for determining the recommended corridor (Section 4).

2 Deployment of the service

2.1 Deployment to HANSA server

The service has been deployed to the HANSA server: `hansa.offis.de` and is exposed on port 54321. The access to the service is possible from localhost or by port tunnelling when logged in via ssh. The following command can be helpful to establish a connection with a port open on own machine:

```
$ ssh -N -f -p 33222 -L 54321:localhost:54321 {{username}}@hansa.offis.de
```

2.2 Running own instance

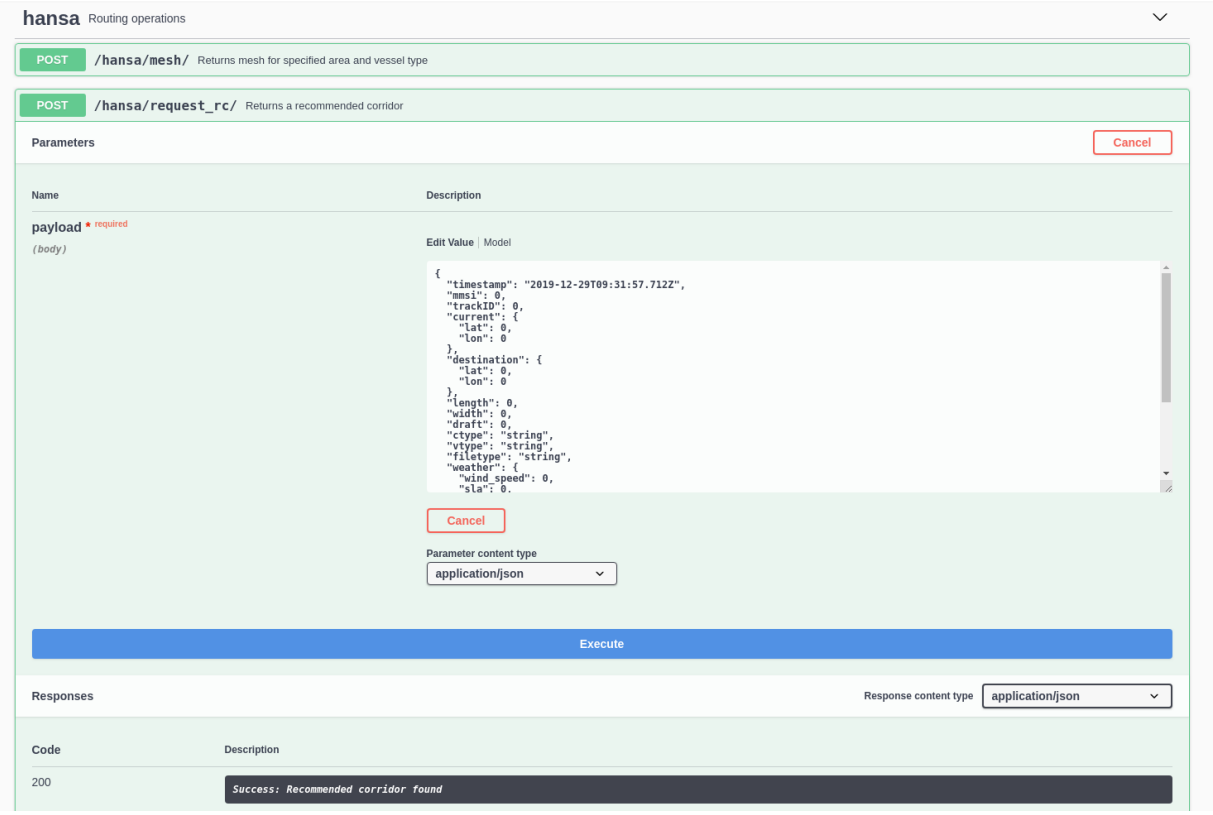
The prerequisite for running the service is installation of Docker. After checkout of the code from the repository, the following command should be used to start the service:

```
$ docker-compose up -d
```

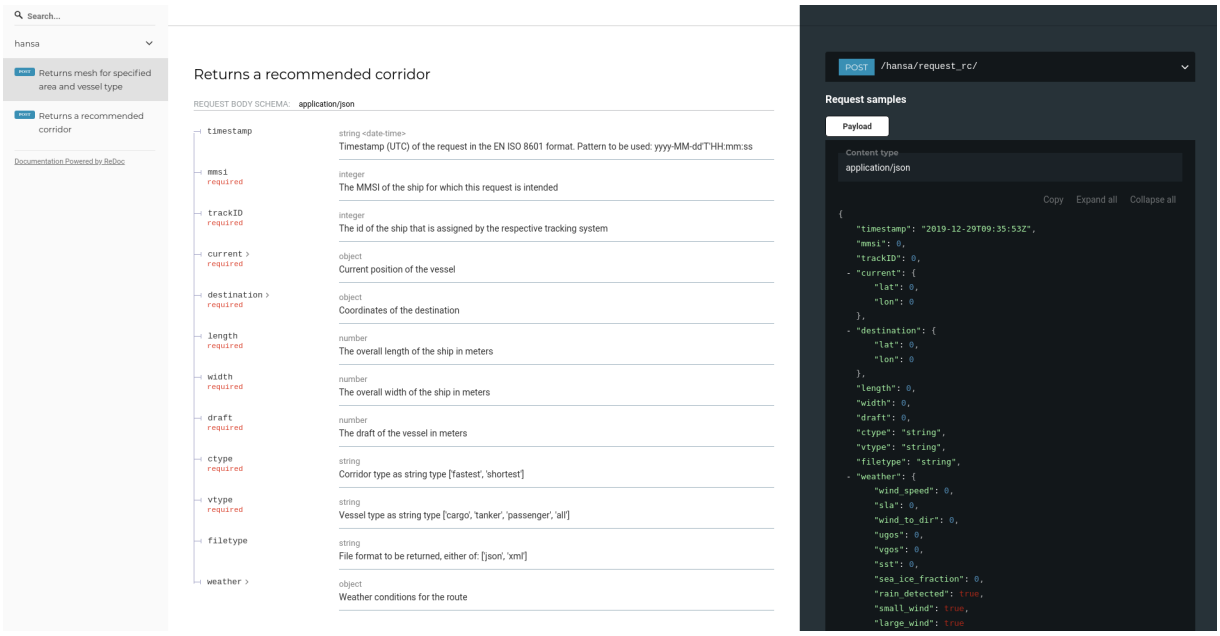
It will start separate docker containers for new instance of Neo4j database and the HANSA web service.

Documentation:

- `http://localhost:54321/` - easily test API from browser



- `http://localhost:54321/docs` - documentation and request samples for exposed services



3 Mesh Service

A web service that returns a waypoints mesh with edges for a given area.

Address:
`[POST] http://localhost:54321/hansa/mesh`

Parameters:

- `ll` - coordinates of lower left corner of desired area (required);
- `ur` - coordinates of upper right corner of desired area (required);
- `vtype` - vessel type ['cargo', 'tanker', 'passenger', 'all'] (required)

Request:

```
$ curl -d @test_request_mesh.json -X POST -H "Content-Type: application/json" http://localhost:54321/hansa/mesh/
```

The content of the file `test_request_mesh.json` is as follows:

```
{
        "ll": {
                "lat": 55,
                "lon": 1
        },
        "ur": {
                "lat": 60,
                "lon": 10
        },
        "vtype": "cargo"
}
```

Response:

```
{
  "edges": [
    {
      "start_id": 5047,
      "dist": 29709.9247865797,
      "count": 1,
```

```
      "end_id": 5,
      "cnt": 1,
      "max_time": 3,
      "min_time": 3,
      "avg_time": 3,
      "stddev_time": null,
      "graph": "cargo"
    },
    {
      "start_id": 1226,
      "cnt": 3,
      "end_id": 5,
      "min_time": 1,
      "max_time": 2,
      "count": 1,
      "dist": 23463.816595246,
      "stddev_time": 0.577350269189626,
      "graph": "cargo",
      "avg_time": 1.33333333333333
    },
    {
      "stddev_time": null,
      "graph": "cargo",
      "avg_time": 5,
      "start_id": 1330,
      "cnt": 1,
      "end_id": 5,
      "min_time": 5,
      "max_time": 5,
      "dist": 129152.926681784,
      "count": 1
    },

    {...}

  ],

  "waypoints": [
    {
      "lon": 5.9202,
```

```
      "graph": "cargo",
      "lat": 58.46583,
      "id": 5
    },
    {
      "graph": "cargo",
      "lat": 59.28458,
      "id": 8,
      "lon": 5.314402
    },
    {
      "lon": 9.427547,
      "lat": 55.043575,
      "graph": "cargo",
      "id": 32
    },

    {...}

  ]
}
```

4 Recommended Corridor Service

A web service that returns recommended corridor for given set of parameters. In response, RTZ is returned in either JSON or XML format. Dijkstra's algorithm is used to determine the shortest path. Depending on the `ctype` parameter, the service returns either the shortest route or the fastest.

Address:
`[POST] http://localhost:54321/hansa/request_rc`

Parameters:

- `timestamp` - Timestamp (UTC) of the request in the EN ISO 8601 format. Pattern to be used: yyyy-MM-dd'T'HH:mm:ss;
- `mmsi` - MMSI of the ship for which this request is intended (required);
- `trackID` - ID of the ship that is assigned by the respective tracking system (required);
- `current` - Current position of the vessel (lat, lon) (required);
- `destination` - Coordinates of the destination (lat, lon) (required);
- `length` - Overall length of the ship in meters (required);
- `width` - Overall width of the ship in meters (required);
- `draft` - Draft of the vessel in meters (required);
- `ctype` - Corridor type as string type ['fastest', 'shortest'] (required);
- `vtype` - Vessel type as string type ['cargo', 'tanker', 'passenger', 'all'] (required);
- `filetype` - File format to be returned, either of: ['json', 'xml'] (required);
- `weather` - Weather conditions for the route (optional);

4.1 Response in JSON

Request:

```
$ curl -d @test_request_corridor_json.json -X POST -H "Content-Type: application/json" http://localhost:54321/hansa/request_rc/
```

The content of the file `@test_request_corridor_json.json` is as follows:

```
{
    "mmsi": 91284,
```

```
    "trackID": 838138,
    "current": {
        "lat": 12.123,
        "lon": 21.133
    },
    "destination": {
        "lat": 39.212,
        "lon": 50.212
    },
    "length": 48.32,
    "width": 24.213,
    "draft": 12.0,
    "ctype": "shortest",
    "vtype": "cargo",
    "filetype": "json"
}
```


Response:


```
{
  "rcAsRTZ": {
    "route": {
      "routeInfo": {
        "@routeName": "Shortest route"
      },
      "waypoints": {
        "waypoint": [
          {
            "@radius": 1,
            "position": {
              "@lat": 53.121895,
              "@lon": 14.381755
            },
            "leg": {
              "@portsideXTD": 0.01,
              "@geometryType": "Orthodrome",
              "@starboardXTD": 0.01
            },
            "@name": "WP4230"
          },
          {
```

```
    "@name": "WP86",
    "leg": {
      "@portsideXTD": 0.01,
      "@geometryType": "Orthodrome",
      "@starboardXTD": 0.01
    },
    "position": {
      "@lon": 14.577963,
      "@lat": 53.413773
    },
    "@radius": 1
  },
  {
    "position": {
      "@lon": 14.5786,
      "@lat": 53.428524
    },
    "@radius": 1,
    "@name": "WP1884",
    "leg": {
      "@geometryType": "Orthodrome",
      "@starboardXTD": 0.01,
      "@portsideXTD": 0.01
    }
  },
  {
    "@name": "WP2651",
    "leg": {
      "@portsideXTD": 0.01,
      "@starboardXTD": 0.01,
      "@geometryType": "Orthodrome"
    },
    "@radius": 1,
    "position": {
      "@lat": 53.43444,
      "@lon": 14.580749
    }
  },
  {
    "leg": {
```

```
      "@portsideXTD": 0.01,
      "@starboardXTD": 0.01,
      "@geometryType": "Orthodrome"
    },
    "@name": "WP5275",
    "@radius": 1,
    "position": {
      "@lon": 14.584461,
      "@lat": 53.447414
    }
  },
  {
    "@name": "WP1623",
    "leg": {
      "@portsideXTD": 0.01,
      "@starboardXTD": 0.01,
      "@geometryType": "Orthodrome"
    },
    "position": {
      "@lat": 54.390064,
      "@lon": 18.668442
    },
    "@radius": 1
  },
  {
    "@radius": 1,
    "position": {
      "@lat": 54.39,
      "@lon": 18.6685
    },
    "leg": {
      "@portsideXTD": 0.01,
      "@geometryType": "Orthodrome",
      "@starboardXTD": 0.01
    },
    "@name": "WP2933"
  },
  {
    "position": {
      "@lat": 54.40318,
```

```
      "@lon": 18.675606
    },
    "@radius": 1,
    "@name": "WP2021",
    "leg": {
      "@geometryType": "Orthodrome",
      "@starboardXTD": 0.01,
      "@portsideXTD": 0.01
    }
  },
  {
    "@name": "WP3058",
    "leg": {
      "@portsideXTD": 0.01,
      "@geometryType": "Orthodrome",
      "@starboardXTD": 0.01
    },
    "@radius": 1,
    "position": {
      "@lon": 19.436436,
      "@lat": 54.78463
    }
  },
  {
    "position": {
      "@lon": 20.526917,
      "@lat": 55.331482
    },
    "@radius": 1,
    "leg": {
      "@starboardXTD": 0.01,
      "@geometryType": "Orthodrome",
      "@portsideXTD": 0.01
    },
    "@name": "WP4787"
  },
  {
    "position": {
      "@lon": 30.628035,
      "@lat": 59.795376
```

```
            },
            "@radius": 1,
            "@name": "WP4377",
            "leg": {
              "@portsideXTD": 0.01,
              "@geometryType": "Orthodrome",
              "@starboardXTD": 0.01
            }
          }
        ]
      }
    }
  },
  "timestamp": "2019-12-29'T'10-00-33",
  "trackID": 838138,
  "mmsi": 91284
}
```


4.2 Response in XML

Request:

```
$ curl -d @test_request_corridor_xml.json -X POST -H "Content-Type: application/json"
http://localhost:54321/hansa/request_rc/
```

The content of the file `@test_request_corridor_xml.json` is as follows:


```
{
    "mmsi": 91284,
    "trackID": 838138,
    "current": {
        "lat": 12.123,
        "lon": 21.133
    },
    "destination": {
        "lat": 39.212,
        "lon": 50.212
    },
    "length": 48.32,
    "width": 24.213,
    "draft": 12.0,
```

```
    "ctype": "fastest",
    "vtype": "cargo",
    "filetype": "xml"
}
```

Response:

```
<?xml version='1.0' encoding='utf8'?>
<route>
  <routeInfo routeName="Fastest route]"/>
  <waypoints>
    <waypoint name="WP4230" radius="1.00000">
      <position lat="53.121895" lon="14.381755"/>
      <leg geometryType="Orthodrome" portsideXTD="0.01000000" starboardXTD="0.01000000"/>
    </waypoint>
    <waypoint name="WP86" radius="1.00000">
      <position lat="53.413773" lon="14.577963"/>
      <leg geometryType="Orthodrome" portsideXTD="0.01000000" starboardXTD="0.01000000"/>
    </waypoint>
    <waypoint name="WP1884" radius="1.00000">
      <position lat="53.428524" lon="14.5786"/>
      <leg geometryType="Orthodrome" portsideXTD="0.01000000" starboardXTD="0.01000000"/>
    </waypoint>
    <waypoint name="WP2651" radius="1.00000">
      <position lat="53.43444" lon="14.580749"/>
      <leg geometryType="Orthodrome" portsideXTD="0.01000000" starboardXTD="0.01000000"/>
    </waypoint>
    <waypoint name="WP2629" radius="1.00000">
      <position lat="53.424484" lon="14.588984"/>
      <leg geometryType="Orthodrome" portsideXTD="0.01000000" starboardXTD="0.01000000"/>
    </waypoint>
    <waypoint name="WP4532" radius="1.00000">
      <position lat="53.895565" lon="14.271922"/>
      <leg geometryType="Orthodrome" portsideXTD="0.01000000" starboardXTD="0.01000000"/>
    </waypoint>
    <waypoint name="WP4054" radius="1.00000">
      <position lat="54.6153" lon="14.210274"/>
      <leg geometryType="Orthodrome" portsideXTD="0.01000000" starboardXTD="0.01000000"/>
    </waypoint>
    <waypoint name="WP4449" radius="1.00000">
      <position lat="54.679813" lon="18.962265"/>
```

```
      <leg geometryType="Orthodrome" portsideXTD="0.01000000" starboardXTD="0.01000000"/>
    </waypoint>
    <waypoint name="WP4787" radius="1.00000">
      <position lat="55.331482" lon="20.526917"/>
      <leg geometryType="Orthodrome" portsideXTD="0.01000000" starboardXTD="0.01000000"/>
    </waypoint>
    <waypoint name="WP4377" radius="1.00000">
      <position lat="59.795376" lon="30.628035"/>
      <leg geometryType="Orthodrome" portsideXTD="0.01000000" starboardXTD="0.01000000"/>
    </waypoint>
  </waypoints>
</route>
```