Routiak - users manual

Richard Búri

May 2024

Contents

1 Installing the web application **2**
- 1.1 Bridge for communication . . . 2
- 1.2 Kontajner webovej aplikácie . . . 2
- 1.3 Web Application Container . . . 2
- 1.4 Valhalla container . . . 3

2 Using the web application **4**
- 2.1 Registration . . . 4
- 2.2 Login . . . 4
- 2.3 Logout . . . 5
- 2.4 Upload routes . . . 5
- 2.5 View Uploaded Files . . . 6
- 2.6 View all routes in uploaded file . . . 6
- 2.7 Displaying individual routes in the uploaded file . . . 8
- 2.8 View error uploaded files . . . 9
- 2.9 Re-run error files . . . 9
- 2.10 Warning about bad route pinning to the road network . . . 10

1 Installing the web application

In this section, we will show you how to install the web application for use.

1.1 Bridge for communication

Two *Docker* containers need to be downloaded for the web application to work properly. These containers will communicate with each other, so it is necessary to create a bridge for communication.

Create

To create a bridge, enter the following command in the terminal:

```
docker network create BRIDGE_NAME --driver bridge
```

The following command creates a bridge named *web_server*:

```
docker network create web_server --driver bridge
```

Check

Information about the bridge can be found using the command

```
docker inspect BRIDGE_NAME
```

Information such as the name of the bridge and the containers that use it will be listed. If the bridge has not been created and does not exist, an error message will be displayed: *Error: No such object: BRIDGE_NAME*

1.2 Kontajner webovej aplikácie

1.3 Web Application Container

This section is dedicated to installing the web application itself, which we will work with later

Creating *Docker image*

To create an image, it is necessary to open a terminal in the folder where the *Dockerfile* file is located. The image is created using the command:

```
docker build -t IMAGE_NAME .
```

If the terminal was not opened in the folder with the file *Dockerfile*, it is possible to write the path to the folder where the file *Dockerfile* is located instead of a dot. The following command creates an image named *dp_webapp*

```
docker build -t dp_webapp .
```

Starting the container

After creating the image, we start the container using the command:

```
docker run -dit --name CONTAINER_NAME --network BRIDGE_NAME -p PORT:PORT
-in PATH:/CONTAINER_PATH IMAGE_NAME
```

To run a container from the image *dp_webapp* to run on port *8090*, connecting to a bridge called *web_server* and linking the folder *C:\Users \richard.buri\Desktop\docker\DP1* to the inner container folder named *DP1*, we run the following command:

```
docker run -dit --name dp_webapp --network web_server -p 8090:8090
-in C:\Users\richard.buri\Desktop\docker\DP1:/DP1 dp_webapp
```

Container access

The container can be accessed either using *Docker desktop* or using the command:

```
docker container exec -it CONTAINER_NAME /bin/bash
```

Additional commands

For the application to function properly, it is necessary to turn on and set up the database. It is also necessary to start the application. The database can be turned on with the following command:

```
service mysql start
```

The following command sets up the database:

`mysql < db.sql` To start the application, use the following command, where the *PORT* must be the same as when starting the container:

```
PORT = 8090 npm run start
```

1.4 Valhalla container

In this section, we will show how to enable the *Valhalla* container to which we will send requests to attach routes to the road network. The container can be linked to a folder on the device where the maps are stored, or with a one-line command that will download the maps automatically. The container may take several minutes to run for the first time. The container is ready for use after typing *INFO: Found config file. Starting valhalla service!*

Starting with a one-line command

```
docker run -dit --name CONTAINER_NAME --network BRIDGE_NAME
-p PORT:PORT -e tile_urls=MAP_URL ghcr.io/gis-ops/docker-valhalla/valhalla:latest
```

Start with Linked Folder

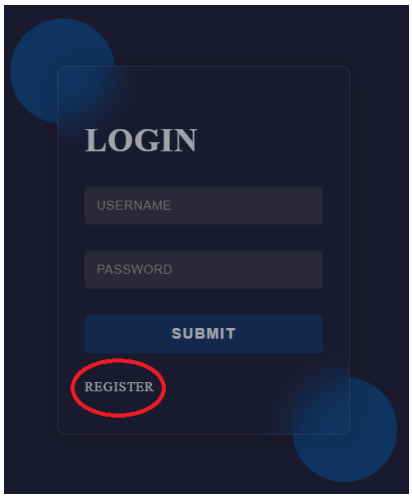
```
docker run -dit --name CONTAINER_NAME --network BRIDGE_NAME -p PORT:PORT
-v PATH\custom_files:/custom_files ghcr.io/gis-ops/docker-valhalla/valhalla:latest
```

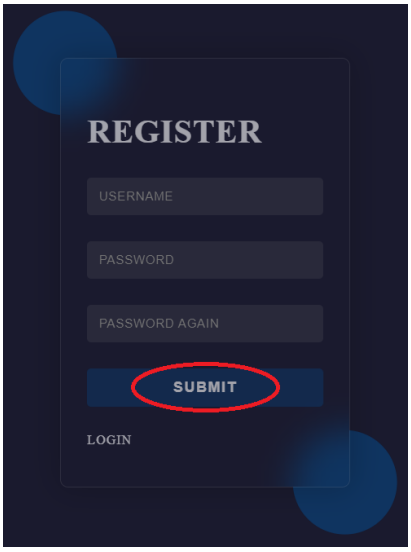
Note: Default setting for
PORT = "8002",
CONTAINER _NAME = "valhalla".
If the container is started with different settings, it is necessary to change the values of the variables in the source code of the web application. Specifically, for *PORT* in the file *map_match.py* in the functions *map_match* the variable *port* and for *CONTAINER_NAME* in the file *upload.js* in the functions *handleUploadAndUnzip* the variable *valhalla_container_name*.

2 Using the web application

2.1 Registration

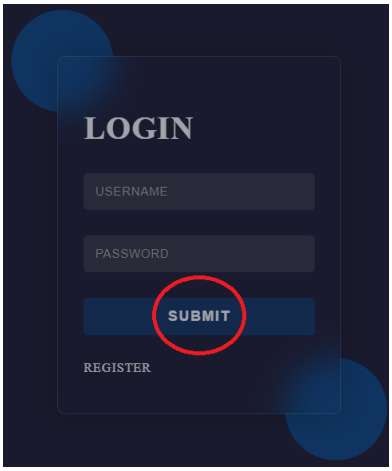
After opening the application, it is necessary to click on the registration button *REGISTER*. After turning off the data, click the *SUBMIT* button.





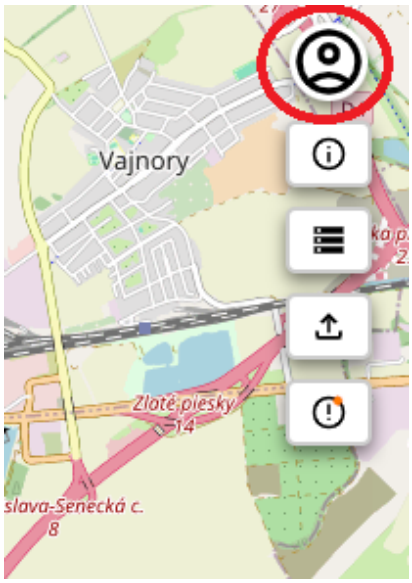
2.2 Login

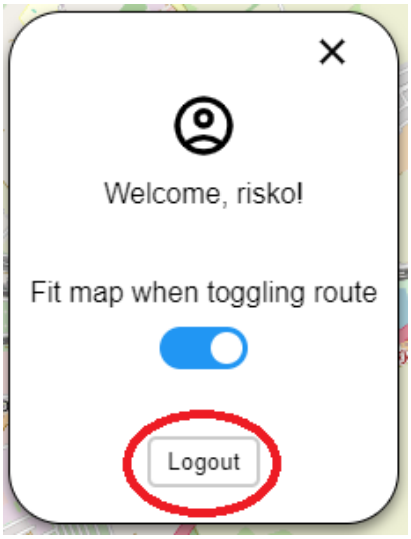
After opening the application and filling in the data, click the *SUBMIT* button.



2.3 Logout

To log out, click the *Profile* button in the upper right part of the map and then click the *Logout* button.





2.4 Upload routes

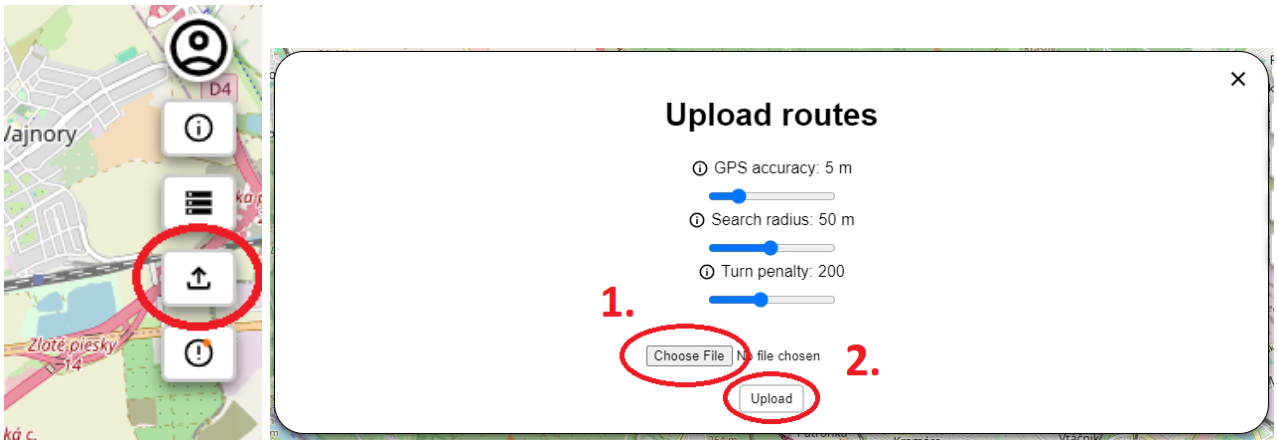
Click the *Upload file* button in the upper right part of the map. After opening the window, select the parameters for attaching routes to the road network. After moving the mouse over the image ⓘ, an explanation for each parameter is displayed. To select a ZIP file, click the *Choose File* button and select the file. The uploaded file must be a ZIP file with the folder structure in it respected. The ZIP file should contain folders named *Drive* or *Walk*. Based on these folders, the route will be attached to the road network.

```
|-- Drive
| |-- 33718.csv
| |-- 31783.geojson
|-- Walk
| |-- 93293.gpx
| |-- 44352.csv
| |-- 764555.geojson
| |-- 354852.gpx
```

Only *csv*, *geojson* and *gpx* files can be found in a ZIP file. The *Csv* file must have two columns whose names contain the string *lat* and *lon*, for example:

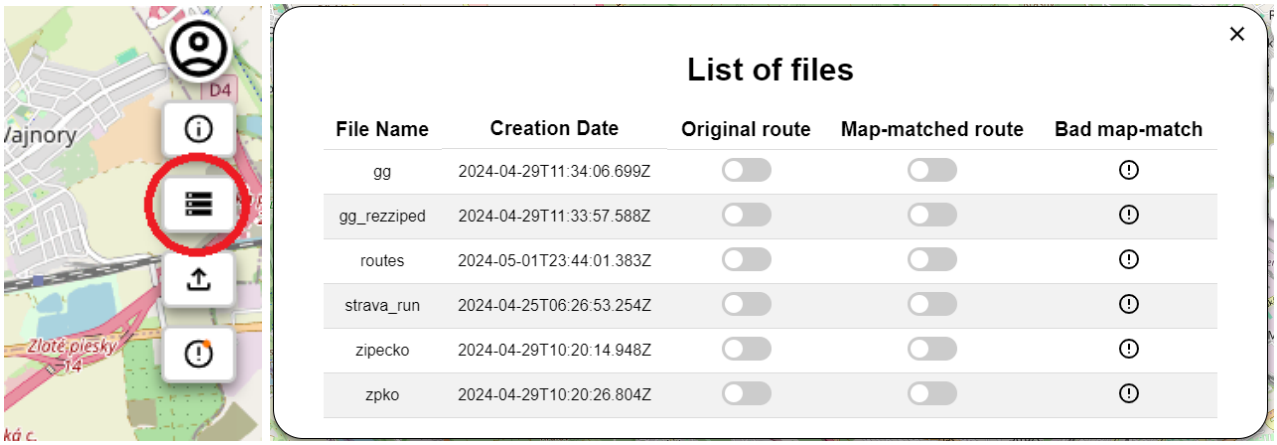
```
lon,lat INDEX,TAG,DATE,TIME,LATITUDE N/S,LONGITUDE E/W,HEIGHT,SPEED
17.07299,48.151611 1,T,240215,144813,48.166046N,17.178055E,132,0.4,356
17.073095,48.151878 2,T,240215,144815,48.165985N,17.178095E,127,5.8,54
17.073084,48.15184 3,T,240215,144816,48.165962N,17.178116E,130,4.0,97
```

To confirm, click the *Upload* button and wait. After processing the ZIP file, you will see a window announcing the processing result.



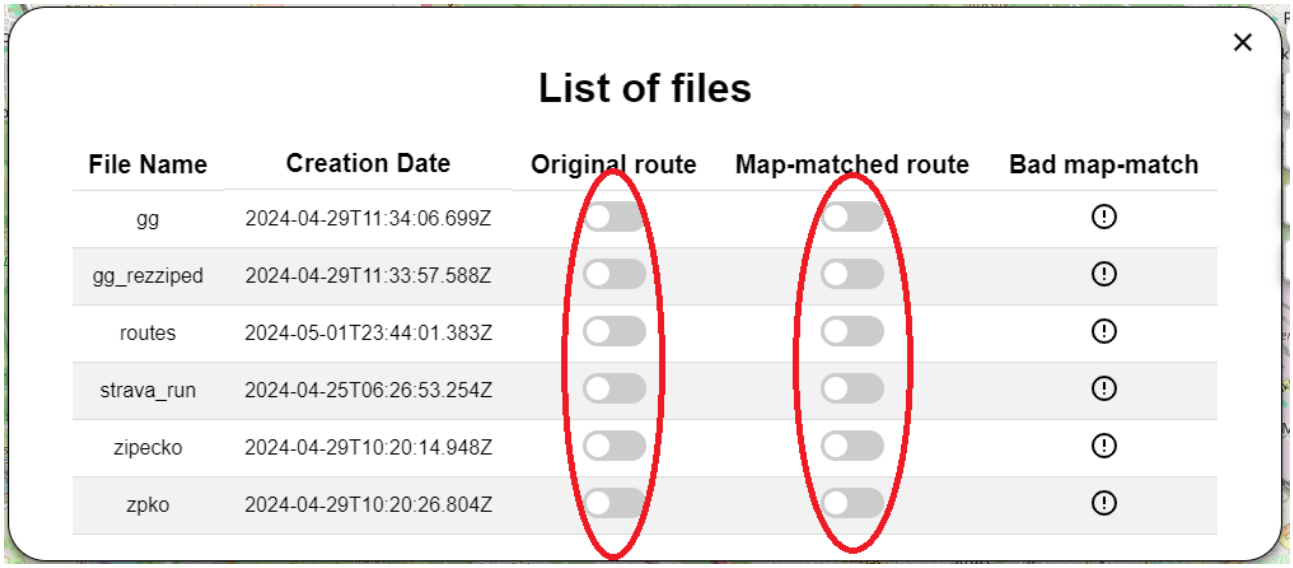
2.5 View Uploaded Files

Click the *Show Files* button. A window will open showing the uploaded files in a table along with the upload date and switches to show the routes on the map.



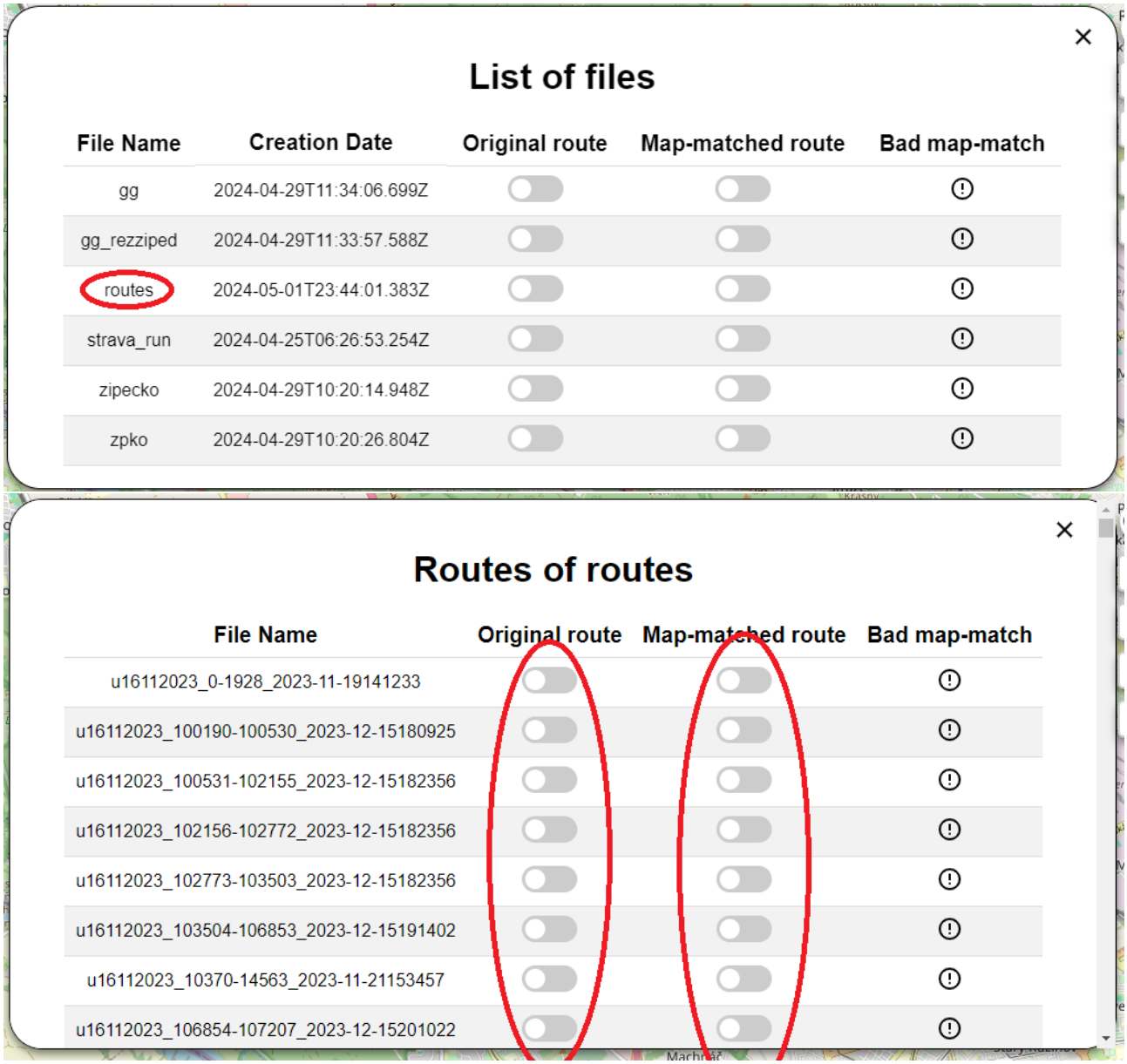
2.6 View all routes in uploaded file

Open the window with uploaded files according to the instructions in section 2.5. After the window opens, click the switch.


List of files
File Name
Creation Date
Original route
Map-matched route
Bad map-match
gg
2024-04-29T11:34:06.699Z
gg_rezziped
2024-04-29T11:33:57.588Z
routes
2024-05-01T23:44:01.383Z
strava_run
2024-04-25T06:26:53.254Z
zipecko
2024-04-29T10:20:14.948Z
zpko
2024-04-29T10:20:26.804Z

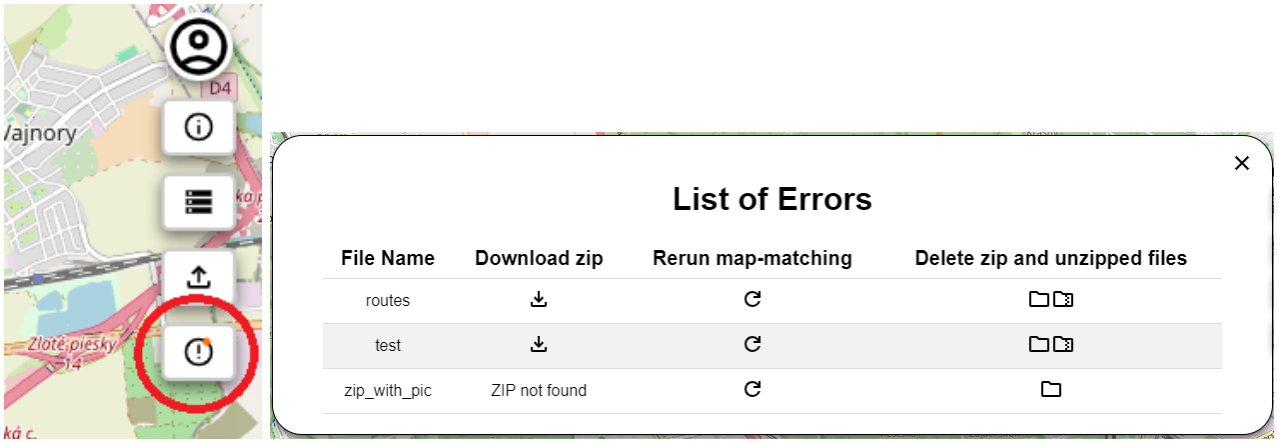
2.7 Displaying individual routes in the uploaded file

Open the window with uploaded files according to the instructions in section 2.5. After the window opens, click on the name of the file whose routes you want to view. After opening the window with the list of routes that the selected file contains, click on the switch.



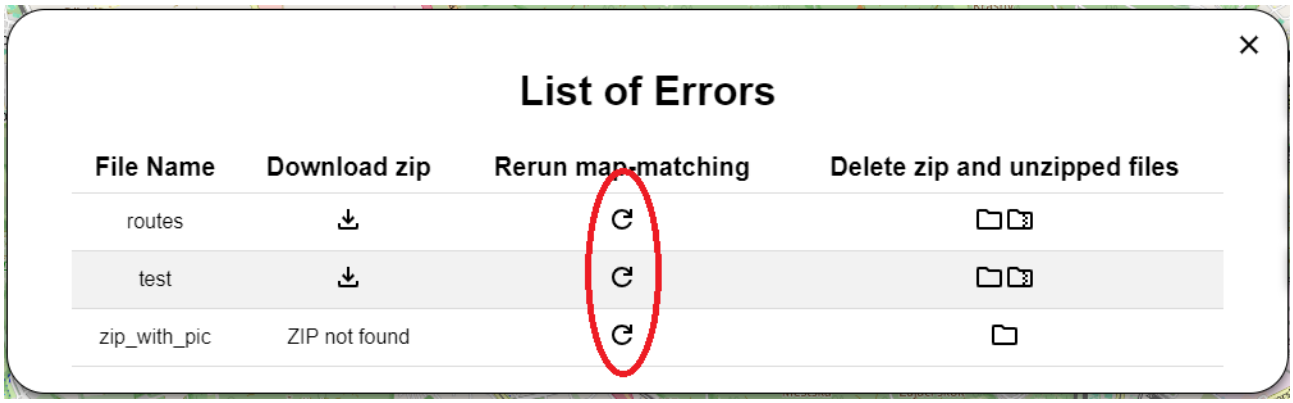
2.8 View error uploaded files

Click the *Error files* button at the top right of the map. A window will open showing the error files in a table.



2.9 Re-run error files

Open the window with error files according to the instructions in section 2.8. Click the restart button. If the uploaded ZIP file could not be unzipped, the button will not be displayed.



2.10 Warning about bad route pinning to the road network

Open the window with uploaded files according to the instructions in section 2.5. After the window opens, click on the notification button.