Automatic Identification System (AIS)

Introduction to the Backend Architecture – Context

For this take home assignment, I have deployed a PostgreSQL database on a Kubernetes cluster to store vessel data from the Automatic Identification System (AIS). The data are collected from Spire and the American Marine Cadastre Hub. One day of data is already entirely loaded into the database: 24-Jan-2024, several other days might be added in the coming weeks.

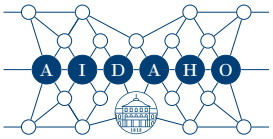
Within PSQL, the TimescaleDB extension is used to convert the dynamic AIS table into a time-partitioned hypertable. The data model is defined in the `.sql` file below.

Each vessel has exactly one row in `ais_static` with fields such as IMO number, call sign, and draught, while the dynamic table `ais_dynamic` stores time-series position updates with geocoordinates, speed, and course for those vessels.

```
--
    ================================================================================


-- 1) Static vessel metadata (one row per MMSI)
--
    ================================================================================


CREATE TABLE IF NOT EXISTS ais_static (
  mmsi                INTEGER          PRIMARY KEY,
  imo                 INTEGER,
  name                TEXT,
  call_sign           TEXT,
  flag                TEXT,
  draught             REAL,
  ship_type_code      SMALLINT,
  ship_type           TEXT,
  length              REAL,
  width               REAL,
  eta                 TIMESTAMP WITH TIME ZONE,
  destination         TEXT,
  static_updated_at   TIMESTAMP WITH TIME ZONE
);
```



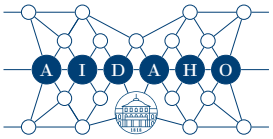

```
--
    ============================================================================

-- 2) Dynamic AIS messages as a Timescale hypertable
--
    ============================================================================


CREATE TABLE IF NOT EXISTS ais_dynamic (
  created_at          TIMESTAMP WITH TIME ZONE,  -- original
     ingestion timestamp
  msg_timestamp       TIMESTAMP WITH TIME ZONE NOT NULL,  -- AIS
     message time
  position_updated_at TIMESTAMP WITH TIME ZONE,
  mmsi                INTEGER NOT NULL,
  latitude            DOUBLE PRECISION,
  longitude           DOUBLE PRECISION,
  speed               REAL,       -- SOG
  course              REAL,       -- COG
  heading             SMALLINT,
  status              SMALLINT,  -- navigational status
  maneuver            SMALLINT,
  accuracy            BOOLEAN,
  rot                 SMALLINT,  -- rate of turn
  collection_type     TEXT,
  PRIMARY KEY (msg_timestamp, mmsi)
);
-- Convert the dynamic table into a hypertable on its message
   timestamp
SELECT create_hypertable(
  'ais_dynamic',
  'msg_timestamp',
  if_not_exists => TRUE
);
-- index on message time alone (helps where-clause on timestamp)
CREATE INDEX ON ais_dynamic (msg_timestamp);

-- index on mmsi alone (helps lookups per ship)
CREATE INDEX ON ais_dynamic (mmsi);

-- composite if you always filter both together
CREATE INDEX ON ais_dynamic (mmsi, msg_timestamp);
```



There is a data ingestion job that loads CSV files into the database. Upserts are used for static data and deduplicated bulk inserts with deduplication.

Important note: Access to the data is via PostgREST via the endpoint `https://aidaho-edu.uni-hohenheim.de/aisdb/`, providing a RESTful API that allows you to query the two tables with SQL-like Syntax.

```
GET /<table_name>?limit=5
GET /<table_name>?
     select=<col1>,<col2>,<col3>
     &<time_variable>=gte.2024-01-24T00:00:00Z
```

where `<col1>, <col2>, <col3>, <time_variable>` as well as `<table_name>` are placeholders. A documentation of the PostgREST Syntax can be found here or can be extracted from a ChatBot of your liking.

This infrastructure provides a real geo-temporal AIS dataset, modern database extensions, and a Kubernetes-based deployment. It is provided in this Gitlab-Repository.

Task 1: Initiate a new git repository

a. Initiate a new R project and `git` repository locally and on the AIDAHO-Gitlab.

   In your Call the repository `AIDAHO_IDS_THAS_2025` and use the `main` branch. Other branches will be ignored.

   Add your team colleagues as well as Johannes Bleher (jbleher) as members with the role *owner* to the repository.

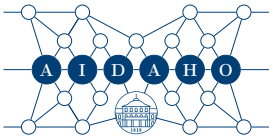
b. Generate the following folder structure :

```
AIDAHO_THAS_2025
├── 00_docs
├── 01_data
├── 02_code
├── 03_report
│   └── graphs
```

c. Store this assignment text in `00_docs`. Your report goes into `03_report`. Store the graphs of your report also separately in the graphs folder.

Task 2: Get an Overview

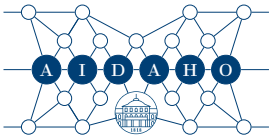
Get hands-on experience querying our Kubernetes-hosted PostgreSQL/AIS database through its PostgREST API. You'll discover what tables and fields are available, inspect sample



records, and compute basic summary statistics. Be smart, do not download the entire table into your R, use the SQL capabilities of the API endpoint and therewith the database to do the heavy lifting. Use R to retrieve the condensed information.

Try to avoid making queries without proper filters to the large table `ais_dynamic` as this will force the database to restart.

1. In your report, start with a short introduction on how AIS data are collected and what information they contain. You will find plenty of information online. But here is a good introduction.
2. What are the available endpoints of the PostgREST-API?
3. In your report, provide a short overview over the data. In order to do so report on the following tasks.
   (a) Briefly describe the AIS data. What information do the respective fields in the table contain?
   (b) How many rows does each table have? How many vessels are recorded?
   (c) Under how many distinct flags do the vessels operate? Which flag is occurring the most?
   Hint: Have a look at the documentation of Automatic Group By
   (d) Provide an overview for each column in the `ais_static` table. How are the data distributed? Describe the data.
   Note: PostgREST only supports the following aggregate functions `avg()`, `count()`, `max()`, `min()`, and `sum()`.
   (e) The table `ais_dynamic` is too large to generate descriptives. Also PostgREST does not offer an endpoint for the `tablesample` procedure of Postgres. Therefore, write a separate script named `ais_dynamic_sample_points.R` and store it in your `02_code` folder. Make sure to adhere to good programming practices – if necessary introduce a functions folder in which you store R-functions. Source the functions in your script. In your script, perform the following tasks:
      i. Iterate over random 5min intervals of Januar 24, 2024.
      ii. Take from each random interval at most 100 observations.
      iii. Stop if you have gathered more than 1000 observations.
      iv. Store the `.csv` file in your repository in the `00_data` folder.
      v. Generate a table of descriptive statistics in your repository to get an overview of the variables.



vi. Display the sampled points on a map using `leaflet`. Indicate the speed of the vessel with a color scheme. Make the graph look nice and integrate it into your report. Also reflect on the sampling procedure in the light of the sampling techniques you have studied in the lecture.

vii. Does the picture change if you only use points with the `collection_type = 'satellite'`? Why could that be?

(f) Write script named `ais_dynamic_individual_paths.R` with which you extract the data from both tables and visualize individual vessels using leaflet. Make sure to adhere to good programming practices. Answer the following questions in your report:

i. Have a look at the data for MMSI=2579999. Do you suspect any data quality issues? If you search online for further information on the MMSI? How are the data generated? What might happen here?
NB: Make use of the Option `clusterOptions = markerClusterOptions()` for this call to leaflet.

ii. Have a look at the MMSI=412420898 and zoom in. Are there any additional issues visible?

4. In order to forecast positions, we assume the following notation

- Latitude ϕ_n
- Longitude λ_n
- Speed v_n (in knots)
- Course θ_n (in degrees from north)

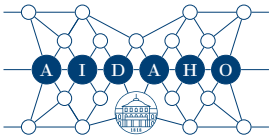
The position can be forecasted on the following formulas for a given time step Δt assuming constant speed. First, we project with the speed and the course the change in position in longitude and latitude (in nautical miles) with

$$\Delta x = v_n \, \Delta t \, \sin(\theta_n), \quad \Delta y = v_n \, \Delta t \, \cos(\theta_n).$$

Second, we need to transform this change back to the change in position in degrees and add it to the previous position,

$$\phi_{n+1} = \phi_n + \frac{\Delta y}{60}, \quad \lambda_{n+1} = \lambda_n + \frac{\Delta x}{60 \, \cos(\phi_n)}.$$

Write a function that performs the calculations for a data frame that contains time ordered data of latitude, longitude, current speed and the current course. Calculate one minute predictions for the MMSI=412420898 – make sure your unit for Δt



corresponds to the velocity measure in knots. Compare your forecast with the true course visually with a leaflet map and integrate it into your report.

Also write a function that calculates the mean squared prediction error (MSPE) for different Δt with

$$\text{MSPE} = \frac{1}{N}\sum_{i=1}^{N}\Big[d\big((\hat{\phi}_i, \hat{\lambda}_i), (\phi_i, \lambda_i)\big)\Big]^2$$

where $d(\cdot)$ is the great-circle distance

$$d\big((\phi_i, \lambda_i), (\hat{\phi}_i, \hat{\lambda}_i)\big) = 2R \arcsin\left(\sqrt{\sin^2\Big(\tfrac{\hat{\phi}_i - \phi_i}{2}\Big) + \cos(\phi_i)\,\cos(\hat{\phi}_i)\,\sin^2\Big(\tfrac{\hat{\lambda}_i - \lambda_i}{2}\Big)}\right)$$

with $R = 3440$ nautical miles being Earth's radius.

What happens when you increase Δt? Calculate the MSPE for MMSI=412420898 for three Δt. Also calculate the MSPE if your prediction would be the last available observation. Report the MSPEs in your report.

5. Can you come up with a model that performs better for 10 minute predictions than the one based on the procedure in 4.?

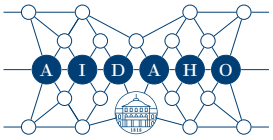
Task 3: Static Sample-Points Dashboard

For this assignment integrate the files hosted in the GIT repo

`https://aidaho-edu.uni-hohenheim.de/gitlab/jbleher/shiny_thas2025`

into your THAS-repository. In this repository a `docker-compose.yaml` file is located that sets up a NGINX sever as well as a shiny app. After integrating the files into your project, clone your repository on your server.

1. In your `02_code` folder, create `sample_dashboard.R`. It should:
   (a) Read in the points from your sampling function in Task 2.3 e) into a `data.frame`. If you have not been able to code Task 2.3, use the `sample.csv` in the repo.
   (b) Build a `leaflet` map and save it via `htmlwidgets::saveWidget()` in a HTML website `sample_points.html`.
   (c) Connect the `sample_points.html` to the document root of the NGINX server.
2. Test and debug your system so that the website is served under `https://<your-server>/sample_points.html`.



Task 4: R Shiny Application

Make the Shiny app in the provided repo work by adjusting a few things.

1. Connect the shiny app files via the `docker-compose.yaml` to the shiny container from the registry of the Gitlab repo.
2. Have a look at the provided `server.R, ui.R` and `global.R` in the public repo.
3. Modify the helper functions in the subfolder `functions` to point at your PostgREST URL and to call your own individual path and prediction routines.
4. Adjust the `server.R` file so that the position predictions are also displayed.
5. Adjust the config-file of the NGINX so that you can access the under the endpoint `GET /mmsi-search`.
6. Add a controlfield to `ui.R` to adjust the time step Δt for the prediction algorithm. Adjust the `server.R` so that the inputed value is used.
7. Verify in your browser: `https://<your-server>/mmsi-search`.

This assignment is due on June 27, 2025, end-of-day. Please hand in your report via ILIAS and save it in your Gitlab repository. Assignments without a duely submitted report are not graded. Reference all team members with matricle numbers in your report. Also reference your Gitlab repository.