

ReNEW

D3.4 Green Resilient IWT Dataspace components solutions and tools v1



Funded by
the European Union

Disclaimer



Funded by
the European Union

Funded by the European Union. Views and opinions expressed are, however, those of the author(s) only and do not necessarily reflect those of the European Union or the European Climate, Infrastructure and Environment Executive Agency (CINEA). Neither the European Union nor the granting authority CINEA can be held responsible for them.

The authors, reviewers, and contributors participating in the ReNEW Consortium took all the necessary measures in good faith to provide the information in the most accurate, consistent, and lawful manner. However, neither the ReNEW consortium, its members, nor any party mentioned here above shall not be responsible nor liable for any errors or omissions in the content of this document or for any type of losses, damages or costs resulting from the use of the data mentioned in this document.

Document Info

Document

Title	Resilience-centric Smart, Green, Networked EU Inland Waterways
Acronym	ReNEW
Call	HORIZON-CL5-2021-D6-01
Type of Action	HORIZON-IA
Grant Number	101069682
Start date	1 September 2022
Duration	36 months
Document identifier	D3.4 Green Resilient IWT Dataspace components solutions and tools v1
Due Date of Delivery to EC	28-02-2024
Actual Date of Delivery to EC	28-02-202x
Dissemination level	PU - Public
Work package	WP3 - European IWT Data Space and Digital Twin solutions for Resilience and Sustainability
Lead Beneficiary	KONNECTA SYSTEMS

Contributor(s)

Main Contributor(s)	Antonis Antonopoulos (KNT)
Contributor(s) (alphabetical order)	Andreas Alexopoulos (KNT) Thanos Karydis (INLE) Socrates Katsoulakos (KNT) Valia Konstantopoulou (INLE) Stelios Kossieris (ICCS) Antonis Mygiakis (KNT) Eleni Voulgari (KNT)
Reviewers (s)	Nik Delmeire (IWT) Malek Hichem (Akkodis) Mandimby-Nirina Ranaivo-Rrakotondravelona (Akkodis)

History Sheet

Version	Date	Editor	Summary of Modifications
0.1	01/12/2022	Antonis Antonopoulos	ToC
0.2	01/9/2023	Antonis Mygiakis	First draft
0.3	01/11/2023	KNT Team	Final edit of “operational core” Initial content for connectors library
0.4	10/11/2023	Stelios Kossieris	Integration of content related to sensors
0.5	15/11/2023	INLE Team	Edits on data sources
0.6	18/12/2023	KNT Team	Final edit of connectors library
0.7	10/01/2024	Antonis Antonopoulos	Objectives Mapping Overview Conclusion
1.0	26/01/2024	Antonis Antonopoulos	Final edit

Acronyms and Abbreviations	Meaning
AI	Artificial intelligence
AIS	Automatic Identification System
AIVDM	Automatic Identification System VHF Data Message.
API	Application Programming Interface
CEF	Connectivity Europe Facility
CEMT	Conférence Européenne des Ministres des Transports
COG	Course Over Ground
DAG	directed acyclic graphs
ECDIS	Electronic Chart Display and Information System.
ERI	Electronic Reporting
ETA	Estimated Time of Arrival
EuRIS	European River Information System
GeoJSON	Geographic JavaScript Object Notation
GRIWT	Green Resilient Inland Water Transport
HTTP	Hypertext Transfer Protocol
IENC	Inland Electronic Navigational Chart
IHO	International Hydrographic Organization
IMO	International Maritime Organization
IWT	Inland Waterway Transport
JSON	JavaScript Object Notation
KG	Knowledge Graph
LDES	Linked Data Event Streams
MMSI	Maritime Mobile Service Identity
NMEA	National Marine Electronics Association
NTS	Notices to Skippers
OSM	OpenStreetMap
PKCE	Proof Key for Code Exchange
Rest	Representational State Transfer
RIS	River Information Services
SOG	Speed Over Ground
SOLAS	International Convention for the Safety of Life at Sea
TSDB	Time-Series Databases
VTs	Vessel Traffic Services
XML	Extensible Markup Language
YAML	YAML Ain't Markup Language

Table of Contents

OBJECTIVES	9
1.1 MAPPING OUTPUTS AND PROJECT REQUIREMENTS	10
1.2 OVERVIEW AND STRUCTURE	11
2.1 INTRODUCTION	12
2.2 POSITION WITHIN THE RENEW FRAMEWORK	13
2.3 DATA SOURCES REQUIREMENTS AND CONSIDERATIONS	15
Data Inputs	16
Data interfaces	18
Data Integration for static, real-time and historic data	19
3.1 THE CENTRAL CONNECTOR	22
Introduction to Apache Kafka as the Central Connector	22
Central connector architecture overview	23
Potential expansions with Kafka integrations	24
Managing Kafka's Backbone	24
3.2 DATA STORAGE LAYER	26
Timeseries database for dynamic data	26
Knowledge graph for static data and structured metadata	27
NoSQL database for configuration storage	28
Temporary data storage	29
3.3 OPERATIONAL COMPONENTS	30
Web app backend	30
Platform monitoring	31
Processing Units	31
Task Scheduler	32
Model Execution Engine	32
3.4 CONNECTORS LIBRARY	33
4.1 WATER LEVEL SENSORS	34
4.2 EURIS	37
Introduction to River Information Systems and EuRIS	37
EuRIS reference network model	38
EuRIS web services	40
Hydrometeo services	41
Inland Electronic Navigational Chart Service	42
Notifications to shippers	42
EuRIS network data	43
4.3 OPEN STREET MAP (OSM)	45
Object Types in OpenStreetMap (OSM)	46
Export Types: Transforming OpenStreetMap (OSM) Data	47
Overpass API: Querying OpenStreetMap (OSM) Data	48
4.4 AUTOMATED IDENTIFICATION SYSTEM (AIS)	49
Information provided by the AIS	49
Raw signal acquisition and decoding	51

Commercial service providers.....	52
5.1 'EURIS API CALLER' MICROSERVICE	54
5.2 'HYDROMETEO_DATA' MICROSERVICE	55
5.3 'OSM DATA MAINTAIN' MICROSERVICE	56
5.4 'EURIS WEBHOOK DATA INGESTION SERVICE' MICROSERVICE	57
5.5 'AIS SERVICE' MICROSERVICE.....	60
5.6 'INGESTION KG API' MICROSERVICE.....	63
5.7 'KG - KAFKA CONNECTOR' MICROSERVICE.....	64
6.1 GENERAL DESCRIPTION OF THE DEPLOYMENT	65
6.2 REPOSITORY AND DEPLOYMENT	68
6.3 THE KNOWLEDGE GRAPH	70
6.4 TIMESERIES DATA STORAGE	72
6.5 USER INTERFACE TO VISUALIZE AVAILABLE DATA.....	73

List of Tables

TABLE 1 ADHERENCE TO RENEW'S GA DELIVERABLE & TASKS DESCRIPTIONS	10
TABLE 2 SAMPLE RETURNED PAYLOAD BY THE HYDROMETEO WEB SERVICE.	41
TABLE 3 SAMPLE NOTIFICATION TO SHIPPERS RETRIEVED VIA EURIS API	43
TABLE 4 SAMPLE 'BERTH' OBJECT	44
TABLE 5 SAMPLE 'TERMINAL CALL RESPONSE' OBJECT	44
TABLE 6 SAMPLE BRIDGE OBJECT	45
TABLE 7 SCHEMA OF THE EXPECTED POST OPERATION AT THE WEBHOOK SERVICE	59
TABLE 8 POSITION REPORT SAMPLE MESSAGE	59
TABLE 9 AIS MESSAGE TYPE 1 "RAW" PACKET DATA EXAMPLE	61
TABLE 10 AIS MESSAGE TYPE 5 "RAW" PACKET DATA EXAMPLE	61
TABLE 11 "MMSI" KAFKA TOPIC SCHEMA	62
TABLE 12 "MESSAGE TYPES" KAFKA TOPIC SCHEMA	62

List of Figures

FIGURE 1: THE DUET T-CELL ARCHITECTURE FOR INTEROPERABILITY [3]	14
FIGURE 2 DIGITAL TWIN DATA INTERACTIONS	16
FIGURE 3 GENERAL LAYOUT OF THE OPERATIONAL CORE OF THE DIGITAL TWIN FOR DATASPACE DYNAMICS	22
FIGURE 4 THE TWO LOCATIONS FOR THE INSTALLATION OF WATER LEVEL AND WATER QUALITY SENSORS.	34
FIGURE 5 THE SENSOR BOX INSTALLED IN ZELZATE MARINA IN THE FRAMEWORK OF SOCORRO PROJECT.	35

FIGURE 6 POLE IN 'EVERSTEIN' LOCATION, WHERE THE DATALOGGER AND THE SOLAR PANEL WILL BE INSTALLED.	36
FIGURE 7 EURIS FAIRWAY SECTIONS DEFINITION [17]	39
FIGURE 8 EURIS INTERFACE TO MANUALLY LOCATE AND DOWNLOAD ELECTRONIC MAPS	42
FIGURE 9 GIT REPOSITORY: 'EURIS API CALLER' MICROSERVICE	55
FIGURE 10 GIT REPOSITORY: 'HYDROMETEO_DATA' MICROSERVICE	56
FIGURE 11 GIT REPOSITORY: 'OSM DATA MAINTAIN' MICROSERVICE	57
FIGURE 12 CONFIGURATION OF WEBHOOK URL IN EURIS PORTAL	58
FIGURE 13 GIT REPOSITORY: 'EURIS WEBHOOK DATA INGESTION SERVICE'	58
FIGURE 14 GIT REPOSITORY: 'AIS SERVICE' MICROSERVICE	60
FIGURE 15 GIT REPOSITORY: 'INGESTION KG API'	63
FIGURE 16 GIT REPOSITORY: 'KG - KAFKA CONNECTOR'	65
FIGURE 17 MAPPING OF DEVELOPED CONNECTORS TO RESPECTIVE DATASOURCES AND THE DATA PROCESSING LAYERS OF THE PIPELINE	67
FIGURE 18 THE RENEW GITLAB REPOSITORY	69
FIGURE 19 SCREENSHOT OF DOCKER CONTAINERS RUNNING ON SERVER	70
FIGURE 20 KNOWLEDGE GRAPH NODES LABELS AND THEIR COUNT	70
FIGURE 21 HYDROMETEO SENSOR PROPERTIES	71
FIGURE 22 VESSEL PROPERTIES	72
FIGURE 23 DISPLAY OF VESSEL DATA TRENDS IN THE INFLUXDB INTERFACE	73
FIGURE 24 SCREENSHOTS OF THE USER INTERFACE, (A) OVERVIEW OF THE AVAILABLE NODES, SHOWING THEIR GEOGRAPHICAL DISTRIBUTION, (B) A LOCK ACQUIRED FROM OSM, (C) VESSELS IN DOURO RIVER AND (D) BRIDGES IN GENT.	74

1 Executive summary

This report summarizes our efforts to develop a Green Resilient Inland Water Transport (GRIWT) Dataspace and a library of tools that facilitate DT Service and Application development, aiming to support the sustainability and resilience of inland water transport (IWT).

In this respect, we explored the complexity of data dynamics within the IWT ecosystem, identifying and integrating pivotal data sources such as environmental sensors and geographic systems. These components form the foundation of our digital twin data pipeline, enabling a deeper understanding and interaction with the IWT environment.

The architectural blueprint presented in this report reflects our commitment to innovation and efficiency. By integrating diverse open-source components, we have developed a robust framework that serves as the operational core of the digital twin, supporting various functionalities from platform monitoring to advanced modelling.

A key aspect of our strategy is the creation of a connector library, facilitating seamless data integration from multiple sources. This library enhances data interpretability and usability, enabling automation and smart monitoring within the IWT domain.

Our efforts resulted in the development and deployment of a prototype operational system, showcasing the practical applicability of our concept. This system represents our methodology and dedication to practical solutions, demonstrating the efficiencies that can be realized through our framework.

This report represents our contribution to the development of a Dataspace and a generic Digital Twin framework, aimed at supporting a future where water transport operations are more efficient, reliable, and aligned with principles of sustainability and resilience.

2 Introduction

2.1 Objectives

The objectives of this report are to architect and implement a suite of data access and connectivity solutions, termed 'solution accelerators', described in Section 4 essential for the operational efficacy of the ReNEW digital twin in a dataspace dedicated to fostering sustainable and resilient inland water transportation. These solution accelerators are designed to seamlessly ingest data across varying temporal scales, from real-time to extended durations, thereby underpinning the digital twin's data pipeline and interface. By leveraging protocols and ontologies related to the IoT, shipping, logistics, and supply chain, the technical solution accelerators can provide solutions to business matters, thus becoming part of the broader spectrum of business solution accelerators, facilitating addressing the issues and challenges that LLs are facing.

The report aims to detail the strategic selection and integration of diverse data sources—including IoT devices, environmental sensors, and external datasets like weather and geospatial information—into a coherent system that enhances the interpretability and utility of data for the digital twin platform. By compiling best practices and developing a versatile connector library, the report seeks to ensure smooth deployment and integration of these technologies into the Living Labs, ultimately facilitating an advanced, scalable framework that supports dynamic data mapping and effective decision-making processes in the management of inland waterways.

2.2 Mapping outputs and project requirements

Table 1 maps the ReNEW's Grant Agreement commitments, both within the formal Deliverable and Task description, against the project's respective outputs and work to perform.

Table 1 Adherence to ReNEW's GA Deliverable & Tasks Descriptions

Deliverable D3.4: Green Resilient IWT Dataspace components solutions and tools v1		
Library Connectors v1.0, and Solution Accelerators v1.0; prototypes and report		
T3.2: Green Resilient IWT Dataspace components, solutions, and tools		
Create the required data access and connectivity acceleration resources and assure a smooth deployment and integration of the technologies into the Living Labs.		
ReNEW GA Element Title	ReNEW GA Element Outline	Respective Document Chapter (s) and Description
ST3.2.1 Green Resilient IWT Dataspace components, solutions, and tools	Produce a library of tools that facilitate DT Service and Application development. Review existing open-source technologies and tools and utilise outputs of research projects, customizing them to fit the ReNEW objectives, as well as developing new tools specific to IWT needs primarily focused on the LL user needs as expressed in the user stories. At the outset, the ReNEW Digital Twins and Data Spaces solutions library will include tools for: •DT modelling, mainly behavioural and rule modelling, •DT Data management, including collection, transmission, storage, processing, fusion & visualization, •DT Connection tools that connect the physical and virtual/digital words as well as the rest of the DT parts. Various connectors will be designed to integrate with external systems, legacy applications, IoT devices, etc. for the Living Labs and the resilient and sustainable inland waterways resources.	In Ch.3 is described the stack of open tools that have been selected to compose an operational core to facilitate DT Service and Application development. These components have been selected to offer data management, support modelling & simulation, support Big Data Analytics and AI applications, interconnect the DT parts and provide application service tools. In Ch.5 are described the connectors to integrate with external systems and connect the physical - virtual/digital words. In Ch.6 is described the customization and composition of the selected components to form an operational deployment.

	• DT Services delivery and analytics tools such as: (a) services platforms that integrate IoT, Big Data, Machine Learning, (b) application service tools that include monitoring, optimization, diagnostics, etc., (c) analysis and prognosis tools that enable predictive maintenance of IWT resources, reducing downtimes by utilizing advanced analytics methods and machine learning, deep learning, neural networks, etc.	
ST3.2.2 Data integration and data space sharing components and connectors, solution Accelerators	Create the solution accelerators which are an essential part of the ReNEW architecture. These components will be set to ingest data over a long or a short (e.g., real-time) time intervals. Design the solution accelerator templates and implementing these to meet the LL requirements. The solution accelerator template structure will consider the emerging IoT standards like NGSI-LD. Consider industry standards such as GS1, TIC4.0, IPCSA etc., and ontologies as the ones developed in the CEF projects of DTLF, Fenix and FEDeRATED. Implement the necessary components for dynamically mapping data from IoT sensors or other external data sources (e.g., public administration IT infrastructure) with Living Labs elements.	In Ch. 2, is conducted an analysis of the use cases pertinent to the ReNEW project from the data requirements perspective. In this context various protocols and ontologies are evaluated for their applicability in the ReNEW use cases. In Chapter 3, we describe a flexible architecture that prioritizes efficiency and adaptability for data ingestion. This architecture serves as a solution accelerator, enabling rapid and scalable data integration. This is demonstrated in the integration of numerous data connectors in the deployment described in Ch. 6.
ST3.2.3 Library of Data Connectors	Access to the physical infrastructure will be established by deploying distributed and microservice-based APIs, which will allow data filtering and will indicate how data can be written to and read from the message queue, forming the Digital Twin backends' heart. Compile and produce a combination of components and best practices to encapsulate and process data from other systems, thereby creating a connector Library, and implement the connectors to the data sources to ensure data interpretability for the DT solution platform. The library will facilitate the ease to integrate data from different sources, including, but not limited, to IoT data as well as external data such as weather, smart city and citizen science, geo-tagging data etc.	Ch. 2 outlines the data requirements, while Ch. 4 details the relevant data sources for the use cases. Ch. 5 describes the development of a connectors library, enabling integration of these sources into the Digital Twin dataspace. Ch. 6 showcases the deployment of these connectors and their interaction with a messaging queue, which constitutes the core backend of the Digital Twin platform.

2.3 Overview and structure

Chapter 2 situates this study within the broader ReNEW dataspace and digital twinning framework, setting the stage for understanding its contribution to the field. It defines the data requirements by categorizing them based on their level of interactability and dynamic nature. This systematic approach aids in identifying and addressing the specific needs and challenges associated with the ReNEW project, ensuring that the digital twin is equipped to handle various data types and flows effectively.

Chapter 3 presents an architecture designed to serve as the operational core of the digital twin's data pipeline and dataspace interface. It explores various open-source components, assessing their viability and integration into the core backend, storage layer, and operational components. The chapter further explores solutions tailored for essential functions such as platform monitoring, modeling, scheduling, and the backend of the user interface. This thorough investigation ensures that the proposed architecture is robust, scalable, and capable of supporting the complex requirements of the digital twin framework.

Chapter 4 provides insights into the selected data sources for integration. It begins by outlining plans for deploying water level sensors under the ReNEW project, strategically positioned near the Living Labs (LLs). Various connectivity methods relevant to these sensors are discussed. The chapter then examines the European River Information System (EuRIS), detailing its reference network model and data sharing services. OpenStreetMap, valued for its geographic information, is also discussed, focusing on its data schema and querying capabilities via its API. Finally, the operational aspects of AIS (Automatic Identification System) are explored, along with considerations for its integration and interfaces within the digital twin framework.

Chapter 5 outlines the microservices developed to gather data from the designated data sources, forming the connector library.

Chapter 6 focuses on integrating the foundational Digital Twin Dataspace components with the connectors' library. We aim to develop a prototype operational system based on the components discussed in Chapter 3. This system will be capable of collecting, processing, and granting access to the data sources outlined in Chapter 4. We will provide an overview of the steps taken to merge these components seamlessly, highlighting the technical complexities and strategies employed for successful integration. Through this process, we aim to demonstrate the practical implementation of our digital twin framework, showing its effectiveness in addressing real-world data challenges and facilitating efficient data utilization.

Chapter 7 presents the conclusions of this report.

3 Dataspace integration requirements

3.1 Introduction

In the domain of maritime operations, data is harvested from an extensive array of sources including onboard systems like the ship's bridge data network, established automation mechanisms, emerging 'e-ship' technologies, performance tracking, Automatic Identification System (AIS), Vessel Traffic Services (VTS), and external sources such as weather, environmental monitoring, port, traffic and

infrastructural information. This diverse data landscape introduces challenges related to data ownership, diversity, and integrity, but also interoperability and integration which are critical issues that need to be addressed by a robust data processing framework, such as the one proposed for the ReNew project.

In this chapter we introduce an operation core which is a compilation of tool that composes a data pipeline for the Data management, including collection, transmission, storage, processing, fusion & visualization. This core is also capable of hosting modelling features, both analytical and data driven (machine learning, deep learning, neural networks, etc).

Initially, reference is made to an earlier deliverable where the outline of the overall Renew architecture is set and the relevant to this deliverable aspects are described. Subsequently, the initial data source requirements necessary for integration into the platform, are identified aiming to support the use cases defined within the project up to this point. Finally, the document provides a comprehensive analysis of the proposed operational core, examining each of its components in detail.

3.2 Position within the ReNEW framework

The ReNEW project deliverable D3.1 [1] defines the basic architectural characteristics of the framework and the infrastructure components where the result of the present deliverable belong. The foundational principles of the architecture are expounded, referencing the current state of the art in Digital Twins and Data Spaces. The document proposes a set of building blocks for a generic digital twin architecture within a dataspace context. These include core digital twin components [4], dataspace connectors, and robust, efficient data transfer building blocks that adhere to open standards for interoperability. The combination of these elements is expected to lead to the development of a solution accelerator. Additionally, the document outlines a generic approach for integrating data and algorithms into the digital twin. This topic is scheduled for further elaboration in the next version of the document, which will benefit from more specific insights and applications derived from the scope of the Living Labs.

In Figure 1 is depicted the conceptual architecture of the DUET (Digital Urban European Twins) system. This architecture highlights that, in addition to several essential components that connect various elements, the digital twin is essentially a distributed system composed of multiple systems. This particular setup poses challenges related to robustness, scalability, and interoperability. In the following section, we will explore the architecture by examining its various components.

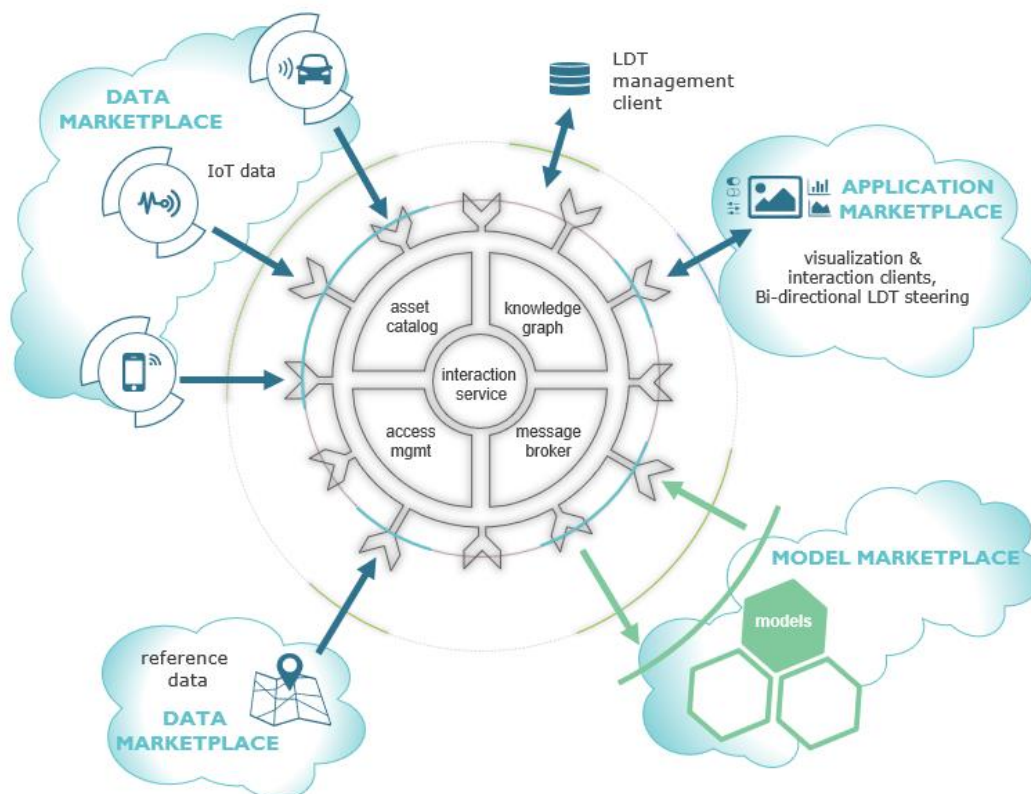


Figure 1: The DUET T-Cell architecture for interoperability [3]

Integrating data into the digital twin necessitates a comprehensive approach to various Data Services functionalities, including the acquisition and ingestion of data, streaming, and transformation. Additionally, it encompasses critical aspects such as managing metadata and providing data storage solutions. In the following sections, we will explore the capabilities deemed most vital within the context of the ReNEW project.

Data ingestion

Accessing and utilizing data from various sources presents a set of challenges. These scenarios illustrate different circumstances:

1. When the data source is an external entity providing a static dataset, the data can be utilized as needed or imported into a storage solution. If necessary, it may also undergo transformation to conform to a specific standard format.

2. In cases where an external entity supplies a dynamic dataset, such as sensor data, one approach is to directly integrate the data stream into the digital twin. This method relies on the data being readily available with minimal delay. Alternatively, it might be more effective to replicate the data to a location closer to its point of use, already in a usable format. Technologies like Linked Data Event Streams (LDES)[5] or others, depending on the provider's capabilities in linked data and event streaming, can facilitate this process.
3. For dynamic datasets published by an entity within the digital twin ecosystem, adopting LDES for both data publishing and consumption is advised to achieve optimal performance and data availability.

Data storage

While directly accessing data from the publisher's services is feasible, digital twins frequently have specific requirements regarding data formats, availability, and latency. Consequently, a more dependable strategy is to duplicate the data near its point of use. This could be adjacent to the simulation models consuming the data or near where the data is to be displayed.

To accommodate various needs, one can employ suitable services that are straightforward to implement (for instance, through Docker). When necessary, the LDES standard can also be utilized to facilitate the replication of data to these services.

Data processing and transformation

Data frequently requires preprocessing, which may include steps like filtering, transforming, aggregating, anonymizing, and calibrating, before it becomes usable. In cases where the data publisher does not provide these processed data streams independently, such functions can be integrated into Linked Data Event Streams (LDES) workflows but can be also handled internally.

Metadata management (including Ontology Management and Model repositories)

For effective utilization of the Digital Twin, users must be able to find, store, and repurpose datasets, simulation/AI models, and other resources within the digital twin framework. This necessitates a robust infrastructure for managing assets, encompassing ontology management and model repositories.

3.3 Data sources requirements and considerations

In the context of the digital twin, as illustrated in Figure 2 we can identify two fundamental types of data interaction, namely "Data Inputs" and "Data Interfaces", which are discussed in the following paragraphs.

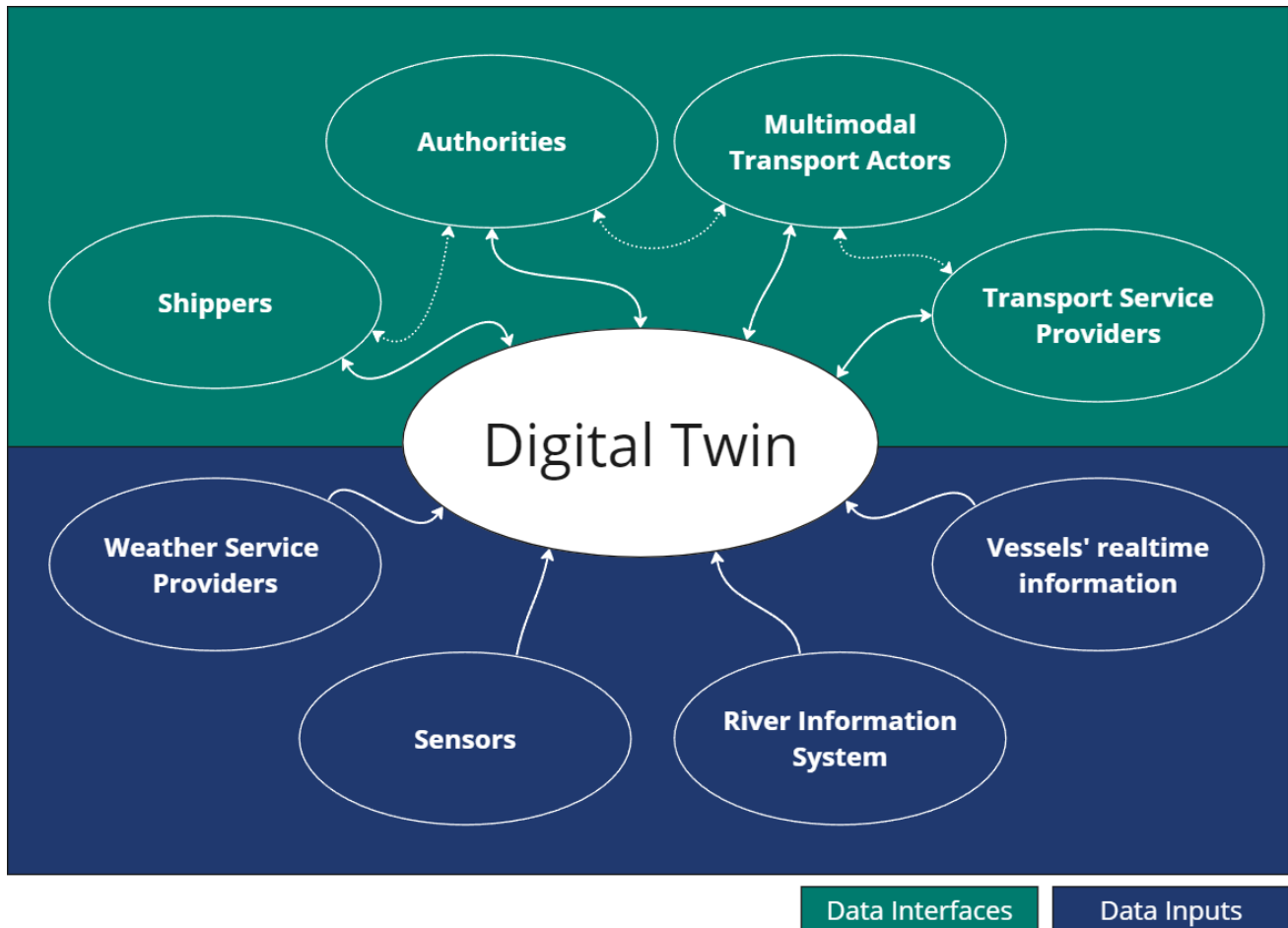


Figure 2 Digital twin data interactions

Data Inputs

The initial type of data interaction involves feeding information into the digital twin, drawing from a diverse range of sources such as sensor readings and topological information. These inputs are vital, furnishing the digital twin with up-to-the-minute data that allows it to accurately mirror and comprehend its physical counterpart. For example, sensor information quantifies the present conditions and dynamics of the physical system, while topological information enhances the understanding of its physical layout. Integrating this data provides the digital twin with the necessary insight to operate efficiently, delivering a detailed reflection of the actual system and facilitating a variety of analytical and decision-making activities.

In this domain of data provision to the digital twin, we can categorize the input into two specific types: static and dynamic data.

Static data

Static data refers to information that does not frequently change, covering aspects such as the geographic positioning of infrastructure elements and vessel registrations. This static information lays the groundwork, enabling the digital twin to grasp the context of its operational environment. As of this report, efforts to incorporate the following sources are underway.:

- The European River Information System (EURIS) acts as an invaluable source of data concerning Europe's waterways, especially its abundance of static information. This static data comprises particulars about permanent infrastructures like bridges, locks, ports, and other crucial infrusrtuctures. Understanding the importance of this foundational information for various stakeholders, EURIS offers connectors to facilitate the efficient retrieval of such data through its platform's APIs. By incorporating these precise and comprehensive structural descriptors, the developed digital twin will be able to reflect the physical and constructed environment. This integration serves as the foundation upon which dynamic interactions and simulations can be constructed, ensuring that our digital twin remains firmly grounded in the realities of Europe's waterway infrastructures.
- OpenStreetMap (OSM) serves as a valuable global repository of geographical and infrastructural data, continuously updated by a vast network of contributors. Its open-source nature ensures a comprehensive collection of information, including details often overlooked by other mapping platforms. In the context of our digital twin, leveraging OSM is essential to access supplementary data beyond the scope of EURIS. While EURIS provides specific static data on Europe's waterways and related infrastructures, OSM complements this by offering information on surrounding road networks, urban layouts, landmarks, and other pertinent geographical features. Integrating OSM data enriches our digital twin's understanding of the environment, facilitating a more comprehensive and interconnected virtual representation. Moreover, the ongoing updates from the OSM community ensure that our digital twin remains synchronized with real-world changes, maintaining its accuracy over time.

Dynamic data

On the other side, dynamic data encompasses information that undergoes frequent updates and fluctuations. This category predominantly includes metrics reflecting real-time conditions and operational parameters, such as water levels, weather conditions, vessel operational states, and other variables subject to variation during system operation. Dynamic data plays a pivotal role in endowing the digital twin with adaptability, responsiveness, and the capacity to make informed decisions based on the evolving conditions it monitors. By amalgamating both static and dynamic data, the digital twin maintains a holistic perspective of the physical system it emulates, bolstering its capability to conduct real-time analysis and facilitate effective decision-making processes.

As of the writing of this report, integration efforts are underway for the following sources:

- EURIS' Real-Time Data: The European River Information System (EURIS) provides valuable real-time updates regarding Europe's waterways, including data on water flow rates, vessel movements, lock schedules, and more. This dynamic input ensures that the digital twin can

react and predict based on immediate conditions, reflecting the current state of waterways accurately.

- Weather Service Providers: Integrating real-time meteorological updates from weather service providers improves the digital twin's accuracy and responsiveness. Information on temperature, precipitation, wind patterns, and more allows the twin to anticipate disruptions, adjust operations based on current weather conditions, and provide timely recommendations to stakeholders.
- Water Level and Water Quality Sensors: Understanding water depth and quality is crucial for waterway operations. By incorporating sensors that monitor water levels, the digital twin can assess navigability, predict potential flooding or receding water levels, and optimize vessel movements. Similarly, water quality sensors provide insights into parameters such as pH levels, turbidity, dissolved oxygen, and contaminants, informing about environmental health and influencing decisions related to waterway usage.
- AIS Receivers: The Automatic Identification System (AIS) is vital for maritime safety and navigation. By integrating data from AIS receivers, the digital twin gains access to real-time information about ship positions, headings, speeds, and other vessel-specific details. This data aids in monitoring vessel traffic, assessing congestion in ports or waterways, predicting potential collision scenarios, and optimizing route planning. Additionally, AIS data contributes to enhanced maritime situational awareness and operational efficiency.

Data interfaces

A digital twin operates within a complex ecosystem, interacting with various stakeholders such as shippers, authorities, multimodal transport actors, and transport service providers. This interaction serves two main purposes:

- Acquiring Information: Digital twins need data from these stakeholders to gain a comprehensive understanding of the environment. For example, a digital twin for a shipment may require data from transport providers regarding routes, schedules, and capacities, or from authorities regarding regulatory constraints.
- Providing Information: Digital twins also act as rich data sources, offering insights into the current state, predictions about future states, or recommendations based on simulations and analysis. For instance, transport service providers can benefit from a digital twin's insights in optimizing routes or maintenance schedules.

Given the diverse interactions, standardizing data exchange is crucial. This is where the principles of data spaces come into play. By ensuring data sovereignty, secure data exchange, standardized interfaces, and clear usage rules, the challenges of data exchange in such a diverse ecosystem can be effectively addressed. It's not only about the smooth transfer of data but also about ensuring data trustworthiness, maintaining integrity, and providing clarity to all stakeholders about data usage. Incorporating data space principles into the digital twin ecosystem ensures a more robust, reliable,

and efficient interaction among all involved entities. It fosters trust and facilitates cooperation, essential for maximizing the potential benefits of digital twins.

Apart from the concept of dataspace, where data-sharing principles arise, the existence various protocols and principles that must be taken into account. The guidelines issued by initiative such as INSPIRE [22] which are specifically applicable to environmental geodata, enhance interoperability and increase data availability.

In addition, it's essential to align with CEF-supported projects like DTLF, Fenix, and FEDeRATED, which offer frameworks and ontologies for enhancing digitalization in transport and logistics. These collaborations facilitate interoperability and efficient data sharing across Europe's transport sector, laying a foundation for our developments and ensuring our solutions are compatible and supportive of a more connected, efficient, and sustainable logistics ecosystem.

Incorporating standards and frameworks from organizations like GS1, TIC4.0, and IPCSA is critical for the IWT digital twin project. GS1's global standards for business communication, including its widely recognized barcoding system, TIC4.0's efforts to establish digital transformation standards in terminal operations, and IPCSA's focus on port community system operators for enhanced efficiency and security, provide essential guidelines for interoperability and data sharing. Engaging with these entities supports the creation of a digital twin that is not only compatible with existing systems but also contributes to the broader goals of efficiency and security in logistics and port operations.

Data Integration for static, real-time and historic data

This paragraph outlines the essential functionality required for data access and connectivity resources, crucial for the seamless deployment and integration of DT technologies within the Living Labs. A virtual or digital environment is established to serve as a centralized repository and platform, collecting, integrating, and processing data from diverse sources associated with the use cases. The aggregated data becomes readily available for analysis, monitoring, and informed decision-making, facilitating various applications.

The dataspace for the digital twin comprises a complex mix of data from various sources, as highlighted earlier. The precision and reliability of this integrated data lay the foundation for an authentic digital representation of the real-world system. To achieve this, we categorize the integration into three distinct types:

- Scheduled Replication: Certain data sources are mirrored onto our platform based on predefined schedules. The ingested data, which can encompass a wide variety of parameters, is methodically stored within a time-series database. This approach ensures that the digital twin has consistent and periodic updates from these sources, allowing for tracking patterns and changes over time.
- Advanced Interface Retrieval: Some sources offer advanced interfaces, permitting on-demand access to both static and dynamic data. Whether it's real-time information or historical data,

the retrieval process is designed to align closely with the dataspace concept. This means data is fetched directly from the source as and when needed, ensuring the most up-to-date and relevant information feeds into the digital twin.

- On-Demand Data Requests: For data that isn't frequently used but remains crucial for specific analyses or scenarios, an on-demand retrieval system is in place. Rather than storing this data continuously, it's requested directly from the source when necessary. This approach ensures efficient use of storage and computational resources while guaranteeing access to vital data when required.

To manage this complexity of data streams, a comprehensive registry of all data, their sources, and associated metadata is maintained. This is encapsulated within a knowledge graph, functioning similar to a data catalog. This knowledge graph offers a structured overview of the available data and provides rich contextual insights, streamlining data discovery, understanding relationships, and enhancing the efficacy of the digital twin's operations.

Central to our data ingestion mechanism is the incorporation of the LDES (Linked Data Event Streams) specification. LDES provides a standardized approach to represent a series of time-ordered data events, particularly beneficial for systems that handle dynamic, real-time data. By adopting the LDES specification, we can efficiently capture, represent, and process changes or events in the data sources over time. In the context of our digital twin's dataspace, using LDES facilitates continuous synchronization with dynamic sources, ensuring that the digital twin always reflects the most recent state of the real-world system. Furthermore, with LDES, we achieve better compatibility with external systems and platforms that also adopt this specification, promoting interoperability and seamless data exchange.

NGSI-LD is a standard developed by ETSI (European Telecommunications Standards Institute) for context information management, which is particularly relevant for IoT (Internet of Things) applications. It enables systems to manage, update, and query context information, enhancing interoperability between various IoT systems and platforms.

GS1: A not-for-profit organization that develops and maintains global standards for business communication, best known for its barcoding standard used worldwide in supply chains.

TIC4.0: Stands for "Terminal Industry Committee 4.0". It's an initiative aimed at establishing a set of standards to facilitate digital transformation in container terminal operations, focusing on interoperability and data sharing.

IPCSA (International Port Community Systems Association): An association that represents the interests of Port Community System operators. It's focused on enhancing efficiency, security, and the electronic flow of information within ports and logistics sectors.

Ontologies developed in CEF projects like DTLF, Fenix, and FEDeRATED:

These refer to projects supported by the CEF (Connecting Europe Facility), which aims to boost energy, transport, and digital infrastructure across the EU.

DTLF (Digital Transport and Logistics Forum): An initiative by the European Commission to bring together stakeholders from different transport and logistics communities to foster digitalization in the field.

Fenix: A project aimed at developing a European federated architecture for data sharing in the logistics and transport sector.

FEDeRATED: This project seeks to create an interoperable ecosystem for logistics data sharing, improving efficiency and sustainability in the supply chain.

4 Operational Core of the Digital Twin's data pipeline and dataspace interface

Figure 3 presents the logical components constituting the operational core of the Digital Twin's data pipeline and dataspace interface. These components, implemented using state-of-the-art technologies, represent specific functionalities within the Dataspace framework. The following paragraphs offer descriptions of each component.

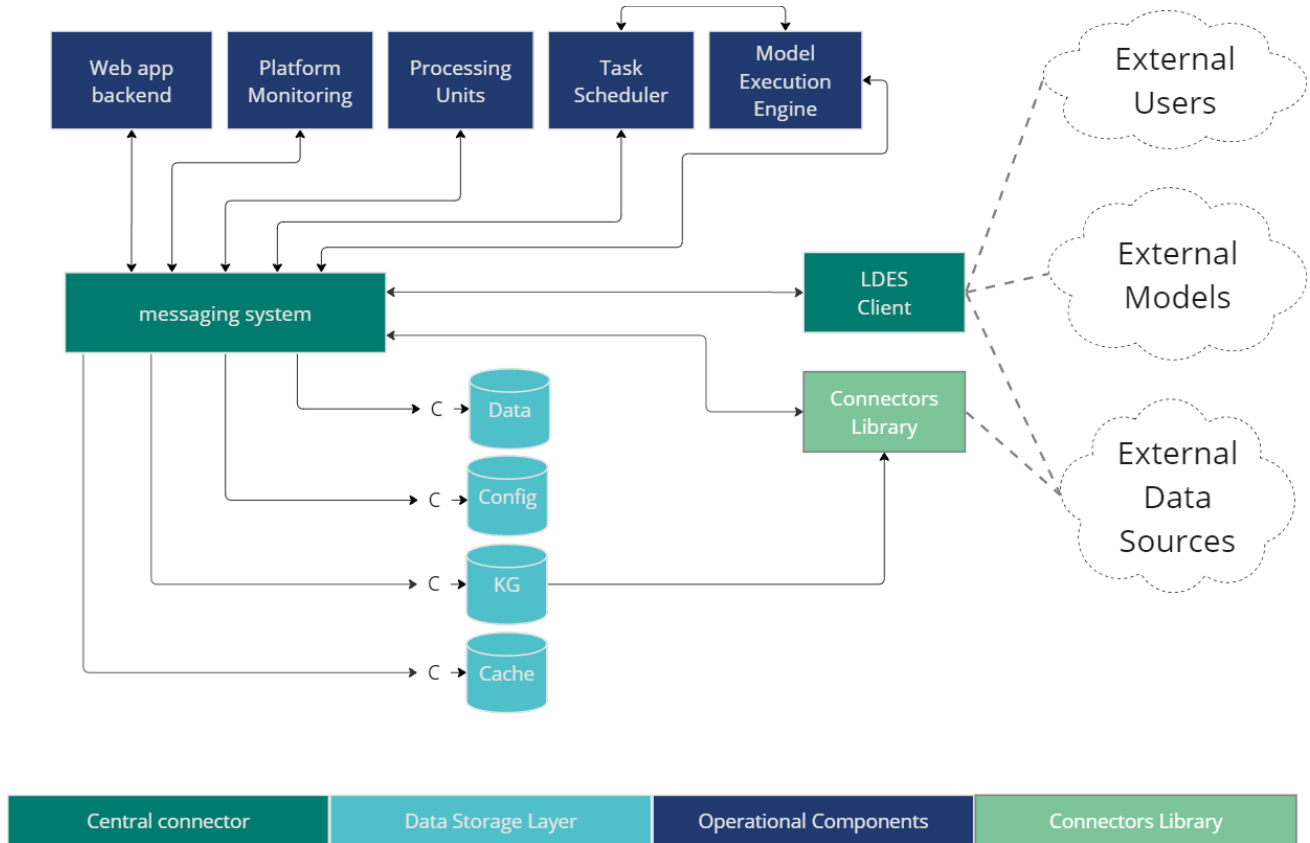


Figure 3 General Layout of the Operational Core of the Digital Twin for Dataspace Dynamics

4.1 The Central Connector

The Central Connector has the purpose of interconnecting the components of the system while ensuring security and transparency. Allows various components of the operational core to exchange data and information via message exchanges. The technology of choice for the implementation in the ReNew is Apache Kafka that implements a publish-subscribe (pub-sub) model.

The following paragraphs briefly describe the functionality of Apache Kafka, its adaptation for the Central Connector architecture, the potential it offers for additional integrations, and the selected components to complement Apache Kafka's operation.

Introduction to Apache Kafka as the Central Connector

Apache Kafka [6], a powerful distributed streaming platform, plays a pivotal role as the central connector within our data pipeline architecture. Renowned for its high performance, Kafka excels in handling vast volumes of data in real-time, making it an ideal backbone for complex data ecosystems.

Its design principles focus on fault tolerance, scalability, and durability, ensuring that data flows seamlessly and efficiently between disparate systems and applications.

As the central connector, Kafka orchestrates the movement of data throughout our architecture, acting as a unified platform that bridges the gap between data producers and consumers. By leveraging Kafka's robust publish-subscribe model, our system can accommodate diverse data sources, from logs and metrics to transactions and event streams, without compromising on throughput or latency.

The choice of Kafka is strategic, aligning with our objectives to build a resilient, scalable, and flexible data infrastructure. It supports our needs for real-time data processing and analytics, enabling us to make informed decisions swiftly. Kafka's widespread adoption and active community further assure us of its reliability and evolving capabilities, making it a cornerstone of our data pipeline architecture.

Central connector architecture overview

In our data pipeline architecture, Apache Kafka serves as the keypoint, centralizing the data flow between various components of the system. At its core, Kafka operates as a distributed messaging system, designed to facilitate high-throughput, low-latency data streaming.

Data within Kafka is organized into topics, which can be thought of as categories or channels for data streams. Each topic is further divided into partitions, allowing for data to be spread across multiple nodes in the Kafka cluster. This partitioning mechanism is crucial for achieving scalability, as it enables parallel data processing and consumption.

Producers are the entities that publish data to Kafka topics, while consumers subscribe to these topics to read the data. Kafka maintains a decoupled relationship between producers and consumers, allowing them to operate independently at their own pace and scale.

The Kafka cluster consists of one or more servers known as brokers. These brokers manage the storage and dissemination of data, ensuring that messages are retained and available for consumption. Brokers also handle tasks such as partitioning, replication, and consumer group coordination.

Kafka ensures data durability and system reliability through replication. Each partition can be replicated across multiple brokers, providing redundancy and protecting against data loss in the event of a node failure.

Consumers can be organized into groups to allow for distributed data processing. Kafka ensures that each partition is consumed by only one member of the group, enabling load balancing and parallel processing of data streams.

As shown in Figure 3, data flows into Kafka initially- the raw input- from the connectors of the Connectors Library and the LDES client. Once ingested, the data is stored in topics, ready to be

consumed by downstream systems. These systems are the data storage services and data processing applications. Kafka's architecture supports flexible and dynamic scaling, allowing the system to adapt to changing workloads by adding or removing brokers and adjusting partitioning as needed.

Overall, Kafka's architecture empowers our data pipeline to efficiently manage and process the vast streams of data that flow through our digital ecosystem, providing a robust foundation for real-time analytics and decision-making processes.

Potential expansions with Kafka integrations

A reason for selecting Kafka is the expandability that this component offers.

Kafka can integrate with stream processing engines like Apache Flink, Apache Storm, and Kafka's own Kafka Streams API. This allows for complex data processing and transformation in real time. These engines subscribe to Kafka topics, perform operations such as filtering, aggregation, and enrichment on the streaming data, and then publish the results back to Kafka or forward them to other systems. This integration supports a wide range of use cases, from simple data transformations to complex event-driven applications and real-time analytics, enhancing the overall capability of our data pipeline.

Data stored in Kafka can be directly accessed by analytics and business intelligence tools to generate insights and visualizations. This integration enables stakeholders to make data-driven decisions based on real-time information. Tools such as Apache Superset, Grafana, and Tableau can connect to Kafka either directly or through intermediate data stores populated by Kafka, providing a flexible approach to data analysis and reporting.

To ensure the health and performance of the Kafka ecosystem, it is integrated with monitoring and management tools such as Prometheus, Grafana, and Confluent Control Center. These tools provide visibility into Kafka's operations, including metrics on throughput, latency, and system health, facilitating proactive management and scaling of the data pipeline.

Kafka supports integration with security protocols and tools to ensure data privacy and compliance with regulatory requirements. Features such as SSL/TLS encryption, SASL authentication, and ACL-based authorization can be leveraged to secure data in transit and control access to sensitive information.

The integration of Kafka with other components in the data pipeline architecture, streamlines data flow and enhances the scalability, flexibility, and security of the entire data ecosystem, supporting a wide array of data-driven applications and analytics.

Managing Kafka's Backbone

In the architecture of Apache Kafka, ZooKeeper acts as the central coordinator, ensuring the stability and scalability of the Kafka ecosystem. ZooKeeper is responsible for managing and

maintaining Kafka brokers, defining partitions, and keeping track of the status of nodes in the Kafka cluster. It acts as the backbone of Kafka, providing a distributed configuration service, synchronization service, and naming registry for the Kafka cluster. This ensures that Kafka's distributed components remain synchronized, making the system resilient and fault-tolerant.

ZooKeeper's ability to maintain a consistent state across the Kafka cluster is vital for managing configurations, such as topic names and partition counts, which are essential for the smooth operation and scalability of Kafka. It enables Kafka to effectively manage cluster metadata and coordinate distributed processes, making it an indispensable component in the Kafka architecture.

On the other hand, Kafdrop serves as an intuitive web UI that simplifies the monitoring and management of Kafka clusters. It provides a clear and concise overview of Kafka topics, partitions, consumer groups, and messages. With Kafdrop, administrative users can easily navigate through their Kafka infrastructure, inspect topic configurations, and even view messages within topics, all through a user-friendly interface. This accessibility makes Kafdrop an invaluable tool for developers and administrators alike, offering insights into the operational aspects of Kafka clusters without the need for deep command-line expertise.

Kafdrop's integration with Kafka complements ZooKeeper's backend management by offering a front-end perspective that enhances visibility into Kafka's data flow and system health. It allows for real-time monitoring of Kafka's performance, making it easier to identify bottlenecks or issues within the cluster.

Together, ZooKeeper and Kafdrop form a comprehensive management suite for Apache Kafka, combining the robust backend coordination and configuration management provided by ZooKeeper with the accessible, real-time monitoring capabilities of Kafdrop. This synergy ensures that Kafka clusters are well-coordinated and resilient but also transparent and manageable, facilitating efficient streaming data operations. For these purposes, both components are included in our stack.

4.2 Data storage layer

As results from the discussion about the data requirements, in section -457079280.426, the ReNEW ecosystem is composed of two different kind of data to be shared and processed between its internal software applications and components. In addition, storage must also be considered for platform internal data such as configuration, cache, schema, etc. In this paragraph are described the data storage components that have been included in our stack, the technologies selection and the context for each of them.

This data, upon its ingestion in the platform it is stored into a database for future retrieval. Due to the different nature for the static and dynamic data, two different discrete storage mechanisms are foreseen described in the following two sections.

Timeseries database for dynamic data

As discussed in section 2.3.1, we are considering two types of data incoming to the platform: dynamic and static. In this section, we focus on the storage of dynamic data, which comprises metrics that vary over time. This category encompasses data for which variations occur at different frequencies. For example:

- The status of a bridge may change once per day, weather forecasts, depending on model runs, are typically valid for windows of three hours.
- Real-time vessel information, such as speed and other sensor-based data, may be available on a minute-by-minute basis.

Among the available solutions, Time-Series Databases (TSDBs), NoSQL Databases, and Relational Databases stand out for their unique capabilities in handling dynamic data sets.

Time-Series Databases (TSDBs) like InfluxDB, Prometheus, and TimescaleDB are tailor-made for time-stamped or time-series data. These databases are optimized for storing sequences of data points indexed in time order, making them ideal for metrics that vary over time [7] . TSDBs excel in fast data ingestion and efficient queries across time ranges, offering functionalities such as automatic data expiration, downsampling, and aggregation. This makes them particularly well-suited for applications involving monitoring, IoT device data, and any scenario where tracking changes over time is critical.

NoSQL Databases, including MongoDB, Apache Cassandra, and Redis, offer a flexible approach to handling unstructured or semi-structured dynamic data. With their schema-less nature, these databases can accommodate a wide variety of data formats and structures, which is beneficial for dynamic data that doesn't fit neatly into traditional table schemas. NoSQL databases are known for their scalability, both horizontally and vertically, allowing them to manage large volumes of data with varying access patterns efficiently. Depending on the specific NoSQL database, they can offer

different data models such as key-value, document, wide-column, or graph, each catering to particular types of dynamic data and use cases.

Relational Databases, such as PostgreSQL, MySQL, and Microsoft SQL Server, have been the backbone of data storage for decades and can also be adapted to store dynamic data. While not as inherently flexible as NoSQL databases for unstructured data, relational databases provide strong ACID transaction guarantees, complex query capabilities, and a well-understood data modeling paradigm. Extensions like TimescaleDB for PostgreSQL enhance traditional relational databases with time-series capabilities, combining the best of both worlds: the robustness and query power of SQL with optimizations for time-series data. This makes relational databases a viable option for dynamic data when the need for complex transactions, joins, and precise data integrity is paramount.

Considering that we foresee a different storage mechanism for all the information not being referenced by a timestamp, the most appropriate solution for timestamped data is to use a time series database.

The initial development is based InfluxDB [8], which is known for its high write throughput and efficient storage, making it ideal for high-volume environments. It's well-suited for real-time analytics and applications in IoT, DevOps monitoring, and sensor data analysis. However, there are considerations to move to TimescaleDB, which is an extension of PostgreSQL, combining the reliability and functionality of a relational database with the scalability of a time-series database. This provides the best of both worlds: complex SQL queries, joins, and transactional integrity, alongside efficient time-series data management.

Knowledge graph for static data and structured metadata

A knowledge graph is used to store topological and infrastructural data, which is the main type of static data. As described below, along with the static information, eg the position of a lock or the existence of a vessel, in this section is also considered the latest status for the objects, while the full history is considered for in the previous section.

Initially, the consideration was to leverage a specialized database like PostGIS, known for its advanced geographical computation capabilities [11]. However, upon closer examination, it became evident that our project's requirements for geographical calculations were minimal, rendering the sophisticated features of a GIS (Geographical Information System) database underutilized.

On the contrary, the need to understand and navigate the complex relationships within infrastructural data emerged as a significant requirement. This aspect is particularly crucial for modeling the intricate interdependencies and interactions within the system's components. A graph database, with its inherent ability to represent and query relationships efficiently, emerged as a more suitable choice. Graph databases excel in delineating connections between entities, making them ideal for identifying and analyzing cascading effects within the infrastructure.

The adoption of a graph database would facilitate a deeper semantic understanding of the infrastructure's network, enhancing our ability to model, visualize, and analyze the system's dynamics. This approach aligns with the project's objectives to accurately represent the interconnected nature of the infrastructure components and to provide insightful analytics on their interrelations and potential impacts.

Furthermore, graph databases, like Neo4j, also offer capabilities for handling geographical calculations to a certain extent. Although not as specialized as dedicated GIS databases like PostGIS, Neo4j provides spatial functions that can support basic geographical queries and operations. This feature allows for the integration of location-based data within the graph model, enabling queries that consider spatial relationships and distances between nodes. This dual functionality of Neo4j enhances its utility in projects that require both complex relationship mapping and some level of geographical data processing, offering a versatile solution that accommodates a broader range of analytical needs.

The decision to adopt Neo4j for storing static data and structured metadata was significantly influenced by two main factors: the task owner's prior experience with the tool and the collaborative efforts of Work Package 1 (WP1) partners who are actively engaged with Neo4j. The task owner's familiarity with Neo4j ensures an efficient and knowledgeable application of the database, leveraging its capabilities to the fullest. Concurrently, the ongoing work and expertise of WP1 partners in utilizing Neo4j contribute valuable insights and proven methodologies, facilitating a synergistic approach to data management. This collective expertise and the collaborative environment underscore the rationale behind choosing Neo4j, aligning with the project's objectives for robust and effective data storage solutions.

In the Neo4j database, we currently store data pertaining to infrastructure points and vessels. For both types of objects are also stored the associated metadata. This includes critical details about each node's characteristics and functionalities. Additionally, we update the latest dynamic information such as the latest status of locks, when applicable, to reflect real-time operational conditions and live operational data for vessels, including their current position, course, draft, and speed, as obtained from the Automatic Identification System (AIS). This integration of both static and dynamic data within Neo4j facilitates a holistic view of maritime operations, enhancing situational awareness and enabling more informed decision-making processes. However, it is noted that only the latest information is storage while the whole history is stored in the timeseries database.

NoSQL database for configuration storage

In the architecture of our system, the choice of technology for configuration storage is pivotal, impacting not only the performance and reliability of the system but also its adaptability to changing requirements. Our decision-making process involved a comprehensive evaluation of several potential technologies, each assessed on a spectrum of critical criteria.

- Scalability: The ability to handle growing amounts of configuration data without degradation in performance.
- Performance: The speed at which the system can retrieve and store configuration data, directly influencing the responsiveness of our system.
- Flexibility: The capacity to accommodate diverse and evolving configuration data structures without requiring significant schema modifications.
- Ease of Use: The simplicity of setup, maintenance, and daily operations, which affects the operational overhead and resource allocation.
- Reliability and Availability: Ensuring high availability and resilience of the configuration storage, minimizing downtime and data loss.

Our evaluation included a variety of database management systems, from traditional relational databases to modern NoSQL solutions.

The technology of choice was the MongoDB, mainly due its schema-less nature, that allows for storing configuration data in a flexible, JSON-like format [10]. This flexibility is crucial for accommodating the varied and evolving structures of configuration data in our system. In addition, MongoDB offers robust performance capabilities, particularly for read-heavy operations common in configuration retrieval. Its horizontal scalability, achieved through sharding, ensures that our system has the potential to scale. With features like built-in replication and automatic failover, MongoDB provides the resilience needed to maintain high availability of our configuration data, a non-negotiable requirement required for system's reliability. MongoDB's compatibility with various programming languages and platforms, coupled with its intuitive query language, facilitates easy integration with our existing systems and simplifies development efforts.

Following the evaluation process, MongoDB emerged as a suitable option for our configuration storage requirements. Its combination of adaptability, efficiency, and dependability, along with considerable community backing, suggests it as a reliable technology that will support the steady functioning and development of our system. Choosing MongoDB is aimed at enhancing our present operations and establishing a solid base for anticipated growth and new developments.

Temporary data storage

Temporary data storage was initially considered for two purposes:

- Caching frequently accessed data reduces the need to repeatedly fetch it from slower, persistent databases or external data sources.
- Storing data required for preprocessing purposes until the calculations are completed.

In addition to these purposes, multiple additional needs were covered by the existence of this convenient layer.

In exploring options for cache storage, we considered various technologies, with a focus on performance, stability, and integration capabilities. Redis stood out as a fitting choice due to its efficient performance in fast-paced transaction environments and its capability to store data in-memory. Its versatility in supporting different data structures like strings, hashes, lists, and sets provides the flexibility our diverse caching needs demand. Additionally, Redis's reliability, supported by features such as data replication, persistence options, and atomic operations, ensures the safety and consistency of our cache data. The broad use and active community around Redis offer ample support and resources, making it an attractive option for our caching requirements. By choosing Redis, we expect to improve our system's response times and enhance the overall experience for users through quick data retrieval.

4.3 Operational Components

Web app backend

The "Web App Backend" microservice within the operational core of the digital twin facilitates communication with the frontend application. As of the time of this report, development of this component has not yet commenced. This component will be custom-made and could be developed using programming languages such as Python or Go.

This microservice should be specifically designed to cater to user interactions and frontend configurations, ensuring a seamless and responsive user experience. Its core functionalities include managing user authentication and authorization, personalizing user settings, and handling frontend-specific configurations.

This backend microservice acts as a gateway for the frontend, providing a structured way to access and manipulate data stored across the system's databases. It processes requests from the frontend, such as data queries or updates, and communicates with other microservices as needed to gather the requested information or perform the required actions. This design enables a clean separation of concerns, where the Web App Backend focuses on user-facing functionalities while other microservices handle more specialized tasks like data analysis or sensor data processing.

User management is a crucial aspect of this microservice, involving secure handling of user credentials, profiles, and access rights. It ensures that users can securely log in, manage their accounts, and have access only to the data and functionalities authorized for their role. This is vital for maintaining data security and privacy within the digital twin environment.

Additionally, the Web App Backend manages configurations for the frontend application, such as layout preferences, data visualization settings, and notification preferences. These configurations allow the frontend to be highly customizable, catering to the diverse needs of different users, from engineers and environmental scientists to policy-makers and the general public.

By focusing on these user-centric tasks, the Web App Backend microservice enables the frontend application to provide a tailored, efficient, and secure interface for interacting with the waterway digital twin, enhancing the overall user experience and effectiveness of the digital twin as a decision-support tool.

Platform monitoring

For the platform monitoring components of the waterway digital twin's information system, Grafana and Kibana can be employed to provide comprehensive visualization and analytics capabilities. Grafana can be utilized for its robust dashboarding and alerting features, allowing for the real-time monitoring of system metrics, logs, and alerts once the system is operational. It can enable the creation of highly customizable dashboards that can display data from various sources, making it an ideal tool for overseeing the health and performance of the microservices architecture and underlying infrastructure. On the other hand, Kibana, as part of the Elasticsearch stack, can be leveraged for its powerful log analytics and visualization capabilities. It can provide deep insights into the system's operational data by allowing for the exploration, analysis, and visualization of log data in a user-friendly manner. Together, Grafana and Kibana can form a potent combination for platform monitoring, offering a holistic view of the system's state. This can enable rapid identification and resolution of potential issues, ensuring high availability and reliability of the digital twin platform, and supporting proactive maintenance and optimization efforts.

Processing Units

The processing units within the operational core of the digital twin are a cornerstone of its data handling and analysis capabilities. These units consist of custom code, potentially written in languages such as Go or Python, designed to interface directly with the messaging system (the central connector, §4.1), to receive continuous data streams. Upon receiving data, these processing units engage in a variety of computational tasks—ranging from basic calculations on input data to more complex operations like outlier detection and pattern recognition. These capabilities are crucial for dynamically interpreting the vast amounts of data generated by the waterway ecosystem and for transforming raw data into actionable insights.

A key feature of these processing units is their ability to not only analyze data but also to publish processed results back onto Kafka. This creates a feedback loop that can be leveraged to trigger other tasks within the system, such as alerting mechanisms or the activation of specific response protocols in real-time, based on the processed data insights.

To enhance their performance and efficiency, the processing units are backed by a caching system, typically implemented using Redis, as detailed in the "Data Storage Layer" section of the architecture, §4.2. This cache serves as a high-speed data storage layer that temporarily holds frequently accessed data and intermediate processing results. By reducing the need to repeatedly access the primary data storage for the same information, the cache significantly speeds up data retrieval times for the processing units, thereby accelerating the overall data processing workflow.

This combination of custom processing code, integration with the messaging system, and backing by a high-performance cache like Redis, equips the processing units with the flexibility and power needed to support a wide range of computational tasks. These tasks are essential for the real-time monitoring, analysis, and management of the waterway digital twin, enabling the system to detect critical events, identify trends, and initiate actions based on sophisticated data-driven insights.

Task Scheduler

The task scheduler component within the operational core of the digital twin is essential for orchestrating various tasks that need to be executed on a schedule, in addition to those triggered by specific events. This includes critical operations such as running simulations and periodic data ingestion from external sources to keep the digital twin updated with the latest information. While these scheduling needs have been initially addressed within individual microservices, the complexity and interdependence of tasks across the system highlight the necessity for a centralized management point for more efficient and reliable task orchestration.

Airflow, an open-source workflow management platform, emerges as a robust solution for this central task scheduling requirement. Airflow allows for the scheduling, coordination, and management of complex workflows through directed acyclic graphs (DAGs) of tasks. It provides a flexible and dynamic platform for defining tasks and their dependencies in Python, offering a programmable environment that can easily integrate with the existing microservices architecture.

With Airflow, tasks such as running simulations or fetching data from external sources can be defined as part of a workflow, with the ability to set precise timing and frequency for their execution. Airflow's rich user interface provides visibility into the scheduling and execution of tasks, enabling monitoring and troubleshooting workflows with ease. Additionally, Airflow's extensive plugin system allows for easy integration with a variety of external systems and services, ensuring that the task scheduler can adapt to the evolving needs of the digital twin platform.

Implementing Airflow as the centralized task scheduler offers a scalable and maintainable approach to managing scheduled tasks, ensuring that critical operations are performed reliably and efficiently. This centralization also facilitates better resource management, error handling, and oversight of all scheduled activities, contributing to the overall robustness and resilience of the waterway digital twin's information system.

Model Execution Engine

The Model Execution Engine is a crucial component within the waterway digital twin's ecosystem, designed to run simulations that predict waterway and vessel conditions in response to various environmental events and operational scenarios. A pertinent example, driven by the requirements from the living labs, is the execution of hydrological models that forecast water levels as influenced by weather conditions and the operation of hydroelectric plants. While the detailed development and management of the Model Execution Engine itself lie beyond the scope of this report, its integration within the digital twin architecture is vital for the system's functionality.

From the perspective of developing dataspace tools and components, our focus is on establishing a robust data interface for the Model Execution Engine. This interface will be designed in collaboration with the providers of the models to ensure seamless data exchange, allowing for the input of necessary environmental and operational data into the models and the retrieval of simulation outputs. The design of this interface will prioritize flexibility, scalability, and interoperability to accommodate a wide range of models and simulation requirements.

The data interface will facilitate the feeding of real-time and historical data into the models, enabling them to simulate current conditions as well as forecast future scenarios. This includes data on water quality, flow rates, weather conditions, vessel traffic, and any other relevant parameters that models require for accurate simulations. The output from these models, in turn, will be made available through the data interface to other components of the digital twin, supporting decision-making processes, predictive analytics, and strategic planning.

The collaboration with model providers will ensure that the data interface is tailored to the specific needs of each model, taking into account the data formats, frequency of updates, and computational requirements. This cooperative approach will ensure that the Model Execution Engine can be effectively integrated into the broader digital twin architecture, enhancing the system's ability to simulate and respond to the dynamic conditions of waterways and vessel operations.

4.4 Connectors Library

The Connectors Library is a vital component of the digital twin's ecosystem, enabling seamless data integration from diverse external sources. Implemented as microservices, these connectors facilitate the acquisition and ingestion of crucial data into the platform, aligning with the specific use cases that drive the digital twin's functionality. The initial set of connectors, detailed in Chapter 4, showcases the variety of data sources tapped into, ranging from meteorological services to vessel tracking systems. Each connector is designed to fetch data, ensuring smooth data flow into the processing and storage layers. This modular and scalable approach allows the digital twin to efficiently adapt to new data needs, maintaining its effectiveness and relevance over time.

5 Datasources

5.1 Water level sensors

Existing sensor technologies within the consortium are used to provide an IoT component for monitoring and optimization of IWT resilience as part of the Digital Twin capabilities and support Living Lab 1. Therefore, in the context of LL1, sensors will be installed for environmental data gathering, upgrading the physical infrastructure and improving Ghent's synchromodality. The new sensors will provide a support 'package' for time of crisis (e.g. low/high water levels) increasing the overall infrastructure's resilience.

Specifically, there will be installed one piezoresistive pressure transmitter sensor for water level monitoring in 'Everstein' location, close to the main part of the Ghent's port (51.1041 N, 3.7250 E), shown in Figure 6. In Figure 4 this location is depicted with the number 1. Also, in the second point at the zelzate marina in the cross border with the Netherlands, a sonde consists of four different water quality sensors and one piezoresistive pressure transmitter sensor will be installed (51.2085 N, 3.8053 E).

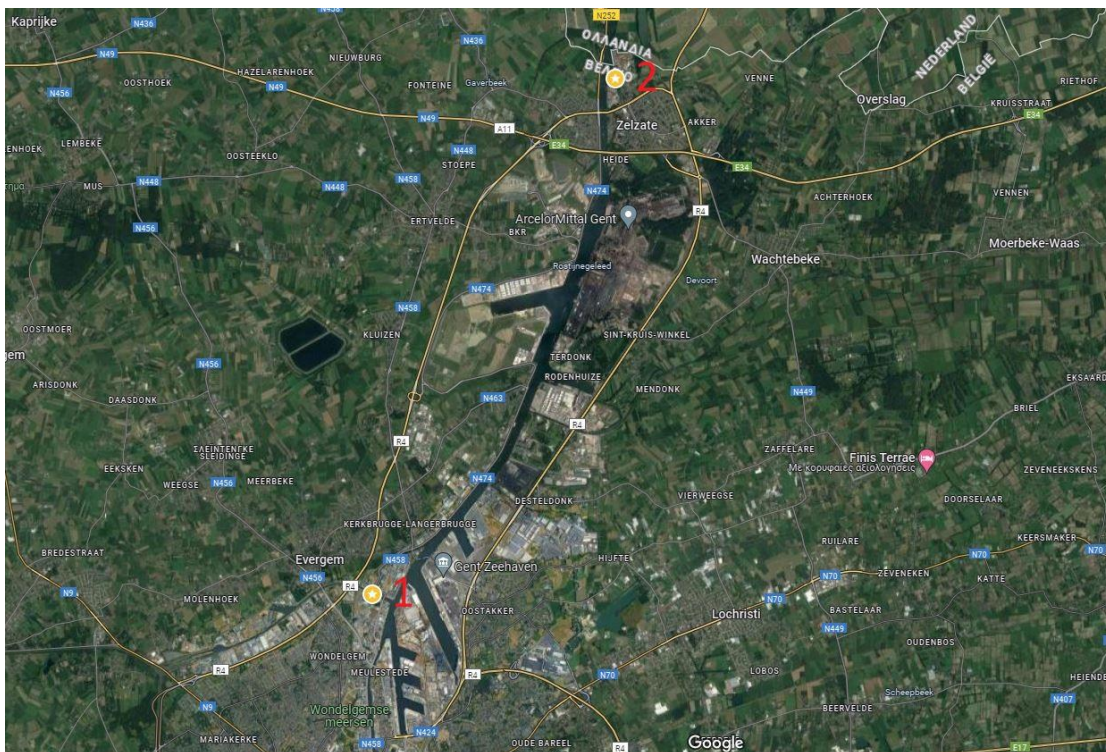


Figure 4 The two locations for the installation of water level and water quality sensors.

At the zelzate marina, a sensor box is already installed in the framework of Socorro European 2 Seas project (Interreg 2 Seas - SOCORRO – Ghent University [13]), as illustrated in Figure 5 Figure 4. Thus,

we would use their electricity connection without need installation of new solar panel. The sonde of sensors will be installed at a pole of approximately 3 meters height, which will be close to the marina and inside the water. The Aqua TROLL 500 is a five-port multiparameter sonde, including four sensor ports and a wiper port. The four water quality sensors which will be installed are: Temperature/Conductivity, Turbidity, Chlorophyll-a, and Dissolved Oxygen contributing to the environmental and ecological impact monitoring of inland transportation in the water ways of Ghent and further. The Aqua TROLL 500 is a fully customizable multiparameter sonde with interchangeable sensor that deliver accurate data and enable simplified calibration, panoramic data and report creation. This flexible instrument is ideal for spot checking and continuous remote monitoring when used with data services. Thus, this state-of-the-art sonde simplify data collection with equipment designed to be reliable, cost effective and easy to use. Also, this in-situ equipment is designed to withstand use in the harshest environments. Features designed to prevent breakage or failure including: interlocking sensors for greater stability, titanium restrictor, fully potted sensors, redundant SD card storage, and multi-chamber design. Moreover, it consists of a mini calibration cup that uses only 50 ml of solution for calibration, reducing calibration cost by 5x over traditional methods. There are three different output options: RS485/MODBUS, SDI-12, and Bluetooth, while its maximum reading rate is 1 reading every 2 seconds. For the needs of the project, the output option of RS-485/MODBUS is going to be used. It operates in temperatures between -5° and 50° C, while its weight is around 1 kg. Furthermore, based on the technical documentation, this sonde is able to provide temperature measurements with accuracy $\pm 0.1^\circ \text{C}$, dissolved oxygen with $\pm 0.1 \text{ mg/L}$, turbidity with $\pm 2\%$ of reading, and Chlorophyll-a with $\pm 0.1 \mu\text{g/L}$.



Figure 5 The sensor box installed in zelzate marina in the framework of Socorro project.

Also, the piezoresistive pressure transmitter PR-36XW KELLER, which will be used for the water level monitoring, is a state-of-the-art sensor providing highly accurate measurements. This sensor has a digital interface RS485, which supports the MODBUS-RTU communication protocol for process values and configuration. It's a high-quality pressure transducer and tried-and-tested mathematical compensation with robust stainless-steel housing. The accuracy is $\pm 0,05$ %FS, while the maximum deviation within the temperature range (0...50 °C) is $\pm 0,1$ %FS.

In both locations, a HD33M-MB.2 Delta Ohm datalogger equipped with a GSM/GPRS module will be used allowing several physical quantities to be monitored remotely. In the framework ReNEW project, the version with input for standard RS485 MODBUS-RTU sensors will be used. Also, thanks to GSM/GPRS transmission, we will not have to remove the datalogger from its position or visit the two locations, where the dataloggers are going to be installed, to download the data measured with the PC. The datalogger stores the measurements in the internal memory and transmits the acquired data via FTP, via e-mail or to an HTTP server (Cloud). Also, we could make a direct GPRS TCP/IP connection with a remote PC. The datalogger functions can be remotely controlled by sending SMS messages. For each detected quantity, we can set two alarm thresholds (high and low threshold), the alarm hysteresis and a delay in the generation of the alarm. The overrun of the thresholds can be signalled by alarm emails or SMS messages. The datalogger operates with 7 up to 30 Vdc power supply voltage. However, in 'Everstein' location, the datalogger will be powered by a solar panel of 10 W, which are going to be installed in the pole, as it is shown in the following picture (Figure 6). The datalogger is equipped with a switched power supply to output the sensors only when the measurement has to be carried out. It operates between -20° and +70 °C with LCD, and between -40° and +70 °C without LCD, while its weight is approximately 1 kg. Its dimensions are: 122 x 120 x 56 mm excluding the antenna.

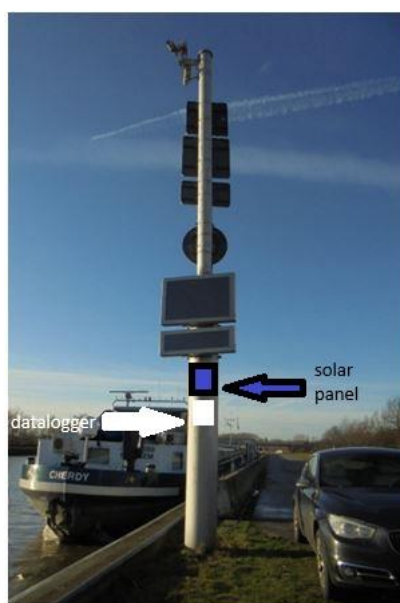


Figure 6 Pole in 'Everstein' location, where the datalogger and the solar panel will be installed.

Water level measurements will increase the infrastructure's resilience of Ghent, but also will be used for the evaluation/validation of space-based altimetry observations which is a challenging task due to the limited access to historical gauge data. Thus, in-situ network will leverage state-of-the-art sensing technologies to enable accurate water level monitoring with high temporal resolution whilst also facilitating remote data transfer.

5.2 EuRIS

The European River Information System (Euris) is a key data source in the context of waterway management and navigation in Europe and thus it is an invaluable resource for the purposes of the waterway digital twin. In the following paragraphs are described the contents of EuRIS and the integration methods.

Introduction to River Information Systems and EuRIS

River Information Services (RIS) offer standardized information services for inland navigation, enhancing safety and efficiency [15]. They facilitate the exchange of traffic and transport-related data among stakeholders. Services like Notices to Skippers (NtS) and Electronic Reporting (ERI) are already in use, with the EuRIS platform advancing RIS capabilities.

RIS aims to improve safety, efficiency, and environmental protection in inland navigation, supported by a legal framework since 2005. Policy-wise, RIS is recognized in various EU documents, including the EC White Paper and the Digital Inland Waterways Activity (DINA) [16] initiative.

According to the RIS guidelines, the following services are defined [19]:

- Fairway information for voyage planning, execution, and monitoring, encompassing geographical, hydrometeorological, and traffic-related data.
- Traffic information services providing vessel positions for tactical or strategic planning.
- Traffic management to optimize infrastructure use and facilitate safe navigation, particularly at VTS Centers, locks, and bridges.
- Calamity abatement services for registering vessels and their transport data, aiding emergency response teams in case of accidents.
- Information for transport management including estimated times of arrival (ETA's) and cargo and fleet management data.
- Statistics and customs services to improve data collection and facilitate waterway charges and port dues calculation and invoicing.

EuRIS [14], a collaborative effort involving 13 European countries, consolidates national information and services into a single accessible platform. This initiative stems from the RIS COMEX project (2016-2021), supported by CEF funding, which aimed to establish Corridor RIS Services. EuRIS marks

a significant milestone in facilitating the operational exchange of RIS data across Europe. Its launch enables traffic management by authorities and enhances transport management for the logistics sector.

EuRIS presents all waterway and traffic related information maps or in tables, providing the following info:

- a real-time traffic image
- position information of authorized vessels
- Notices to Skippers
- Actual water levels, discharges, bridge clearance, water depth
- information about the fairways, bridges, locks, berths, terminals
- the operating times of locks and bridges
- a Route and Voyage Planner
- travel duration and expected arrival times

EuRIS reference network model

In the EuRIS Reference network model, each waterway is represented by one or more fairway sections. A fairway section is a (part of) a waterway where none of the waterway's important features change. Such features include:

- Maximum allowed ship dimensions (length, width, draught)
- Maximum allowed sailing speed
- Maximum allowed CEMT class
- RIS authority
- Waterway name

Moreover, a new fairway section must be started at each intersection of waterways. Two fairway sections of the river 'Leie' near Kortrijk (BE) are depicted in Figure 7. The yellow section has a maximum allowed ship dimensions of 110m x 11.5m x 2.8m, while the red section has a maximum allowed ship dimensions of 86m x 9.6m x 2.5m.

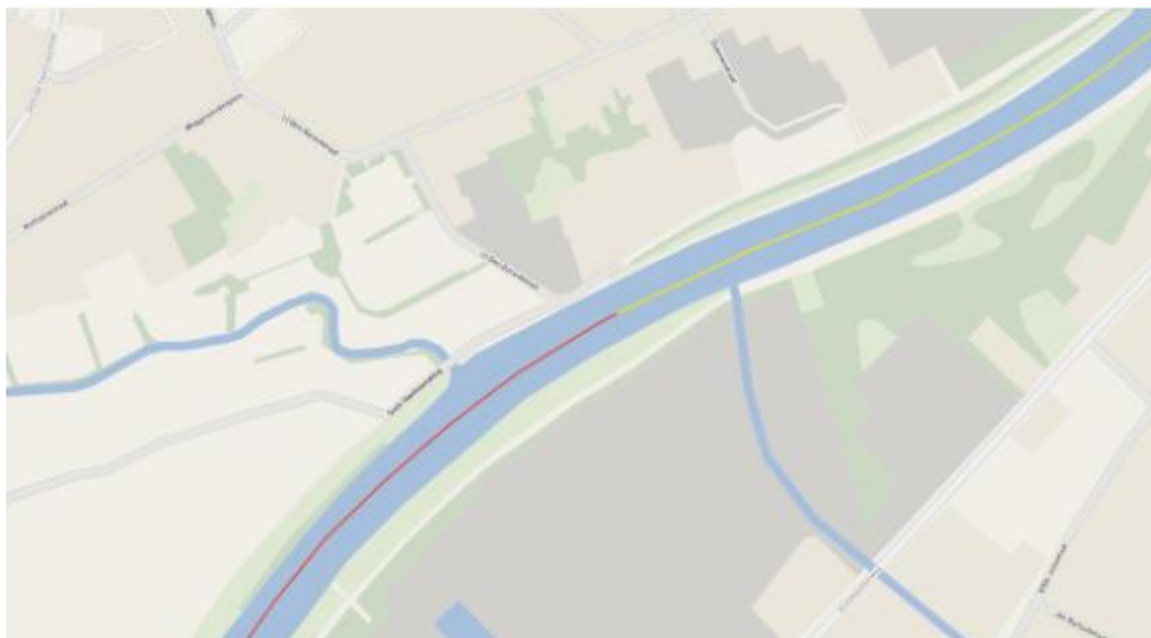


Figure 7 EuRIS fairway sections definition [17]

Each fairway section is uniquely identified by:

- The country code (CNTRYCODE)
- The fairway code (FW_CODE) conform the RIS Index fairway section code
- A sequence number (SEQ_NR)

The combination of the above fields results in the SECTIONREF code, which is used to link nodes and other objects to fairway section. The usage of a sequence number for each fairway section allows all countries to split their existing RIS Index fairway sections into smaller fairway sections without having to update their entire RIS Index.

The **RIS Index** is a flat table of important RIS related locations. The most important attributes are:

- ISRS code: this is the primary key and used in many RIS related messages to identify an object.
- Fairwaysectioncode: internal code of a part of the Fairway
- Objectcode: internal code to identify an object
- Hectometer: hectometer code along the Fairway axis
- Objectname: name of the object
- Function: type of the object

The RIS index is used to lookup locations for route and voyage calculation.

EuRIS web services

The EuRIS provides a wide array of services and data, many of which are accessible through web services, including Open APIs. This accessibility ensures that users can easily integrate EuRIS data into their own systems and applications. The range of data available through these web services encompasses:

- Reference Data: This includes comprehensive datasets such as the RIS Index and detailed information on the waterway network, providing essential baseline information for navigation and planning.
- Fairway Objects: Data on critical infrastructure elements like locks, bridges, berths, and terminals are made available, offering insights into the physical components of the waterways that affect navigation and logistics.
- Dynamic Fairway Information: Real-time updates on water levels, bridge clearances, and water discharge rates among other dynamic factors, ensure that navigators have the latest information to make informed decisions during their voyages.
- Dynamic Traffic Information: Information on vessel positions and traffic density helps in understanding current traffic conditions, facilitating smoother navigation and congestion management.
- Transport Information: Tools and data for route planning, voyage information, and other transport-related services support efficient and effective voyage management.

Authentication

APIs which return privacy sensitive data are protected. Authentication is required to access these APIs. EuRIS uses OAuth 2.0 Authorization Code Flow (with PKCE) and Client Credentials Flow. Access to the endpoints which can return privacy sensitive data is limited.

ODATA

Most endpoints support OData v4 (Open Data Protocol version 4), which is a web protocol for building and consuming RESTful APIs that expose and consume data over the internet. It provides a standardized way to create, read, update, and delete (CRUD) operations on data, making it easier to develop interoperable and scalable APIs. Sorting, filtering and paging functionality is supported.

Geographic Service

Euris offers web services that return data based on geographic queries, eg area define by coordinates. The available endpoints, differentiated by returned object type, such as Notice to Skippers, bridges, etc, are listed at [18]. Geographic service support timeaware features.

Hydrometeo services

The hydrometeo services provide information about water levels, volumetric flows and velocity. A sample returned payload is shown in Table 2.

Table 2 Sample returned payload by the hydrometeo web service.

```
{
  "value": 343.0,
  "hydrometeoProvider": "NtS WRM - BG - Bulgarian Ports Infrastructure C",
  "hydrometeoProviderId": "fbdec042-4f91-490d-8979-fd268a46b2c9",
  "measuredAt": "2024-02-17T06:00:55+01:00",
  "receivedAt": "2024-02-17T07:09:40.1227097+01:00",
  "dataReferenceLevel": "ZPG",
  "dataUnit": "cm",
  "linkID": "f1b977f1-34e1-4cb3-824c-08f96509701e",
  "linkName": "Water level @ gauge Oryahovo ",
  "linkDataProviderCode": "WAL @ BGOKH00001G000606780",
  "dataStatus": 0,
  "alarmStatus": 0,
  "visibility": 3,
  "hideWebsiteExternal": False,
  "originator": "ИАППД",
  "locationID": "198bce2b-3511-430a-82be-149b57fc01a1",
  "locationName": "gauge Oryahovo ",
  "locationCode": "BGOKH00001G000606780",
  "locationDataProviderCode": "BGOKH00001G000606780",
  "lat": 43.74105,
  "lon": 23.9629716666667,
  "dataProvider": "NtS WRM - BG - Bulgarian Ports Infrastructure Company",
  "dataOwner": None,
  "parameterID": "b9441f12-d5f8-4db0-b1f7-f060e40ffb15",
  "parameterName": "Water level",
  "parameterCode": "WAL",
  "parameterDataProviderCode": "WAL",
  "unit": None,
  "referenceLevel": None,
  "refreshInterval": None,
  "definedParameterID": "f4425786-fb63-4ef6-94da-dddcd8a456aa",
  "definedParameterName": "Water level",
  "definedParameterCode": "WAL",
  "visibilityRadius": 1
}
```

Inland Electronic Navigational Chart Service

An IENC (Inland Electronic Navigational Chart) Map is a digital navigational chart specifically designed for the requirements of inland waterway navigation. IENCs provide detailed and standardized information about inland waterways and related infrastructure, such as rivers, canals, locks, bridges, and navigational aids, to support the safe and efficient navigation of vessels in inland waters.

These digital charts are developed in accordance with international standards set by organizations like the International Hydrographic Organization (IHO) and the Inland ENC Harmonization Group (IEHG) to ensure consistency, reliability, and interoperability across different regions and navigation systems. IENCs are used in conjunction with electronic chart display and information systems (ECDIS) or other compatible navigation systems onboard vessels.

Preview	Map	Description	(Sub)Area	Modified
	2P7D1114			08/04/2023 7:35
	2P7D1408			08/04/2023 7:35
	1H7TI480			12/10/2021 15:55
	1H7TI380			12/10/2021 15:54

Figure 8 EuRIS interface to manually locate and download electronic maps

EuRIS provides APIs to list and download IENC in its binary format, zipped, shown in Figure 8.

Notifications to shippers

EuRIS provides APIs to get Notifications to shippers, filtered by dateStart, dateEnd, dateIssue, messageType, countryCode, fairways, limitationCodes, originator, number, organisation, internalId, targetAudiences. A sample Notification to shippers retrieved via EuRIS API is provided in Table 3.

Table 3 Sample Notification to shippers retrieved via EuRIS API

```
{
  "headerID": "3e28be93-9f49-461a-b724-3e56a59381c9",
  "sectionID": "f8dc05fd-51a5-42e9-b8f3-116ee75d0a03",
  "internalId": null,
  "organisation": "Rijkswaterstaat",
  "dateIssue": "2024-02-17T06:42:20.0659968+01:00",
  "dateStart": "2024-02-16T00:00:00+01:00",
  "dateEnd": "2024-02-16T00:00:00+01:00",
  "state": 80,
  "messageType": 1,
  "title": "blockage - Vaarweg vanaf het IJmeer via de Randmeren naar het
            Ketelmeer, Nijkerkersluis (Nijkerk)",
  "number": "2024/989/1",
  "countryCode": "NL",
  "country": "Netherlands",
  "relatedFairways": [
    "Nijkerkernauw"
  ],
  "targetaudiences": [
    "___|empty|___"
  ],
  "modificationTimeInVisuRIS": "2024-02-17T05:54:40.806946+01:00",
  "multilanguageTitles": "{\"en\":\"blockage - Vaarweg vanaf het IJmeer via de
            Randmeren naar het Ketelmeer, Nijkerkersluis (Nijkerk)\"}",
  "multilanguageTooltips": "{\"en\":\"blockage\"}",
  "limitations": ["OBSTRU"],
  "hindranceState": 0,
  "stateMessage": "Published",
  "messageTypeMessage": "fairway message",
  "hindranceStateMessage": "Undetermined",
  "year": "2024",
  "versionId": "ea790446-b385-455a-a380-5e4841be4cb7",
  "originator": "Waterkamer",
  "missingData": false
}
```

EuRIS network data

The EuRIS network data contains spatial and important attributes for the following types of objects:

- Areas of Competence: primary used internally by the EuRIS partners (geofencing)

- Berths: places where the vessels can moor. Sample object is shown in Table 4.
- Bridges: information concerning bridges and their openings. Sample object is shown in Table 6.
- Fairwaysection: part of the fairway with the same sailing characteristics. The line projection must follow the water flow direction from source to mouth.
- Locks: information concerning lock complexes and lock basins. Barrages are also included.
- Nodes: used to connect the fairwaysections
- Radio calling points: Also called radio reporting points, they have been established in certain busy waterways and port approaches to assist traffic control. On passing these points or crossing a defined line vessels are required to report on VHF to a Traffic Control Centre.
- Sailingspeed: optional table with additional sailing speed rules depending on draught, width, length and the category of the ship.
- Terminals: A terminal covers that area on shore that provides buildings and constructions for the transfer of cargo or passengers from and to ships. Sample object is shown in Table 5.

Table 4 Sample 'berth' object

```
{
  "locode": "BEAAB11817L005800522",
  "termRef": "BEAAB11817T019000522",
  "sectionRef": "BE1181700000",
  "objectname": "Ligplaats Aalst LO kmp 52.2",
  "originalObjectname": "Ligplaats Aalst LO kmp 52.2",
  "nationalObjectname": "Ligplaats Aalst LO kmp 52.2",
  "hectom": "00522",
  "waterwayName": "Dender",
  "bank": "LB",
  "address": "Overkant: Frits De Wolfkaai, 9300, Aalst",
  "adnMode": 0,
  "berthCategory": null,
  "shipCategory": "30000",
  "reservable": false,
  "myBerth": false,
  "weborganisationId": null,
  "privacyClass": null,
  "occupancy": null,
  "bankMessage": "Left bank",
  "adnModeMessage": "Disallowed",
  "shipCategoryMessages": ["Inland vessel"],
  "berthCategoryMessages": null,
  "reservableMessage": "No",
  "myBerthMessage": "No",
  "occupancyMessage": null,
  "linkId": null,
  "defaultPrivacyClass": 0,
  "owned": false}
```

Table 5 Sample 'Terminal Call Response' object

```
{
  "objectid": 0,
  "objectname": "Kaai De Witte te Herdersem",
  "locode": "BEAAB11818T013100570",
```

```
{
  "loname": "Aalst",
  "wW_NAME": "Dender",
  "hectom": "00570",
  "status": "8",
  "fnction": "trm11",
  "load_TYPES": null,
  "comname": null,
  "hasgeometry": true,
  "statusMessage": "Private",
  "load_TYPESMessage": "",
  "risFunctionMessage": "Bulk Terminal",
  "originalObjectName": "Kaai De Witte te Herdersem",
  "linkID": null,
  "defaultPrivacyClass": 0,
  "privacyClass": 2,
  "owned": false
}
```

Table 6 Sample bridge object

<pre>{ "isrs": "ATASC00001BRGA121599", "isrsCB": null, "name": "Straßenbrücke Aschach", "nationalObjectName": "Straßenbrücke Aschach", "nameCB": null, "waterwayName": "Danube", "waterwayNameCB": null, "fairwayCode": "AT0000100004", "wwAuthorit": "Schifffahrtssaufsicht Linz", "wwAuthoritCB": null, "hectom": 21599, "hectomCB": 0, "route": "Donau", "routeCB": null, "location": "Aschach", "locationCB": null, "remark": null, "comCha": null, "comName": null, "comNameCB": null, "centroper": null, "refGauge": "ATWHE00001G000221440", "refGaugeCB": null, "tideDependent": 0, "heightRef": null, "heightRefCB": null, "referenceLevel": null, "referenceLevelCB": null,</pre>	<pre> "referenceLevelName": "HDC", "referenceLevelNameCB": null, "vpInpoint": "0", "staticHeights": [8.59], "dynamicHeights": [], "operable": false, "warning": false, "maxStaticHeight": 8.59, "maxDynamicHeight": 0, "country": null, "countryCB": null, "centroperMessage": null, "tideDependentMessage": null, "heightRefMessage": null, "heightRefCBMessage": null, "referenceLevelNameMessage": null, "referenceLevelNameCBMessage": null, "linkedWaterLevelName": null, "linkedWaterLevelNameCB": null, "linkedWaterLevelId": "00000000-0000-0000-0000-000000000000", "linkedWaterLevelIdCB": "00000000-0000-0000-0000-000000000000", "hasGeometry": true, "nameFolded": "Strassenbruecke Aschach", "nationalObjectNameFolded": "Strassenbruecke Aschach", "countryCode": "AT", "countryCodeCB": null }</pre>
--	--

5.3 Open Street Map (OSM)

OpenStreetMap (OSM) stands as an invaluable geospatial data resource, playing a pivotal role in open access to detailed and up-to-date information about our surroundings. It serves as more than just a map. It is a dynamic and collaborative platform where a global community of contributors continuously updates and enriches the spatial data. This living atlas encapsulates a wide spectrum of information, including road networks, land use patterns, points of interest, and natural features.

The collaborative nature of OSM ensures that the data is frequently updated, making it a reliable and contemporary source for a variety of geospatial applications.

The accessibility and openness of OSM make it a valuable asset for researchers, developers, and organizations seeking accurate and detailed geospatial information. By tapping into OSM's diverse dataset and leveraging tools like Overpass Turbo, our system gains the ability to extract and utilize specific subsets of this vast repository, contributing to a more nuanced understanding of our ever-evolving geographic landscape.

Object Types in OpenStreetMap (OSM)

OpenStreetMap (OSM) serves as a digital canvas capturing the intricate details of our physical world through a diverse array of object types. At the core of OSM lies a rich tapestry of catalogued features, each serving as a digital representation of various geographical elements. The underlying data schema of OSM has been the basis of open geospatial protocols aiming to harmonize various data sources and build application on top of them[20]. The term "OSM type" is central to understanding these object types, representing the classification assigned to geographic objects or features within the map dataset. This section describes these object types, highlighting their role in encapsulating the intricacies of human-made structures and natural formations.

1.1..1 Definition of OSM Type

In OpenStreetMap, an "OSM type" is the term used to describe the type of geographic object or feature in a map dataset. It identifies the basic shape or object, such as a point, line, or polygon, and allows differentiation between features like buildings, streets, parks, and more. OSM types are typically represented by codes in the dataset, like "N" for node points.

Examples of OSM types include:

- Node: A single point in space, representing a standalone feature such as a building, church, or restaurant. Nodes are identified by unique codes in the dataset.
- Way: An ordered list of node points that collectively form a linear feature, such as roads, footpaths, or railway lines. Ways provide a structured representation of these linear elements.
- Relation: An ordered list of ways, nodes, and other relations, facilitating the grouping of multiple objects. Relations are used for representing complex entities like bus routes, postal code boundaries, or administrative regions.
- Polygon: A closed region bounded by multiple line segments (ways), representing enclosed features like parks, lakes, mountains, cities, and more. Polygons bring spatial cohesion to diverse features across the map.

1.1..2 Human-Made Structures

Within the expansive activity of human ingenuity, OSM catalogues in detail human-made structures, acting as a repository for the cartography of bridges, water towers. Bridges, symbols of connectivity and engineering prowess, find their digital manifestation in OSM. These structures, ranging from simple footbridges to complex highway overpasses, are not merely dots on a map but integral components of the spatial narrative woven by OSM. Similarly, water towers, iconic silhouettes on city skylines, are catalogued within OSM, contributing to a comprehensive depiction of the urban landscape. These structures serve as vital components of water distribution systems, and their inclusion in OSM enriches the dataset with critical infrastructure information.

1.1..3 Waterways in OpenStreetMap

Within the vast spectrum of OSM object types, waterways play a crucial role in depicting the fluid networks that traverse our landscapes. Waterways are a category of linear features that capture various aquatic elements, providing a detailed representation of rivers, streams, canals, and other water-related features. These waterways contribute to a nuanced understanding of the hydrological landscape, helping map users visualize the flow and connectivity of water bodies.

1.1..4 Natural Formations

Beyond human-made structures, OSM extends its reach to embrace the natural wonders of our planet. Springs, sources of freshwater emerging from the Earth, are among the natural formations meticulously documented in OSM. These features, often hidden gems in the landscape, add depth to the cartographic narrative by highlighting the natural hydrological diversity present in different regions.

1.1..5 Comprehensive Nature of OSM Data

The diversity of object types within OSM is a testament to the platform's commitment to curating a comprehensive and inclusive dataset. OSM goes beyond simple cartography; it endeavours to capture the essence of our surroundings by representing the intricate interplay between human interventions and the natural environment. The cataloguing of bridges, water towers, springs, and a myriad of other features contributes to the richness of the spatial dataset, making OSM a dynamic and evolving resource for understanding the complexities of our world. The collaborative efforts of the global OSM community ensure that this digital tapestry remains vibrant, reflecting the ever-changing landscape of our planet.

Export Types: Transforming OpenStreetMap (OSM) Data

To harness the wealth of information contained within OSM, our system employs an export mechanism that extracts raw data with the help of OverPass API and transforms it into a standardized, interoperable format known as GeoJSON. GeoJSON facilitates seamless integration with a wide array of geospatial tools and platforms, providing a versatile means to visualize and analyse the geographical features captured by OSM. Notably, LineString geometries in GeoJSON are employed to represent linear features like bridges and waterways. The process of transforming raw

OpenStreetMap (OSM) data into a usable format is a critical step in harnessing the richness of geographical information.

1.1..1 Exporting OSM Data to GeoJSON

A curated selection of object types is identified for extraction from OSM, encompassing diverse features such as human-made structures, waterways, amenities, bodies of water, and natural formations. Each category represents a thematic grouping of OSM features, setting the stage for a targeted export process. Object types can be added or excluded easily through variable configuration modification.

The transformation process involves:

- Enriching OSM Data: The OSM data, retrieved based on the specified object types, is enriched with additional information, such as unique identifiers and thematic categorizations.
- GeoJSON Structure: The enriched OSM data is then structured into the GeoJSON format. For line features like bridges and waterways, LineString geometries are utilized, providing a coherent representation of linear elements. Other features, such as amenities, bodies of water, and natural formations, may use Point or Polygon geometries as appropriate.
- Feature Collection: The transformed GeoJSON data is organized into a Feature Collection, ensuring a cohesive and standardized output that can be easily integrated into various geospatial tools and platforms.

In essence, the export types guide the extraction and transformation process, enabling the creation of a GeoJSON representation that retains the thematic richness of the original OSM data. This GeoJSON output becomes a versatile and interoperable resource for subsequent geospatial analysis and visualization, contributing to a more nuanced understanding of the geographical features within the OpenStreetMap dataset.

Overpass API: Querying OpenStreetMap (OSM) Data

The extraction of OpenStreetMap (OSM) data is facilitated through the OverPass API, a robust and versatile tool designed for querying and retrieving specific subsets of geospatial information. This section explores the functionalities of the OverPass API, outlining its role in dynamically formulating queries and extracting targeted data from the expansive OSM dataset.

1.1..1 OverPass API: A Gateway to OSM Data

The OverPass API serves as a pivotal gateway to the wealth of information stored in OpenStreetMap. Unlike OverPass Turbo, which provides a user-friendly interface for crafting queries, the OverPass API allows for programmatic access, enabling seamless integration into custom scripts and applications.

Key Features of the OverPass API include:

1. **Dynamic Query Formulation**: The OverPass API allows for the dynamic formulation of queries based on specific parameters such as object types, geographical bounding boxes, and

thematic criteria. This flexibility ensures precise data extraction tailored to project requirements.

2. **Programmatic Access:** As an API, OverPass enables programmatic interaction with OSM data, facilitating automation and integration into larger geospatial workflows. This is particularly advantageous for projects requiring real-time or periodic updates of OSM information.
3. **Efficient Data Retrieval:** OverPass API efficiently retrieves only the necessary OSM data based on the formulated query. This minimizes bandwidth usage and ensures that the extracted dataset aligns closely with the project's focus, optimizing both speed and resource efficiency.

1.1..2 Querying OSM Data Using the OverPass API

In practical terms, the querying process involves constructing a well-defined OverPass query, specifying the object types, geographical scope, and any additional filters. The OverPass API processes this query and returns the relevant OSM data in a structured format, ready for further processing. The OverPass API provides a foundation for seamless interaction with OSM data, enabling developers and researchers to harness the richness of the OSM dataset programmatically. Whether extracting data for analysis, visualization, or integration into a broader geospatial system, the OverPass API stands as a robust tool for accessing and utilizing OpenStreetMap's extensive geographic information.

5.4 Automated Identification System (AIS)

Information provided by the AIS

In the context of a waterway digital twin, the Automatic Identification System (AIS) serves as a critical input data source, providing real-time information essential for creating an accurate and dynamic representation of maritime activities [27].

The International Maritime Organization's (IMO) International Convention for the Safety of Life at Sea [28][23] requires operating AIS transmitters on all international cargo vessels of more than 300 tons displacement, all cargo vessels of more than 500 tons displacement, and all passenger vessels.

AIS serves the primary purpose of enabling ships to observe marine traffic within their vicinity while ensuring their visibility to other vessels. This functionality relies on a dedicated VHF AIS transceiver, enabling ships to monitor local traffic through an AIS-enabled chartplotter or computer monitor while simultaneously transmitting their own ship information to other AIS receivers. Shore-based facilities such as port authorities may opt for AIS receivers only, allowing them to monitor local traffic without transmitting their own location data. While this system provides reliable viewing of AIS-equipped traffic, its effectiveness is constrained within the VHF range, typically spanning 10 to 20 nautical miles [24]

However, global coverage and tracking of AIS signals are achieved through a combination of terrestrial AIS receiver stations and satellite-based AIS receivers. Terrestrial AIS Networks consist of a network of VHF receiver stations positioned along coastlines, ports, and waterways around the

world. Each station can receive AIS signals from vessels within its line-of-sight range, and by linking these stations through a global network, a comprehensive picture of maritime traffic in coastal areas is formed. To overcome the line-of-sight limitation and extend AIS coverage to open oceans and remote areas, satellites equipped with AIS receivers are used. These satellites can capture AIS signals from vessels far beyond the reach of terrestrial networks, providing truly global coverage. As the satellite orbits the Earth, it collects AIS data from different parts of the world, including mid-ocean where terrestrial coverage is not possible.

AIS data includes a range of information transmitted by vessels, such as identification details, location coordinates, speed, course, and navigational status. It encompasses 27 message types, each serving unique functions in maritime navigation and safety. These messages range from dynamic vessel position reports (Types 1, 2, 3) and static voyage-related information (Type 5) to more specialized functions like aid-to-navigation reports (Type 21) and search and rescue coordination (Type 9). They include data on vessel identity, position, course, speed, navigational status, and safety-related messages, along with system management communications like base station reports (Type 4) and binary messages for customized data exchange (Types 6, 8, 25) [26].

Out of these message types, the following provide information relevant to the use cases under consideration:

- **Type 1, 2, 3** - Position Report Class A: These messages are used by Class A transceivers and provide standard position information, such as the vessel's current position, speed over ground, and course over ground.
- **Type 5** - Static and Voyage Related Data: This message includes static information transmitted by Class A AIS transceivers, such as the vessel's IMO number, call sign, dimensions, type of ship, destination, and estimated time of arrival.

Consolidating the data of these message type the following information is made available:

- Static Information:
 - **MMSI (Maritime Mobile Service Identity):** A unique nine-digit number identifying the vessel.
 - **Vessel Name:** The official name of the vessel.
 - **IMO Number:** A unique identifier for ships, assigned by the International Maritime Organization (IMO).
 - **Call Sign:** The international radio call sign assigned to the vessel.
 - **Type of Ship:** Information about the type or class of the vessel (e.g., cargo, tanker, passenger, etc.).
 - **Dimensions:** The size of the vessel, including length and beam.
 - **Location of Position-Fixing Antenna:** The location of the AIS transceiver on the vessel, which can affect the accuracy of the position data.
- Dynamic Information:

- **Position:** The latitude and longitude of the vessel.
- **Time Stamp:** The time when the message was sent.
- **Course Over Ground (COG):** The direction in which the vessel is moving relative to the geographic North.
- **Speed Over Ground (SOG):** The speed of the vessel relative to the ground.
- **Heading:** The direction in which the vessel is pointed, typically from the vessel's compass.
- **Navigational Status:** Indicates the operational condition of the vessel (e.g., at anchor, under way, not under command, etc.).
- **Rate of Turn:** The rate at which the vessel is turning, which can be critical for understanding maneuvering intentions.
- Voyage-Related Information:
 - **Destination:** The vessel's intended destination.
 - **Estimated Time of Arrival (ETA):** The expected time the vessel will arrive at its destination.
 - **Draught:** The depth of water needed to float the vessel, indicating the current load condition.
 - **Cargo Type:** An optional detail that can indicate the type of cargo the vessel is carrying, especially relevant for hazardous materials.
- Safety-Related Messages:
 - These can include short safety-related messages that the vessel can broadcast to nearby ships and shore stations, providing alerts or important information relevant to navigation or safety.

Raw signal acquisition and decoding

The raw signal is built upon a modified version of the NMEA 0183 standard, which is a well-established framework for data exchange in maritime navigation devices. The NMEA 0183 protocol is known for its extensive use and detailed documentation, available for reference at the NMEA organization's resources.

The NMEA sentences used for AIS purposes are AIVDM and AIVDO. AIVDM is used for sending messages, while AIVDO is used for received messages. These sentences form a two-layered communication system, with the first layer being the NMEA protocol and the second layer involving the encoding of field 6, as mentioned below. A typical AIVDM data packet is dissected as follows:

- Field 1: Denoted as !AIVDM, this field categorizes the packet as an AIVDM type.
- Field 2: Indicates the total number of fragments into which the message is divided. Due to NMEA 0183's limitation of 82 characters per sentence, longer messages need to be segmented.

- Field 3: Specifies the sequence number of the current fragment, starting from one. A message where both the fragment count and fragment number are one signifies a self-contained sentence.
- Field 4: This is reserved for a sequential message identifier used for multi-sentence messages and is omitted when not applicable.
- Field 5: Represents the radio channel code, where 'B' typically signifies AIS Channel B, operating at 162.025Mhz. Occasionally, codes '1' and '2' appear, though not formally defined in the standards.
- Field 6: Contains the encoded data payload, the substance of the message, which will be explained in subsequent sections.
- Field 7: Shows the number of fill bits used to pad the data payload for proper alignment to a 6-bit boundary. This number can range from 0 to 5 and interacts intricately with the byte alignment requirements of over-the-air AIS messages.

Following the data fields, the sentence is concluded with an NMEA 0183 checksum, denoted by an asterisk (*), which ensures the integrity of the transmitted data.

The 6th field contains the payload in an ASCII-encoded 6-bit vector. The first 6 bits of the payload define the message type. Depending on the message type, the respective pattern should be applied for decoding the remaining message.

This decoding approach should be applied when reading data from a raw source such as the serial output of AIS transceiver or a Terrestrial AIS Network endpoint.

Commercial service providers

Commercial AIS data providers offer a range of interfaces and services to access AIS data for maritime tracking and analytics. In general, they offer processed information at a higher level compared to the raw data received by an AIS receiver

Providers like Spire Maritime and ORBCOMM boast a comprehensive set of APIs that facilitate the integration of AIS data into user applications. Spire Maritime provides a constellation that delivers real-time AIS data with a latency of less than one minute, ensuring an up-to-date and accurate view of global shipping activities [29][30].

Similarly, ORBCOMM provides flexible APIs and AIS data sets tailored for applications such as search and rescue, surveillance and security, IUU fishing, environmental monitoring, and more. Their solutions are described as versatile and suitable for diverse applications[31]

Another provider, OrbitSailor, offers AIS data services through various subscription tiers, ranging from basic tracking to analytics and position reporting. Their services include real-time tracking, satellite AIS data, and time-stamped reports, with higher-tier subscriptions offering value-added services such as analytics for vessel behavior and transit monitoring of forbidden areas [32].

MarineTraffic, another AIS data provider, suggests that when choosing an AIS data provider, it's crucial to consider factors like the range of AIS coverage, the frequency of position updates, data cleanliness, ease of use through the platform, ongoing support, and the trustworthiness of the service provider [33].

Commercial AIS data providers typically offer a variety of APIs, including REST (Representational State Transfer) and GraphQL, to access their data services. These APIs are designed to cater to different user needs and preferences for data retrieval and manipulation.

REST APIs are the most common type of API provided by AIS data service providers. REST is a web standards-based architecture that uses HTTP requests to access and use data. The RESTful model is stateless, and it allows users to access and manipulate resources using a set of predefined operations. In the context of AIS data, REST APIs allow users to retrieve information about vessels, voyages, and maritime traffic by sending HTTP requests to the provider's server, which then returns the data typically in JSON or XML format.

REST APIs are known for their simplicity and ease of use. They are stateless, meaning that each call from a client to the server must contain all the information needed to understand and complete the request. This approach is highly scalable and flexible, allowing for easy integration with a wide variety of platforms and services.

GraphQL APIs, on the other hand, offer a more flexible and efficient approach to interacting with AIS data services. GraphQL is a query language for APIs and a runtime for executing those queries with existing data. It provides a more efficient, powerful, and flexible alternative to the REST API. It allows clients to specify exactly what data they need, which can reduce the amount of data transferred over the network and decrease the number of requests required to fetch data.

With GraphQL, users can create complex queries that gather data from multiple sources and return it in a single request. This can be particularly useful for maritime data applications that need to aggregate information from various data points, such as vessel position, cargo details, and port data.

Both REST and GraphQL APIs provided by commercial AIS data services come with comprehensive documentation, including how to authenticate, make requests, and handle responses. They also often provide support for developers during the integration process and may offer client libraries or SDKs to simplify the use of their APIs in various programming environments.

6 Connectors Library

The Connectors Library is an integral part of the dataspace, designed to facilitate the acquisition and ingestion of data from a variety of external data sources into the platform. These connectors are implemented as microservices, each tailored to interact with specific data sources, ensuring that data is seamlessly integrated into the platform for subsequent processing and storage. The library serves

as a link between the digital twin and the external world, enabling the system to access real-time and historical data necessary for its operations.

The development of these connectors is driven by the specific use cases defined by the LLs, which dictate the types of data and the sources from which this data needs to be acquired. For instance, connectors might be designed to fetch data from hydrological stations for water levels and flow rates, or from vessel tracking services for maritime traffic data. Each connector is responsible for retrieving data and preprocessing it as necessary to conform to the data formats and structures expected by the digital twin's data processing layers.

In this chapter is presented the first version of the Connectors Library. It includes a comprehensive description of each connector that has been developed thus far. This description covers the data source each connector interacts with, the specific data it retrieves, the method of data acquisition (e.g., API calls, etc.), and any initial data processing steps performed by the connector. The chapter also outlines the protocols and standards adhered to for data exchange, ensuring compatibility and interoperability within the digital twin architecture.

This modular approach to data acquisition, facilitated by the Connectors Library, enhances the flexibility and scalability of the digital twin and allows for the incremental expansion of the system as new data sources are identified and integrated. By decoupling data acquisition from the core processing and storage components, the library enables the digital twin to adapt to evolving data requirements and use cases, ensuring its long-term relevance and utility.

6.1 'EuRIS API Caller' microservice

The 'EuRIS API Caller' microservice functions as an integrator, operating in both on-demand and scheduled modes to bridge the Euris API service with the knowledge graph API. This component facilitates communication between the two APIs, ensuring the timely retrieval and integration of data into the knowledge graph repository. Operating in on-demand mode, the microservice responds to immediate requests, fetching data from the Euris API and transmitting it to the knowledge graph API in real-time. Additionally, in scheduled mode, the microservice executes periodic tasks, enabling regular updates and synchronization between the Euris API and the knowledge graph. Specifically, the scheduled process is configured to run once per week, ensuring consistent data updates and alignment between the two systems. The service retrieves and processes data in a structured manner, prioritizing the extraction of static data such as berths, terminals, bridges, locks, sublocks, and associated areas of competence. This involves gathering waterway names to establish connections, ensuring comprehensive coverage and accuracy within the knowledge graph repository. For further details regarding the data retrieval and processing methodology, stakeholders can refer to the EuRIS network data chapter. By orchestrating this seamless integration and synchronization process, the 'EuRIS API Caller' microservice significantly contributes to the comprehensive and utilitarian nature

of the knowledge graph repository. In Figure 9 is shown screen of the Git repository of the 'EuRIS API Caller' microservice.

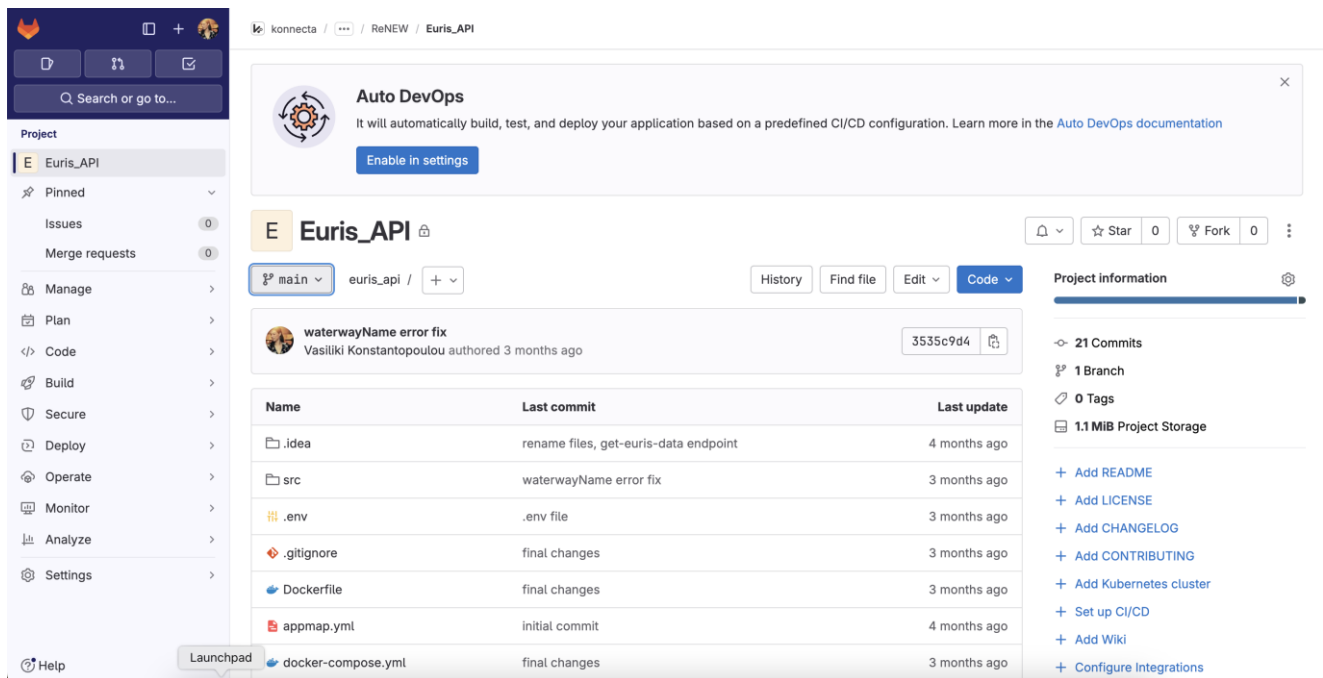
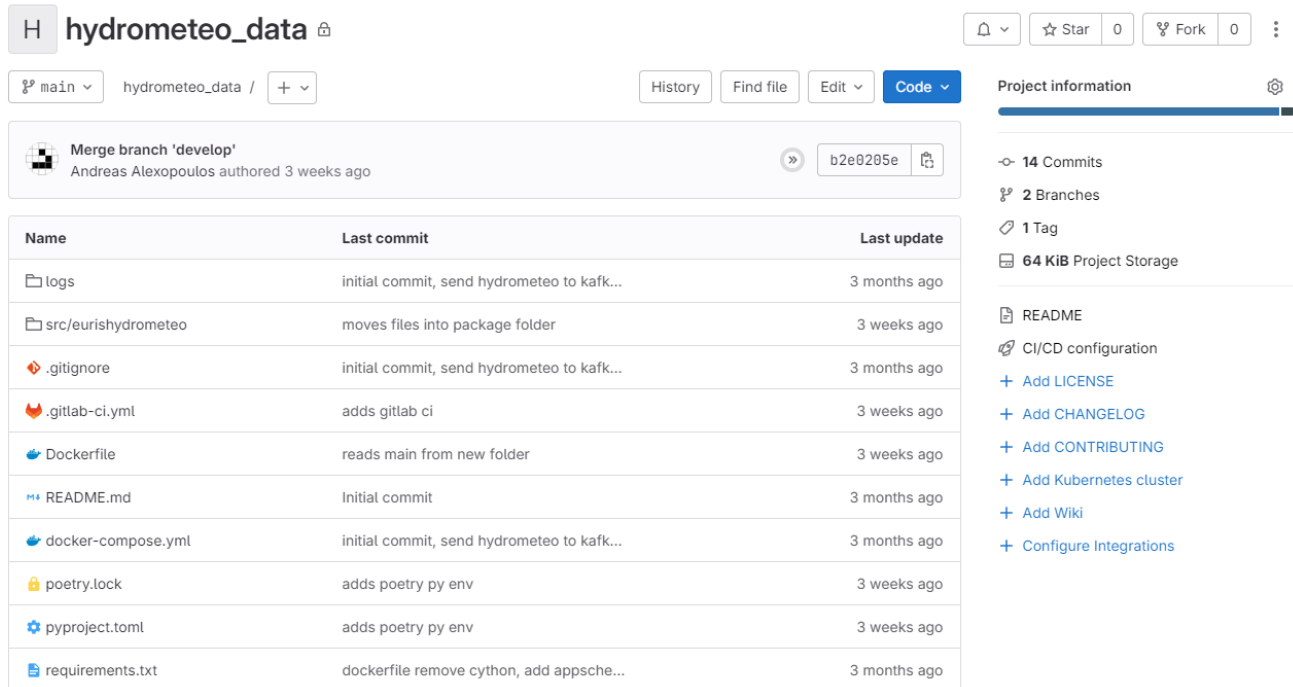


Figure 9 Git repository: 'EuRIS API Caller' microservice

6.2 'hydrometeo_data' microservice

The Hydrometeo_data microservice, implemented in Python, serves as a tool for accessing hydrometeorological data from a designated Euris endpoint. Operating on a 60-minute interval, requests are sent to the 'GetHydrometeoData' EurIS endpoint retrieving data fields including hydrometeoProviderId, value, dataUnit, measuredAt, and alarmStatus, structured into a dictionary format. This parsed data is then transmitted via a dedicated Kafka topic, facilitating downstream accessibility. Additionally, supplementary data fields such as hydrometeoProvider, lat & lon coordinates, linkID, linkName, and linkDataProviderCode are integrated into a post request sent to the knowledge graph API, enriching the repository with comprehensive information. Through adherence to stringent technical specifications and seamless integration with Kafka and knowledge graph APIs, the microservice ensures the availability and enrichment of critical hydrometeorological data for stakeholders within the ecosystem. In Figure 10Figure 9 is shown screen of the Git repository of the "hydrometeo_data" microservice.



hydrometeo_data

Merge branch 'develop'
Andreas Alexopoulos authored 3 weeks ago

Name	Last commit	Last update
logs	initial commit, send hydrometeo to kafk...	3 months ago
src/eurishydrometeo	moves files into package folder	3 weeks ago
.gitignore	initial commit, send hydrometeo to kafk...	3 months ago
.gitlab-ci.yml	adds gitlab ci	3 weeks ago
Dockerfile	reads main from new folder	3 weeks ago
README.md	Initial commit	3 months ago
docker-compose.yml	initial commit, send hydrometeo to kafk...	3 months ago
poetry.lock	adds poetry py env	3 weeks ago
pyproject.toml	adds poetry py env	3 weeks ago
requirements.txt	dockerfile remove cython, add appsche...	3 months ago

Project information

- 14 Commits
- 2 Branches
- 1 Tag
- 64 KiB Project Storage
- README
- CI/CD configuration
- [Add LICENSE](#)
- [Add CHANGELOG](#)
- [Add CONTRIBUTING](#)
- [Add Kubernetes cluster](#)
- [Add Wiki](#)
- [Configure Integrations](#)

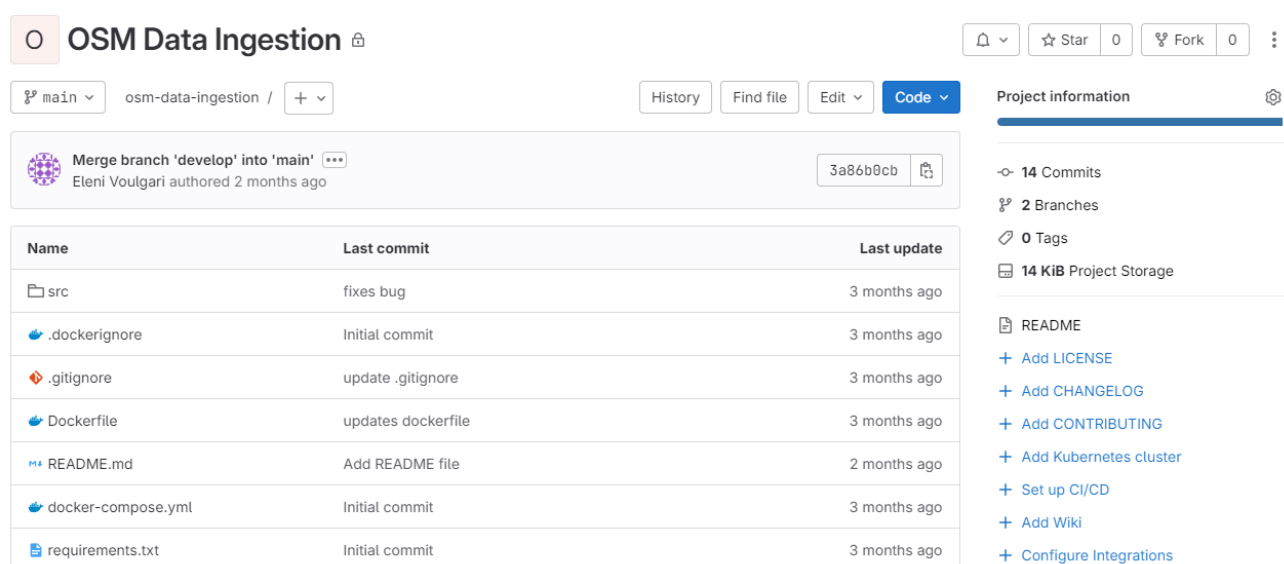
Figure 10 Git repository: 'hydrometeo_data' microservice

6.3 'OSM Data maintain' microservice

The 'OSM Data Maintenance' microservice orchestrates the interaction with OpenStreetMap (OSM) data to ensure its precision, pertinence, and integration into our DT framework. At the core of the 'OSM Data Maintenance' microservice is the detailed classification of OSM object types. The term "OSM type" serves as a key element, facilitating the categorization of these objects into fundamental shapes—points, lines, polygons—representing diverse features on the map. This microservice is responsible for managing the extraction, transformation, and maintenance of various object types, encompassing both human-made structures and natural formations. The export types, carefully defined within the 'OSM Data Maintenance' microservice, guide the transformation of raw OSM data into the GeoJSON format. This intricate process involves enhancing OSM data with supplementary information, structuring it into GeoJSON features, and organizing the output into a Feature Collection. Thematic categorizations, like those outlined in the initial code snippet, enable the microservice to extract and represent specific features, such as bridges, waterways, amenities, bodies of water, and natural formations. The 'OSM Data Maintenance' microservice relies on the OverPass API as a crucial gateway to OSM data. Unlike OverPass Turbo, the OverPass API provides programmatic access, empowering the microservice to dynamically formulate queries based on predefined parameters. This technical capability enables the microservice to efficiently retrieve essential OSM data,

optimizing resource usage and ensuring that the extracted dataset aligns precisely with the thematic focus of our project.

One distinctive feature of the 'OSM Data Maintenance' microservice is its ability to update OSM data on-demand. This means that the microservice can be triggered to refresh the dataset in real-time, accommodating the evolving nature of OSM information. This capability ensures that our geospatial system operates with the most current and relevant data, meeting the demands of dynamic analytical scenarios and staying synchronized with the dynamic changes in the OSM landscape. In Figure 11 Figure 9 is shown screen of the Git repository of the “OSM Data maintain” microservice.



OSM Data Ingestion

main osm-data-ingestion /

Merge branch 'develop' into 'main'
 Eleni Voulgari authored 2 months ago

3a86b8cb

Name	Last commit	Last update
src	fixes bug	3 months ago
.dockerignore	Initial commit	3 months ago
.gitignore	update .gitignore	3 months ago
Dockerfile	updates dockerfile	3 months ago
README.md	Add README file	2 months ago
docker-compose.yml	Initial commit	3 months ago
requirements.txt	Initial commit	3 months ago

Project information

- 14 Commits
- 2 Branches
- 0 Tags
- 14 KiB Project Storage
- README
- + Add LICENSE
- + Add CHANGELOG
- + Add CONTRIBUTING
- + Add Kubernetes cluster
- + Set up CI/CD
- + Add Wiki
- + Configure Integrations

Figure 11 Git repository: 'OSM Data maintain' microservice

6.4 'Euris Webhook Data Ingestion Service' microservice

EuRIS provides to shipowners the functionality to assign a URL to receive available AIS information for their vessels. The respective EuRIS form is shown in Figure 12. To facilitate the development of our service, we have requested a shipowner to register our webhook under their vessel. This allows us to receive updates on its status.

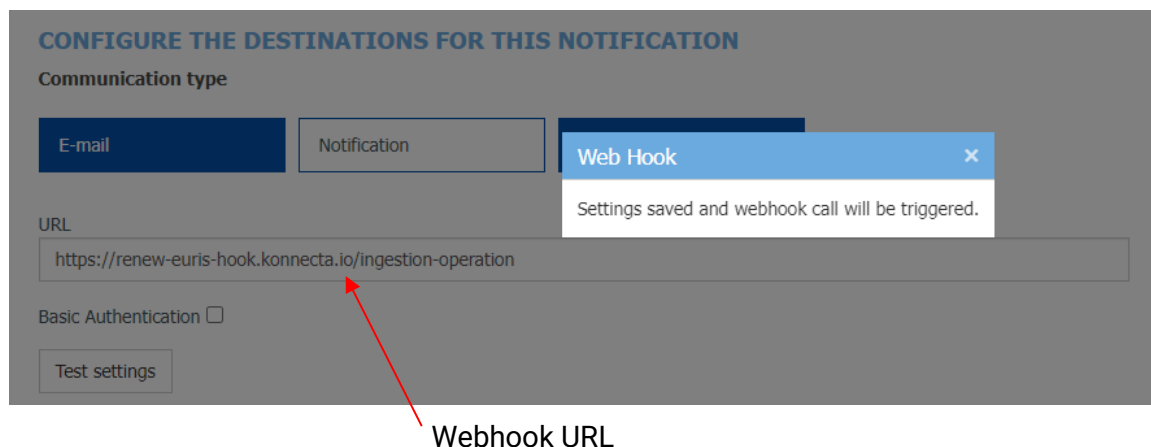


Figure 12 Configuration of Webhook URL in EuRIS portal

The EURIS Webhook Data Ingestion service publishes a REST (Representational State Transfer) endpoint. The data received by the endpoint are being ingested into the service and then be published to a KAFKA topic. The endpoint accepts an object in JSON (JavaScript Object Notation) format via a POST operation, as shown in Table 7. The objects sent to the endpoint are vessel position or ETA change reports. A sample is shown in Table 8.

In Figure 13Figure 9 is shown screen of the Git repository of the “Euris Webhook Data Ingestion Service” microservice.

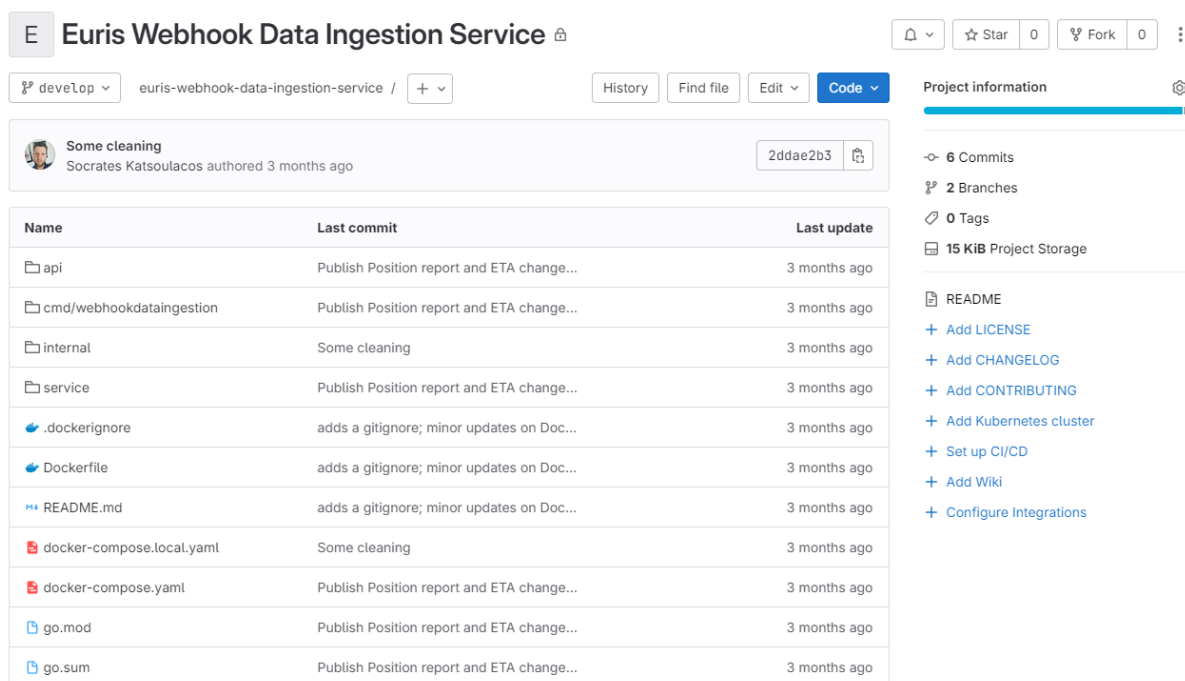


Figure 13 Git repository: 'Euris Webhook Data Ingestion Service'

Table 7 Schema of the expected POST operation at the webhook service

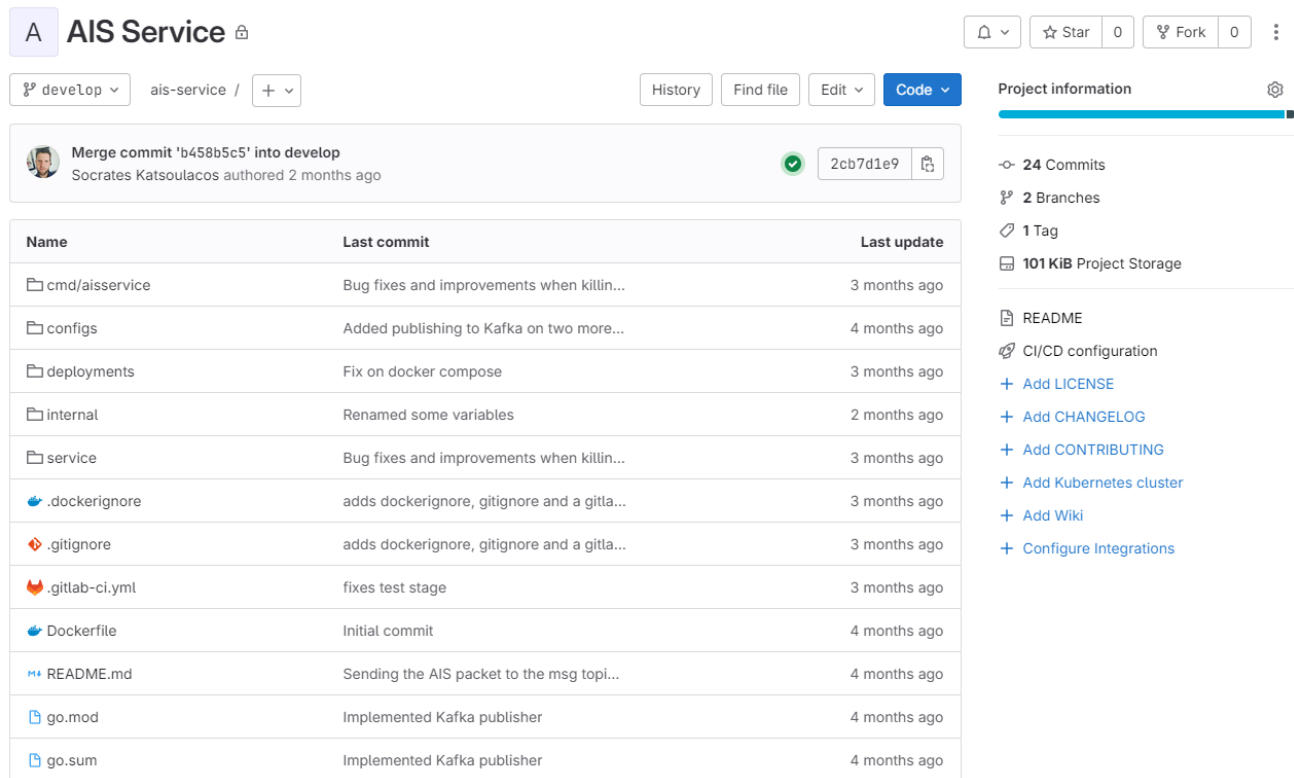
```
{
  ReportType, (string)
  Lat, (string)
  Lon, (string)
  MMSI, (string)
  ObjectName, (string)
  ShipName, (string)
  CalculatedETA, (time)
  PortOfDeparture, (string)
  PortOfDestination, (string)
  TrackName, (string)
  VoyageNr, (string)
  TimeStamp (time)
}
```

JSON Examples:

Table 8 Position Report sample message

```
{
  ReportType: "PositionReport",
  Lat: "51.05169",
  Lon: "4.36317",
  MMSI: "205342890",
  ObjectName: "Ligplaats Willebroek RO kmp 24.3 (2)",
  ShipName: "BLAUWE REIGER",
  TimeStamp: "2024-01-28T15:58:38.9562622+01:00"
}
```

6.5 'AIS Service' microservice



The screenshot shows the GitHub repository for the 'AIS Service' microservice. The repository is named 'AIS Service' and is in the 'develop' branch. It has 24 commits, 2 branches, and 1 tag. The project storage is 101 KiB. The repository includes a README, CI/CD configuration, and various files like .dockerignore, .gitignore, .gitlab-ci.yml, Dockerfile, go.mod, go.sum, and go.sum. The last commit was made 2 months ago by Socrates Katsoulacos.

Name	Last commit	Last update
cmd/aiservice	Bug fixes and improvements when killin...	3 months ago
configs	Added publishing to Kafka on two more...	4 months ago
deployments	Fix on docker compose	3 months ago
internal	Renamed some variables	2 months ago
service	Bug fixes and improvements when killin...	3 months ago
.dockerignore	adds dockerignore, gitignore and a gitla...	3 months ago
.gitignore	adds dockerignore, gitignore and a gitla...	3 months ago
.gitlab-ci.yml	fixes test stage	3 months ago
Dockerfile	Initial commit	4 months ago
README.md	Sending the AIS packet to the msg topi...	4 months ago
go.mod	Implemented Kafka publisher	4 months ago
go.sum	Implemented Kafka publisher	4 months ago

Figure 14 Git repository: 'AIS Service' microservice

APDL, LL2 partner, maintains a TCP (Transmission Control Protocol) server with AIS data at the vicinity of the Douro river. The AIS Service interacts with this TCP server, fetching AIS (Automatic Identification System) messages from a TCP socket continuously.

A TCP socket allows devices to exchange data in a network and ensures the successful delivery of the data from one device to another. The service has been implemented with a retry/reconnect mechanism to guarantee that the connection stays alive all the time so that data can be read from it.

Once an AIS message is fetched the service will parse the messages of type **1** and **5** (discarding other types) and read the data we are interested at from each one.

The following are the fields from **AIS message type 1** (Position Report) that are being used:

- SOG,
- Lat,
- Lon,
- Course,

- Heading, and
- Navigation Status

A sample of AIS message type 1 is shown in Table 9.

Table 9 AIS Message type 1 “raw” packet data example

```
{
  Header: {MessageID: 1 RepeatIndicator: 0 UserID: 263401660}
  Valid: true NavigationalStatus: 15 RateOfTurn: -128 Sog: 8.4
  PositionAccuracy: true
  Longitude: -8.755675 Latitude: 41.07676333333333 Cog: 20.4 TrueHeading: 511
  Timestamp: 13
  SpecialManoeuvreIndicator: 0 Spare: 0 Raim: true
  CommunicationStateNoItDMA: {CommunicationState: 33281}
}
```

The following are the fields from **AIS message type 5 (Ship Static Data)** that are being used:

- ETA,
- DraftAft, and
- Destination Port

A sample of AIS message type 1 is shown in Table 10

Table 10 AIS Message type 5 “raw” packet data example

```
{
  Header: {MessageID: 5 RepeatIndicator: 0 UserID: 263441683}
  Valid: true AisVersion: 2 ImoNumber: 9903595 CallSign:CSZU2 Name:DORIS Type: 52
  Dimension: {A: 12 B: 13 C: 5 D: 8}
  FixType: 1 Eta: {Month: 6 Day: 13 Hour: 8 Minute: 0}
  MaximumStaticDraught: 6 Destination:LEIXOUS Dte: false Spare: false
}
```

Finally, the service will publish the “raw” packet of these AIS messages (type 1 and 5) to two separate KAFKA topics, as well as publish the data we are interested at (see below) to another third topic – MMSI Data, as show below.

In Table 11 is the JSON schemas that are being used to publish data to the `MMSI` Kafka topic, separating values one by one and characterizing them by a `tag` id.

In Table 12 is the JSON schema of the `Message Types` Kafka topic, where the whole received AIS packet is published to be used in case of further processing required.

MMSI (i.e. only interested data from messages) and Message Types (packet) are shown in Table 11 and Table 12, respectively.

Table 11 "MMSI" Kafka topic schema

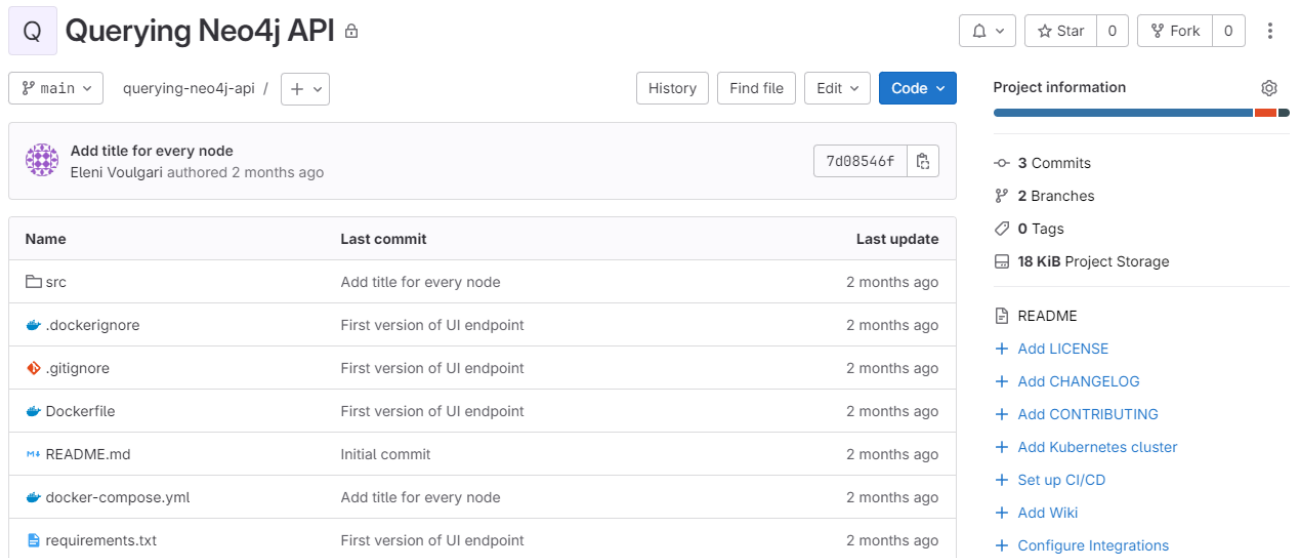
```
{
  mmsi, (string)
  tag, (string)
  metric, (string)
  timestamp (time.Time) //timestamp of message retrieval
}
```

Table 12 "Message Types" Kafka topic schema

```
{
  packet, (object)
  timestamp (time.Time) //timestamp of message retrieval
}
```

In Figure 14Figure 15 is shown screen of the Git repository of the "AIS Service" microservice.

6.6 'INGESTION KG API' microservice



Querying Neo4j API

main querying-neo4j-api

Add title for every node
Eleni Voulgari authored 2 months ago

Name	Last commit	Last update
src	Add title for every node	2 months ago
.dockerignore	First version of UI endpoint	2 months ago
.gitignore	First version of UI endpoint	2 months ago
Dockerfile	First version of UI endpoint	2 months ago
README.md	Initial commit	2 months ago
docker-compose.yml	Add title for every node	2 months ago
requirements.txt	First version of UI endpoint	2 months ago

Project information

- 3 Commits
- 2 Branches
- 0 Tags
- 18 KiB Project Storage
- README
- + Add LICENSE
- + Add CHANGELOG
- + Add CONTRIBUTING
- + Add Kubernetes cluster
- + Set up CI/CD
- + Add Wiki
- + Configure Integrations

Figure 15 Git repository: 'Ingestion KG API'

The 'INGESTION KG API' microservice is a component, designed to facilitate the ingestion and integration of structured data into a knowledge graph. This microservice is implemented as a RESTful API using the Flask web framework in Python. It comprises of an endpoint ("*/post-data*") implementing HTTP Method: *POST* and consumes: *application/json*.

Key features of the Ingestion KG API include:

1. **Data ingestion and graph update:** The microservice is responsible for receiving JSON data, validating it against predefined schemas and subsequently updating a Neo4j knowledge graph with the processed information.
2. **CSV label file generation:** Upon receiving JSON data, the microservice extracts relevant information and generates a CSV file, containing the type (label) of each piece of data that is to be inserted in Neo4j database.
3. **Label assignment:** The microservice assigns new labels to nodes within the knowledge graph using Cypher queries.
4. **Properties update:** It processes data to create a structured JSON file, which is then utilized to update properties of nodes in the knowledge graph.
5. **Error Handling and logging:** The microservice incorporates error-handling mechanisms to ensure the integrity of data processing and graph updates. Detailed logs provide insights into the execution flow and potential issues.

Integration with External Systems

The microservice is designed to seamlessly integrate with external systems, especially those producing JSON data adhering to the specified schemas. It acts as a connector, enabling the smooth flow of data from external sources to a Neo4j knowledge graph.

The 'INGESTION KG API' microservice bridges external data sources and a Neo4j knowledge graph. Its modular functionalities ensure flexibility and adaptability to varying data formats, contributing to the overall success of the data integration initiative. In Figure 15 is shown screen of the Git repository of the "INGESTION KG API" microservice.

6.7 'KG - Kafka connector' microservice

The kafka-to-kg microservice serves as a critical component in transferring data from designated Kafka topics, which are filled from the 'AIS Service' microservice to a REST API that interfaces with the knowledge graph. Functioning as a Kafka consumer, the microservice subscribes to three distinct Kafka topics, each providing varied AIS messages. Upon receiving messages, the microservice processes them based on their source and format, extracting pertinent fields such as locode (containing the MMSI code of the vessel), Sog (speed over ground), Latitude, Longitude, Cog (course over ground), TrueHeading, ImoNumber, CallSign, Name, Type, Dimension, Eta (estimated time of arrival), MaximumStaticDraught, and Destination. These extracted fields are then forwarded to the REST API, facilitating communication with the knowledge graph component. By seamlessly integrating Kafka message consumption with REST API communication, the microservice enables the enrichment of the knowledge graph repository with real-time vessel-related data, thereby enhancing its utility and comprehensiveness for stakeholders. In Figure 16 is shown screen of the Git repository of the 'KG - Kafka connector' microservice.

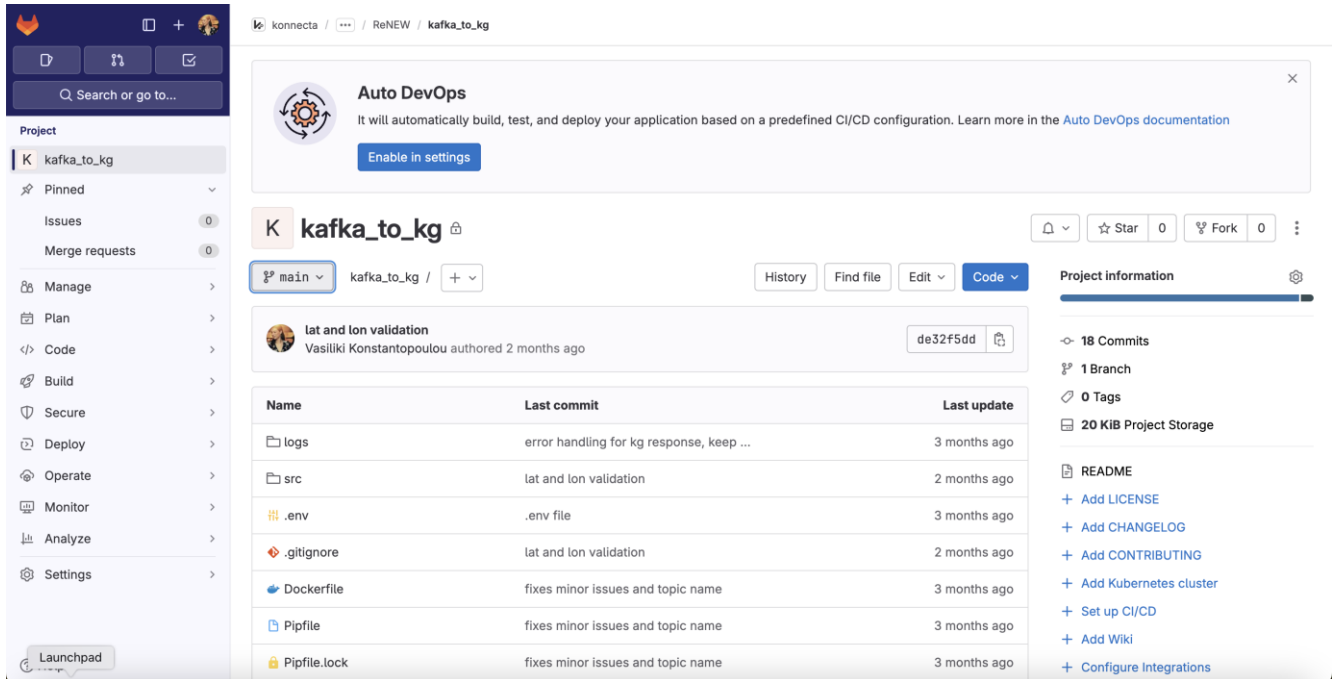


Figure 16 Git repository: 'KG - Kafka connector'

7 Deployment of an operational compilation of the developed dataspace components

7.1 General description of the deployment

Chapter 4 outlines the foundational elements designed to empower the digital twin for optimal functionality within a dataspace environment. Chapter 6 delves into the connectors developed to facilitate integration with identified data sources deemed essential for addressing the Living Labs (LLs) use cases.

This chapter presents the integration of the foundational components of the Digital Twin Dataspace with the connectors library, aiming to realize a prototype operational system. This system is engineered to collect, process, and provide access to the data sources discussed earlier, thereby demonstrating an initial practical application.

Figure 17 presents a structured overview of the data flow and component interaction within the deployed solution, starting from various data sources to the final user interfaces. On the left, data sources such as the EuRIS API, OSM Data, and APDL meteorological data are outlined, each providing different types of information like static geodata, infrastructure status, and hydrometeorological data.

These are then fed into the ingestion layer, where services like the EuRIS API Caller and AIS Service standardize and prepare the data for further processing.

Moving through the pipeline, the transport-transform layer is represented, primarily facilitated by Kafka, which acts as a central hub for data streaming and message queuing. This layer is responsible for transforming the ingested data into a suitable format for storage and querying. Services such as Kafka_to_kg and Kafka_to_influx indicate the flow of data towards specific storage solutions.

In the storage layer, databases like Neo4j and InfluxDB, optimized for time-series data, are shown to house the processed information, making it persistent and ready for analysis. Finally, the user interfaces on the right, including the Querying KG API and GraphQL, illustrate the endpoints available for users to interact with and retrieve information from the system. These interfaces provide the means to query the data efficiently and are likely designed for different user roles or purposes.

The color coding and associated percentages in the figure serve as indicators of the completion status for each component.

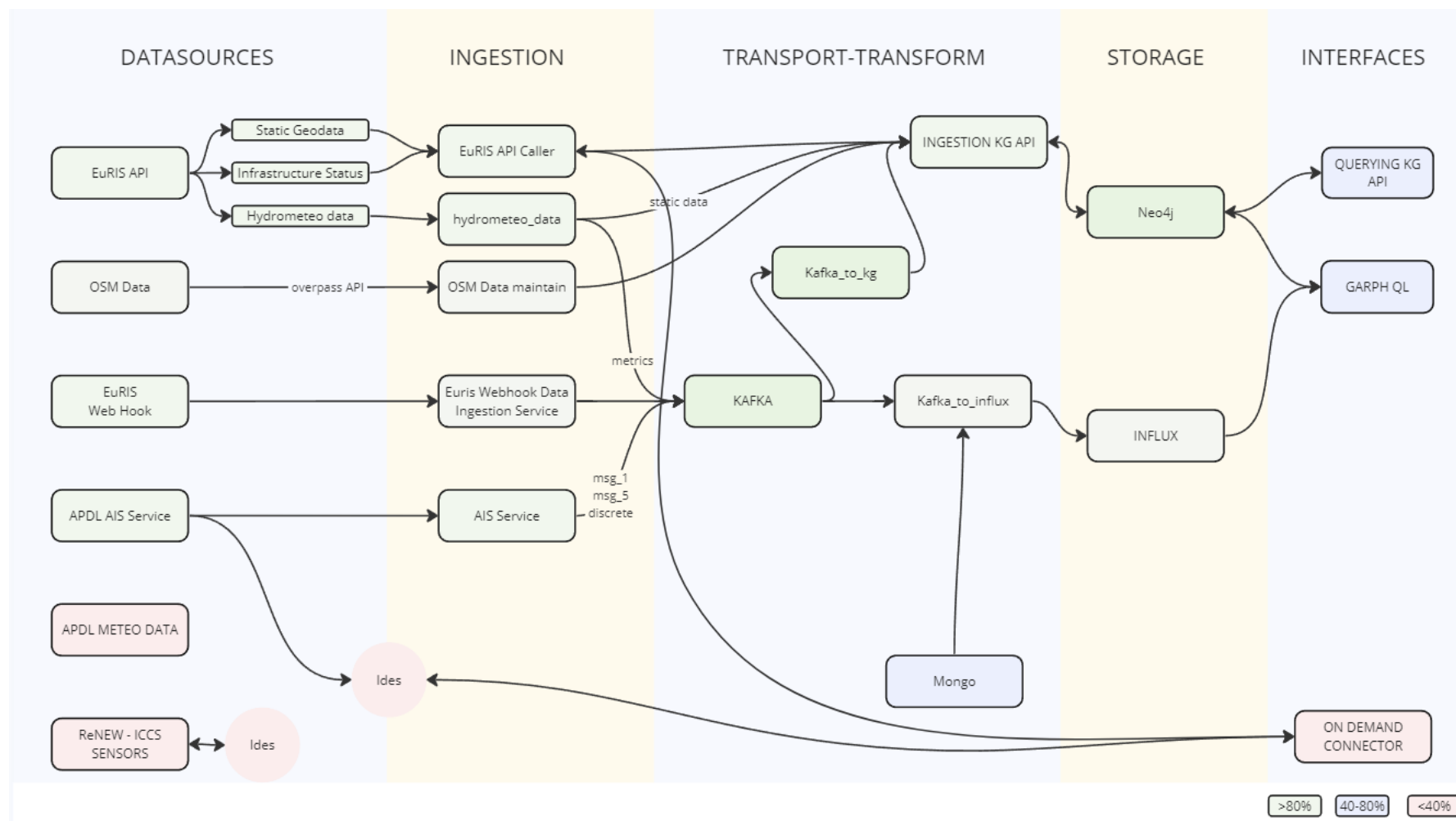


Figure 17 Mapping of developed connectors to respective datasources and the data processing layers of the pipeline

7.2 Repository and deployment

The diverse components developed, depicted as projects in Figure 18, have been containerized, ensuring each component is encapsulated within its own Docker container. These container images have been compiled and are hosted within the Container Registry specific to our GitLab repository, facilitating easy access and deployment. This repository is intended to be open to the other partners of the project, enhancing the collaboration toward our common aim.

The ReNEW project, also shown in Figure 18, features a Docker Compose YAML file that outlines the execution of these various components within a Docker environment. Upon executing this Docker Compose file on a designated server, a structured process unfolds where each containerized service is initiated in a sequential manner, adhering to the defined dependencies and order of operations. Figure 19 provides a screenshot of the Docker containers actively running for development and evaluation purposes on a server in Konnecta's infrastructure..

ReNEW



**Funded by
the European Union**

R

ReNEW 

Subgroups and projects

Shared projects

Archived projects



K

Kafka Connect 

Kafka Connect for ReNEW project



H

hydrometeo_data 



R

ReNEW Platform 



Q

Querying Neo4j API 



A

AIS Service 



K

kafka_to_kg 



K

Knowledge Graph 



O

OSM Data Ingestion 



N

Neo4j 



E

Euris Webhook Data Ingestion Service 



E

Euris_API 



K

Kafka-2-InfluxDB 

Figure 18 the ReNEW Gitlab repository

• deployer@ubuntu-2004-focal-64-minimal:~\$ docker ps grep renew			
462f1db19228	registry.gitlab.com/konnecta/projects/renew/kafka-connect:latest	"kcg"	3 weeks ago Up 3 weeks (healthy)
eb3377ca089f	registry.gitlab.com/konnecta/projects/renew/kafka_to_kg:latest	"python src/main.py"	3 weeks ago Up 3 weeks
723364bc145b	registry.gitlab.com/konnecta/projects/renew/euris_api:latest	"python src/app.py"	3 weeks ago Up 3 weeks 0.0.0.0:3402->5000/tcp
a73792745fcc	registry.gitlab.com/konnecta/projects/renew/hydrometeo_data:latest	"python src/eurishyd..."	3 weeks ago Up 3 weeks
2efadfe7c69	registry.gitlab.com/konnecta/projects/renew/knowledge-graph:latest	"python /srv/app/src..."	3 weeks ago Up 3 weeks 0.0.0.0:3401->5050/tcp
864dd495d4dc	registry.gitlab.com/konnecta/modules/influxdb2/kafka-2-influxdb2:latest	"k2i -vv"	3 weeks ago Up 3 weeks (healthy)
6be52d39aee	registry.gitlab.com/konnecta/projects/renew/kafka-2-influxdb2-1	"/app/aissrv"	3 weeks ago Up 3 weeks 8080/tcp
83c491a4d68a	registry.gitlab.com/konnecta/projects/renew/ais-service:latest	"tini -g -- /startup..."	3 weeks ago Up 3 weeks (healthy) 7473/tcp, 0.0.0.0:3421->7474/tcp, 0.0.0.0:3422->7687/tcp
1ccc8251a2e9	registry.gitlab.com/konnecta/projects/renew/neo4j:latest	"start-kafka.sh"	3 weeks ago Up 3 weeks (healthy)
5a516dd5c678	registry.gitlab.com/konnecta/projects/renew/neo4j-1	"python /srv/app/src..."	3 weeks ago Up 3 weeks 0.0.0.0:3403->5050/tcp
be79dc6ab7e7	5b5ea1807970	"/kafdrop.sh"	5 weeks ago Up 5 weeks 0.0.0.0:3430->9001/tcp
131999df960c	registry.gitlab.com/konnecta/projects/renew/kafdrop-1	"/docker-entrypoint..."	2 months ago Up 2 months 0.0.0.0:3405->80/tcp
f352a6795c3c	wurstmeister/zookeeper:latest	"/bin/sh -c '/usr/sb..."	2 months ago Up 2 months (healthy) 22/tcp, 2181/tcp, 2888/tcp, 3888/tcp
a0e50c539fe5	influxdb:2.7.1-alpine	"/entrypoint.sh infl..."	2 months ago Up 2 months (healthy) 0.0.0.0:3420->8086/tcp
2aa18855783c	registry.gitlab.com/konnecta/projects/renew/influxdb-1	"/app/euriswebhookda..."	2 months ago Up 2 months 0.0.0.0:3400->8080/tcp
	registry.gitlab.com/konnecta/projects/renew/euris-webhook-data-ingestion-service:latest		
	registry.gitlab.com/konnecta/projects/renew/euris-webhook-data-ingestion-service-1		

Figure 19 Screenshot of Docker containers running on server

7.3 The knowledge Graph

As described in §4.2, a Neo4j knowledge graph database is employed to store static information, primarily metadata, pertaining to the waterway infrastructure and vessels, along with their latest state where applicable. This state encompasses factors such as lock conditions, level sensor values, vessel positions, speed courses, destinations, etc.

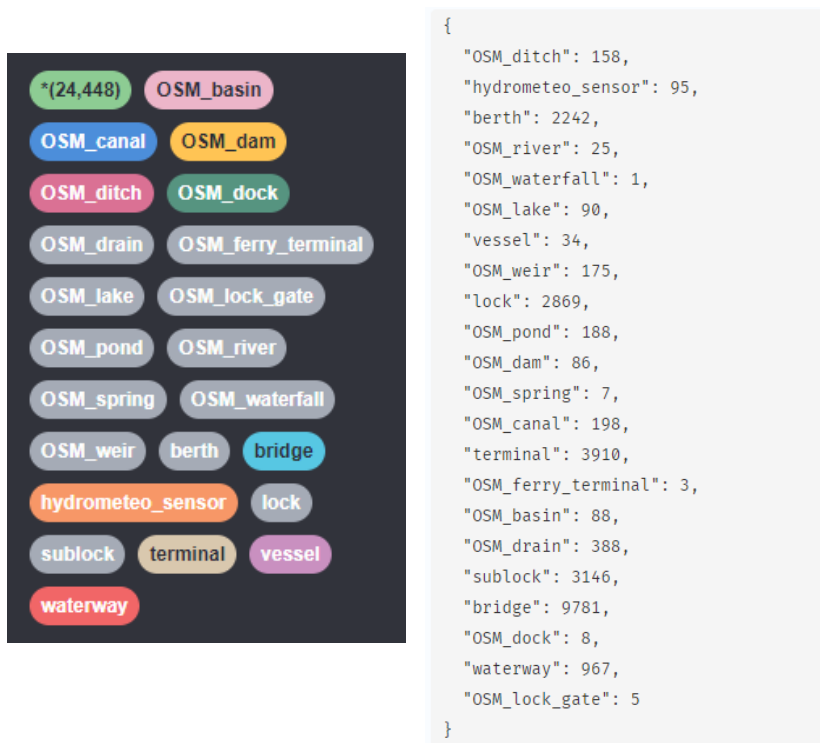


Figure 20 Knowledge graph nodes labels and their count

In Figure 20, the labels of nodes existing in the knowledge graph, along with their respective counts, are presented. Three distinct types of nodes are identifiable: those originating from OSM (OpenStreetMaps), the 'vessel' nodes generated from AIS data ingestion, and the remaining nodes originating from Euris. Within this category, nodes correspond to various infrastructure elements such as berths, locks, terminals, bridges, etc., as well as hydrometeorological sensors.

In Figure 21 are depicted some hydrometeo sensor nodes along with the properties of one of them. As discussed, properties include metadata such as sensor position, location code, sensor owner, sensor type, but also the latest measured value.

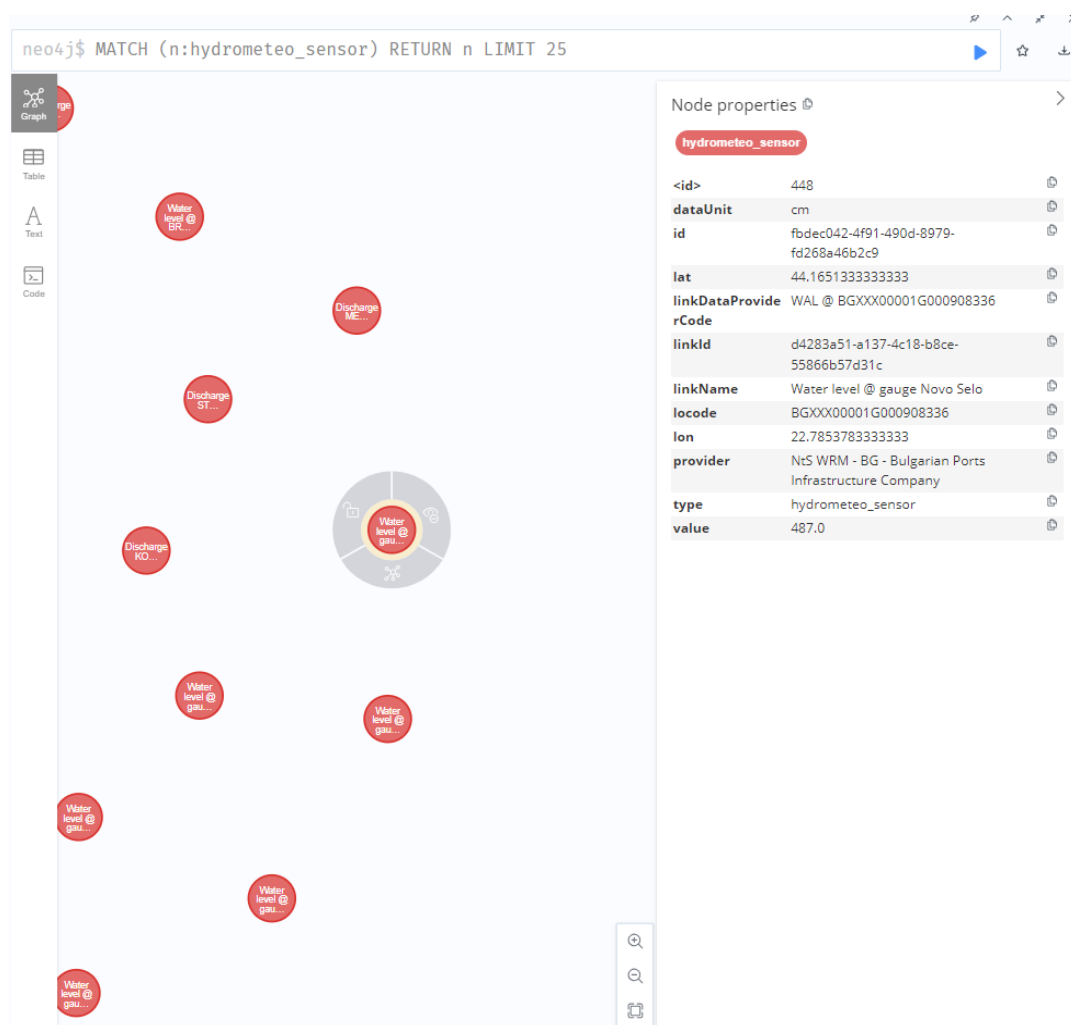


Figure 21 Hydrometeo sensor properties

In Figure 22 are depicted some vessel nodes along with the properties of one of them. The properties include metadata such as name, MMSI, call sign, dimensions. In the properties are also included the latest available operational values, such as position, speed over ground (SOG), etc.

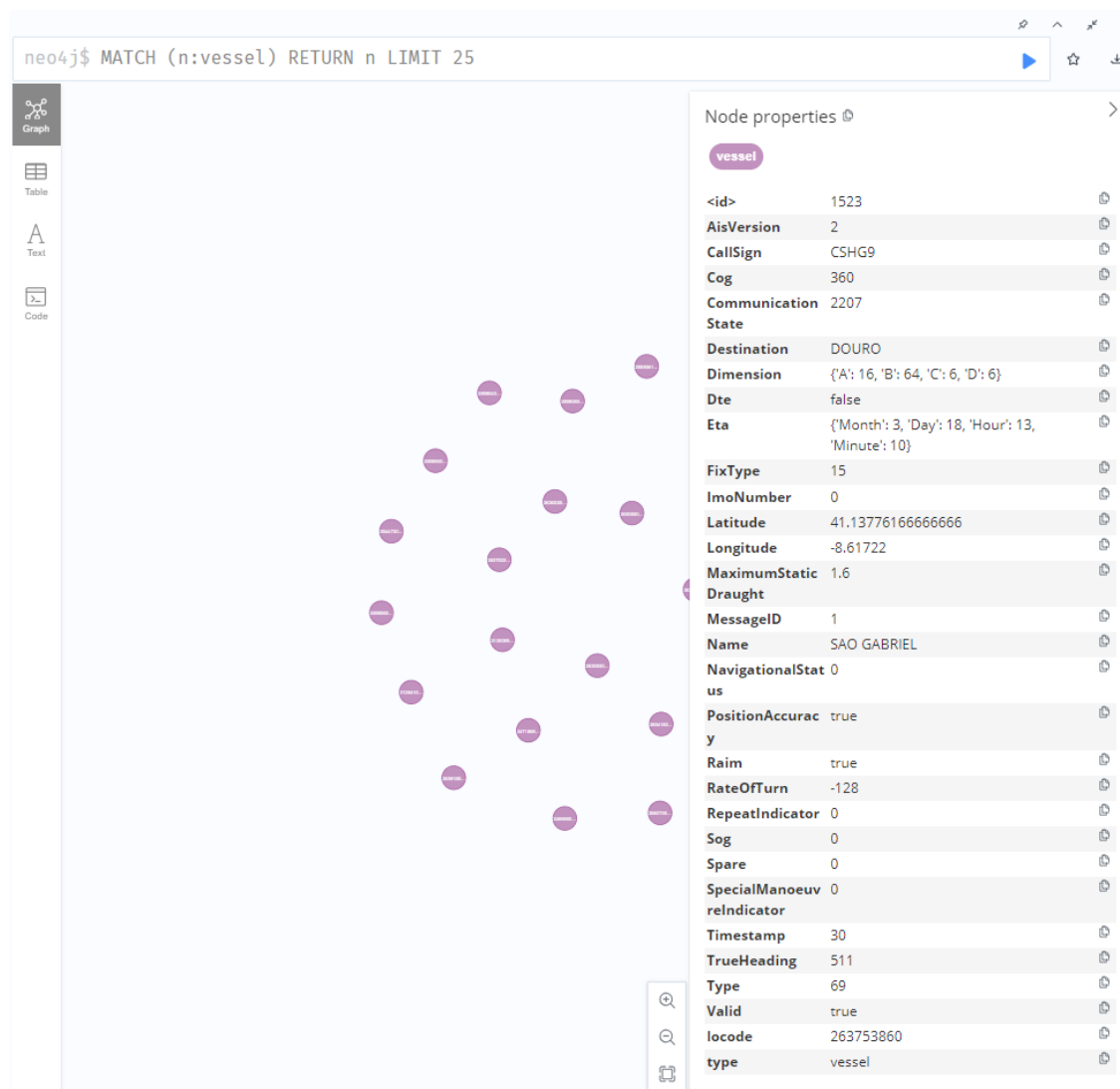


Figure 22 Vessel properties

7.4 Timeseries data storage

As described in §3.2, an Influx timeseries database is used to store all metrics ingested into the platform. Current, the available data are vessel metrics, as received by the AIS, and hydrometeo data as acquired by the EuRIS services. In Figure 23 is shown a dashboard created in the influxdb interface, showing data for specific vessel, identified by its CallSign. We see the latest value in gauge indicator for Speed Over Ground (SOG), Course Over Ground (COG), numeric indicators for Latitude, Longitude and Heading, and a chart for SOG.

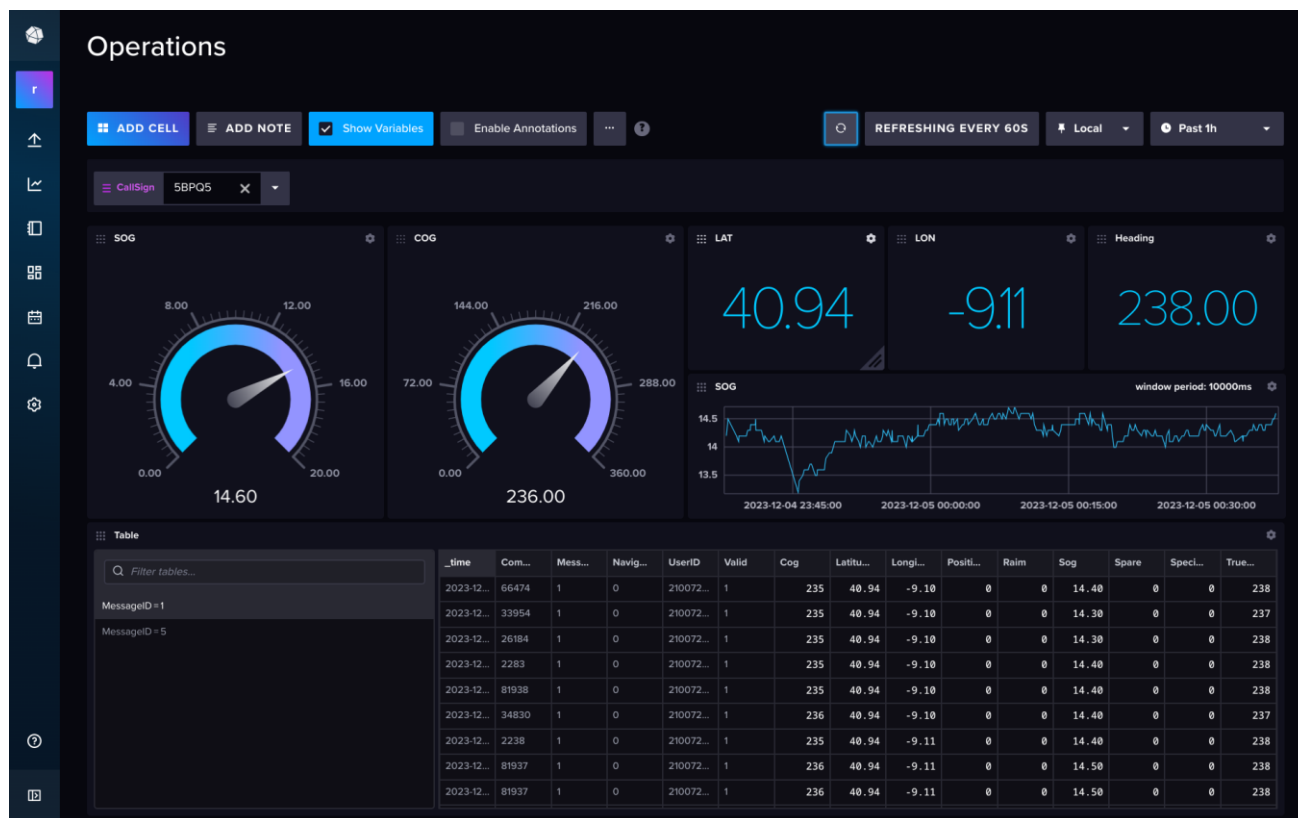


Figure 23 Display of vessel data trends in the InfluxDB interface

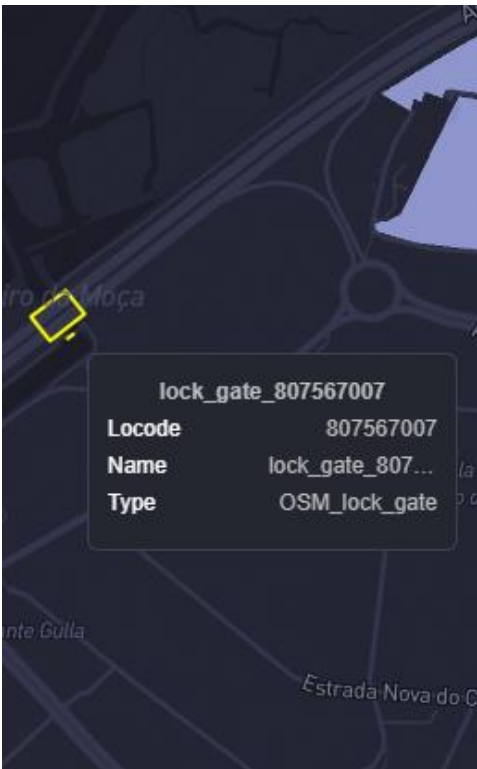
7.5 User interface to visualize available data

The user interface for visualizing waterway infrastructure and vessel data is developed using Angular, a powerful front-end framework, along with Mapbox consuming OpenStreetMaps tiles for mapping functionalities. Leveraging Mapbox's capabilities, the interface provides users with an intuitive and interactive experience for exploring waterway-related information. One notable feature is the clustering functionality, which dynamically groups entities when zooming out to provide a clearer and more manageable view of the data. This is a feature that is planned to be improved to visualize apart from infrastructural entities also traffic [21]. This approach enhances the user experience by presenting a comprehensive overview of the waterway system while maintaining performance and responsiveness. Through Angular's component-based architecture, the interface offers modular and reusable components, ensuring maintainability and scalability as the application evolves.

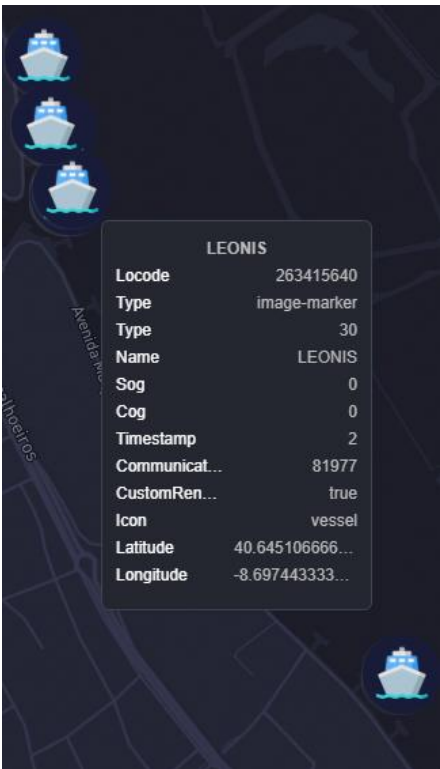
In Figure 24 are shown various aspects of the user interface. In (a) is shown the geographical distribution of the maintained nodes, the number of which is in the order of 25000. In (b), (c) and (d) are shown details of specific areas along with the properties of the respective objects.



(a)



(b)



(c)



(d)

Figure 24 Screenshots of the user interface, (a) overview of the available nodes, showing their geographical distribution, (b) a lock acquired from OSM, (c) vessels in Douro river and (d) bridges in Gent.

8 Conclusions

This report wraps up our work to create a Green Resilient Inland Water Transport (GRIWT) Dataspace and the corresponding Digital Twin components, aimed at enhancing the sustainability and resilience of inland waterways. Our approach focused on utilizing emerging technologies to improve the operations of inland water transport, making it more adaptable and automated.

We explored the IWT ecosystem's data dynamic, identifying key data sources like environmental sensors and geographic systems to feed into our digital twin's data pipeline. This integration forms the bedrock of our digital twin, enabling an enriched interaction with the IWT environment and facilitating a nuanced understanding of its complexities.

The architecture we've outlined in this report reflects our effort to combine various open-source components into a solid framework that acts as the backbone of the digital twin. This framework supports a range of functions critical for modern inland water transport operations, from monitoring to advanced modeling.

An important part of our approach was the development of a connector library to facilitate easy integration of data from various sources. This library is crucial for making the data more interpretable and usable, leading to improvements in monitoring and automation within the inland water transport domain.

The deployment of a prototype system highlighted the practical application of our digital twin framework, showing the potential benefits and efficiencies it can bring to the field. This prototype is a direct result of our methodical approach and commitment to delivering workable solutions.

In sum, this report not only meets its set goals but also contributes to the field of inland water transport. It showcases the potential of digital twin technology to make water transport operations more efficient and sustainable, laying the groundwork for future advancements in the management of inland waterways.

9 References

- [1] D3.1 Framework Architecture & core infrastructure components
- [2] DUET: Digital Urban European Twins | [Digitalurbantwins.com](https://digitalurbantwins.com)
- [3] DUET T-Cell Architecture for Interoperability | <https://www.digitalurbantwins.com/technical-approach>
- [4] DUET: A Framework for Building Secure and Trusted Digital Twins of Smart Cities | Lieven Raes, Philippe Michiels, Thomas Adolphi, Chris Tampère, Athanasios Dalianis, Susie Mcaleer, Pavel Kogut | https://www.researchgate.net/publication/349602444_DUET_A_Framework_for_Building_Secure_and_Trusted_Digital_Twins_of_Smart_Cities

- [5] Linked Data Event Streams (Living Standard) | Pieter Colpaert | <https://w3id.org/ldes/specification>
- [6] <https://kafka.apache.org/>
- [7] Shafer, Ilari & Sambasivan, Raja & Rowe, Anthony & Ganger, Gregory. (2013). Specialized storage for big numeric time series.
- [8] <https://www.influxdata.com/>
- [9] <https://zookeeper.apache.org/>
- [10] <https://www.mongodb.com/>
- [11] <https://postgis.net/>
- [12] <https://www.timescale.com/>
- [13] <https://www.ugent.be/en/research/explorer/eu-trackrecord/interreg/socorro.htm>
- [14] <https://www.eurisportal.eu/>
- [15] <https://risdefinitions.org/>
- [16] <https://www.eumonitor.eu/9353000/1/j9vvik7m1c3gyxp/vkin9vxe8iyv>
- [17] https://data.eurisportal.eu/doc/DataModel%20EuRIS%20-%20Explanation_v1.1.pdf
- [18] https://www.eurisportal.eu/doc/api/?urls.primaryName=ArcGIS_v1#/
- [19] Zwicklhuber, T., Kaufmann, M. (2023). EURIS (European River Information Services System) – The Central European RIS Platform. In: Li, Y., Hu, Y., Rigo, P., Lefler, F.E., Zhao, G. (eds) Proceedings of PIANC Smart Rivers 2022. PIANC 2022. Lecture Notes in Civil Engineering, vol 264. Springer, Singapore. https://doi.org/10.1007/978-981-19-6138-0_75
- [20] Veeckman, C., Jedlička, K., De Paepe, D. et al. Geodata interoperability and harmonization in transport: a case study of open transport net. Open geospatial data, softw. stand. 2, 3 (2017). <https://doi.org/10.1186/s40965-017-0015-6>
- [21] Jedlička K, Ježek J, Kepka M, Hájek P, Mildorf T, Kolovský F, Beran D. Dynamic Visualization of Volume of Traffic. In: Papers ICC 2015. Brazil: ICA; 2015. p. 1–13. available online at: <http://www.icc2015.org/abstract,536.html>. ISBN 978-85-88783-11-9.
- [22] INSPIRE TWG. Data Specification on Transport Networks – Technical Guidelines. 2014. INSPIRE Infrastructure for Spatial Information in Europe. On-line, http://inspire.ec.europa.eu/documents/Data_Specifications/INSPIRE_DataSpecification_TN_v3.2.pdf. Accessed 8 Feb 2017.
- [23] [https://www.imo.org/en/About/Conventions/Pages/International-Convention-for-the-Safety-of-Life-at-Sea-\(SOLAS\),-1974.aspx](https://www.imo.org/en/About/Conventions/Pages/International-Convention-for-the-Safety-of-Life-at-Sea-(SOLAS),-1974.aspx)
- [24] https://en.wikipedia.org/wiki/Automatic_Identification_System
- [25] http://www.navcen.uscg.gov/enav/AIS/AIS_carriage_reqmts.htm
- [26] <https://gpsd.gitlab.io/gpsd/AIVDM.html>
- [27] <http://www.unece.org/fileadmin/DAM/trans/doc/finaldocs/sc3/ECE-TRANS-SC3-176e.pdf>
- [28] <https://www.navcen.uscg.gov/sites/default/files/pdf/AIS/SOLAS.V.19.2.1-5.pdf>
- [29] <https://spire.com/wiki/ais-tracking-data-your-ultimate-guide/>
- [30] <https://spire.com/maritime/>
- [31] <https://www.orbcomm.com/en/solutions/maritime/ais-data>

- [32] <https://orbitsailor.com/>
- [33] <https://www.marinetraffic.com/blog/seven-things-you-should-consider-when-choosing-ais-provider/>