



Project Acronym: **SmartShip**

Project Full Title: **A data analytics, decision support and circular economy – based multi-layer optimization platform towards a holistic energy efficiency, fuel consumption and emissions management of vessels**

Project Duration: **60 months (01/04/2019 – 31/03/2024)**

DELIVERABLE 5.1 (final version):

Decision support module and multi-layer optimization tools and technologies

Work Package **WP5 – SmartShip Decision Support and multi-layer optimization module**

Task **T5.3- Design and Development of Data analytics and Decision Support module**

Document Status **“final v1.1”**

Due Date: **31.03.2023**

Submission Date: **19.04.2023**

Lead Beneficiary: **BlueSoft (BLS)**

Dissemination Level	
Public	X

Authors List

Leading Author				
First Name		Last Name	Beneficiary	Contact e-mail
Jakub		Rola	BLS	jakub.rola@bluesoft.com
Co-Author(s)				
#	First Name	Last Name	Beneficiary	Contact e-mail
1	Anna-Maria	Anaxagorou	ITML	aanaxagorou@itml.gr
2	Ioannis	Kontopoulos	HUA	kontopoulos@hua.gr
3	Antonios	Makris	HUA	amakris@hua.gr
4	Konstantinos	Tserpes	HUA	tserpes@hua.gr
3	Hernan	Ruiz Ocampo	CERC	h.ruizocampo@pontsbschool.com
4	Artemis	Flori	DANAOS	af.drc@danaos.gr
5	Dimitris	Panos	BLS	dimitris.panos@bluesoft.com
6	Rafal	Szustkiewicz	BLS	rafal.szustkiewicz@bluesoft.com

Reviewers List

Reviewers			
First Name	Last Name	Beneficiary	Contact e-mail
Marc	Bonazountas	EPSILON	bonazountas@epsilon.gr
Vassilis	Prevelakis	TUBS	prevelakis@ida.ing.tu-bs.de
Vlatka	Katusic	CERC	v.katusic@pontsbschool.com
Hernan	Ruiz Ocampo	CERC	h.ruizocampo@pontsbschool.com

Legal Disclaimer

SmartShip has received funding from the European Union's Horizon 2020 research and Innovation programme under the Marie Skłodowska-Curie grant agreement No 823916. The responsibility for the content of this publication lies with the authors. It does not necessarily reflect the opinion of the funding agencies or the European Commission (EC). Funding Agencies or the EC are not responsible for any use that may be made of the information contained therein.

Executive Summary

The objective of this document is to demonstrate the first version of SmartShip framework implementation, and show how the developing process has been managed. The focal point is demonstration of development process and the description of the component API's

Table of Contents

Executive Summary	3
1. Introduction	7
1.1 Scope and objectives of the deliverable	7
1.2 Structure of the deliverable	7
1.3 Relation to Other Tasks and Deliverables	7
2. Use cases and requirements	7
2.1 Use case 1: Weather routing optimization	7
2.1.1 Short description	7
2.1.2 Requirements	7
2.2 Use case 2: Route monitoring	8
2.2.1 Short description	8
2.2.2 Requirements	8
2.3 Use case 3: Condition based maintenance	8
2.3.1 Short description	8
2.3.2 Requirements	8
2.4 Use case 4: Visualization	9
2.4.1 Short description	9
2.4.2 Requirements	9
3. Maritime operation software	9
3.1 Overview	9
3.1.1 METIS	9
3.1.2 DEEPSEA	10
3.1.3 DANAOS SEAROUTES	14
3.1.4 DANAOS WAVES	14
3.1.5 DANAOS WAVES and DANAOS SEAROUTES	16
3.2 Summary	18
4. SmartShip Architecture - simplified	19
4.1 SmartShip components diagram	19
5. Developing process	20
5.1 Agile and scrum history	20
5.2 Scrum in the SmartShip development process	21
5.2.1 Roles	21
5.2.2 Events	21
5.2.3 Artefacts	22
5.2.4 Tools	22
6. SmartShip platform components	23
6.1 Data storing	23

6.1.1	API	23
6.1.2	Technological stack	24
6.1.3	Licences	24
6.2	Weather routing component	24
6.2.1	API	27
6.2.2	Technological stack	28
6.2.3	Licences	28
6.3	Predictive maintenance component	28
6.3.1	API	28
6.3.2	Technological stack	30
6.3.3	Licences	30
7.	Deployment and testing environment	30
7.1	Tools used for deployment process	30
7.1.1	Jenkins Pipelines	31
7.2	Server parameters	32
1.1.1	Hardware and Software specifications	32
8.	Conclusions	33
9.	References	33

List of Figures

Figure 1	METIS Data Flow [1]	10
Figure 2:	Cassandra's Notification dashboard	12
Figure 3:	Cassandra's Emission Tracking	12
Figure 4:	Weather optimal route vector	16
Figure 5:	Optimal route plotted by DANAOS tools	17
Figure 6:	Weather optimal route visualized by DANAOS tools	17
Figure 7:	SmartShip simplified component diagram	19
Figure 8:	SmartShip backlog	22
Figure 9:	SmartShip data component API	23
Figure 10:	Ship information data structure in SmartShip ream	24
Figure 11:	Weather routing algorithm	25
Figure 12:	Alternate route recommendation on route Gibraltar–Genoa	25
Figure 13:	Alternate route recommendation on route Gibraltar–Cagliari	26
Figure 14:	Alternate route recommendation on route Napoli–Amman	26
Figure 15:	Weather route optimization component endpoint	27
Figure 16:	Body response of weather route optimization module	28
Figure 17:	Predictive maintenance API	29
Figure 18:	Predictive maintenance response example	29

List of tables

Table 1	DeepSea solutions	12
Table 2	DANAOS WAVE data source	16
Table 3	Data base server specification	37
Table 4	Component server specification	37

List of Acronyms and Abbreviations

Term	Description
AIS	Automatic Identification System
AMS	Alarm Monitoring System
Aol	Area of Interest
CBM	Condition-based (predictive) maintenance
CE	Circular Economy
DHCP	Dynamic Host Configuration Protocol
ECR	Engine Control Room
ECDIS	Electronic Chart Display and Information System
ETA	Estimated Time of Arrival
ETSI	The European Telecommunications Standards Institute
FOC	Fuel Operational Consumption
GRE	Generic Routing Encapsulation
IoT	Internet of Things
IMO	International Maritime Organisation
ISG	Industry Specification Group
PCS	People-Centric Sensing
RDBMS	Remote onboard database Server
SOG	Speed Over Ground
STW	Speed Through Water
VDR	Voyage data recording
TCE	Time Charter Equivalent

1. Introduction

1.1 Scope and objectives of the deliverable

Deliverable D5.1 presents the SmartShip platform development process and its result. It contains a detailed analysis of the entire process that led to developing the components of the SmartShip.

1.2 Structure of the deliverable

This deliverable demonstrates the SmartShip platform, and keeps the following structure:

- **Section 2** quotes the use case description and requirements from deliverable D2.1.
- **Section 3** provides a short overview of market ready tools for vessel's management and route optimization.
- **Section 4** shows the interaction between the SmartShip platform components and briefly describes SmartShip architecture.
- **Section 5** describes the developing process of the components and SmartShip implementation of the SCRUM framework
- **Section 6** consists the illustration of the component
- **Section 7** describes the SmartShip DevOps process

1.3 Relation to Other Tasks and Deliverables

This deliverable expands upon already existing SmartShip work packages, WP2 Requirements elicitation, use case scenarios and roadmaps for integrated vessel management, and WP3 Smartship Circular-Economy based functional architecture design. The first part of this document presents the summary of WP2's deliverable D2.1 and WP3's deliverable D3.1. Work package 2, deliverable D2.1, identified the use cases and background for Smartship and work package 3 deliverable D3.1 defined the architecture and the framework in which SmartShip is operating.

2. Use cases and requirements

The following chapter consists of a short description of use cases analysed in deliverable D2.1 combined with the requirements the SmartShip platform should fulfill.

2.1 Use case 1: Weather routing optimization

2.1.1 Short description

This use case refers to the design and representation of the best-fit weather routing advice to the master on-board considering weather information along the plotted voyage plan and adapted to individual vessel characteristics and cargo specifications. Weather optimal routing is a multi-variable decision support mechanism against a bundle of objectives consisting of on-schedule arrival (passage time), charter party clauses compliance (fuel consumption, speed, allowance variations), fuel savings and energy efficiency, and cost savings and TCE earnings maximisation.

2.1.2 Requirements

#1.1 Multi-variable routing optimization algorithmic analysis adds new factors to existing considerations (weather conditions) based on information about navigational restrictions, notice to mariners, and other constraints.

#1.2 Benchmarking and normalisation of existing algorithmic-based weather routing optimization with common route patterns based on AIS data analysis (external reference) and own fleet historical navigational/operational data (internal reference).

2.2 Use case 2: Route monitoring

2.2.1 Short description

This use case liaises with the first one and addresses the continuous readjustments of routing advice along voyage execution. By default, the progress of the vessel in accordance with the voyage and passage plan is closely and continuously monitored, while any changes made to the plan are marked and recorded

2.2.2 Requirements

- #2.1 Ongoing monitoring of voyage performance.
- #2.2 Alerting mechanism and warnings to the master for deviations and possible voyage under-performance.
- #2.3 Risk assessment of master navigational decision along the route execution and cause analysis of any deviation from the system-generated optimal route advice.
- #2.4 Dynamic voyage performance comparison, triggered by the user anytime along the voyage, between system route advice and master course plotting.

2.3 Use case 3: Condition based maintenance

2.3.1 Short description

This use case will capitalise on the technology-driven fleet performance monitoring framework of DANAOS. The company handles operational efficiency optimization as a continuous descriptive analysis of historical information (hindsight) to come up with insights and assessment of the reasoning behind what happened in the past (diagnostic analytics). The reason is to find a pattern of prediction of what will happen in the future (predictive analytics) and plan the right strategy to make it happen or prevent it from not happening (prescriptive analytics). The company has developed an intelligence platform (DANAOS fleet performance monitoring platform) to retrieve data from different sources capitalising on Internet of Things philosophy (IoT) while performing artificial intelligence (AI) and machine learning techniques.

2.3.2 Requirements

- #3.1 Real-time key machinery monitoring.
- #3.2 User-defined configuration of functions in data processing.
- #3.3 Multi-index data frame time-series generation and routine plotting for functional performance of vessel components.
- #3.4 Alert mechanism for error/anomaly detection and failure prediction.
- #3.5 Performance report of machinery/equipment for assistance in decision-making for an efficient maintenance plan and spare parts – consumables procurement plan.

2.4 Use case 4: Visualization

2.4.1 Short description

Use case 4 efficiently illustrates and represents the aforementioned use cases. The main concept is that alternative routes vessels must follow, the route deviations that occurred, and the weather conditions in each case must be presented to the officers on board the vessels in a user-friendly way.

2.4.2 Requirements

#4.1 Fully interactive environment.

#4.2 Intuitive menu.

#4.3 Friendly to user navigation.

#4.4 Representation of information in user defined and multi-format building Dashboard layouts.

3. Maritime operation software

Every software development process starts with examining the currently available software that provides similar capabilities. Following sections describe in the short form the some market available software.

3.1 Overview

Each section should briefly describe the product's launch, market share, industry significance, collaboration with academia, other essential features, etc.

3.1.1 METIS

Maritime companies are lacking the ability to monitor and therefore analyse basic real-time data of their fleet performance [1] and nowadays, recognize the necessity of installing and Operating Ship Performance Monitoring System to be able to have accurate and reliable, real-time information about their fleet navigational performance, fuel oil consumption, etc.

Generally, companies confront the «making or buy » dilemma. METIS white paper (2018) provides an answer relating to the internal development requirements which include 10 primary requirement concerns a) human resources, b) technology, c) technology adoption, d) hardware infrastructure, e) software development tools, f) versioning, g) bugs fixing h) hosting i) support j) certifications (ISO 9001, ISO 20000, ISO 27001) and h) Marine Type Approval.

METIS Cyberspace Technology SA is an innovative company specializing in IoT and Cloud Computing adapted to the maritime sector. The platform is vessel centric; it uses Artificial Intelligence (AI) and virtual assistants. The company source of data can be:

- Manual inputs by the personnel
- External information providers (for example, weather data)
- Real-time monitoring systems

METIS is an easy to use system. It ensures maximum efficiency of engines, optimising routes, and minimising fuel consumption.

METIS agents communicate with the personnel of the maritime company and the crew through the collaboration platform used in each Maritime company, and interact with the users through the same channels, like virtual colleague assistants, exchanging text messages in plain English. METIS Agents

are adaptable to the procedures of each company. The role of the METIS platform is to support the personnel in their scope of work. Figure 1 depicts METIS data flow.

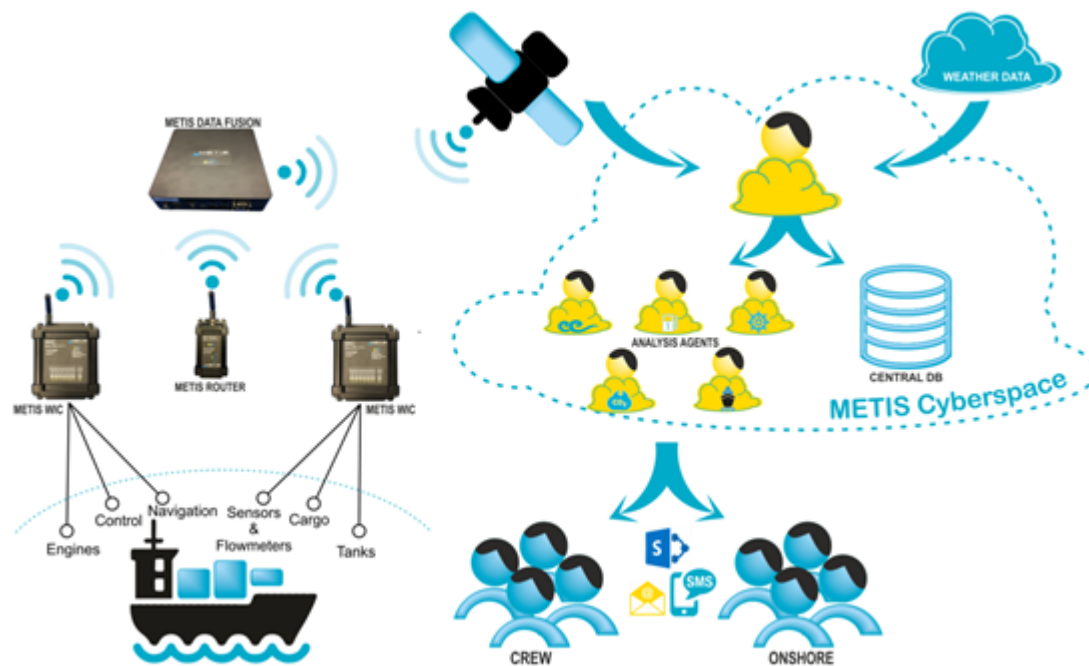


Figure 1 METIS Data Flow [1]

METIS is an aggregation of smart devices and cloud-based microservices. The smart devices GET, FETCH, and CLEAN the data from any equipment, instrument, and sensor on board the vessels. Then transmits the data to the METIS Cyberspace, where the microservices implement all the functionality of the Analysis Agents and the interoperability with third-party systems to get additional data or share the analysis results. The users interact with the system through well-established channels like emails, Skype, Yammer, SharePoint, Teams, and other collaboration tools.

METIS encapsulates three major services to maritime companies, Data Acquisition, Qualitative Analysis and Quantitative Information for further investigation on Critical aspects.

Data acquisition

GET – FETCH – CLEAN and STORE any measurements directly from the on board equipment, sensors, and crew. METIS is designed according to maritime standards and fulfills any issues for data acquisition on board the vessels. METIS has Wireless Networks and Smart Devices such as METIS WIC (Wireless Intelligent Collector), a smart device.

3.1.2 DEEPSEA

Shipping companies nowadays recognize the necessity of installing and operating Ship Performance Monitoring systems. They need accurate and reliable, real-time information about their fleet navigational performance, fuel oil consumption, etc. We are experiencing worldwide an increase in the number of maritime companies with forward-thinking managerial teams eager to adopt value-adding intelligent IT systems. This trend is led by their awareness that the only way to be competitive is to be technologically ahead. We realise that companies unwilling to adopt this approach risk falling far behind or being left by the wayside. Maritime companies often face a fundamental dilemma: *“Is it advisable to DEVELOP our own system internally, or can we fulfil our requirements by selecting one of the solutions available on the market?”*

As shipping faces the dual challenge and opportunity to decarbonise and digitalise simultaneously, DeepSea's technology enables its clients to make data-driven decisions to lead the industry forward.

DeepSea collects real-time information on a vessel and compares that across fleets. Furthermore, DeepSea can optimise ship performance and reduce the maritime industry's environmental impact. The elements that DeepSea is focusing on and implementing are described below:

TAILOR-MADE FLEET OPTIMISATION SOLUTION

- MONITOR
 - Connect with your vessels in real-time to make correct decisions.
- EVALUATE
 - Complex data, simple insights: Deep learning models draw meaning from ship data in real-time. The result is powerful, actionable insights.
 - Analyse Hull Fouling trends by creating a model of your specific vessel, and harnessing DeepSea massive database, you can visualise the effect of hull fouling on vessel fuel consumption.
- CONTROL
 - Detect Anomalies: Critical values, such as fuel overconsumption, are closely monitored. Alerts are generated so you can know when something unusual happens.
 - Power balancing: Increase energy efficiency, and save fuel, by intelligently power balancing your generator engines and boilers.
 - Empower your team on the front line: Optimised, transparent vessels mean optimised, transparent relationships.

Table 1 DeepSea solutions

	Functions
Technical quality assurance	<ul style="list-style-type: none"> ● Real-time hull and key machinery monitoring ● Unjustified fuel consumption detection (99.6% prediction accuracy)
Environmental compliance	<ul style="list-style-type: none"> ● Real-time fuel type detection ● Fuel change on ECA Zones/Areas
Decision making support	<ul style="list-style-type: none"> ● Hull cleaning simulation ● Anomaly detection
Charter party monitoring	<ul style="list-style-type: none"> ● Performance violations ● Claims assistance

Cassandra is one of the decarbonisation solutions that DeepSea provides as a market-ready tool for optimisation and real-time monitoring. Specifically, it is a vessel monitoring and decision-making assistant that utilises state-of-the-art Artificial Intelligence algorithms. It aims to monitor each vessel's hull and key machinery in your fleet in real-time and around the clock. It offers you peace of mind and enables better decision-making through evidence-based insights.

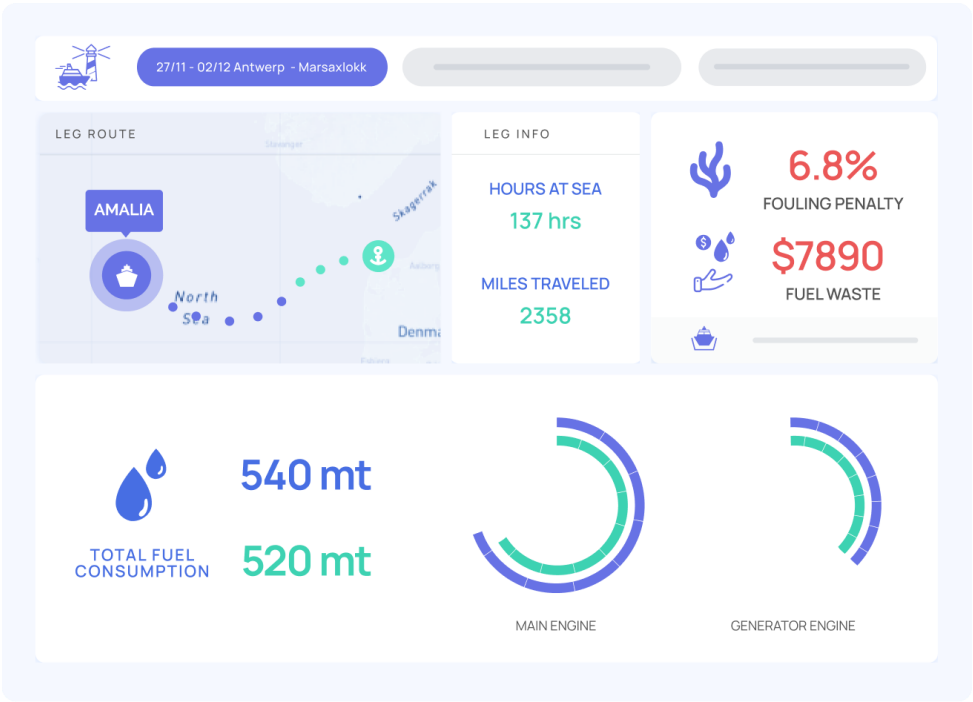


Figure 2: Cassandra's Notification dashboard



Figure 3: Cassandra's Emission Tracking

Deepsea Platform

Deepsea platform changes the typical data-flow in Maritime Companies. Deepsea secures the reliability of data and the direct access of proper stakeholders at the right time, based on their responsibilities and authorization levels, avoiding delays and interference.

Artificial Intelligence helps systems establish direct communication with people, in a natural, human-like way. Deepsea Platform created virtual assistants which take over specific, complicated, and time-consuming tasks, optimizing and simplifying the operations at a minimum cost. Deepsea, provides a collaboration platform between machines and humans aiming to optimize work procedures.

The data sources can derive from the following:

- Manual inputs by the personnel
- External information providers (for example, Weather data)
- Real-time monitoring systems

Additionally, Deepsea provides the solution for real-time monitoring based on a wireless network of smart collectors to secure the reliability of the data and the proper information for a sustainable and powerful analysis system. Deepsea Agents communicate with the personnel of the Maritime Company and the crew through the collaboration platform used in each Maritime company. The system interacts with the users through the same channels, like virtual colleagues-assistants, exchanging messages, watching, alerting, answering questions, and responding to requests.

Deepsea Data Flow

Deepsea is an aggregation of smart devices and cloud based microservices. The smart devices GET, FETCH and CLEAN the data from any equipment, instrument and sensor on board the vessels and transmit them to the Deepsea Cyberspace where the microservices implement all the functionality of the Analysis Agents and the interoperability with third party systems to get additional data or share the results of the analysis. The users interact with the system through well-established channels like emails, Skype and other collaboration tools.

Deepsea provides three main services to the Maritime Company, the Data Acquisition, the Qualitative Analysis and the Quantitative Information for further investigation on critical aspects.

Data Acquisition

The scope of the Data Acquisition system is to GET, FETCH, CLEAN and STORE any measurements directly from the onboard equipment and sensors and even from the crew.

Get data from any equipment, sensor, and instrument.

- Fetch data with different formats, modulation, accuracy, and time stamp.
- Keep only the valuable data.
- Store and secure them for a very long period.

Deepsea is designed according to the Maritime standards and fulfills any issues for data acquisition on board the vessels like:

- Diversity of equipment makers and data sources
- Significant Low Installation Cost and Time
- Cyber threats
- High demands in data storage capacity
- Maintainability of monitoring equipment

The current level of the equipment interconnection on board is very low. A major issue for any Analysis system is the Information Fusion in a system with many devices with heterogeneous interfaces. Deepsea implements a solution to this problem based on Wireless Networks and Smart Devices. Deepsea uses a smart device with enough processing power to collect and analyse analog and digital signals and transmits the results through a robust mesh wireless network.

Above description based on the DeepSea web page [2]

3.1.3 DANAOS SEAROUTES

DANAOS SEAROUTES is a Decision Support System that produces optimal route planning based on vessel location, weather information and destination optimising bunker consumption and passage time according to the market condition.

The tool is a centralised solution, gathering all necessary data in a central location, analysing the data and producing a detailed optimal route planning that is sent to both the vessel and the vessel's managing company for further action.

Since the required data have a small footprint, and all processing takes place remotely, there is no need to install special equipment and data-gathering devices in each vessel.

The system calculates the optimal route based on detailed mathematical calculations performed at a dense array of sequential nodes placed on the course (every 5-10 miles).

Data Acquisition

Searoutes uses, as its primary source of input, an automated email, sent daily from the vessel to the central system, that contains the following information:

- Current Position
- Time of Positioning
- Destination point / port
- Vessel information (optional).

Additionally, SeaRoutes receives, on a daily basis, weather information from the meteorological and global forecasting centre of the National Oceanic and Atmospheric Administration (NOAA). The data collected contain the wave spectrum based on the wave forecasting model (WAM) that is produced on a daily basis and contains wind, wave, and air movement spectrum data.

3.1.4 DANAOS WAVES

DANAOS WAVES® is a maritime Fleet Performance System and maritime data analytics platform that aggregates and analyses data gathered from multiple sources with the ulterior view of creating a comprehensive and analytical ships management platform.

The platform utilises a many devices, systems and data acquisition techniques, integrates them using several different quality and integration methods, performs analysis based on these data and produces route planning, reports and fleet performance information.

The system integrates all of the above information from heterogeneous sources of information. It combines them to generate graphical reports, fill dashboards, produce maps and comprehensive information regarding the entire fleet of a maritime operator. The information produced can be used for fleet awareness, real-time monitoring, optimal route planning, benchmarking, risk assessment and executive information reporting. All data is gathered and maintained centrally in the vessel's managing company headquarters.

Data Acquisition

The WAVES platform uses several types of data acquired from various sources to integrate them and provide a single integrated point of information. The data received and utilized by WAVES can be broken down into the following four sections.

- Vessel Data

Data is gathered from all sorts of devices, sensors, and systems, already in-place for each vessel. Most of the globally leading systems are supported like LAROS, MARORKA, ENRIRAM, etc. Additionally, data is gathered from various ship sensors like engine RPM, engine output power, fuel flowmeter, fuel operating temperatures, etc.

- Structured Data

The information is gathered internally from the company's operations, human resources, and financial departments. The data include financial reports, cost of systems, human resource availability and cost, operational data, and fuel cost and consumption

- Semi-structured Data

The platform utilises information in various internal and external documents in well-known document formats like PDF, XLS, DOC, etc. These documents include but are not limited to fuel analysis forms, port calls, international safety notices, etc.

- Unstructured Data

The platform gathers data from system logs, device generated reports and other sources of information produced within the organisation.

The following table indicates most sources of information utilised by the platform.

Table 2 DANAOS WAVE data source

Source	Type of Information	Information	Sampling Period
Vessel	System	DANAOS tools LAROS MARORKA ENRIRAM	Depending on each system
	Sensor/Device	Engine Data Fuel Data Destination Data Vessel Data	Depending on each sensor
Organisation	Structured Data	HR DATA FINANCIAL DATA OPERATION COSTS	Daily
	Semi-Structured Data	Documents Reports	Daily
	Unstructured Data	Logs Reports	On demand

3.1.5 DANAOS WAVES and DANAOS SEAROUTES

The following section presents the solution present in both software tools developed by DANAOS

Weather route optimization

Both SEAROUTES and WAVES utilise the WAM model of wave height, airspeed, and wind direction. The weather information is received daily from NOAA's meteorological and global forecasting centre. The information received contains:

- a. The air velocity field regarding the whole planet.
- b. The sea wave field worldwide.

The sea wave is not described as a single harmonic wave having a certain height, direction, and period. Instead, the wave is described with a spectrum at each sea position and time. This spectrum represents the distribution of wave energy (alternatively of wave height) over wave directions and frequencies at a fixed location and time.

The tool, to reduce the problem's dimensionality and efficiently use the wave forecasting for the optimal trajectory algorithm, applies a certain simplification and transformation to the wave spectrum.

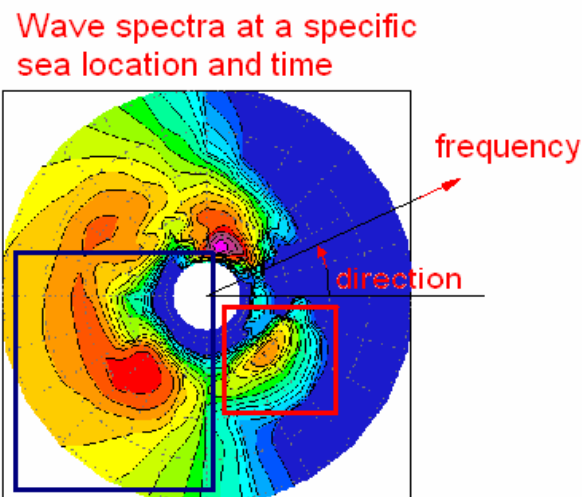


Figure 4: Weather optimal route vector

Figure 4 shows the transformation of the smooth function to a function that consists of two distinct Dirac functions, representing the wind wave and the swell wave, respectively. The red and blue squares surround these two different waves. Thus, each one is characterised by a specific direction/period/height. In physical meaning, the wind wave is driven directly from the wind, so its direction is almost parallel to it. After the wind waves leave the wind field that generates them, they travel far away as swell waves, even in low wind fields. So both kinds of waves often coexist in a particular location/time.

Route Visualization

Figure 5 illustrates an optimal route plotting on the map used by both SEAROUTES and WAVES.

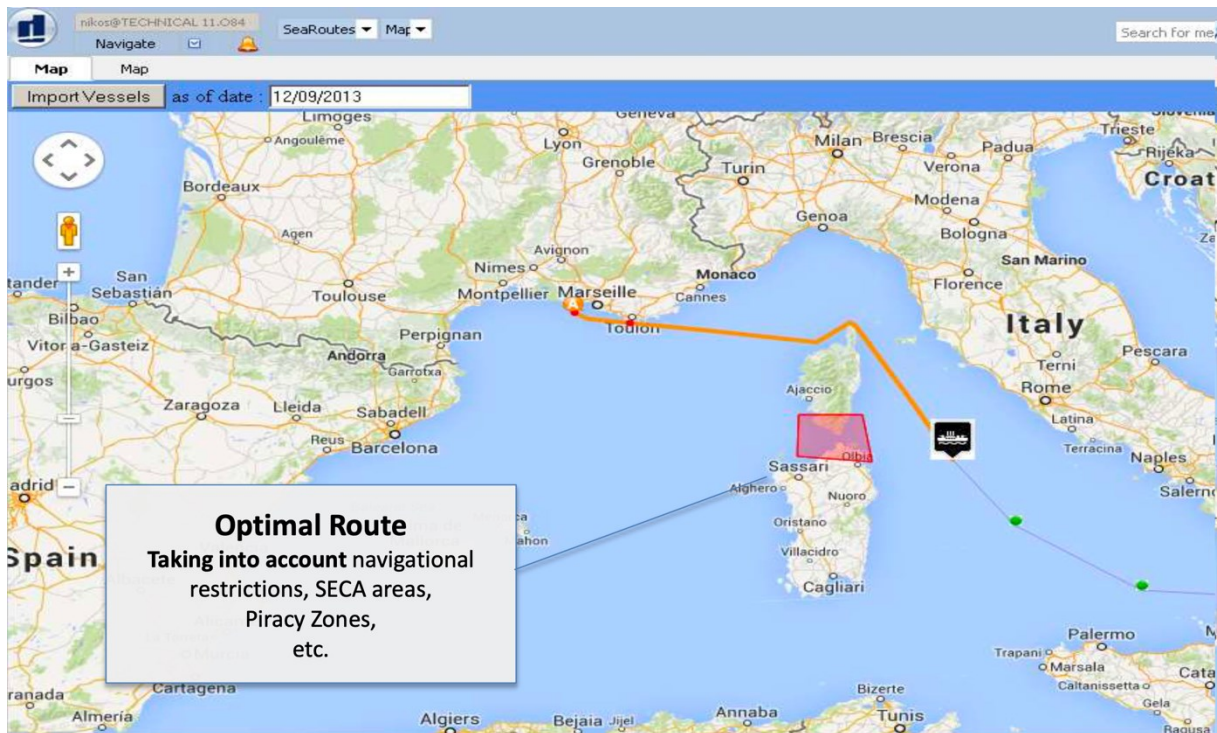


Figure 5: Optimal route plotted by DANAOS tools

Figure 6 depicts optimal route planning taking into consideration the current weather situation.

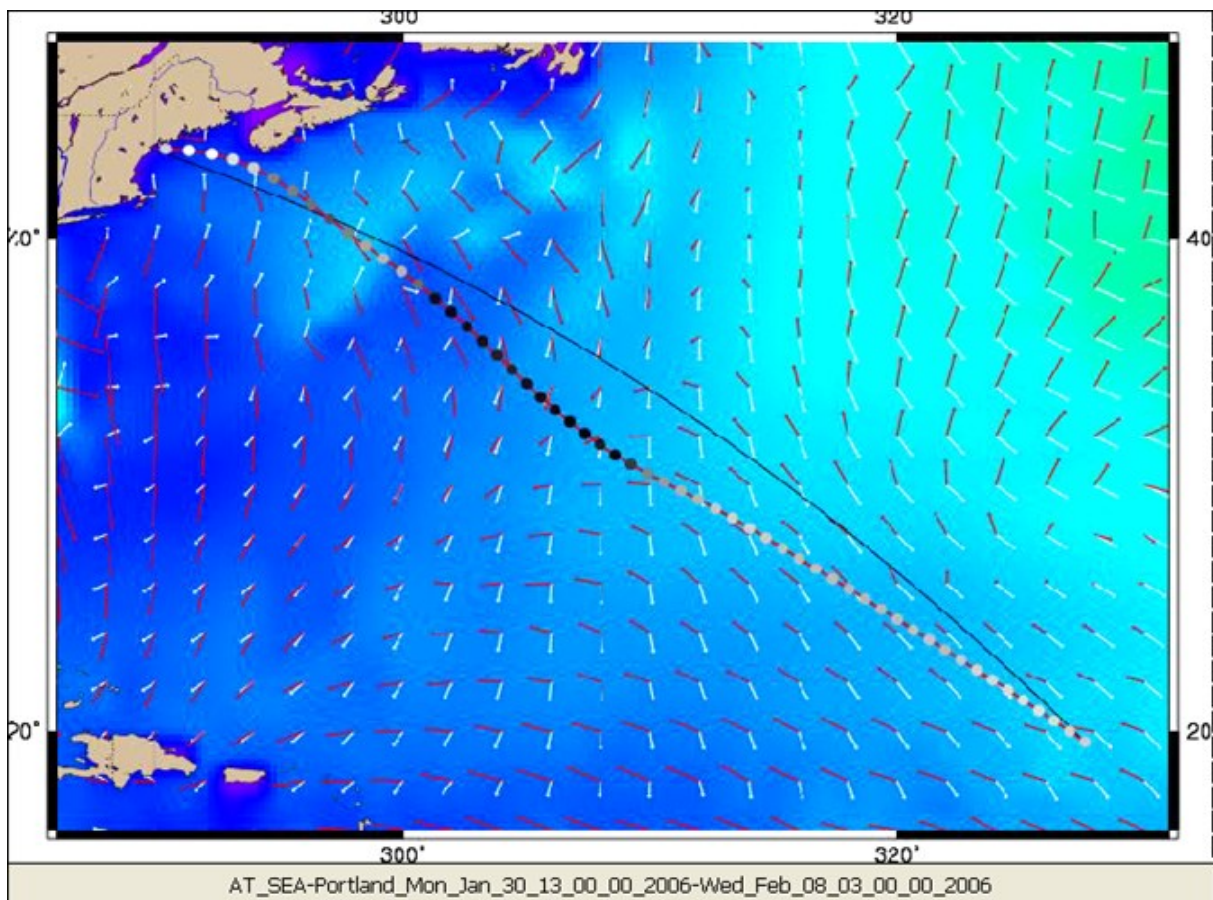


Figure 6: Weather optimal route visualized by DANAOS tools

3.2 Summary

The working hypothesis is that many of the aforementioned functions are already developed, and in the scope of this RISE project, there is no space to develop such a complex platform for the maritime industry. Therefore we should add only these identified components that can provide additional functions for the sector.

4. SmartShip Architecture - simplified

Microservices architecture is an approach to software development that involves breaking down large monolithic applications into smaller, independent services. Each service in a microservices architecture is designed to perform a specific business function and communicates with other services through well-defined APIs.

A key benefit of microservices architecture is its ability to scale and deploy services independently of each other, which can result in increased agility, faster time-to-market, and improved fault tolerance. It also enables developers to work on smaller, more manageable codebases, making it easier to maintain and update applications. However, microservices architecture also has some disadvantages. One of the biggest challenges is the increased complexity of managing and coordinating multiple services, as well as the overhead of managing the infrastructure required to deploy and operate these services. Additionally, testing, debugging, and monitoring of microservices can be more difficult due to the distributed nature of the architecture.

SmartShip develops and delivers a holistic framework for energy efficiency and emissions control incorporating technological advancements, information harvesting, short/mid-term decision support tools, and human intellect, thus materialising the next-generation paradigm for the maritime industry. SmartShip is based upon microservice architectural design comprising concrete components to achieve state-of-the-art deployment, which is detailed described in deliverable D3.1.

Next section below shows only interaction between components which are described in chapter 6 of this document.

4.1 SmartShip components diagram

This section shows the interaction between the components described in chapter 6 of this document. Figure 7 shows the simplified component diagram. It presents the interaction between the components. More information about component API can be found in chapter 6 of this document.

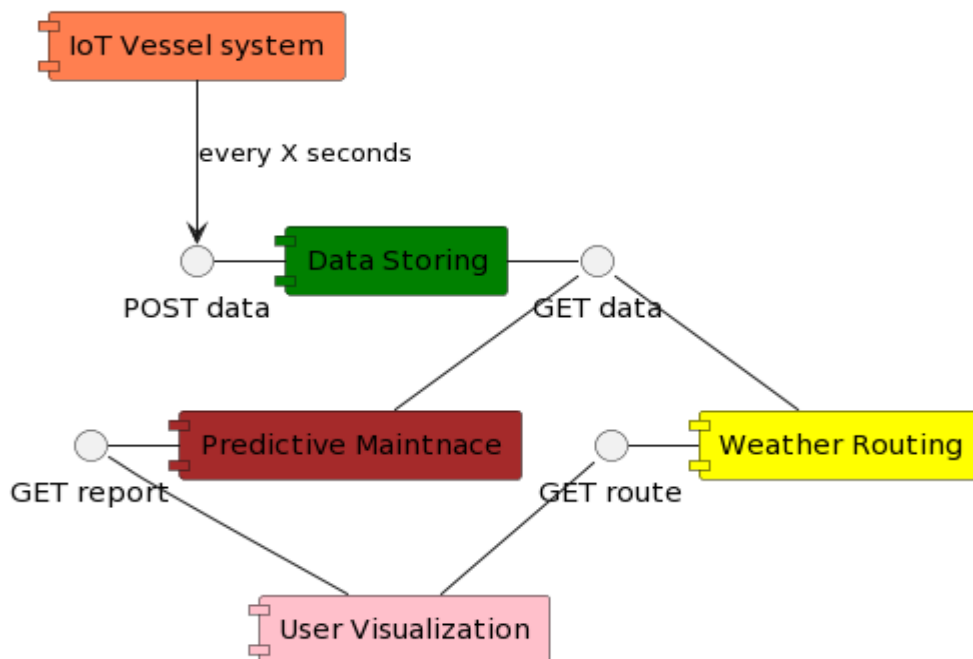


Figure 7: SmartShip simplified component diagram

5. Developing process

5.1 Agile and scrum history

In February 2001 during the Snowbird meeting in Wasatch the agile manifesto (<https://agilemanifesto.org/>) was created. It contained the twelve rules of the agile management process which were distilled from management frameworks like SCRUM, DSDM Adaptive Software Development, Crystal, etc. Twelve rules of agile approach are:

- Our highest priority is to satisfy the customer through early and continuous delivery of valuable software
- Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.
- Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
- Business people and developers must work together daily throughout the project.
- Build projects around motivated individuals. Give them the environment and support they need and trust them to get the job done.
- The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.
- Working software is the primary measure of progress.
- Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
- Continuous attention to technical excellence and good design enhances agility
- Simplicity – the art of maximising the amount of work not done – is essential.
- The best architectures, requirements, and designs emerge from self-organising teams.
- At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behaviour accordingly.

The most popular agile framework is SCRUM, it is based on the five values:

- Commitment
- Focus
- Openness
- Respect
- Courage

It was founded on empiricism and lean thinking. Scrum employs an iterative, incremental approach to optimise predictability and to control risk. Scrum engages groups of people who collectively have all the skills and expertise to do the work and share or acquire such skills as needed.

People who are working according to the SCRUM framework compose the SCRUM team, and have one of the roles:

- Developer
- Product Owner
- Scrum Master

The group of developers constitutes the developers' team who are responsible for delivering the product. The Product owner fully understands of product requirements, technical challenges and the product marketplace. Scrum Master is the guardian of the SCRUM methodology in the Scrum Team.

While working in Scrum, there are five periodical Scrum Events:

- The sprint
- Sprint planning
- Daily
- Sprint review
- Sprint retrospective

Scrum Artefacts:

- Product Backlog
- Sprint planning
- Increment

5.2 Scrum in the SmartShip development process

The nature of the RISE project led the consortium to choose the agile approach to the development process. We have decided to base our component development on the Scrum framework and adapt it to our needs. The section below describes our approach to Scrum and how we worked with this framework while developing SmartShip's components.

5.2.1 Roles

Due to the nature of SmartShip, which involves many researchers and knowledge exchange, we have decided not to assign each Scrum role to one person but to one of the partners. Each assigned partner was responsible for taking on the responsibilities of a given role but also being open to engaging any willing researcher.

Role of Scrum Master was taken by DANAOS as it is the leader of the SmartShip project and was already leading the regular meetings.

The Product Owner is one of the researchers from Harokopio University.

The developers team consisted of researchers who were on their secondments.

5.2.2 Events

The sprint planning, sprint retrospective, and sprint review have been conducted only during the consortium's physical and virtual meetings. The work progress was documented in the working stages of deliverables and presented on the forum. The development plan was mainly taken from the proposal, and the priorities and work orders have been established smoothly.

The acceptance of work progress was done collectively by the consortium members, with the veto power given to the tech leader.

Sprints were from two to three mounts, much longer than have been written in the Scrum Guide, but as said before, the nature of the RISE project would not allow the strict following of the framework.

Daily meetings took place every week during the intensive working periods and every month during the rest of the developing months.

5.2.3 Artefacts

The product backlog contains the list of new features, bug reports, enhancements, DevOps jobs, or work requirements needed to build a product. It is constantly changing by all members of the SCRUM team. In the SmartShip project, we have decided to group the items of backlog into batches corresponding to the SmartShip platform's components.

Sprint backlog during the SmartShip project was adjusted to the current capacity of development teams and the current state of the secondments. Many tasks have been active during more than one sprint, but we have tried to bring some value added after each sprint.

Product increment was discussed in our regular online meetings, and we weren't constant with the out of each sprint due to the oscillation in the development team quantity and capabilities

5.2.4 Tools

For the task progress, we used the simple Google sheet where we put the task and subtasks in rows, and we tracked the assignment and status of it. Figure 8, below, presents a sample of it.

Number	Name	Status	Assigned to
1	DevOps	In progress	Dannaos
1.1	Preparation of the server	Done	Danaos
1.1.1	Get hardware requirements	Done	Danaos
1.1.2	Get Software requirements	Done	Danaos
1.1.3	Set up the mashin and install OS	Done	Danaos
1.1.4	Propagate the credential to partners	Done	Danaos
1.2	Install DevOps tools on mashine	Done	BlueSoft
1.2.1	Install Jenkins	Done	BlueSoft
1.2.2	Create the technical account on the code repository	Done	BlueSoft
1.2.3	Create the simple Jenkins pipeline for component deployment	Done	BlueSoft
2	Data storing component	In progress	ITML
2.1	Gathering requirements	Done	ITML
2.1.1	Choose type of database	Done	ITML
2.1.2	Choose the technological stack	Done	ITML
2.2	Establishing data model	Done	ITML
2.2.1	Ship info data model	Done	ITML
2.2.2	Ship data model	Done	ITML

Figure 8: SmartShip backlog

6. SmartShip platform components

The following chapter describes SmartShip components. Each starts with a short description, then shows its interface, and ends with a list of licenses .

6.1 Data storing

Main role of the data storing component is storing and exposing data from IoT systems located on the vessels. Their functionality is basic and acts more or less like a data lake. Taking in consideration the prototype maturity of this component, a basic authorization method [3] is used in this component.

6.1.1 API

API definition can be found on GitHub [4]

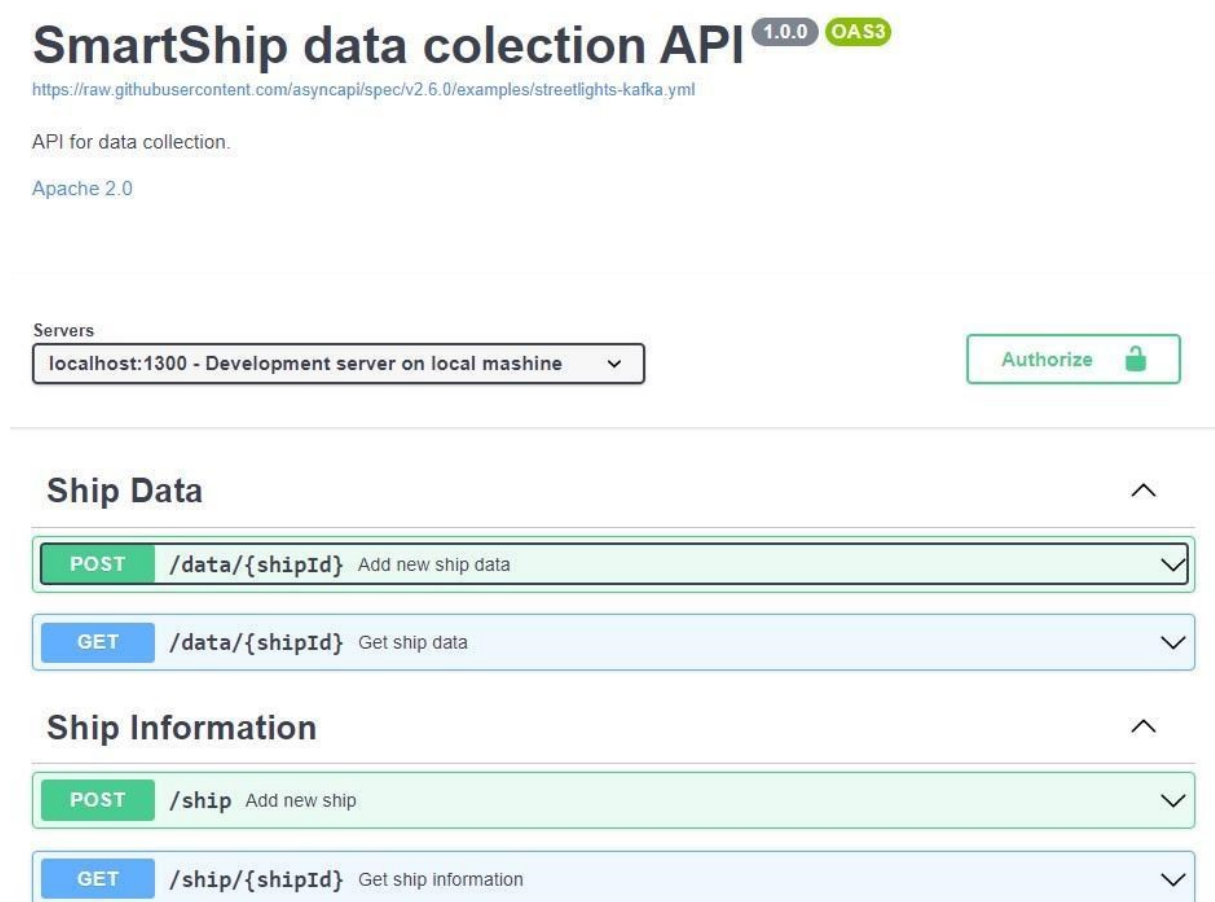


Figure 9: SmartShip data component API

As shown on figure 9, data storing modules expose four endpoints grouped into two tags, “Ship Information” allows adding a new ship to the module (POST request) and get basic information from GET request. Figure 10 shows data structure of Ship Information.

```

"shipInfo": {
  "id": "3fa85f64-5717-4562-b3fc-2c963f66afa6",
  "name": "Very Smart Ship",
  "ownerId": "3fa85f64-5717-4562-b3fc-2c963f66afa6"
},
"metadata": {
  "createdAt": "2023-03-09T09:26:32.782Z",
  "createdBy": "admin@smartship.eu",
  "modifiedAt": "2023-03-09T09:26:32.782Z",
  "modifiedBy": "user@smartship.eu"
}

```

Figure 10: Ship information data structure in SmartShip realm

Property shipInfo.id is assigned by the data storing component and is an identifier of a given ship in all of the components developed in the scope of the SmartShip project.

6.1.2 Technological stack

Data base : MongoDB

It allows for more flexibility in the data structure - some ships can have different sets of data generate by IoT sensors

WebServer: Amazon Corrento JDK 17, Spring Boot 2.7.1

6.1.3 Licences

MongoDB: Server Side Public License (SSPL) v1 [5]

Java: GNU Public License version 2

Spring Boot: Apache 2.0 [6]

6.2 Weather routing component

To calculate an alternate route option based on the adverse weather conditions as defined in ship details, available in storage component , and given an initial departure/destination point, and departure time, the set of centroids along that route is iterated. The following steps are made during the weather routing process and are depicted in Figure 11:

1. Based on the current timestamp, the mean speed provided by the current centroid of the route, and the Haversine distance between the current centroid and the next one along that route, the timestamp of the next centroid is calculated (line 4).
2. The position of the next centroid is used to find the cell of the weather grid at which the vessel will be located.
3. Given the timestamp and the grid cell of the next centroid, the weather conditions from the Copernicus services are retrieved (line 5).
4. If the weather conditions are adverse, the nearest cell of the weather grid in the space-time that satisfies the weather thresholds is found (line 7). The centroid of the weather grid acts as the proposed alternative for the vessel to move to (line 8). If the weather conditions are not adverse, the initial centroid is used for the route (line 10). To speed up the process, the search space is limited to the distance the vessel would have travelled, given the mean speed of the centroid and the three-hour time horizon. To find the nearest cell, the nearest neighbour algorithm is employed, which finds the nearest cell in relation to distance from the route centroid using spatial indexing (R-tree). R-trees allow indexing data values, which are defined in two (or more) dimensions. The search strategy uses the best-first-search to query the index, where the values are provided in order of the increasing distance. We limited the search to a maximum of 10 nearest points.

5. Finally, the next centroid of the initial route is considered the current one and the entire process repeats itself until there are no more centroids along the route.

Input: A set of centroids along the initial route R , Weather Thresholds WT

Output: A set of centroids along the optimized route R_o

```

1: function ROUTEOPTIMIZATION( $R$ [])
2:    $R_o \leftarrow []$ 
3:   for  $i \leftarrow 1$  to  $\text{length}(R)$  do
4:      $t(c_{i+1}) \leftarrow \text{distance}(c_i, c_{i+1}) / v(c_i)$ 
5:      $w \leftarrow \text{getWeatherConditions}(t(c_{i+1}), \text{cell}(c_{i+1}))$ 
6:     if  $w$  exceeds  $WT$  then
7:        $\text{cellCentroid} \leftarrow \text{getNearestNeighbor}(\text{cell}(c_{i+1}))$ 
8:        $R_o.\text{append}(\text{cellCentroid})$ 
9:     else
10:       $R_o.\text{append}(c_{i+1})$ 
11:   return  $R_o$ 

```

Figure 11: Weather routing algorithm

Figures 12, 13, and 14 illustrate three examples of the alternate route recommendation via the weather routing algorithm and the adverse condition thresholds. The red lines indicate the initial route the vessel is supposed to follow and the yellow lines indicate the route proposed by the algorithm.

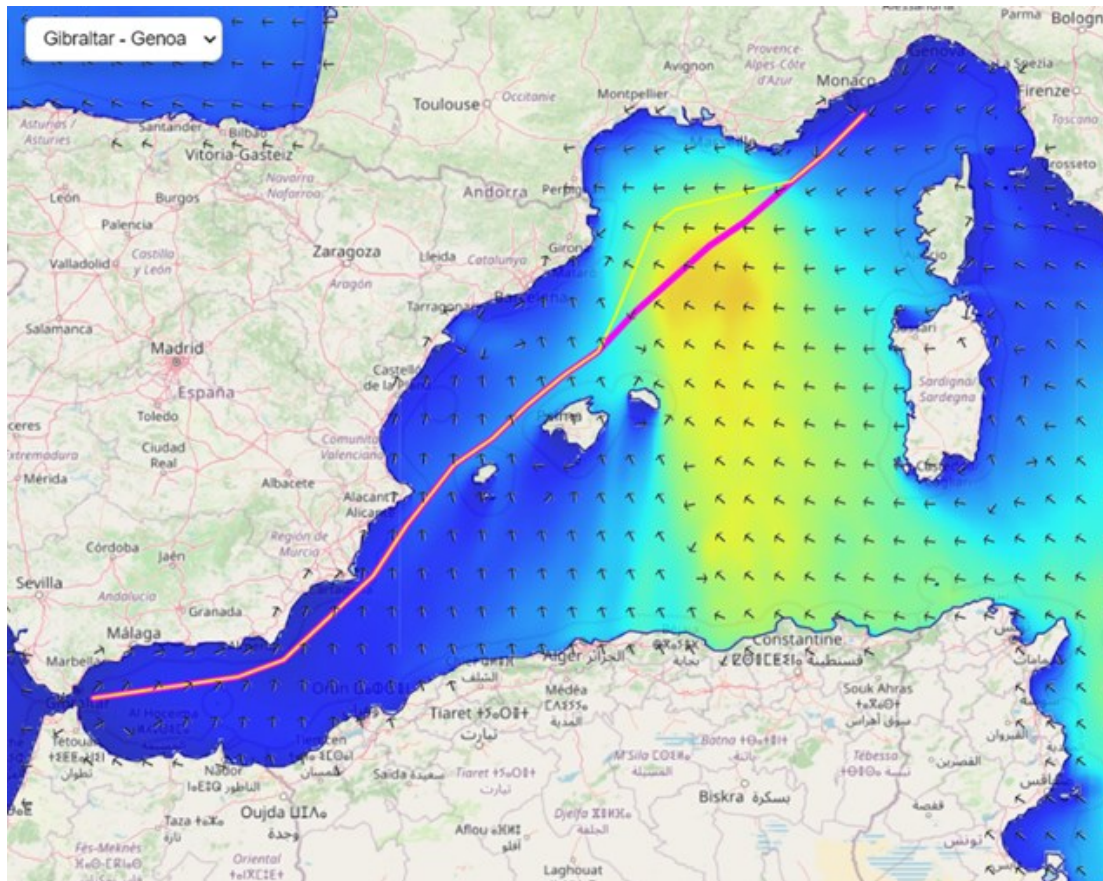


Figure 12: Alternate route recommendation on route Gibraltar–Genoa

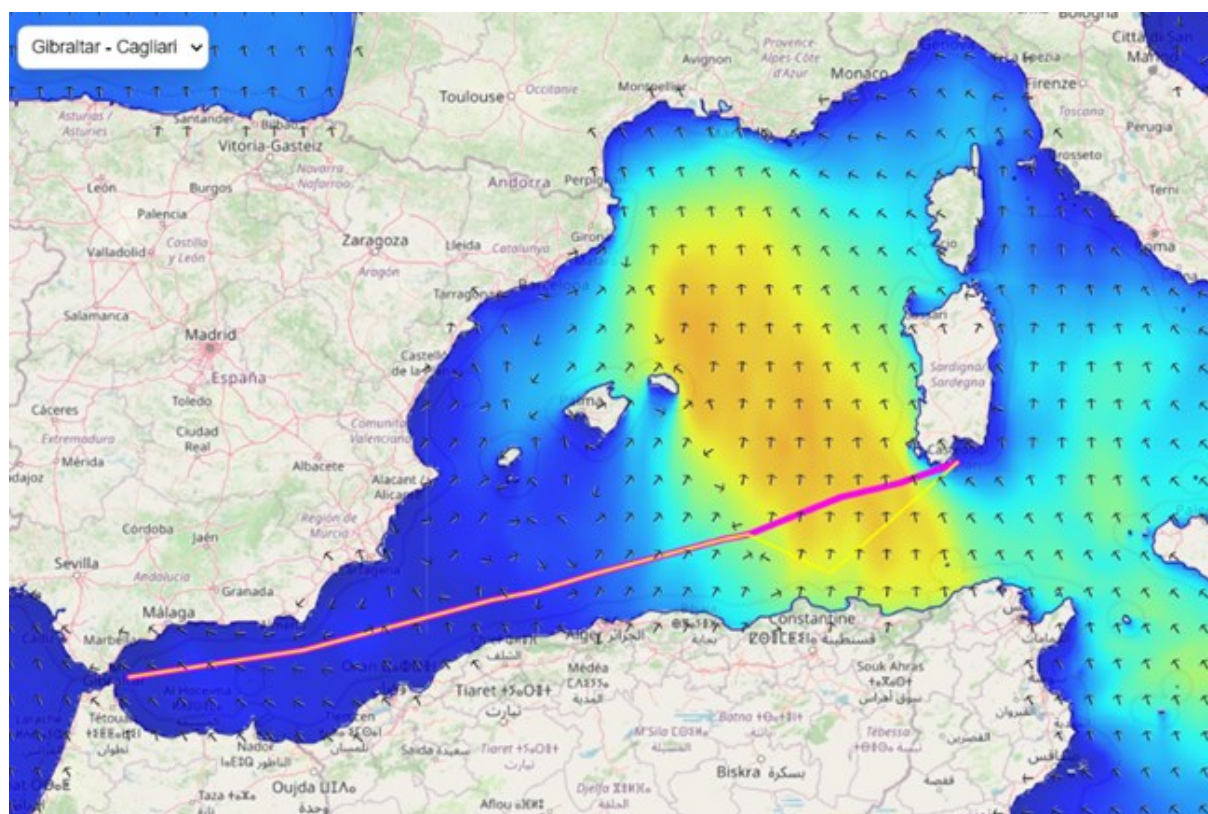


Figure 13: Alternate route recommendation on route Gibraltar–Cagliari

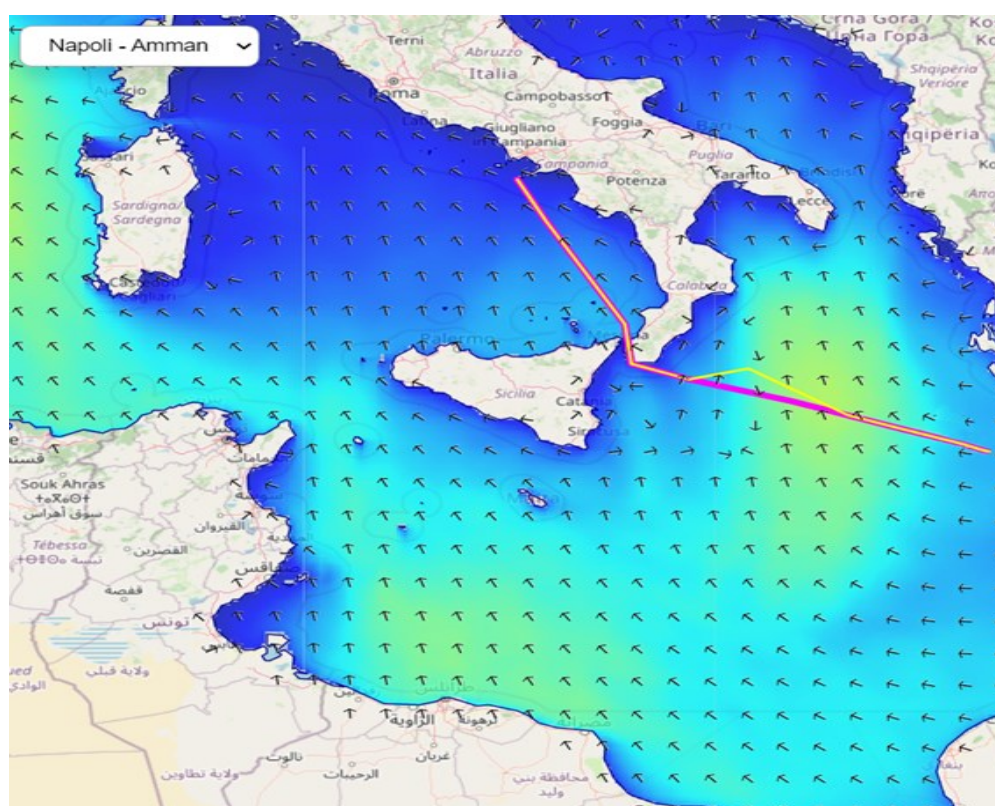
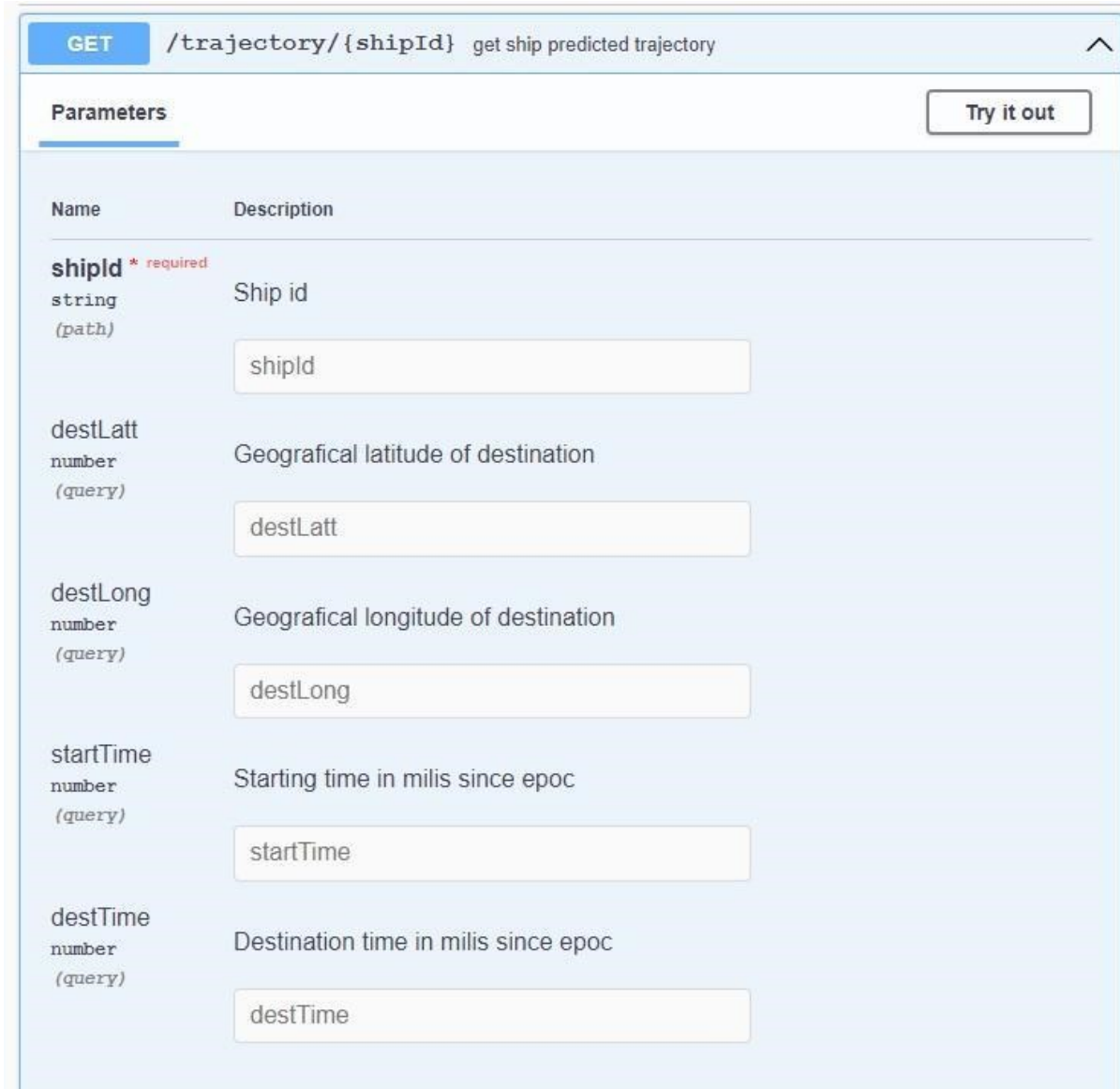


Figure 14: Alternate route recommendation on route Napoli–Amman

6.2.1 API

This component has minimalist API, only one endpoint to receive the optimal vessel route , considering the weather condition from start to finish.



The screenshot shows a web interface for an API endpoint. At the top, there is a blue button labeled 'GET' and the endpoint path `/trajectory/{shipId}` with the description 'get ship predicted trajectory'. Below this is a 'Parameters' section with a 'Try it out' button. The parameters are listed in a table with columns 'Name' and 'Description'. Each parameter has a text input field below it.

Name	Description
shipId * required string (path)	Ship id
destLatt number (query)	Geographical latitude of destination
destLong number (query)	Geographical longitude of destination
startTime number (query)	Starting time in milis since epoc
destTime number (query)	Destination time in milis since epoc

Figure 15: Weather route optimization component endpoint

Figure 16 shows the response body for the request:

```
{
  "id": "string",
  "metadata": {
    "createdAt": "2023-03-10T08:07:04.569Z",
    "createdBy": "admin@smartship.eu",
    "modifiedAt": "2023-03-10T08:07:04.569Z",
    "modifiedBy": "user@smartship.eu"
  },
  "prediction": [
    {
      "latitude": 37.757285,
      "longitude": 23.626491,
      "predictedTime": "2023-03-10T08:07:04.569Z",
      "accuracy": 76
    }
  ]
}
```

Figure 16: Body response of weather route optimization module

6.2.2 Technological stack

This component is written with Python script language due to vast popularity in machine learning and statical oriented developers. Python Fast API framework provide the fast, easy and robust implementation for web service.

- Python 3
- Python Fast API
- Python TensorFlow

6.2.3 Licences

Python 3: PSF License Agreement [7]

Python TensorFlow: Apache License 2.0 [6]

6.3 Predictive maintenance component

This component produces the predictive maintenance report about ship parts. It is based on the data stored in the DataStoring component which are provided by the IoT system at the vessels

6.3.1 API

This component has a minimalistic API which exposes the predictive maintenance report in JSON format. Figure 17 presents the SmartShip predictive maintenance API.

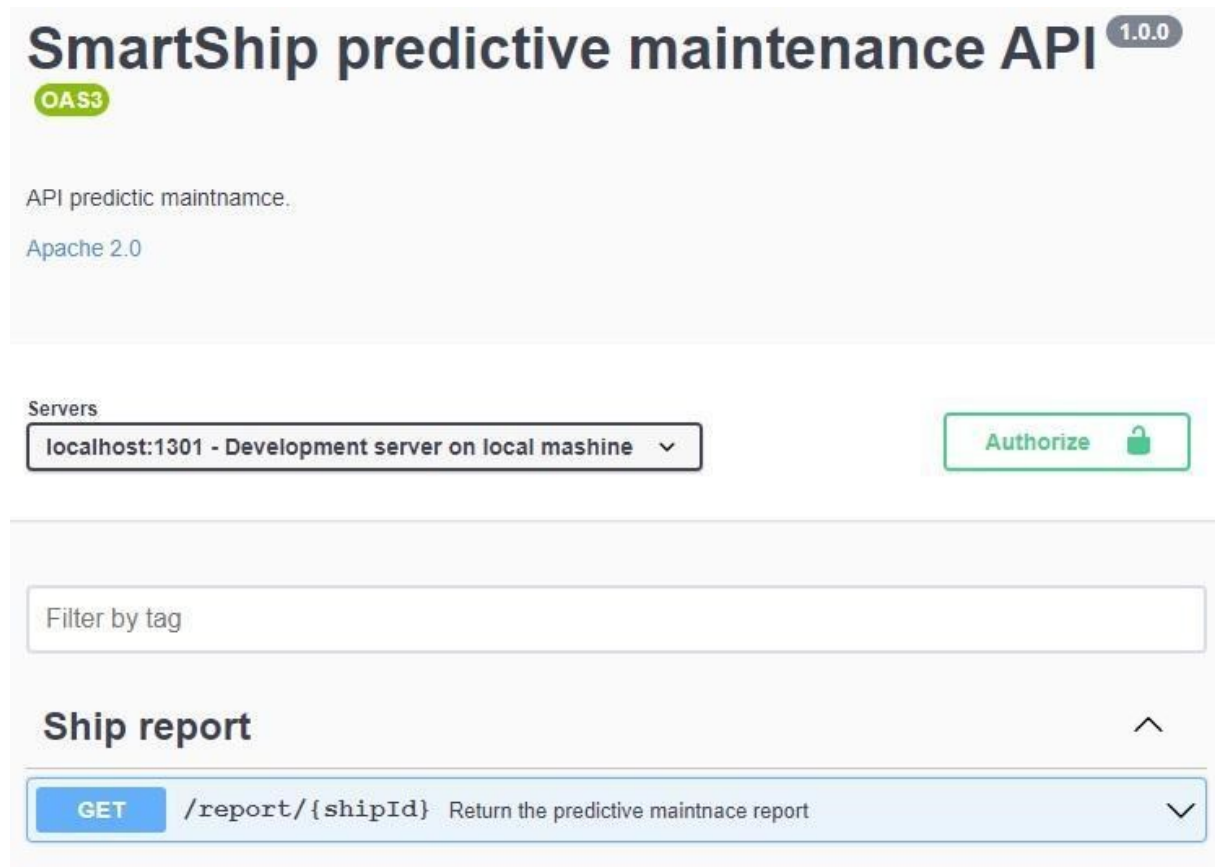


Figure 17: Predictive maintenance API

Figure 18 shows the response in JSON format:

```
{
  "id": "string",
  "metadata": {
    "createdAt": "2023-03-10T09:46:41.204Z",
    "createdBy": "admin@smartship.eu",
    "modifiedAt": "2023-03-10T09:46:41.204Z",
    "modifiedBy": "user@smartship.eu"
  },
  "engineReport": {
    "waterPump": {
      "partId": "3fa85f64-5717-4562-b3fc-2c963f66afa6",
      "daysToChange": 213,
      "daysToService": 213,
      "failureChance": 21
    },
    "airPump": {
      "partId": "3fa85f64-5717-4562-b3fc-2c963f66afa6",
      "daysToChange": 213,
      "daysToService": 213,
      "failureChance": 21
    }
  },
  "cylinerInformation": [
    {
      "partId": "3fa85f64-5717-4562-b3fc-2c963f66afa6",
      "daysToChange": 213,
      "daysToService": 213,
      "failureChance": 21
    }
  ]
}
```

Figure 18: Predictive maintenance response example

6.3.2 Technological stack

Python 3
Python Fast API
Python TensorFlow

6.3.3 Licences

Python 3: PSF License Agreement [7]
Fast API: MIT license [8]
Python TensorFlow: Apache License 2.0 [6]

7. Deployment and testing environment

DevOps is a software development methodology emphasizing collaboration, automation, and continuous improvement. The process combines software development and IT operations to shorten the systems development life cycle and ensure the continued delivery of high-quality software.

The DevOps process revolves around several core principles, including continuous integration, continuous delivery, and infrastructure as code. Continuous integration involves using tools like Jenkins, CircleCI, or Travis CI to build, test, and validate code changes frequently and automatically. This helps catch bugs early in the development process, reducing the time and cost required to fix them.

Infrastructure as code is another critical principle in the DevOps process, using tools like Terraform or CloudFormation. It treats infrastructure components like servers, networks, and storage as code, version-controlled, and managed using the same tools and practices as software code. This approach enables teams to spin up new environments quickly and consistently, reducing manual errors and accelerating delivery times.

DevOps processes involve collaboration, communication, and transparency between development and operations teams. These teams can share knowledge, skills, and expertise by working together, leading to faster problem-solving and better decision-making. DevOps also encourages a culture of experimentation and learning, where teams can try new things, learn from failures, and continually improve their processes.

The benefits of the DevOps process are numerous, including faster time-to-market, higher quality software, improved reliability and uptime, and better customer satisfaction. By enabling teams to work more efficiently and effectively, DevOps can also lead to significant cost savings and increased revenue.

7.1 Tools used for deployment process

During the development process of the SmartShip project, we rely heavily on Jenkins, which is installed on the FORTH server. Jenkins is our primary tool for continuous integration and delivery, allowing us to automate the build, test, and deployment of our code changes. Using Jenkins, we can ensure that we deliver high-quality software quickly and reliably.

We can customise our build pipeline with Jenkins to integrate with other tools such as Git and Docker. It streamlines our development process and helps us to identify and resolve issues quickly. Jenkins also provides detailed reporting and analytics on build performance, making it easier for us to optimize our development process continually.

Overall, Jenkins is a critical tool in our DevOps toolkit that enables us to deliver value to our customers faster and more efficiently. By installing it on the FORTH server, we can ensure a reliable and secure environment for our continuous integration and delivery processes.

7.1.1 Jenkins Pipelines

Jenkins pipelines are a powerful tool for automating software development workflows. A pipeline is a series of stages that define the steps required to build, test, and deploy software. Jenkins pipelines allow us to define and execute these stages in a declarative way, making it easier to manage complex software development workflows. In the SmartShip project, Jenkins pipelines are used to automate various tasks, from building and testing code to deploying applications to production environments. Using pipelines, you can streamline your software development process and ensure consistent, reliable results. Below is an example of a pipeline used to deploy the Python application.

```
pipeline {
  agent any

  environment {
    DOCKER_IMAGE = "SmartShipPredictiveMaintnaceApp:latest"
  }

  stages {
    stage('Checkout') {
      steps {
        checkout([$class: 'GitSCM',
          branches: [[name: 'develop']],
          doGenerateSubmoduleConfigurations: false,
          extensions: [],
          submoduleCfg: [],
          userRemoteConfigs: [[credentialsId: 'my-github-creds',
            url: 'https://github.com/SmartShip/SmartShipPredictiveMaintnaceApp.git']]])
      }
    }

    stage('Compile') {
      steps {
        sh 'pip install -r requirements.txt'
        sh 'python setup.py build'
        sh 'python setup.py install --user'
      }
    }

    stage('Build Docker Image') {
      steps {
        script {
          docker.build(DOCKER_IMAGE)
        }
      }
    }

    stage('Run Docker Container') {
      steps {
        script {
          docker.image(DOCKER_IMAGE).run("-p 80:80")
        }
      }
    }
  }
}
```

```

    }
  }
}

post {
  always {
    script {
      docker.image(DOCKER_IMAGE).remove(force: true)
    }
  }
}
}
}

```

7.2 Server parameters

1.1.1 Hardware and Software specifications

Smartship database and application server specifications are scalable based on the user's volume. Server could be configured/hosted on-premise and cloud (e.g. Microsoft Azure). Minimum server Specifications are depicted in the tables below

Database Server

Table 3 Database server specification

Users	1-5	6-10	11-25	26-50
CPU (cores)	4	6	6	8
RAM (GB)	12	16	32	64
HDD (GB)	250	300	400	500
O/S	LINUX Centos 7 or latest			

Application Server

Table 4 Component server specification

Users	1-5	6-10	11-25	26-50
CPU (cores)	6	6	8	8
RAM (GB)	12	16	16	32
HDD (GB)	200	300	300	350
O/S	Windows server 2012 or latest			

OTHER	IIS.net 4.5
-------	-------------

8. Conclusions

This deliverable reports the design and the development of the decision support module and multi-layer optimization tools and technologies of SMARTSHIP, which is actual the presentation of the SmartShip platform components. The report also contains a detailed analysis of the entire process that led to developing the SMARTSHIP components.

Furthermore, it presents the current state of the art in maritime operation software, available in the market today, similar to SMARTSHIP platform

The deliverable is active throughout the Work Package 5 which contains three tasks; Tasks are related with the above aforementioned sections. All the three tasks are ongoing until M48 of the project, in March 2023.

9. References

[1]. M. Konstantinidis, Artificial Intelligence helps Shipping become even SMARTER & more EFFICIENT, 2017.
[2]. Deepsea, "DeepSea," DeepSea solutions, 15 03 2023. [Online]. Available: https://www.deepsea.ai/ .
[3]. IETF, "Hypertext Transfer Protocol (HTTP/1.1): Authentication," 03 03 2023. [Online]. Available: https://datatracker.ietf.org/doc/html/rfc7235 .
[4]. J. Rola, "GitHub," BlueSoft, 17 02 2023. [Online]. Available: https://github.com/KUBER2/SmartShipAPI/blob/main/SensorDataAPI/dataAPI.yaml .
[5]. MongoDB, MongoDB, [Online]. Available: https://github.com/mongodb/mongo/blob/master/LICENSE-Community.txt . [Accessed 22 03 2023].
[6]. "Apache 2 Licence," [Online]. Available: https://github.com/spring-projects/spring-boot/blob/main/LICENSE.txt .
[7]. Python, "Python licence," [Online]. Available: https://docs.python.org/3/license.html#psf-license . [Accessed 7 03 2023].
[8]. "MIT Licence," [Online]. Available: https://opensource.org/license/mit/ . [Accessed 9 03 2023].