Optimisation of digital catch monitoring and reporting in European Fisheries

A Digital Architecture for Smarter Catch Monitoring in EU Fisheries: The OptiFish Approach

Responsible Author:
Antonis Koukourikos (SCiO)

Co-funded by
the European Union

optifish.eu

Document Information

Grant Agreement No.	**101136674**		
Project Acronym	**OptiFish**		
Project Title	Optimisation of digital catch monitoring and reporting in European Fisheries		
Type of Action	HORIZON Innovation Actions		
Call	HORIZON-CL6-2023-FARM2FORK-01		
Start – Ending date	1 February 2024 – 31 January 2028	**Duration**	48 months
Project Website	optifish.eu		
Work Package	WP4: Data management framework and system architectures		
WP Lead Beneficiary	SCiO		
Relevant Task(s)	T4.1 Requirements specification: Data management framework and system architectures		
Deliverable type [1]	R	**Dissemination level** [2]	PU
Due Date of Deliverable	-		
Submission Date	22 September 2025		
Responsible Author	Antonis Koukourikos (SCiO)		
Contributors	Josean Fernandez (AZTI), Lancelot Blondeel (EV ILVO), Teun De Boer (EFICE)		
Reviewer(s)	-		

[1] *Please consult the Grant Agreement: R: Document, report; DEM: Demonstrator, pilot, prototype, plan designs; DEC: Websites, patents filing, press & media actions, videos, etc.; DATA: Data sets, microdata, etc; DMP: Data management plan; ETHICS: Deliverables related to ethics issues; SECURITY: Deliverables related to security issues; OTHER: Software, technical diagram, algorithms, models, etc.*

[2] *Please consult the Grant Agreement: PU – Public, fully open, e.g. web (Deliverables flagged as public will be automatically published in CORDIS project's page); SEN – Sensitive, limited under the conditions of the Grant Agreement; Classified R-UE/EU-R – EU RESTRICTED under the Commission Decision No2015/444; Classified C-UE/EU-C - EU CONFIDENTIAL under the Commission Decision No2015/444; Classified S-UE/EU-S – EU SECRET under the Commission Decision No2015/444.*

Document History

Version	Changes	Date	Contributor
0.9	1st full version	08/09/2025	Antonis Koukourikos (SCiO)
1.0	Final version ready for submission	22/09/2025	Antonis Koukourikos (SCiO), Josean Fernandez (AZTI), Lancelot Blondeel (EV ILVO), Teun De Boer (EFICE)

OptiFish Consortium

No.	Participant Organisation Name	Short Name	Country
1	EIGEN VERMOGEN VAN HET INSTITUUT VOOR LANDBOUW- EN VISSERIJONDERZOEK	EV ILVO	BE
2	FUNDACION AZTI - AZTI FUNDAZIOA	AZTI	ES
3	BENCO BALTIC DOO ZA SAVJETOVANJE IUSLUGE	BENCO	HR
4	DANMARKS TEKNISKE UNIVERSITET	DTU	DK
5	REFRAME FOOD ASTIKI MI KERDOSKOPIKI ETAIRIA	RFF	EL
6	SCIO IKE	SCiO	EL
7	STICHTING WAGENINGEN RESEARCH	WR	NL
8	UNIVERSITY OF CUKUROVA	UC	TR
9	FISKERIDIREKTORATET	NDF	NO
10	SINTEF OCEAN AS	SO	NO
11	ELECTRONIC FISH INFORMATION CENTRE EUROPE B.V	EFICE	NL
12	Justervesenet	JV	NO
13	VCU ROBOTICS B.V.	VCUR	NL
13.1	VCU TCD B.V.	VCU	NL
14	WAGENINGEN UNIVERSITY	WU	NL
15	ANCHOR LAB KS	ANCHOR	DK
16	DANMARKS PELAGISKE PRODUCENTORGANISATION FORENING	DPPO	DK
17	ZUNIBAL SL	ZUN	ES
18	DANMARKS FISKERIFORENING PRODUCENTORGANISATION	DFPO	DK

Executive Summary

The OptiFish Data Platform provides the technical backbone for a scalable, interoperable, and secure digital infrastructure supporting fisheries Monitoring, Control, and Surveillance across Europe. The platform enables the collection, semantic integration, processing, and dissemination of fisheries data from a heterogeneous landscape of sources, including onboard electronic monitoring systems, sensor networks, vessel tracking systems (e.g. AIS, VMS), and third-party providers.

Following a C4 architectural approach (Context, Container, Component, Code), the design ensures modularity, traceability, and alignment with both technical and stakeholder requirements. The platform supports automated ETL (Extract, Transform, Load) pipelines, role-based access and compliance controls, and domain-specific analytics, while acting as a common interface for tools and services developed across OptiFish activities and use cases.

This paper presents a focused view of the architectural foundations of the OptiFish Data Platform, highlighting key innovations, integration principles, and the roadmap for further development.

Contents

List of Figures

List of Tables

1 Vision and Motivation

Fisheries across Europe are undergoing rapid digital transformation, driven by evolving regulatory frameworks, sustainability demands, and technological advances. In this context, the OptiFish project aims to optimise digital catch monitoring and reporting, enhancing traceability, regulatory compliance, and decision support for a wide spectrum of stakeholders — from fishers and scientists to policy makers and control authorities.

A central challenge in this domain is the fragmentation and heterogeneity of data sources: video streams from electronic monitoring systems, sensor data, vessel logs, satellite inputs, and legacy systems often lack semantic alignment and interoperability. Furthermore, the increasing adoption of AI-based tools and advanced analytics requires robust, secure, and transparent infrastructures to support real-time inference, historical analysis, and actionable insight generation.

The OptiFish Data Platform was conceived as the technical backbone of this transformation. Its goals include:

- Aggregating and harmonising diverse data sources, both in real-time and batch modes.
- Supporting semantic enrichment and standardisation for regulatory and scientific reuse.
- Enabling GDPR-compliant access control, with traceable, role-based permissions.
- Providing open, well-documented APIs to foster integration with analytics, dashboards, and decision support tools.
- Offering intuitive, stakeholder-specific interfaces for data exploration, reporting, and feedback.

The architectural design was grounded in consultations with technical partners and domain stakeholders, ensuring alignment with real-world needs, use cases, and constraints. The platform's modularity and openness also allow it to serve as an extensible foundation for integrating future tools and piloting new services developed under OptiFish and beyond.

2 Architectural Principles and Methodology

The architecture of the OptiFish Data Platform is shaped by the dual need to meet the technical requirements of data-intensive fisheries monitoring and the operational realities of regulatory compliance, real-time analytics, and stakeholder usability. To address this, the platform was designed using a structured and widely adopted methodology: the C4 model.

2.1 Design Drivers

Several guiding principles informed the architectural approach:

- **Modularity**: Each part of the system is designed as an independent module with a single responsibility, facilitating future evolution, substitution, or scaling.

- **Traceability**: Data flows, transformations, and accesses are traceable across all layers of the system — essential for auditing, reproducibility, and compliance.
- **Security & Privacy by Design**: The platform enforces GDPR-compliant access control, minimisation, and auditability mechanisms throughout its components.
- **Scalability**: The architecture supports high-volume ingestion and processing, including video streams, sensor feeds, and satellite data, with cloud-native elasticity.
- **Interoperability**: Semantic enrichment and API-driven design ensure seamless integration with external data sources, tools, and services developed in other WPs.
- **Stakeholder-Centric Interfaces**: Role-specific dashboards and data views are tailored to the needs of regulators, scientists, and fishers.

2.2 Methodological Approach: The C4 Model

To structure the architectural documentation and design, we adopted the C4 model (Context, Container, Component, Code), developed by Simon Brown. It provides a multi-level representation of software architecture with increasing technical detail:

- **Context** (Level 1): Describes the platform as a whole — how it interacts with external systems and users.
- **Container** (Level 2): Breaks the system into major services and storage units (web apps, APIs, databases, etc.).
- **Component** (Level 3): Zooms into each container to show internal services, responsibilities, and interconnections.
- **Code** (Level 4): Describes the internal implementation of each component (not covered in this paper).

By applying this methodology, the OptiFish architecture becomes both transparent to stakeholders and actionable for developers. It fosters shared understanding across technical and non-technical audiences, promotes traceable design decisions, and aligns system structure with functional and non-functional requirements.

2.3 From Requirements to Architecture

The architectural design directly reflects the functional and cross-cutting requirements elicited through questionnaires, partner interviews, and stakeholder consultations (see Section 2). Key requirements translated into concrete design elements, including:

- A modular ETL pipeline to handle data ingestion, validation, cleaning, and semantic enrichment;

- A secure access control system to manage user roles, credentials, and compliance logic;
- Integration of Machine Learning and geospatial services as first-class analytical components;
- A dual-layer storage model, combining raw data preservation with curated warehouse access;
- Public and internal APIs to expose data and computation capabilities for other work packages.

The resulting architecture is future-proof, built to evolve alongside the tools, pilots, and regulatory frameworks developed throughout the OptiFish project.

3 High-level Architecture Overview

The OptiFish Data Platform is designed as a modular, cloud-native system capable of integrating, processing, storing, and exposing data from a diverse ecosystem of sources and stakeholders. The following subsections provide a summary of the main architectural constituents of the platform under the three C4 levels covered by this paper (Context, Container, Component levels).

3.1 Context Level

At the context level, the OptiFish Data Platform is positioned as the central infrastructure enabling the collection, processing, storage, and dissemination of data across the OptiFish ecosystem. More specifically, the OptiFish Data Platform is a cloud-based fisheries data management system that integrates multiple data sources, processes information using AI models, and provides analytics for industry stakeholders and regulators.

Key Actors and Interactions

- Fishers: Use the system for real-time fishing data, quota tracking, and operational insights.
- Regulatory Authorities (e.g., EFCA, national agencies): Access compliance data, reports, and alerts to monitor fisheries activities.
- Marine Scientists: Utilize historical and real-time data for research on fisheries sustainability.

Key external communications

- Vessel Monitoring & Sensor Systems: Provide VMS, AIS, REM, catch, fuel consumption, vessel state and environmental data for analysis.
- External Systems & Data Providers: Integrate satellite imagery, FLUX reports, and other industry datasets.
- Copernicus: Supply environmental data forecast and fill missing historical data gaps.
- EMODnet data

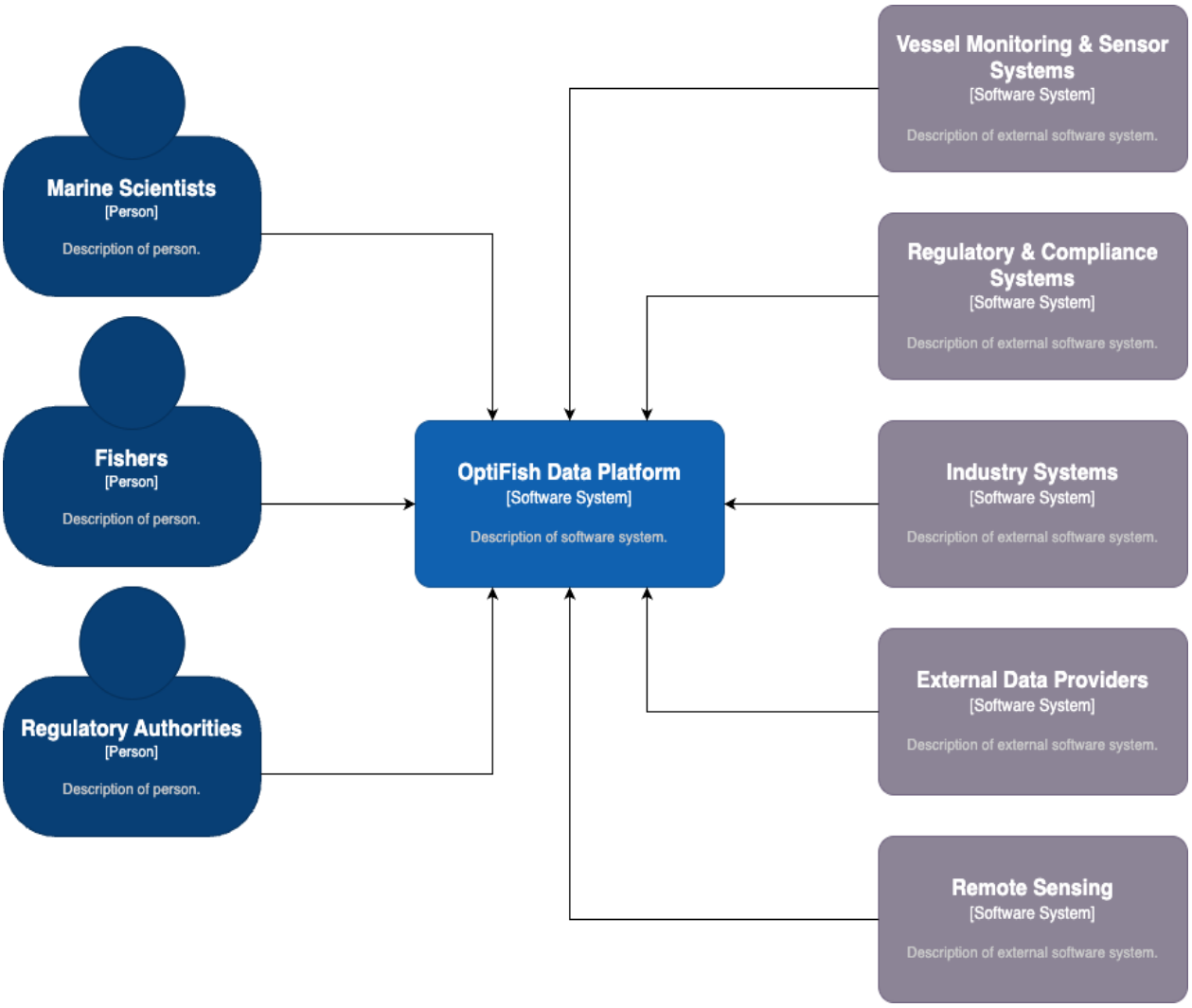

Figure 1. OptiFish Architecture - Context Level.

The system will operate as a centralized data hub, ingesting, processing, analysing, and visualising data for all stakeholder types and end-user personas.

3.2 Container Level

The platform is divided into multiple logical containers, each representing a distinct application or data service with its own responsibility, deployment model, and technology stack. The core containers foreseen in the platform are summarised in the following table.

Table 1: OptiFish Data Platform Architecture C2 Containers.

Container	Description	Related Operation(s)
Web Portals	Provides user dashboards for different stakeholder types	User Interface and Access
API Gateway	Central entry point for external system, sensor data management, and API requests	Data Ingestion, Integration and Programmatic Access
Data Lake	Storage of unstructured data	Data Storage
Data Warehouse	Storage of structured data	Data Storage
ETL Pipeline	Extract/Clean/Transform raw data	Data Processing and Transformation
Analytics Engine	Bootstraps and executes analytical and AI models	Data Processing and Transformation, Analytics and Visualisation
Geospatial Engine	Handles the analysis of geospatial data	Data Processing and Transformation, Analytics and Visualisation
Access Control and Identity Management	Ensures data security, role-based access, and regulatory compliance	Governance
Audit and Logging Service	Tracks data access, modification, and compliance events	Governance
Data Export Service	Allows users to export reports and insights	User Interface and Access, Integration and Programmatic Access

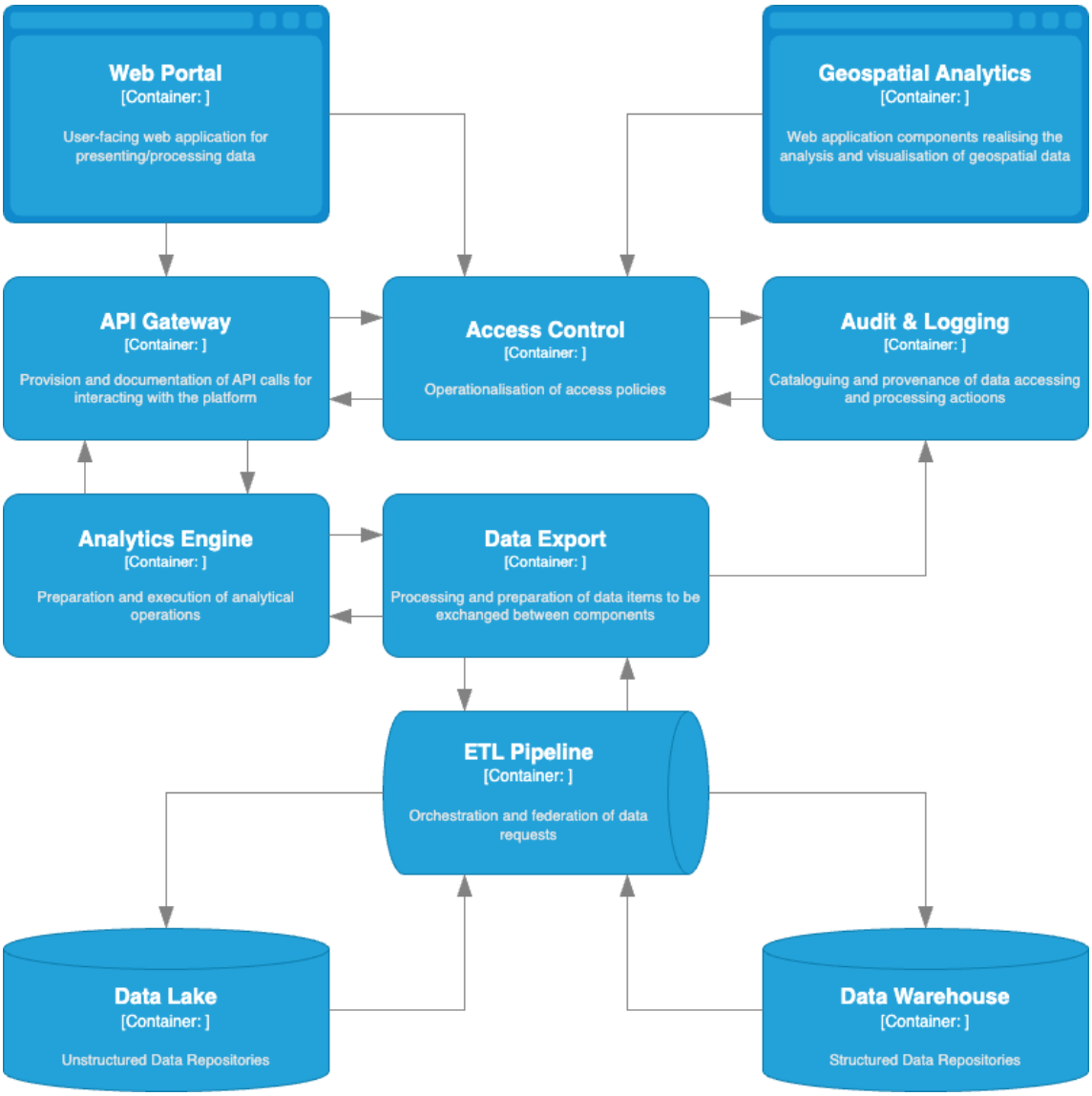

Figure 2: OptiFish Architecture - Container Level

3.3 Component Level

Each container is internally decomposed into core components, each with a single responsibility and well-defined interfaces. Below is a representative sample from selected containers.

Table 2: Data Platform Components: ETL Pipeline Container.

Component	Responsibilities	Technologies
Ingestion Service	Connects to data sources (vessels, EM systems, external APIs)	Apache NiFi, Kafka Connect
Validation Module	Schema and integrity checks on incoming data	JSON Schema, Apache Beam
Data Cleaner	Applies cleaning rules (removal, fixing, standardisation)	Python (Pandas), Spark
Format Converter	Transforms data into standard formats	Apache Camel, custom scripts
Semantic Enricher	Adds metadata, tags, ontology references	Python, RDFLib, SPARQL
Pipeline Orchestrator	Coordinates ETL steps, logs execution	Apache Airflow, Prefect

Table 3: Data Platform Components: API Gateway Container.

Component	Responsibilities	Technologies
Routing Engine	Routes requests to backend services	Spring Cloud Gateway, Kong
Rate Limiter	Controls usage per user/API key	NGINX, Envoy, Redis
Auth Adapter	Validates tokens and forwards user claims	Keycloak Adapter, OAuth2 Proxy
Request Validator	Validates payloads against API specs	OpenAPI Validator, JSON Schema
Monitoring Hook	Captures API metrics and errors	Prometheus, Grafana, ELK

Table 4: Data Platform Components: Access Control Container.

Component	Responsibilities	Technologies
User Directory	Stores users, roles, credentials	Keycloak, LDAP
RBAC Service	Enforces fine-grained permissions	Keycloak Authorization, OPA
Token Service	Issues and verifies JWT tokens	OAuth2, OpenID Connect
Compliance Checker	Enforces data governance constraints	Custom policy engine, OPA
Audit Hook	Logs auth events to audit container	Kafka, Fluentd

Table 5: Data Platform Components: Analytics Engine Container.

Component	Responsibilities	Technologies
Model Executor	Runs ML models on incoming or stored data	Python, TensorFlow, PyTorch
Job Scheduler	Schedules and triggers analytics runs	Apache Airflow, Kubernetes CronJobs
Data Preprocessor	Prepares inputs for model runs	Pandas, Spark
Model Registry	Stores and version-controls ML models	MLflow, DVC
Result Publisher	Sends outputs to Data Export or Warehouse	Kafka, REST API

Table 6: Data Platform Components: Geospatial Engine Container.

Component	Responsibilities	Technologies
Map Renderer	Displays spatial layers, routes, events	Leaflet.js, Mapbox, OpenLayers
Spatial Processor	Computes zones, heatmaps, overlaps	PostGIS, GeoPandas
GeoAPI	Exposes geospatial queries	Flask, FastAPI, Node.js
Layer Manager	Handles base maps and user-defined overlays	GeoServer, QGIS Server

Table 7: Data Platform Components: Web Portals Container.

Component	Responsibilities	Technologies
User Dashboard	Displays charts, alerts, KPIs	React, Vue, Angular
Data Viewer	Allows exploration of datasets	React Table, D3.js
Feedback Form	Sends feedback or corrections	Formspree, Custom Backend
Alert Panel	Displays compliance or safety alerts	WebSockets, MQTT
Profile Manager	Manages user settings and preferences	React + Auth integration

Table 8: Data Platform Components: Data Export Container.

Component	Responsibilities	Technologies
Export Generator	Builds reports in CSV, JSON, PDF	Python, Pandas, JasperReports
Report Scheduler	Automates periodic exports	Quartz, cron, Airflow
Data Formatter	Applies export templates and formatting	Jinja2, XSLT

Download API	Endpoint for users to retrieve files	Flask, SpringBoot

Table 9: Data Platform Components: Audit and Logging Service Container.

Component	Responsibilities	Technologies
Log Collector	Captures logs from all services	Fluentd, Filebeat
Event Store	Stores structured events for auditing	Elasticsearch, Loki
Alert Generator	Flags suspicious or policy-violating actions	Kibana, Grafana Alerts
Log Dashboard	Interface for browsing logs	Kibana, Grafana, Graylog

Table 10: Data Platform Components: Data Lake Container.

Component	Responsibilities	Technologies
Blob Storage	Holds large binary files	S3, MinIO, HDFS
Metadata Index	Indexes content types, timestamps, sources	ElasticSearch, PostgreSQL
Access API	Serves files or streams on demand	FastAPI, MinIO Gateway

Table 11: Data Platform Components: Data Warehouse Container.

Component	Responsibilities	Technologies
Schema Manager	Defines and evolves warehouse schema	Liquibase, Flyway
Query Engine	Supports reporting and analytics	PostgreSQL, ClickHouse
ETL Loader	Loads curated data from ETL pipeline	Apache Sqoop, Spark

4 Conclusions and Roadmap

The described architectural specification has laid the groundwork for a robust, modular, and interoperable data platform. By following the C4 model, we systematically defined the system's context, decomposed it into functional containers, and described the internal architecture of core components. This approach has ensured a high degree of clarity, traceability, and alignment with both user requirements and the technical needs of other work packages. It also helps to clarify component communication points, identify dependencies from external systems and technologies, and select the appropriate candidate frameworks and technologies for development and deployment.

To ensure the successful evolution and deployment of the platform, the following design considerations will be systematically assessed as the platform transitions to its development phase:

- Iterative refinement of the architecture based on implementation experience, stakeholder feedback, and tool integration results.
- Formalisation of semantic services, including ontology and schema management, term mapping, and annotation APIs, to support interoperability and enrich data discovery.
- Development of compliance rule engines that encapsulate regulatory logic and support automated alerting and reporting.
- Scalable deployment planning, particularly for handling video-intensive use cases and multi-vessel data streams in real-world pilots.
- Security and privacy validation, with a focus on ensuring GDPR-compliant user access, data minimisation, and traceability across all platform layers.

Furthermore, the OptiFish data platform will take advantage of already established knowledge and assets for developing the components foreseen in the architecture: Firstly, we will examine bidirectional linkages with the EDITO European infrastructure and particularly its DataLake services and APIs, for ingesting, processing and submitting data - under the relevant access and data exposure limitations will be established. Additionally, the semantic framework to be used for conceptualising and associating formally defined semantics for the data handled by OptiFish will be based on the PoseiDAT open data schemas, extended accordingly to cover the needs of OptiFish.

In summary, the architecture defined in this deliverable provides a flexible and future-proof foundation for the OptiFish Data Platform. It reflects the project’s commitment to technical excellence, user-centric design, and sustainable digital transformation in fisheries management.

END OF DOCUMENT