HiSea Deliverable 4.1

HiSea Backend Architecture guidelines

Work Package Number: 4

Work Package Title: Service Design and Cloud Computing

HiSea Project Information	
Project full title	High Resolution Copernicus-Based Information Services at Sea for Ports and Aquaculture
Project acronym	HiSea
Grant agreement number	821934
Project coordinator	Dr. Ghada El Serafy
Project start date and duration	1st January, 2019, 30 months
Project website	https://HiSeaproject.com/

Deliverable Information	
Work package number	4
Work package title	Service Design and Cloud Computing
Deliverable number	4.1
Deliverable title	HiSea Backend Architecture Guidelines
Description	Summary of HISEA platform system technical requirements necessary to make HiSea platform fully compliant with a cloud development and prepared to use the DIAS infrastructure.
Lead beneficiary	Hidromod
Lead Author(s)	Pedro Galvão and Adélio Silva (Hidromod)
Contributor(s)	Anna Spinosa (Deltares), Danny Pape (ASCORA)
Revision number	V0.3

Revision Date	27/05/2019
Status (Final (F), Draft (D), Revised Draft (RV))	F
Dissemination level (Public (PU), Restricted to other program participants (PP), Restricted to a group specified by the consortium (RE), Confidential for consortium members only (CO))	RE

Document History

Revision	**Date**	**Modification**	**Author**
0.1	26/03/2019	Text updating	Pedro Galvão
0.2	19/05/2019	Text updating	Anna Spinosa
0.3	27/05/2019	Text updating	Adélio Silva

Approvals

	Name	**Organisation**	**Date**	**Signature (initials)**
Coordinator	Ghada El Serafy	Deltares		
WP Leaders	Adélio Silva	Hidromod		

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 821934

Table of Contents

1 Executive Summary

HISEA is a user-centered project aiming to make marine data easily accessible and operational to a broad range of end-users. HISEA platform aims at collecting data from different existing sources like CMEMS, NOAA, etc. and to run algorithms to compute new indicators on marine data.

This deliverable D4.1: HiSea Backend Architecture Guidlines refers to Task 4.1 (HiSea backend architecture requirements). It summarises HISEA platform system technical requirements necessary to make HiSea platform fully compliant with a cloud development and prepared to use the DIAS infrastructure. In particular, this deliverable describe how to host the developed components, how to integrate the services and what are the system requirements needed to run all components implemented by the partners of the HISEA platform.

2 Introduction

The goal of this document is to specify the HISEA platform architecture technical requirements. The platform will host a set of services that supply data to the front-end applications. Data will be gathered from external sources (such as CMEMS and products provided byconsortium partners), processed into standard formats and shared via a common set of services. The objective of the adopted platform architecture is to end up with a product that:

- Integrates the contribution of all the members of the consortium;
- Is easily maintainable;
- Is easily scalable;
- May enable to assure a confortable Service Level Agreement (SLA).

With this set of requirements, the adoption of an architecture based on what is commonly known as microservices seems appropriate, since applications built using microservices possess certain characteristics. In particular, they:

- Are fragmented into multiple modular, loosely coupled components, each of which performs a discrete function;
- Have those individual functions built to align with business capabilities;
- Can be distributed across clouds and data centers;
- Treat each function as an independent service that can be changed, updated, or deleted without disrupting the rest of the application.

A microservice based architecture will allow to split applications into distinct independent services, each managed by individual groups within the consortium. They support the separation of responsibilities critical for building highly scaled applications, allowing work to be done independently on individual services without impacting the work of other developers in other groups working on the same overall application. In short, the HISEA platform can grow as its requirements grow.

It is difficult to talk about microservices without also talking about containers. The services that make up an application can be placed within containers that possess the smallest libraries and executables needed by that service or application, making each container a self-contained package.

Although an application within a container operates independently from those in other containers, it still answers to directives from the kernel or other orchestration tools. Such orchestration tools are essential for managing large numbers of containers. A more detailed explanation of how orchestration is implemented in the HISEA platform is outside of the scope of this document.

The biggest drawback of microservices is that architectures tend to be complex, while individual microservices may be easier to understand and managed, the application as a whole may end up with significantly more moving parts. There are often more components involved, and those components have more interconnections. Those interdependencies increase the application’s overall complexity thus the importance of this document.

3 Overall Technical Architecture

The HISEA platform will perform three main tasks:

1. Acquire data from CMEMS, partner platforms and other external sources;
2. Convert the acquired data to common format in local cache storage;
3. Serve data via common standard services.

The platform itself will not perform major computational efforts to create refined data products as these are produced by the project partners or future partners that wish to use the platform. However, it will act as a data repository for these platforms to obtain data and publish the refined products.

Three different types of data will be served by the platform:

1. Time series (including transects and profiles);
2. Grid Data;
3. Images (Georeferenced or pre-processed).

These data types will be combined and accessible trough a comprehensive user interface trough which it will be delivered the requested information. The proposed platform architecture is designed to allow to achieve a platform backend scalable and flexible enough to accommodate any future requirements.

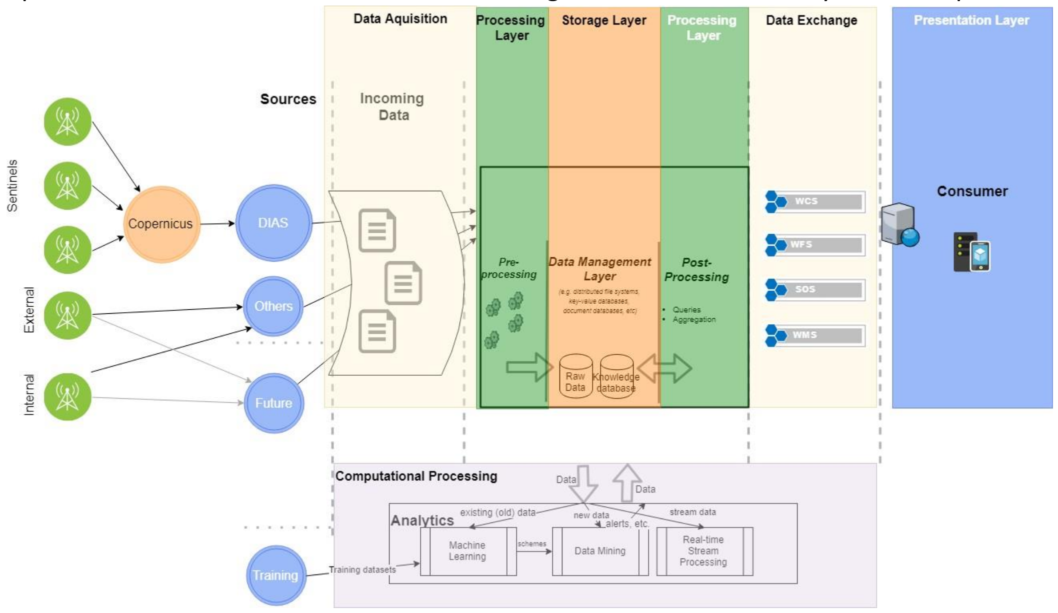

FIGURE 1 - PLATFORM ARCHITECTURE OVERVIEW

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 821934

4 Component Definition

The **Architecture Component Definitions** section describes each architecture component in the System Architecture Model and identifies specific elements that comprise that component. Figure 1 reports an overview of the HiSea platform architecture.

4.1 External Data Sources

External data providers are platforms, systems, and networks already operating andcollecting data from in-site sensors and remote sensing facilities, producing aggregated indicators and databases and running numerical models at the operational level (case of Copernicus products). Data may be divided (based on their parameters) into topics such as meteorology, topography, bathymetry, hydrology, hydrography, geology, physics, chemistry, biology, biodiversity, climate change, etc.

These data may be further divided into:

- Static/archived datasets, collected once or at long irregular intervals, as bathymetry, geology, biodiversity, etc.;
- Dynamic/forecasting datasets, collected and/or produced at short regular intervals (hourly, daily, etc.) continuously updated in databases, as physical, optical, biophysical parameters (real-time sensors, satellites, models).

A selection of data of interest for end-user needs will be reviewed. Some data will be retrieved and stored on the HISEA platform directly (high demand, high availability data for instance) and other data will be cataloged and referenced directly from its source for users to access.

A crucial part of the data used in HiSea will be obtained via DIAS services.

4.2 Data Acquisition

The data acquisition component is responsible for acquiring data from external sources into the HiSea platform. This acquisition is done via two separate routes depending on the size, demand, and type of source:

- Download and storage: this path will keep the data physically in the platform which allows for more control over the response times of the service but will occupy storage space in the platform;
- Index and pointers: with this option, instead of storing the data in the platform the data is maintained is kept in its original locations and, in case of request, the HiSea platform points directly for this original location. This saves space in the HiSea platform but may incur in larger maintenance costs and possible longer response times.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 821934

Any acquired files are transformed by this component into a common set of file formats following the SeDataNet (https://www.seadatanet.org/) common vocabularies and Metadata formats before passing into the pre-processing and storage layer.

A special case in data acquisition will be data that lives in the DIAS cloud-based providers. Since the platform be hosted inside DIAS, pointer sources will be created so that DIAS Copernicus data can be used as local data by the platform.

4.3 Pre-Processing

The pre-processing will run validation simple calculations over the acquired data. These algorithms will be implemented with a plug-in architecture so that the platform can accommodate new validations has the platform grows. After this component, the datasets are ready to be passed into the storage layer.

A good case for the pre-processing component is alarm management. Alarms need to be triggered as soon as possible instead of sitting on the datastore waiting for batch execution. A pre-processing stream will allow alarms to be triggered just as soon as data is acquired.

4.4 Storage layer

The platform will manage a variety of data formats. There are two main types of data to store:

- Time series, data in a single point for a single property over time:
  - Point Time Series – Fixed location (x, y, z) varying time, ex. weather station;
  - Transect Time Series :
    - Gliders Vary x, y, z, and t ;
    - Drifters fixed z, varies x, y, and t;
    - Vertical profiles fixed x, y and vary z and t;
- Gridded data, data on a structure or unstructured grid (1D, 2D or 3D) that varies in time (e.g. data from radar images or model results):
  - Raw data (usually in NetCDF, GeoTIFF or HDF format);
  - Processed data (usually has an image) .

The main objective is to store the data effectively and allow fasts access times. A mix of storage technologies should be explored during the implementation of this component (Structured databases, NoSQL, etc).

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 821934

This goal will be reached against the constraints of cloud storage. The most obvious solution in the future is to move the storage to the cloud but, for such a large platform of data, this solution at this stage of the project is not cost effective. However, the data storage solution will consider the new paradigm of cloud computing to ensure it is compatible with a future deployment via Data and Information Access Service (DIAS).

4.5 Post Processing layer

The Post Processing component will be used to run pre-planned processes on data stored in the platform to produce the end user products. The component is designed to be able to accept new processing services as required and is fully scalable to allow an increase in computational load as the platform and products increase in complexity and processing requirement.

It is assumed that these processes run fast enough to sit between the user requests and the data storage.

The list below describes the type of products that are expected to be generated by the post-processing component:

- Datasets filtering and aggregation (averages, min, max, etc.);
- Indicators;
- Simple calculations and time aggregation;
- Previously implemented Machine Learning algorithms.

4.6 Computational processing

The computational processing layer will run the longer processing algorithms that are too lengthy to run on the post-processing layer. This layer will obtain its data from the Data Exchange component just like any other client.

Two types of processes are expected to run on this component:

- User triggered processes: The user initiates a process supplying some information about the execution (ex. start and end date, the location where execution is processed, etc.)
- Automated processes: Once automated processes are set up, they are triggered either by events (ex. Download complete, calculation finished, alarm triggered, etc.) or on regular intervals

Once user triggered processes are complete the result dataset is sent directly to the user and is only stored temporarily on the platform.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 821934

Automated processes usually generate a new dataset that will be returned into the storage layer via the Data Acquisition component. The exception is report generation processes. The report files created via automated report generation are only stored temporally before being sent to the respective distribution lists.

4.7 Data exchange

The Data exchange components are used to access to the HISEA data through both human and machine-to-machine mechanisms. HiSea will consider the use of standard OCG compliant protocols whenever possible. However most of these protocols contemplate scenarios that might be too complex and unnecessary to accomplish the HiSea platform objectives. In these cases, an adaptation or partial implementation will be used. Some of these standard OCG compliant protocols are for instance:

- **SOS, Sensor Observation Service:** The OpenGIS® Sensor Observation Service standard is applied to time series. It defines a Web service interface which allows querying observations, sensor metadata, as well as representations of observed features. Further, this standard defines means to register new sensors and to remove existing ones. Also, it defines operations to insert new sensor observations.

- **WMS, Web Map Service**: The OpenGIS® Web Map Service Interface Standard (WMS) provides a simple HTTP interface for requesting geo-registered map images from one or more distributed geospatial databases. A WMS request defines the geographic layer(s) and area of interest to be processed. The response to the request is one or more geo-registered map images (returned as JPEG, PNG, etc.) that can be displayed in a browser application. The interface also supports the ability to specify whether the returned images should be transparent so that layers from multiple servers can be combined or not.

- **WFS, Web Feature Service:** The OpenGIS® Web Feature Service Interface Standard (WFS) provides an interface allowing requests for geographical features across the web using platform-independent calls. One can think of geographical features as the "source code" behind a map, whereas the WMS interface or online mapping portals like Google Maps return only an image, which end-users cannot edit or spatially analyze.

- **WCS, Web Coverage Service:** Unlike OGC Web Map Service (WMS), which portrays spatial data to return static maps (rendered as pictures by the server), the Web Coverage Service provides available data together with their detailed descriptions; defines a rich syntax for requests against these data; and returns data with its original semantics (instead of pictures) which may be interpreted, extrapolated, etc., and not just portrayed. Unlike OGC Web Feature Service (WFS), which returns discrete geospatial features, the Web Coverage Service returns coverages representing space/time-varying phenomena that relate a spatiotemporal domain to a (possibly multidimensional) range of properties. As such, WCS focuses on coverages as a specialized class of features and, correspondingly, defines streamlined functionality.

5 Transversal components

This section describes two components of the HiSea platform used by all other parts of the platform, Logging and User Management.

5.1 Authentication and user management

A Single Sign-On (SSO) server will be used to authenticate and authorize users to access the platform. The front end of the application will have a dedicated web page for user registration which stores data in an Lightweight Directory Access Protocol (LDAP) server. Each user will have an associated profile. This profile will be retrieved by the different end-user services which will manage the authorization to data.

5.1.1 GDPR

Within the protection of personal data regulatory framework, HISEA is obliged to protect in the best way all personal data processed and assumes responsibility for its position as processing manager (“controller”).

5.2 Logging and monitoring

In a microservices architecture, it can be especially challenging to pinpoint the exact cause of errors or performance bottlenecks. A single user operation might span multiple services. Services may hit network I/O limits inside the cluster. A chain of calls across services may cause back pressure in the system, resulting in high latency or cascading failures. Moreover, generally it is not possible to know which node a particular container will run in. Containers placed on the same node may be competing for limited CPU or memory.

Collecting metrics and logs is fundamental for a stable platform. Particularly:

- Logs are text-based records of events that occur while the application is running. They include things like application logs (trace statements) or web server logs. Logs are primarily useful for forensics and root cause analysis;
- Metrics are numerical values that can be analyzed. They are used to observe the system in real time (or close to real time) and to analyze performance trends over time.

Two options are available for logging, either log from within each individual service or use a central service (*see* **שגיאה! מקור ההפניה לא נמצא.**).

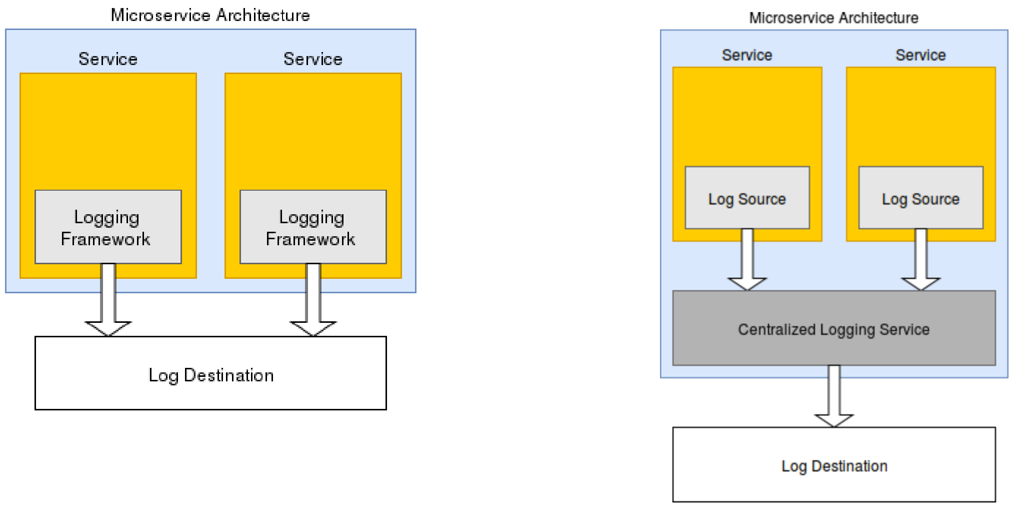

FIGURE 2 – LOGGING OPTIONS

The first one is faster to implement however it might be harder to maintain sinceeach service to be implemented needs its own logging methods. The latest increases the difficulty of changing the logging behavior across multiple services.

In the central service approach, services still need a way to generate logs, but the logging service is responsible for processing, storing or sending logs to a centralized log.

The choice will depend largely on the final number of services that will compose the platform (each of the described components of 3-Overall Technical Architecture will generate multiple services). Metric collection will depend largely on the implementation and orchestration of the HiSea platform (Kubernetes, Docker Swarm) and will be addressed at a later time. However, the log process should:

1. Appends a unique identifier to requests;
2. Identifies the service that generates events;
3. Generates events specific to each microservice as the request enters and leaves the service;
4. Correlates events from each microservice to create an overall view of the request.

The result is a series of log events that contain (atminimum):

1. A request ID, which lets us trace a single request across our architecture;
2. A service ID, which uniquely identifies the service that generated the event;
3. Unique content associated with the event, such as messages, severity levels, class and method names, and stack traces.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 821934