CO-LOCATION OPPORTUNITIES FOR RENEWABLE ENERGIES AND AQUACULTURE FACILITIES, DECISION SUPPORT FOR OPERATIONAL MULTI-USE PLATFORM ACTIVITIES AT COASTAL AREAS"

(RENAQUA Decision Support System)

System Architecture of the RENAQUA Service

15th January 2020

INDEX

List of Acronyms

DIAS	Data Information Access Service
DSS	Decision Support System
HSOM	Health - Safety and Operational Maintenance
IH-MSP	IHCantabria's Platform for Marine Spatial Planning
M2M	Machine to machine communication
Metocean	meteo-oceoanographic
MINECO	Spanish Ministry of Economy, Industry and Competitiveness
MITECO	Spanish Ministry for the Ecological Transition
MRE	Marine Renewable Energy
MUP	Multi-Use Platforms
NCEP	National Centers for Environmental Predictions
NCML	NetCDF Markup Language
NGA	National Geospatial Intelligence Agency
NOAA	National Oceanic and Atmospheric Administration
O&M	Operations & Maintenance
OGC	Open Geospatial Consortium
RENAQUA	RENewable energies and AQUAculture facilities
SDGs	Sustainable Development Goals
TDS	THREDDS Data server
UCLA	University of California, Los Angeles
UI	User Interface
UX	User Experience
WCS	Web Coverage Service
WEkEO	We knowledge Earth Obsevation
WMS	Web Map Service

1 INFRASTRUCTURE

Operational Systems require having a robust and reliable infrastructure. The RENAQUA Service is mainly hosted at IHCantabria's Data Center and partially at the WEkEO DIAS Platform. Figure 1 shows the main sections of the current IHCantabria's System Infrastructure, highlighting the virtual infrastructure that holds the development and production environments. The development or preproduction Servers provide the required environment to design and test the software developments, whereas the Production environment releases, under a versioning control system, mature versions of the Systems.

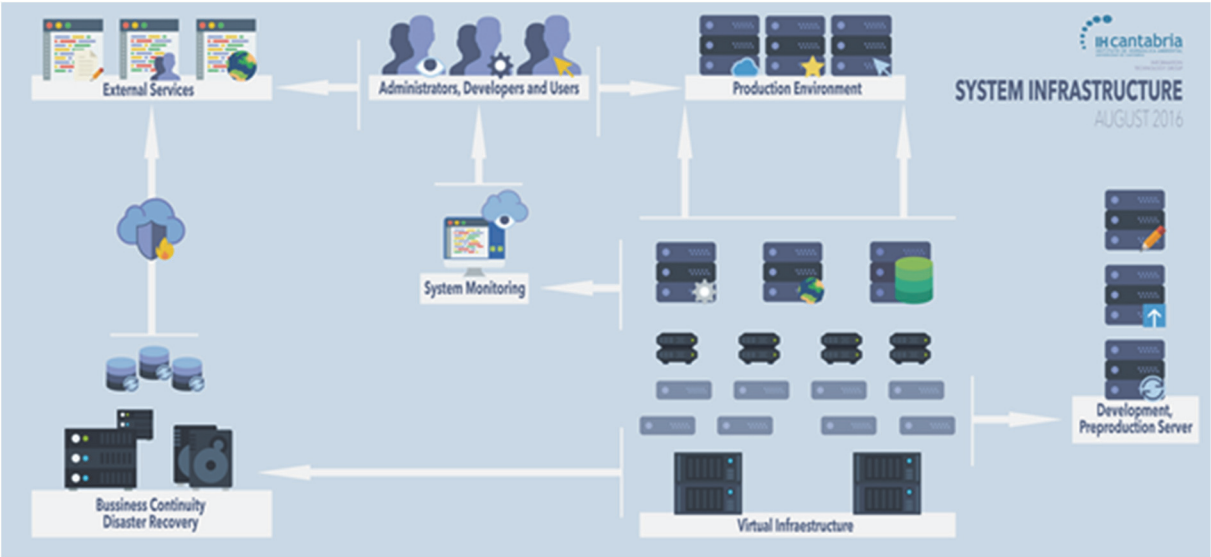

Figure 1. System Infrastructure

Virtualization is a proven software technology that makes it possible to run multiple operating systems and applications on the same server at the same time. Virtual machines act like a real computer but the software executed on these machines is separated from the underlying hardware resources. Virtualization can increase IT agility, flexibility, and scalability while creating significant cost savings. Workloads get deployed faster, performance and availability increases, and operations become automated.

Main components of the virtualized infrastructure are showed in Figure 2. The infrastructure is composed of numerical analysis and processing components, orange elements, storage and management, blue elements, and user interfaces, purple elements.

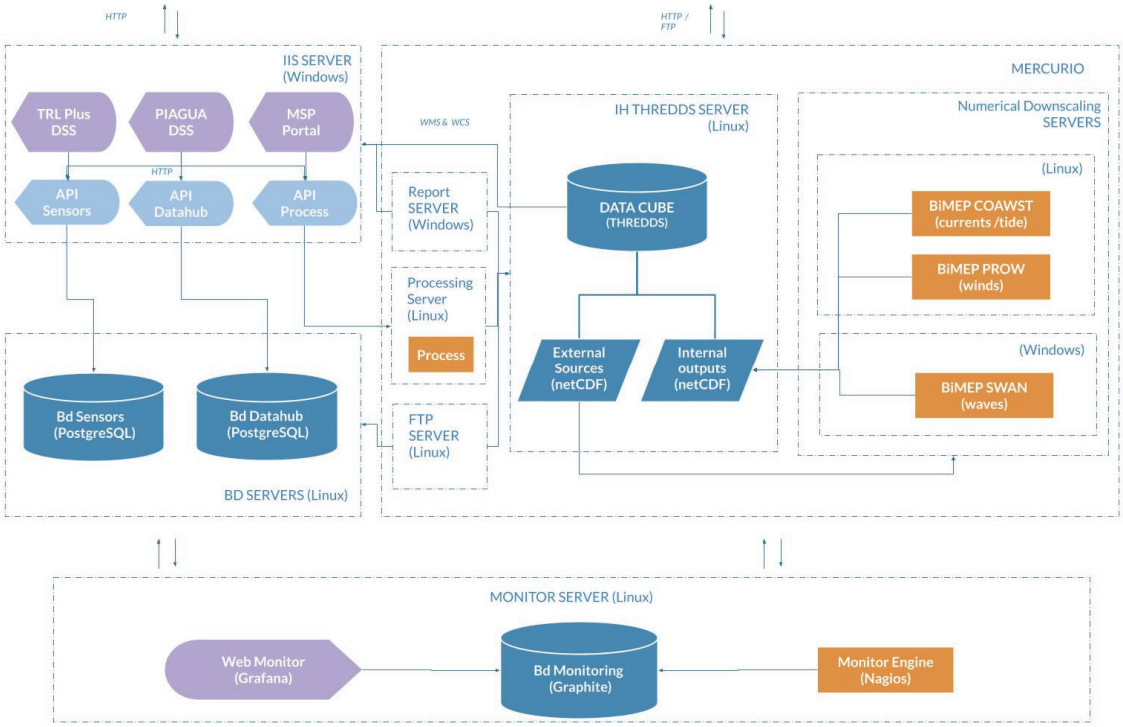

Figure 2. Main components of the Infrastructure

Numerical analysis and processing components:

- BiMEP COAWST – Numerical modelling of currents and tides.
- BiMEP PROW – Statistical modelling of winds.
- BiMEP SWAN – Numerical modelling of waves.
- Process – Analytical processes for tailored information products.
- Monitor Engine (Nagios) – Monitoring of the Infrastructure, numerical modelling. (inputs and outputs) and processes.

Storage and management:

- DATA CUBE (THREDDS) – Manages all the NetCDFs.
- Sensors BD – Manages the information collected from *in situ* sensors.
- Datahub BD – Manages all the products stored in NetCDF format.
- Monitoring BD – Manages all the data collected by the Monitor Engine.
- Sensors API – Interoperability access to the data stored at the Sensors relational database.
- Data hub API – Interoperability access to the data stored at the Datahub relational database.
- Process API – Interoperability access to request processing analysis.

User Interfaces (UIs):

- TRL Plus DSS – UI of the Decision Support System for the BiMEP Platform.
- PIAGUA DSS – UI of the Decision Support System for the PIAGUA Farm.
- MSP Platform – UI of the Decision Support System for Marine Spatial Planning.
- Web Monitor – UI of the Monitoring System.

The following subsections describe more in detail the technologies implemented, the APIs developed, the designed Monitoring System and the use of the WEkEO cloud Service.

2 TECHNOLOGIES

The Infrastructure of the RENAQUA Service is based on a cloud-based architecture. The architecture required to run the service is composed by two main sections: back-end and front-end. Figure 3 shows a graphical representation of the workflow between Front-end and Back-end sections for any standard Web application.

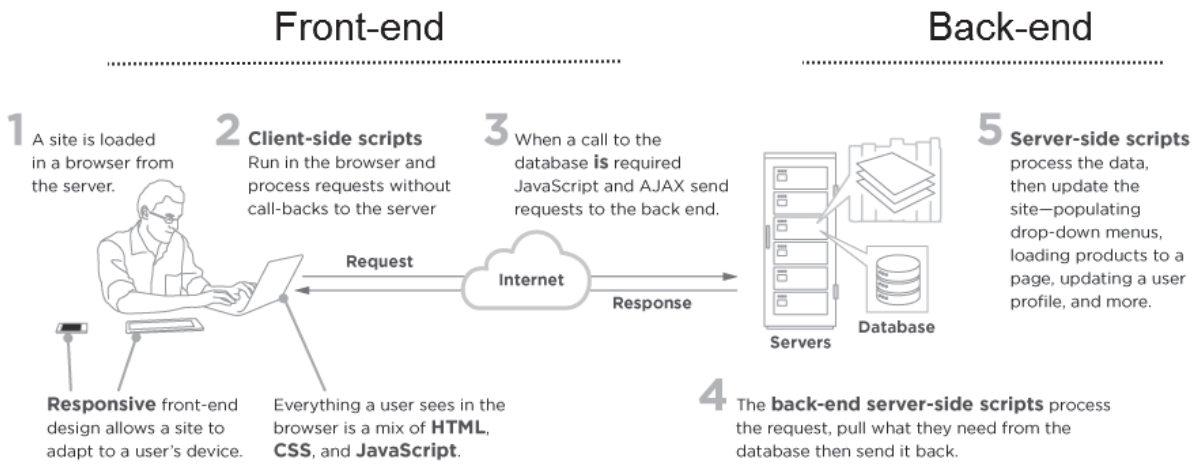

Figure 3. Front-end and back-end sections

The front-end includes the user interface and the client's computer system used for accessing the Geospatial information through interoperability protocols. In order to provide access through any standard Web browser (like Chrome or Firefox) three Web based applications were designed and developed. The document entitled "User Interfaces and User Experience of the RENAQUA Service", describes in detail the User Interface and User Experience of the Web applications developed.

On the other hand, the back-end includes servers, data storage system, virtual machines, backup system, processing system, monitoring system and the required software to provide interoperability protocols to provide access to the data. Back-end is in charge of gathering, perform the analytical and numerical modelling, manage data and provide interoperability protocols.

The main technologies used by the Systems are Javascript (front-end) and python (back-end), see Figure 4.

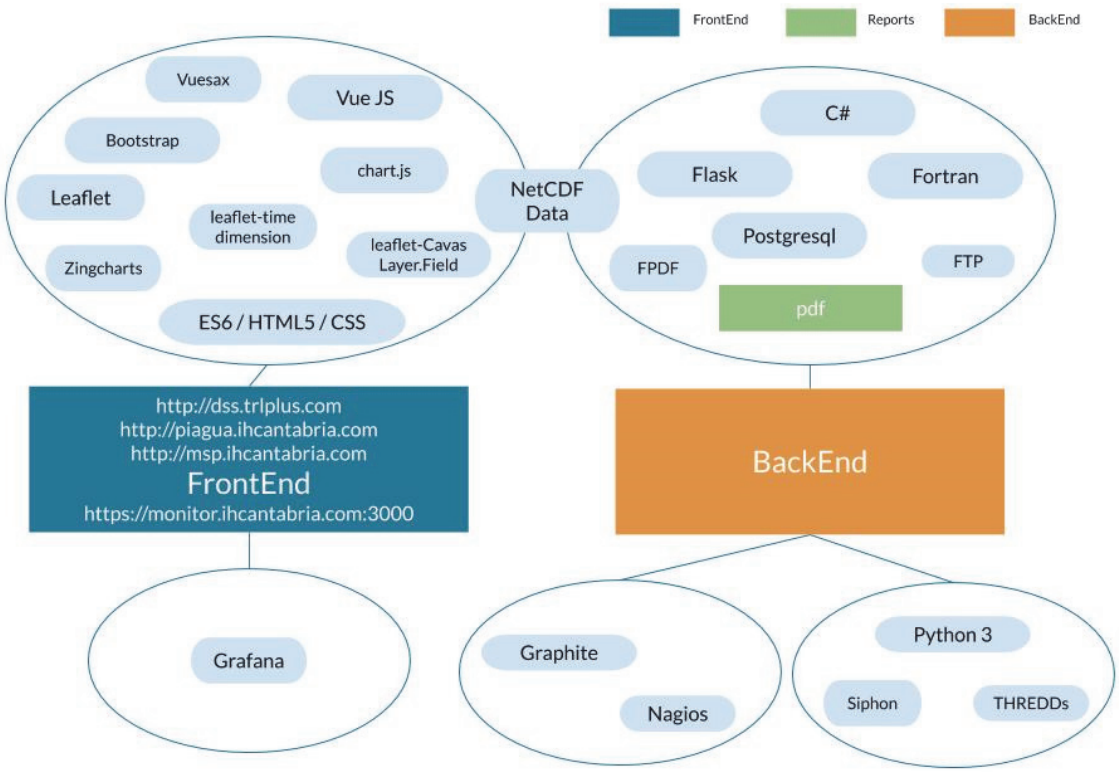

Figure 4. Front-end and back-end Technologies

The technologies are listed and described:

- *Vue JS* (https://vuejs.org/) is an open-source JavaScript framework for building user interfaces and is perfectly capable of powering sophisticated Single-Page Applications when used in combination with modern tooling and supporting libraries.
- *Vuesax* (https://lusaxweb.github.io/vuesax/) is a framework of components based on Vue.js. The framework is focused on facilitating the development of applications, improving the design of the same without removing the necessary functionality.
- *Leaflet* (https://leafletjs.com/) is an open-source JavaScript library used to build interactive web mapping applications. It works efficiently across all major desktop and

mobile platforms, can be extended with lots of plugins, has an easy to use and well-documented API and a simple, readable source code.

- *Boostrap* (https://getbootstrap.com/) is an open source CSS framework for front-end Web development.
- *Chart.js (*https://www.chartjs.org/) is an open source JavaScript charting for designers & developers.
- *Leaflet-time dimension* (https://github.com/socib/Leaflet.TimeDimension) is an open source plugin for leaflet that add time dimension capabilities on a leaflet map.
- *Leaflet-canvas Layer.Field* (https://github.com/IHCantabria/Leaflet.CanvasLayer.Field) is an open source plugin for leaflet that provides a dynamic visualization of rasters (geotiff and asciigrid) as vector field animations. The plugin was developed under the TRL Plus project.
- *Zingchart* (https://www.zingchart.com/) is one of the most feature-rich, fully customizable JavaScript charting library available. ZingChart is built with vanilla JavaScript, but it has integrations for including charts in projects built in Angular, Backbone, Ember, jQuery, and React. Zingchart is a simple JavaScript library for building responsive charts and dashboards.
- *ES6/HTML5/CSS3* JavaScript enables interactive web pages and thus is an essential part of web applications. The vast majority of websites use it, and all major web browsers have a dedicated JavaScript engine to execute it. Alongside HTML and CSS, JavaScript is one of the three core technologies of the World Wide Web.
- *NetCDF* (https://www.unidata.ucar.edu/software/netcdf/) is a Java library and self-describing, machine-independent data formats that support the creation, access, and sharing of array-oriented scientific data.
- *PostgreSQL* (https://www.postgresql.org/) is an open source relational database management system.
- *Python* (https://www.python.org/) is an interpreted, high-level, general-purpose programming language.
- *Fortran* is a compiled imperative programming language that is especially suited to numeric computation and scientific computing.
- *C #* is a programming language developed by Microsoft that runs on the .NET Framework.
- *FTP* or File Transfer Protocol is a standard network protocol used for the transfer of computer files between a client and server on a computer network.
- *Flask* (https://flask.palletsprojects.com/en/1.1.x/) extensible web microframework for building web applications with Python.
- *FPDF* (http://www.fpdf.org/) is a library built to write PDFs

- *Siphon* (https://unidata.github.io/siphon/latest/index.html) is a collection of Python utilities for downloading data from remote data services. Much of Siphon's current functionality focuses on access to data hosted on a THREDDS Data Server. It also provides clients to a variety of simple web services.
- *THREDDS* Data Server (https://www.unidata.ucar.edu/software/thredds/current/tds/) is a web server that provides metadata and data access for scientific datasets, using a variety of remote data access protocols.
- *Grafana* (https://grafana.com/) is an open source snalytics & monitoring solution for databases.
- *Nagios* (https://www.nagios.org/) is an open source server monitoring software.
- *Graphite* (https://graphiteapp.org/) is an open source tool that monitors and graphs numeric time-series data such as such as the performance of computer systems.

3 APIs

Application Programming Interfaces, commonly known by its acronym "APIs", are software elements that act as an abstraction layer, with a set of rules and specifications, which can be used by other software.

The RENAQUA Service has been designed to be accessed by other software through three APIs: Sensors API, Data Hub API and Process API. The following subsection describe each of them.

Sensors API

The Sensors API provides access to *in situ* observations, such as buoys or anemometers. Therefore, through the Sensors API any software developer or technician could access to the real-time data or historical data from in situ sensors. Figure 5 shows the UI to introduce the Sensors API capabilities (methods) to any developer or technician.

Figure 5.Web Interface of the Sensors API

Figure 6 shows the result, JSON format, of a query to the “LatestData” method collected by the anemometer installed at the BiMEP Platform.

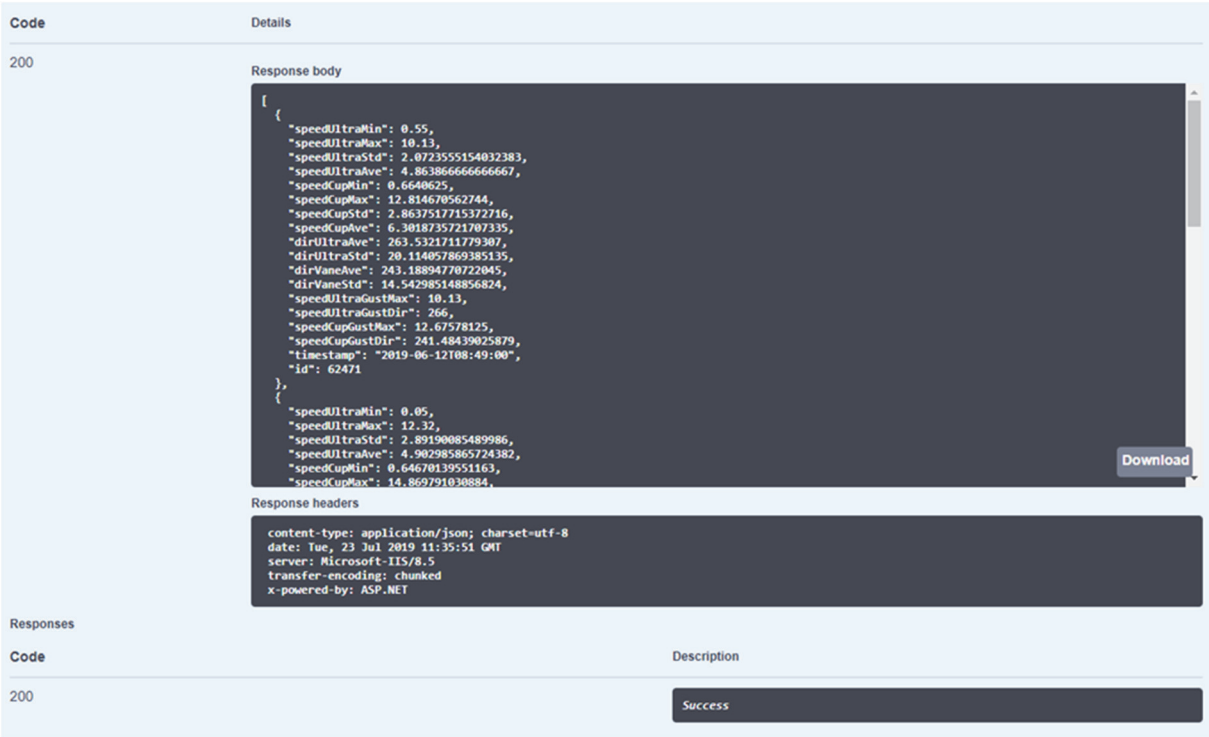

Figure 6. Query from the Sensors API

The Sensors API is based on a Relational database management system, PostgreSQL, that is in charge of the management of observations obtained from the sensors (buoys, gauges, anemometers). Figure 7 shows the relational data model for buoy observations.

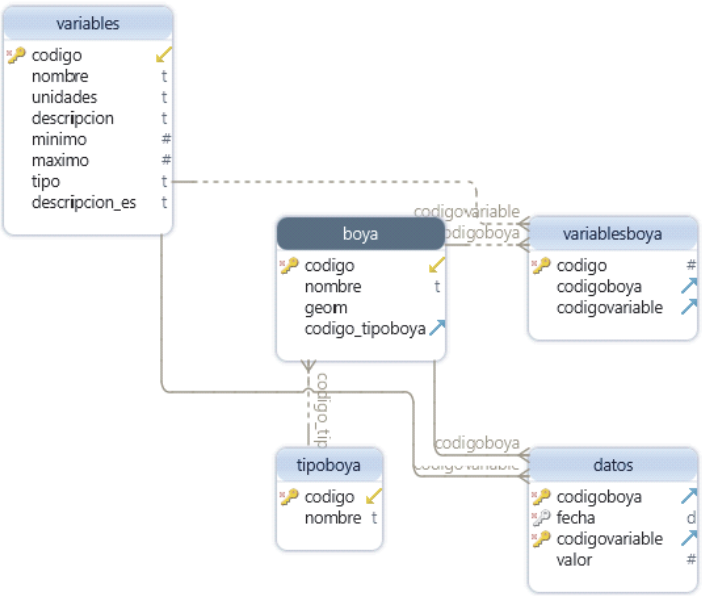

Figure 7. Sensors API – Buoys Relational Database Model

Data hub API

Data hub API facilitates the search and access to metadata of the operational metocean products, which are stored in NetCDF format. The technology implemented to facilitate interoperable access to files stored in NetCDF format is THREDDS Data Server (TDS), which provides geospatial standard protocols established by the Open Geospatial Consortium (OGC) such as WMS, WCS in addition to NetCDF Markup Language (NCML), NetcdfSubset, among others.

The meteocean information is stored as a “Product”. Products are obtained from different “Sources”, data providers such as CMEMS, NOAA, AEMET, etc. One metocean Product could host several “Variables”, which are characterized depending on the “Type of variable” and refer to a specific “Moment”, such as past conditions or short term forecast, among others. Finally, several “Clients” make use of the metocean Products, Clients are applications such as TRL Plus, PIAGUA and the MSP Platform. All these elements are accessible though the Data hub API, see Figure 8.

Figure 8. User Interface Data hub API

Figure 9 shows the result, in JSON format, of a query about the list of products stored in the data hub. Among the results, the characteristics or metadata of the product are listed: spatial, temporal resolution, URL access to the THREDDs catalogue, etc.

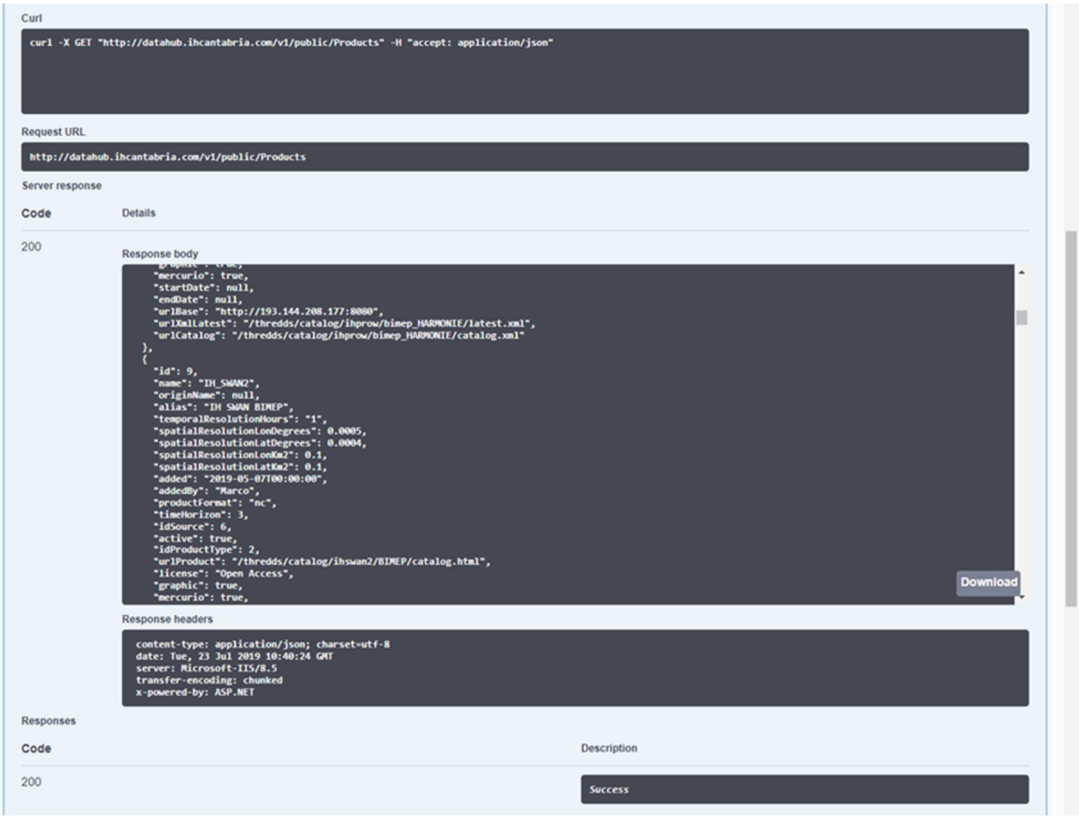

Figure 9. Query from the Data hub API

The API is based on a Relational database management system, PostgreSQL, that is in charge of the management of products, sources, clients, moments, variables and type of variables. Figure 10 shows the relationships among the different features.

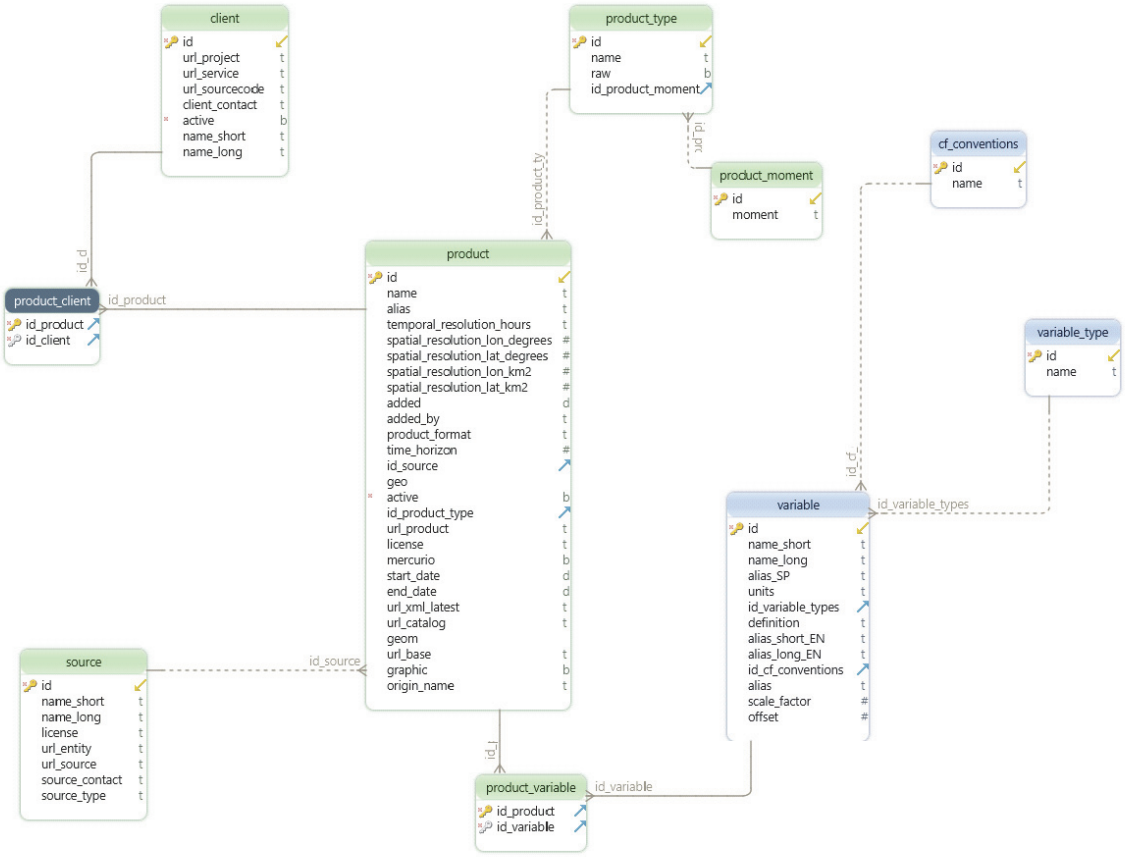

Figure 10. Data hub API Relational Database Model

<u>API Process</u>

The API Process is a set of programming interfaces that allow the request of analytical processes though interoperability protocols. The following programming interfaces have been created making use of flask technology:

- **Biological Suitability**. The Biological Suitability analysis requires the following parameters to obtain the % of suitability worldwide:
    - specie:
        - ‘name’: string | Name of biological specie
        - ‘salinity_max’: float | Maximum threshold of salinity for the specie growth

    - ‘salinity_min’: float | Minimum threshold of salinity for the specie growth
    - ‘temperature_max’: float | Maximum threshold of temperature for the specie growth
    - ‘temperature_min’: float | Minimum threshold of temperature for the specie growth
  - point:
    - ‘lon’: float | Longitude of point requested
    - ‘lat’: float | Latitude of point requested
  - dates:
    - ‘ini’: string (YYYY-MM-DD) | Starting date of period to consider
    - ‘end’: string (YYYY-MM-DD) | Ending date of period to consider

- **Wave Energy Suitability**. The Wave Energy Suitability analysis requires the following parameters to obtain the % of suitability worldwide:
  - config:
    - ‘hs_min’: float | Minimum threshold of significant wave height
    - ‘hs_max’: float | Maximum threshold of significant wave height
    - ‘tp_min’: float | Minimum threshold of peak wave period
    - ‘tp_max’: float | Maximum threshold of peak wave period
    - ‘cge_min’: float | Minimum threshold of available energy flux
  - point:
    - ‘lon’: float | Longitude of point requested
    - ‘lat’: float | Latitude of point requested
  - dates:
    - ‘start’: string (YYYY-MM-DD) | Starting date of period to consider
    - ‘end’: string (YYYY-MM-DD) | Ending date of period to consider

- **Wind Energy Suitability**. The Wind Energy Suitability analysis requires the following parameters to obtain the % of suitability worldwide:
  - config:
    - ‘hs_max’: float | Maximum threshold of significant wave height
    - ‘pow’: float | Minimum threshold of available potential
  - point:
    - ‘lon’: float | Longitude of point requested
    - ‘lat’: float | Latitude of point requested
  - dates:
    - ‘start’: string (YYYY-MM-DD) | Starting date of period to consider

- ‘end’: string (YYYY-MM-DD) | Ending date of period to consider

- **Wave 2 Wire** (W2W). The W2W analysis requires the following parameters to analyse the behaviour of a virtual device at the BiMEP Platform:
  - config Constant module:
    - 'mooring_dyn_type': string | Mooring type (dynamic/quasistatic)
    - 'wind_dyn_type': string | Wind type (dynamic/quasistatic)
    - 'wind_type': string | Wind type (uniform/constant/turbulent)
    - 'wave_height': float | Hs
    - 'wave_period': float | Tp
    - 'wave_heading': float | Direction regarding the platform
    - 'vx_wind': float | wind component x
    - 'vy_wind': float | wind component y
    - 'vz_wind': float | wind component z
    - 'email': string | email to send report
  - config Uniform module:
    - file: file | CSV winds values
    - 'mooring_dyn_type': string | Mooring type (dynamic/quasistatic)
    - 'wind_dyn_type': string | Wind type (dynamic/quasistatic)
    - 'wind_type': string | Wind type (uniform/constant/turbulent)
    - 'wave_height': float | Hs
    - 'wave_period': float | Tp
    - 'wave_heading': float | Direction regarding the platform
    - 'email': string | email to send report
  - config Turbulent module:
    - 'mooring_dyn_type': string | Mooring type (dynamic/quasistatic)
    - 'wind_dyn_type': string | Wind type (dynamic/quasistatic)
    - 'wind_type': string | Wind type (uniform/constant/turbulent)
    - 'wave_height': float | Hs
    - 'wave_period': float | Tp
    - 'wave_heading': float | Direction regarding the platform
    - 'strenght': string | Strength wind (A, B, C)
    - 'speed': int | Speed wind (8, 12, 16, 20)
    - 'dir': int | Platform orientation
    - 'email': string | email to send report

- **Accessibility**.The Accessibility analysis requires the following parameters to analyse the accessibility conditions at the BiMEP Platform:
    - point:
        - 'lon': float | Longitude of point requested
        - 'lat': float | Latitude of point requested

4 MONITORING SYSTEM

The infrastructure is being monitored 24 hours a day, 7 days a week, 365 days a year, ensuring business continuity and disaster recovery; with a Recovery Point Objective (RPO) and Recovery Time Objective (RTO) in accordance with the Service Level Agreement (SLA) for the services provided. The ability to find out what is happening on any operational system at any given time is crucial to provide a 24/7 Service. In this sense, Nagios is an open source system that offers monitoring and alerting services for servers, models, applications and services, see Figure 11.

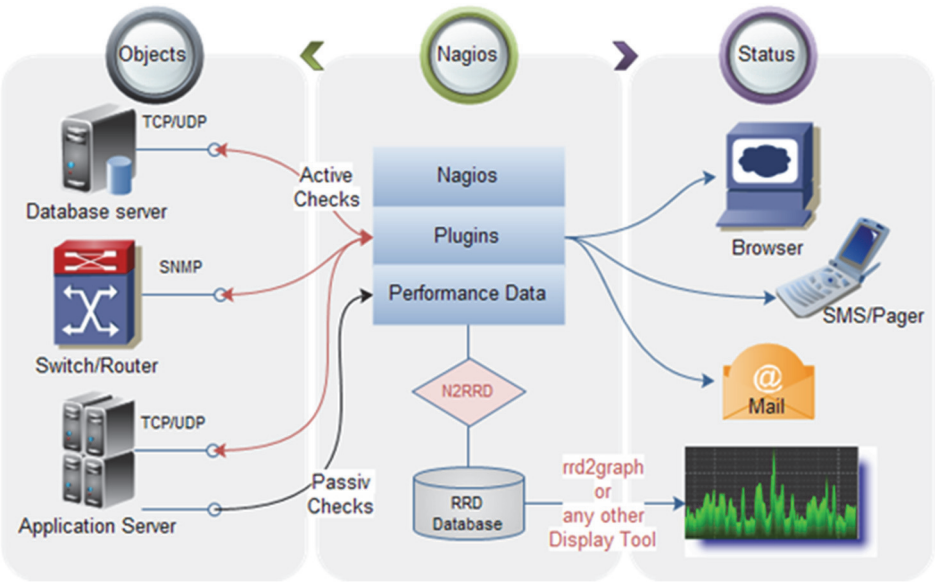

Figure 11. Monitoring System based on Nagios

Therefore, the TRL Plus DSS, the PIAGUA DSS and the MSP Platform are constantly monitored. In order to provide access to the status of the Systems and all their components, a specific dashboard has been designed and developed. The dashboard of the Monitoring System, see Figure 12 showing the BiMEP operational status, provides information about the lots of metocean data (wind, waves and currents), as well as information on the RAM System, CPU, storage, etc.

Figure 12. Dashboard of the Monitoring Service

In addition, the Monitoring System can be configured to receive alerts via mail or through team communication channels, such as slack, see Figure 13.

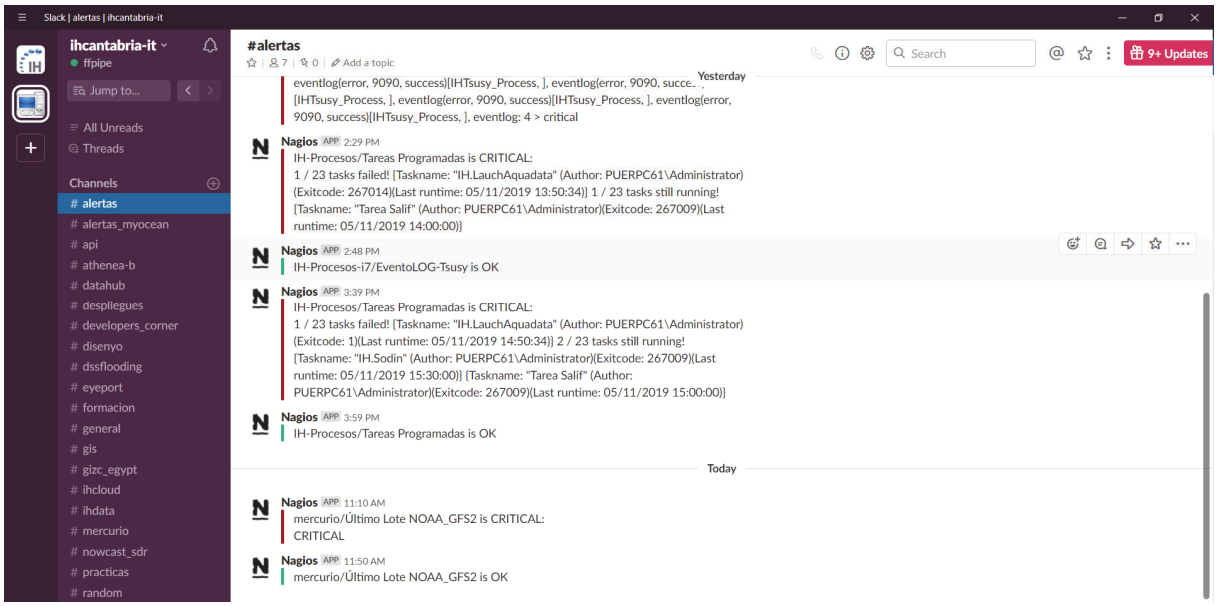

Figure 13. Nagios alerts integrated with Slack

5 WEKEO

Copernicus launched an initiative to facilitate access to Copernicus data and information services called "Data Information and Access Service" (DIAS) on the 20th of June 2018. DIAS is designed to improve users' ability to access as well as process Copernicus data and information by standardizing access to data through five cloud-based platforms: CREODIAS, MUNDI, ONDA, SOBLOO and WEKEO.

The RENAQUA project was selected for the WEkEO Beta testing programme. Two virtual machines were requested to test the RENAQUA Service, see Figure 14.

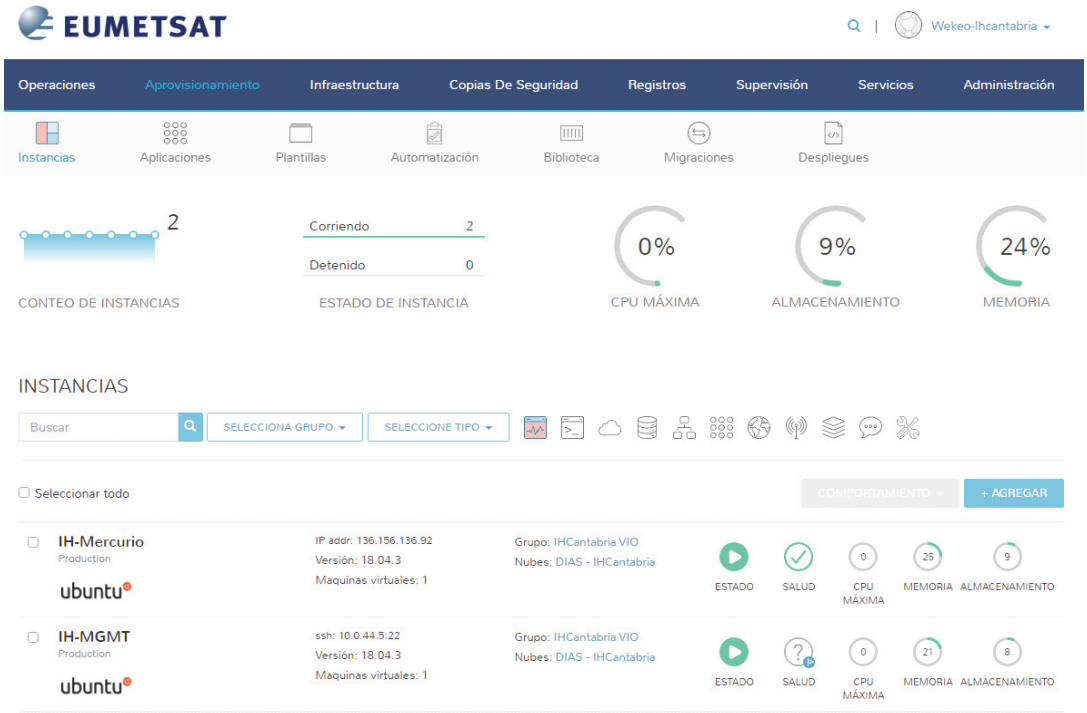

Figure 14. Virtual Machines at the WEkEO Platform

One of the main sections of the BackEnd architecture designed at the RENAQUA Project is the collection, transformation and publication of metocean products through standard interoperability protocols. The middleware in charge of the ETL (Extract, Transform & Load) and interoperability is called “Mercurio”, which collects meteocean data from different data providers, standardizes and shares them making use of THREEDs technology. Then, different Decision Support Systems are able to make use of the standardized products through interoperability access (WCS, WMS, etc.). Mercurio was deployed at the WEkEO platform, see Figure 15, providing access to metocean product from different sources (CMEMS, NOAA, meteoGalicia, Puertos del Estado, etc.).

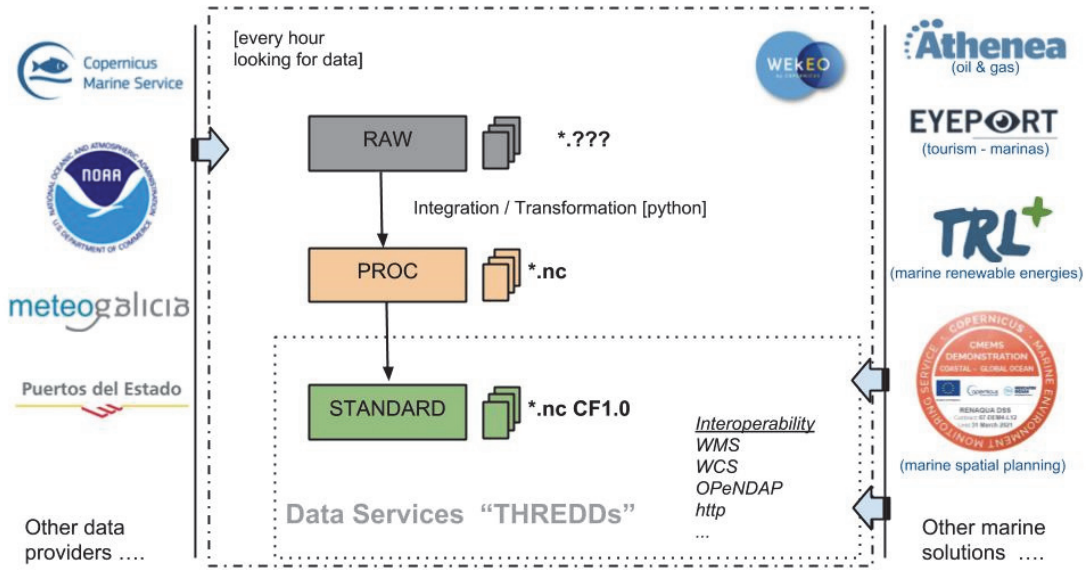

Figure 15. Architecture at WEkEO platform

The deployment of Mercurio at the WEkEO platform allowed analysing its performance. The results of such analysis are listed:

- Scalability is ensured
- It provides infrastructure as a service
- The access to the Copernicus products was twice faster from the WEkEO platform compared with the access from our Data Center.
- The harmonized data access facilitates the use and access of the Copernicus products.
- The platform is not mature enough for production.
- There is no Windows operative system option yet.
- There is a lack of data sets.

The insights obtained from the testing were shared by Enrique Álvarez from *Puertos del Estado*, see Figure 16, at the Data Access and Contributing Mission Workshop that took place in Paris, April 2019.

Figure 16. Feedback provided to the DIAS meeting

In relation with the RENAQUA Service, the main feature that WEkEO provides to the Service is the fast access to Global products. All metocean products must be transformed and standardized in order to be shared through interoperability protocols (THREDDs Data Server). However, Global Products can be extremely large to perform, in a timely and operational manner, such transformations. In this sense, the MSP Platform points to the CMEMS Global products that are currently being collected, standardized and shared at the Mercurio hosted at the WEkEO Platform.