Finanziato dall'Unione europea NextGenerationEU
Ministero dell'Università e della Ricerca
Italiadomani PIANO NAZIONALE DI RIPRESA E RESILIENZA
TECH4YOU

SPOKE 6

Digital transformation and technology transfer

PP 6.1.1 TITLE:

Digital solutions matching the needs of thematic spokes

SCIENTIFIC REFERENT	*Maurizio Cosmai*
Partner	*Engineering Ingegneria Informatica S.p.A.*

REPORT PP 6.1.1

MILESTONE I	*M12*
DELIVERABLE D1.1.2	*Design document for the enabling platform (detailing architecture, standards, core technologies)*
Report Title	-

Index

Versions History

Version	Authors	Date	Remarks
1.00	Engineering Ingegneria Informatica Spa	30/03/2024	First version

1 APPLICABLE & REFERENCE DOCUMENTS

1.1 Applicable

[AD1]. SPK1_Pilot_Crati_Model_update_18-10-2023
[AD2]. SPK1_Pilot_Fires_Model_update_18-10-2023
[AD3]. D2.1.6 Initial modelling of applications

1.2 Reference

[RD1]. UML specs (https://www.uml-diagrams.org/class-reference.html)
[RD2]. OGC (https://www.ogc.org/)
[RD3]. Spatiotemporal Asset Catalog (https://stacspec.org/en/)
[RD4]. OpenEO (https://openeo.org/)
[RD5]. NGSI-v2 Fire experience protocols (fiware-tutorials.readthedocs.io)
[RD6]. Device Measurements (https://github.com/smart-data-models/dataModel.Device/)
[RD7]. Geoserver (https://geoserver.org/)
[RD8]. Apace Airflow project (https://airflow.apache.org)
[RD9]. Orion Context Broker (https://fiware-orion.readthedocs.io/en/master/)
[RD10]. EA Project (Tech4You_plaftorm_design_model.eap)
[RD11]. T4Y back-end open_Api (t4y_api.yaml)

1.3 Acronyms

- COTS Commercial of-the-shelf
- ENG Engineering Ingegneria Informatica SpA
- EP Enabling Platform
- FOSS Free and Open-Source Software
- SU Service User (actor)
- OGC Open Geospatial Consortium
- PP Pilot Project
- PP Project Pilot
- SM Service Manager (actor)
- SO Service Operator (actor)
- T4Y Tech4You
- TBC To Be Completed
- TBD To Be Defined
- TBV To Be Verified
- UC Use Case
- UML Unified Modelling Language
- VAP Value Added Product
- WMS Web Map Service

1.4 Figures index

2 INTRODUCTION

This present document has the objective to describe the logical and physical architecture of the Tech4You Enabling Platform (EP).

The main goal of the Enabling Platform is to integrate a framework with a set of tools, able to support every Value-Added Monitoring service (Thematic Service) provider to generate and publish new content.

The document describes the core components and the methods of interfacing and integrating vertical services to them.

The structure of the document includes the following sections:

- The section 3 describes the Enabling platform general features.
- The section 4 describes the actors of the EP.
- The section 5 describes the significant application scenarios (use cases) of the EP.
- The section 6 describes the architecture design.
- The Section 7 describes the design from the behavioural point of view.
- The Section 8 describes the architecture from the physical point of view.
- The Annex A list the platform requirements.
- The Annex B shows the traceability matrix between requirements and design.
- The Annex C, lists all the interface specifications provided by the EP.
- The Annex D summarize the core and the thematic services available in the platform.
- The Annex E shows a physical schema with the technological involved tools.

2.1 Introduction to UML

To describe the system, it will be used the UML Language. The UML allows to better understand the static and dynamic components of the system. Follows a brief description of UML. Hereafter a breef description is made.

Unified Modelling Language, commonly known as UML, is a versatile and standardized visual language used to model the architecture, design, and implementation of software systems. UML encompasses a range of diagrams and notations that software developers and system architects utilize to depict the structure and behaviour of software projects. It serves as a lingua franca in the software development industry, ensuring clear communication and understanding among varied stakeholders, from developers to business analysts.

One of the primary advantages of UML is its ability to present complex software designs clearly and coherently. By providing a graphical representation, UML allows for a more accessible understanding of how system components interact and are structured. This not only aids in the

conceptual phase but also provides a guide during the development and subsequent documentation stages. The visual aspect of UML is particularly beneficial in identifying potential issues early in the design process, thereby reducing errors and saving development time.

UML diagrams are various, each tailored to represent different aspects of a software system. Use Case diagrams, for instance, are pivotal during the requirements gathering phase, helping to clarify what users expect from the system. Class diagrams delve into the blueprint of the system, detailing the classes and relationships that make up the application's data model. Sequence diagrams and Activity diagrams take a closer look at the behavioural aspects, mapping out object interactions over time and the flow of control or data, respectively.

State diagrams are instrumental in charting the various states within a system entity, essential for dynamic and reactive systems. Lastly, Component and Deployment diagrams focus on the more concrete elements of software systems, specifying the makeup and distribution of physical components and how they are deployed within the system's infrastructure.

3 ENABLING PLATFORM FEATURES

The present chapter describes the Enabling Platform under consideration, and briefly describe the general architecture.

3.1 APPLICATION PURPOSE AND UNKNOLEDGEMENT

Main purpose of the Enabling Platform is to provide a set of core functionality to integrate value-added products within a general-purpose cloud-based platform.

The functionalities groups implemented in the platform (data acquisition, data processing, product publishing, products analysis, products cataloguing) will enable an *easy-to-implement* integration of new value-added thematic services, as provided by third parties.

Qualified definitions of the provided service are:

- THEMATIC SERVICES: value-added services integrated into the platform for Earth monitoring (landslides, fire monitoring, etc.)
- BASIC SERVICES: services provided by the Enabling Platform, that will allow the integration of the thematic services into the platform like data acquisition, data processing, data archiving, data consultation, data cataloguing.

The presented platform is then to be considered as *mash-up* of different open-source tools and facilities, strongly integrated with software "glue", that will give them a coherence and consistence from the user point of view. Every single tool will be described in the next sections.

Please note that the chosen tools could be updated accordingly to the evolution of the user needs and technologies during the project inception, analysis, and design phases.

4 ACTORS

In the present section a detailed description of the actors of the Enabling Platform is made.

As described in the below diagram, the Enabling Platform will provide the following reference users for access and use.

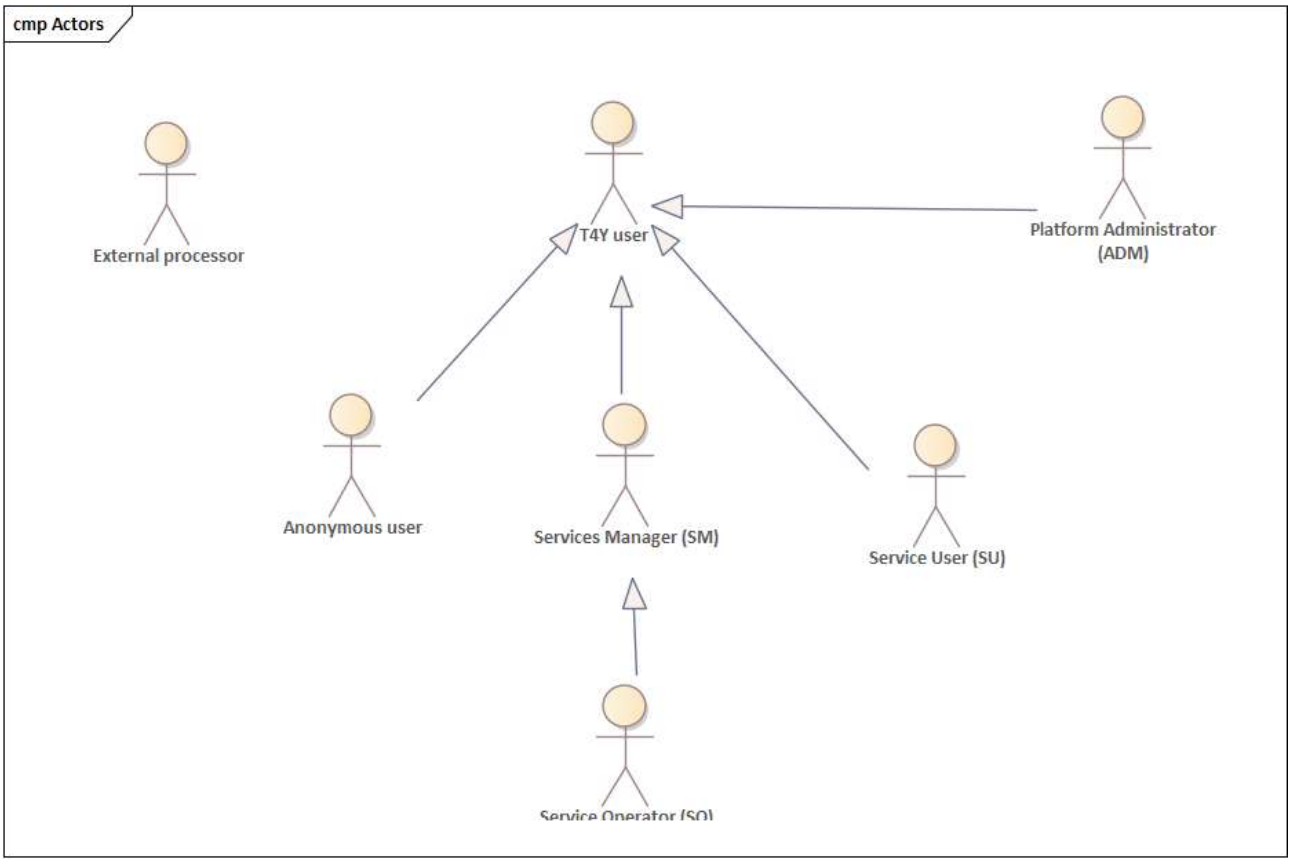


Figure 1 : T4Y Enabling Platform Actors

The actors on the EP are detailed hereafter.

4.1 Anonymous User

User who accesses the public pages of the Geoportal without a login. He accesses the static pages and partially in the geo-navigation page.

4.2 Service User (SU)

User who accesses the thematic services by means of the geo-navigator or analytics tools.

Abilities

- Access the geo-navigator.
- Access the service description.
- Access to smart analytics (trend graphs of measured quantities, etc.)
- Access to service messages
- Access to the platform's support services (error reporting, etc.).

4.3 Service Operator (SO)

The SO is the responsible of the execution of the processing, and the quality of them in the platform. He can run and stop processors, verify the processing status. Restart the workflows. He makes use of the facilities made available by the platform.

Abilities:

- Access own workspace.
- View Thematic services.
- View workflows
- Launch/Abort workflows.
- View processing status (jobs)
- Views processing logs
- Schedule a workflow

4.4 Service Manager (SM)

The SM is responsible of the Thematic Service integrated in the Enabling Platform. It's in charge of him the deployment in the platforms of the processors, their orchestration, and the input and output contents. He makes use of the facilities available in the platform to reach these goals.

Abilities:

- Access his workspace.
- Edit Thematic services.
- Edit workflows configuration.
- Launch/Abort workflows.
- View processing status (jobs)

- Views processing logs
- Schedule workflows
- Access the orchestrator (Airflow) environment.
- Access the deployment environment (via ssh)
- Access the Geospatial service environment.
- Edit the geo-web services to show in the TOC of the navigator.

4.5 Platform Administrator (ADM)

He has in charge of the user managements of the platform.

Abilities

- Users' management
- User profiling management
- Platform metering reports

4.6 External processor (EP)

A not human actor, that interacts with the platform to share Thematic Service products. He makes use of a platform generated key, and authentication protocols, and interacts with the platform using a specific platform protocol.

Abilities

- Accept processing request from the Platform.
- Send the completion status to the platform call-back.
- Send the status and products to the platform

5 USE CASES

In the present chapter a detailed description of platform scenarios has been made, as use cases.

5.1 UC-10 - THEMATIC SERVICES MANAGEMENT

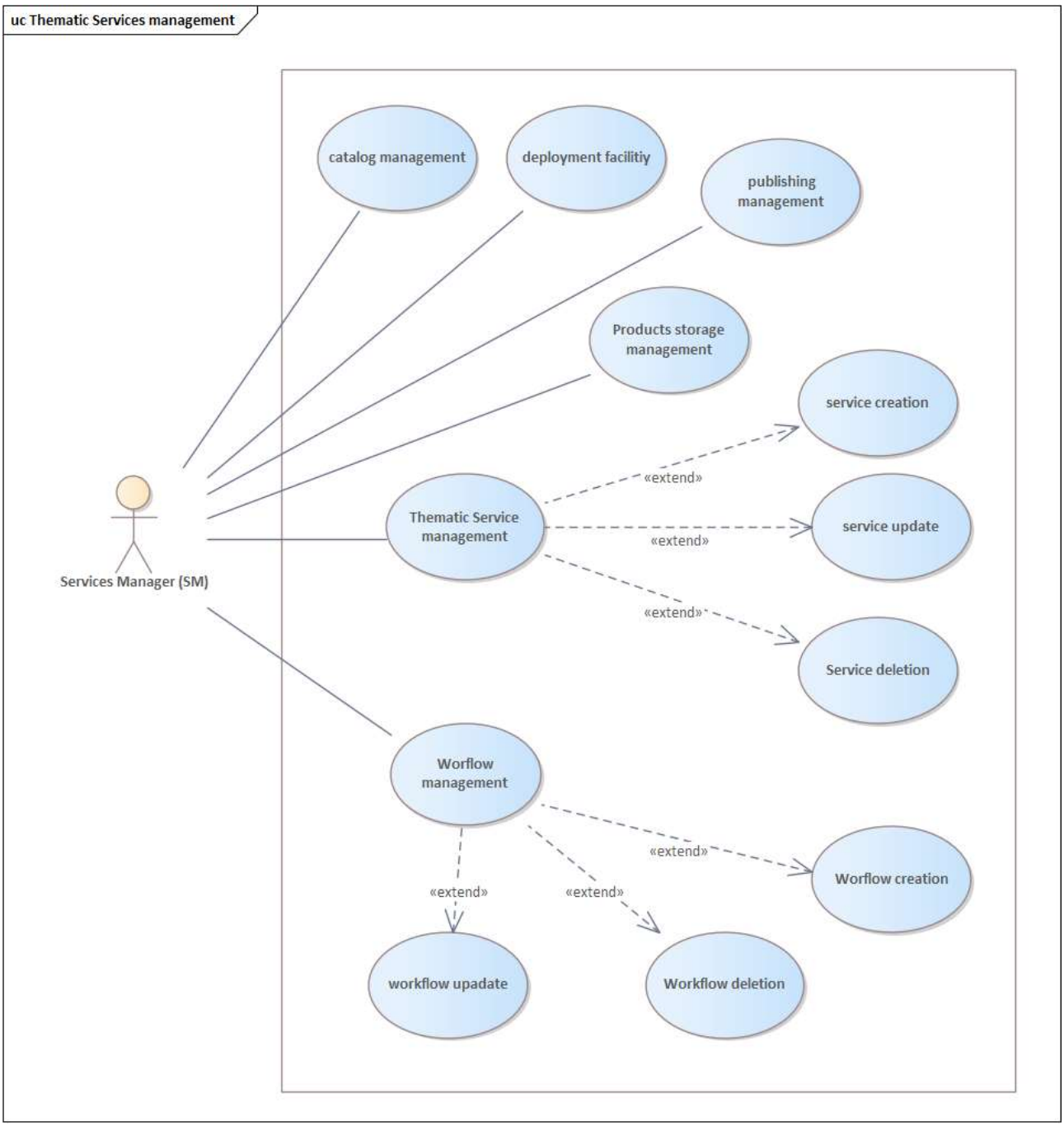


Figure 1 : Thematic service Authoring

5.1.1 UC-10-10: Thematic Service creation

Actors

- Service Manager (SM)

Preconditions

- The user is logged into the platform as Service Manager (SM).

Postconditions

- A new Thematic Service is created.

Description

In this scenario, the user creates a new thematic service. Please beware, parts of the steps could be made manually by the platform managers.

Steps

1. The user accesses the T4Y Geoportal and accesses his workplace.
2. The user registers a new THEMATIC service by entering the service metadata (title, description, etc.).
3. Upon confirmation of creation, the platform performs the following actions:
    - Creates a deployment user virtual machine on the application server to install processors (Docker containers with a specific template). Access is via SSH protocol.
    - Creates a new vector product database on the data server. By default, it will contain certain information such as logs and expected configurations.
    - Creates a new subfolder on the data server for saving products (shapefiles, json, raster, etc.).
    - Creates a new Geoserver workspace on the application server to manage data publishing (OGC-WMS, WMTS, WFS protocols).
    - Creates a new catalog table to manage the service metadata (OGC-Catalog Services for the Web - pyCSW).
    - Sets up an initial workflow project in Airflow, associated with the service (initially empty).
    - Creates a workspace on the T4Y Geoportal, where the following information will be included:
        - Access to input data from sensors (access via REST connection).
        - Access to the output data folder (access via FTP, WebDav, CLI S3 protocols).
        - Access to the output data database (via JDBC protocols and PostGIS client).
        - Access to the processor installation environment (via SSH).
        - The workflows, the processes in execution, the logs, external processors.
4. If step 3 is successful, a notification is sent to the user.
5. From this point on, the user (service manager) will be able to create new workflows associated with the created service.

Requirement reference

RQ-PLT-GEN-010 Thematic services integration

5.1.2 UC-10-20: Workflow creation (internal processor)

Actor

- Service Manager (SM)

Preconditions

- The user is logged into the platform as Service Manager (SM).

Postconditions

- A new processing workflow is created.

Description

In this scenario, the user crates a new workflow, integrating a processing script internally deployed as a docker container.

Steps

1. The user accesses the development area via SSH and creates a new docker processing container, according to the specifications set by T4Y (volumes, environment variables, parameters, container name, etc.).
2. The user accesses the Airflow workflow manager via the web and opens an existing project or creates a new one.
3. Through the authoring module, the user creates the processing chain, inserting the call to the container in the deployment environment, and after testing the call, saves the workflow.
4. The user accesses their T4Y workspace and creates a new workflow object by linking the project just created in Airflow.
5. The user launches the processing from the workspace or Airflow and waits for the outcome.
6. The Airflow processing job invokes the processor implemented as a service container.
7. The processor accesses REST services to extract data from sensors, processes the results, and saves them in the output folder. File management is its responsibility, so it must take care of deleting temporary data.
8. The processor generates the metadata file and saves it in the catalog (REST API service).

Requirement reference

RQ-PLT-GEN-050 New thematic services processing orchestration

5.1.3 UC-10-30: Workflow creation (external processor)

Actor:

- Service Manager (SM)

Preconditions

- The user is logged into the platform as Service Manager (SM).

Postconditions

- At the end of this scenario a new processing with an external processor has been executed.

Description

In this scenario, the user creates a processing workflow calling an external processor.

Steps

1. The user prepares an external processor to the platform that responds to precise REST execution protocols.
2. Accesses their workspace and adds the reference (endpoint) to the external processor's registry.
3. When the processor has finished, it must make a callback call to the context broker to confirm that it has finished with the outcome. The data will be saved in the shared folder.
4. The rest of the procedure is like the previous one.

Requirement reference

RQ-PLT-GEN-050 New thematic services processing orchestration

5.1.4 UC-10-40: Publishing of a new generated Thematic product

Actor:

- Service Manager (SM)

Preconditions

- The user is logged into the platform as Service Manager (SM).

Postconditions

- At the end of this scenario a new Thematic Service is published.

Description

In this scenario, the user creates a new thematic service.

Steps

1. The user accesses the own Geoserver workspace and create a new data store linked to the product dataset.
2. The user creates a new service layer (WMS), linked to a datastore.
3. The user accesses the “my Workspace” area and add the new endpoint to the TOC.
4. The user accesses the geoportal and selects the geo-navigator where they can view the data of interest in cartography.

Requirement reference

RQ-PLT-GEN-020 Geospatial navigation tool

5.2 UC-20 - WORKFLOW EXECUTION

The present section will describe the workflow processing execution scenarios. The user has in charge the running of the processing workflows.

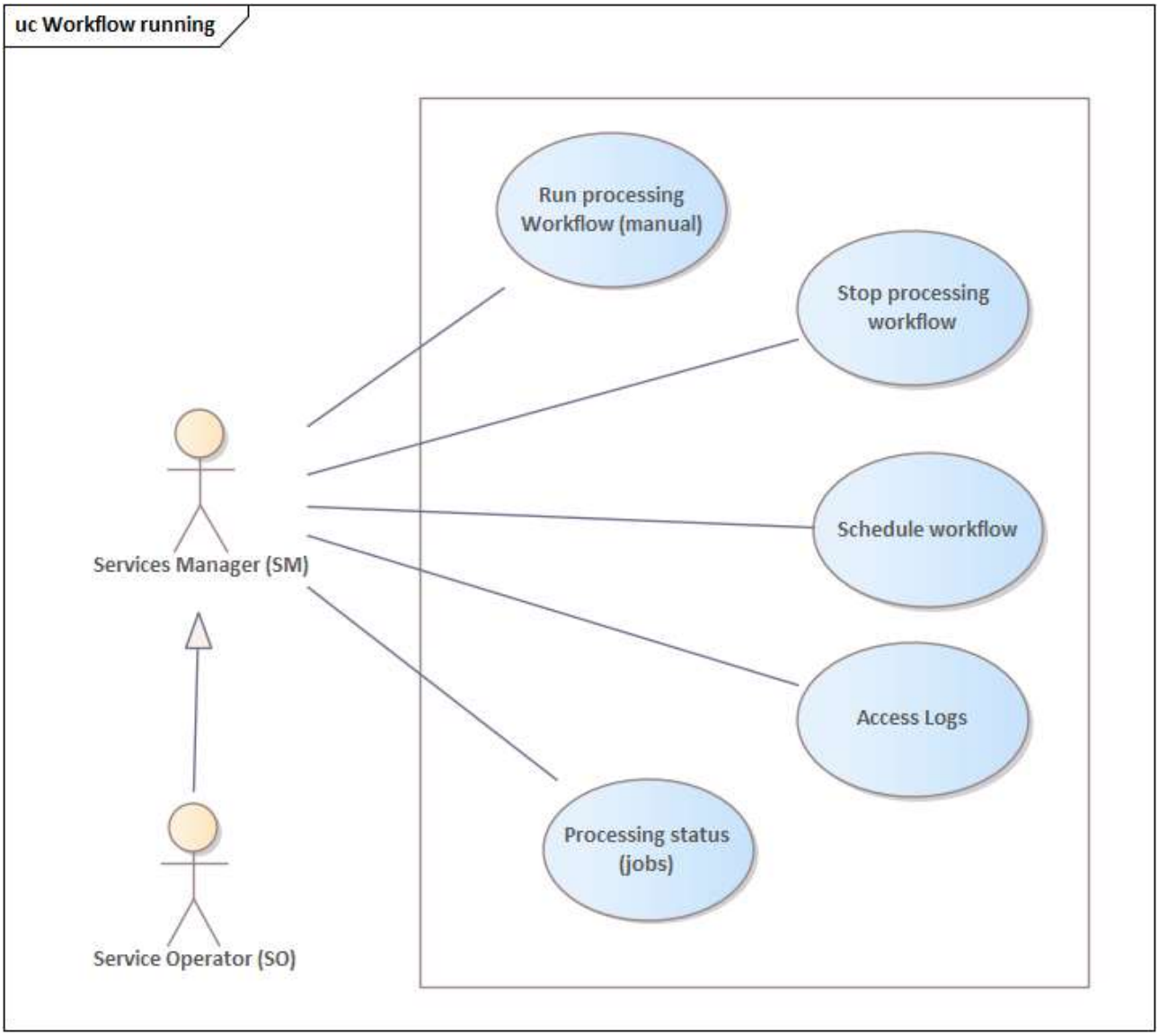


Figure 2 : Workflow execution scenario

5.2.1 UC-20-10: Run processing workflow (manual mode)

Actor

- Service Manager (SM)
- Service Operator (SO)

Preconditions

- The user is logged into the platform as Service Manager (SM).
- A processing workflow has been created in the platform.

Postconditions

At the end of this scenario a processing workflow has been executed.

Description

In this scenario, the user manually run a new processing workflow and stop it manually before the natural ending.

Steps

1. The user (Service manager) accesses his workspace (or directly Airflow), views the list of workflows associated with the service, and launches the workflow of interest.
2. The workflow is executed and returns log messages related to the status.

5.2.2 UC-20-20: Running a processing workflow (scheduled mode)

Actor:

- Service Manager (SM)
- Service Operator (SM)

Preconditions

- The user is logged into the platform as Service Manager (SM).
- A processing workflow has been created in the platform.

Postconditions

At the end of this scenario a processing workflow has been executed.

Description

In this scenario, the user manually run a new processing workflow and stop it manually before the natural ending.

Steps

1. The user (Service manager) accesses his workspace (or directly Airflow),
2. The user accesses the workflow associated with the service and select the schedule tool.
3. At a defined tile the platform launches the scheduled workflow.
4. The workflow is executed and returns log messages related to the status.

Requirement reference

RQ-PLT-GEN-020 Geospatial navigation tool

5.3 UC-30: DATA NAVIGATION

The present section, analyse in deep the Data Navigation scenario of the platform.

The aim of this scenarios is to access the UI where to navigate datasets and show the smart graphs that describe the status of the monitoring.

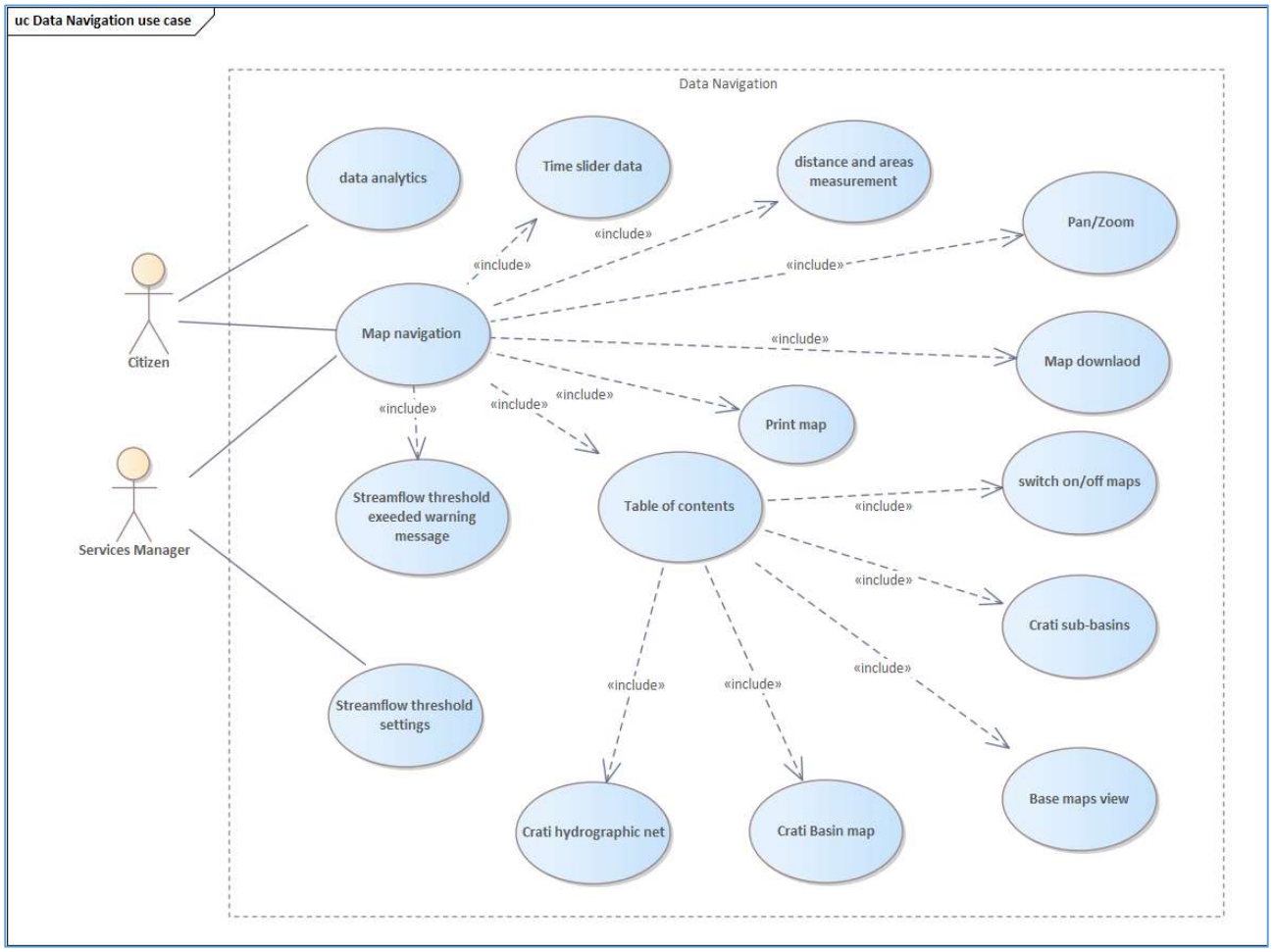


Figure 3 : Geodata Navigation

Main role: To user access data layers and navigate them using the webgis platform.

Description

This scenario includes the access to the user interface of the platform. It allows users to navigate cartography, extracting the results of the generated CRATI products. Furthermore, the platform will allow the user to receive an asynchronous warning message in case of exceeding the threshold defined by the service products.

An in-depth analysis of individual use cases follows:

5.3.1 UC-30-10: Data Geo-Navigation

Actors

all

Pre-Conditions

- The user is logged in.
- The user accesses the navigation map page.

Post-Conditions

- The navigates the t4y products.

Description

Access the geo navigator and view the output data.

Steps (main path)

1. The user selects a layer and activate it. The layer is shown in the map.
2. The user pan/zoom the map, the extent of the map changes.
3. The user selects the measure tool and draw a line. The measure of the length is shown.
4. The user accesses the Table of Content, select a layer, and activate it. The layer is shown in the map.
5. The user modifies the transparency of the current layer using a slider. The layer changes its transparency.
6. The user selects the legend tool, and the legend is shown.
7. The user selects and area and press the print tool. The platform shows the preview. The user confirms it. A pdf file is created and downloaded by the platform.

Requirements

RQ-PLT-GEN-020 Geospatial navigation tool

6 LOGICAL VIEW

The present section underlines the platform from a structural point of view. The architecture diagram shown in the Figure 5, represents a layered software platform for geospatial data integration and analytics, primarily in the context of IoT (Internet of Things).

6.1 CONTEXT ANALYSIS

From the context analysis point of view (see Figure below), we can consider the **T4Y Enabling Platform** as a black box, interacting with external actors according to specific application logics. Particularly, by analysing the inside of the black box, we can find a subcomponent responsible for the implementation of the thematic custom services present and managed by the platform.

These services are like plug-in components that must be inserted into specific guides for accessing the platform's services. Externally the platform, there will be interactions with human actors for the creation, management, and use of the services themselves.

There will also be software actors such as sensors, data catalogues, as well as external processing services that will have to interact with the platform according to codified standards.

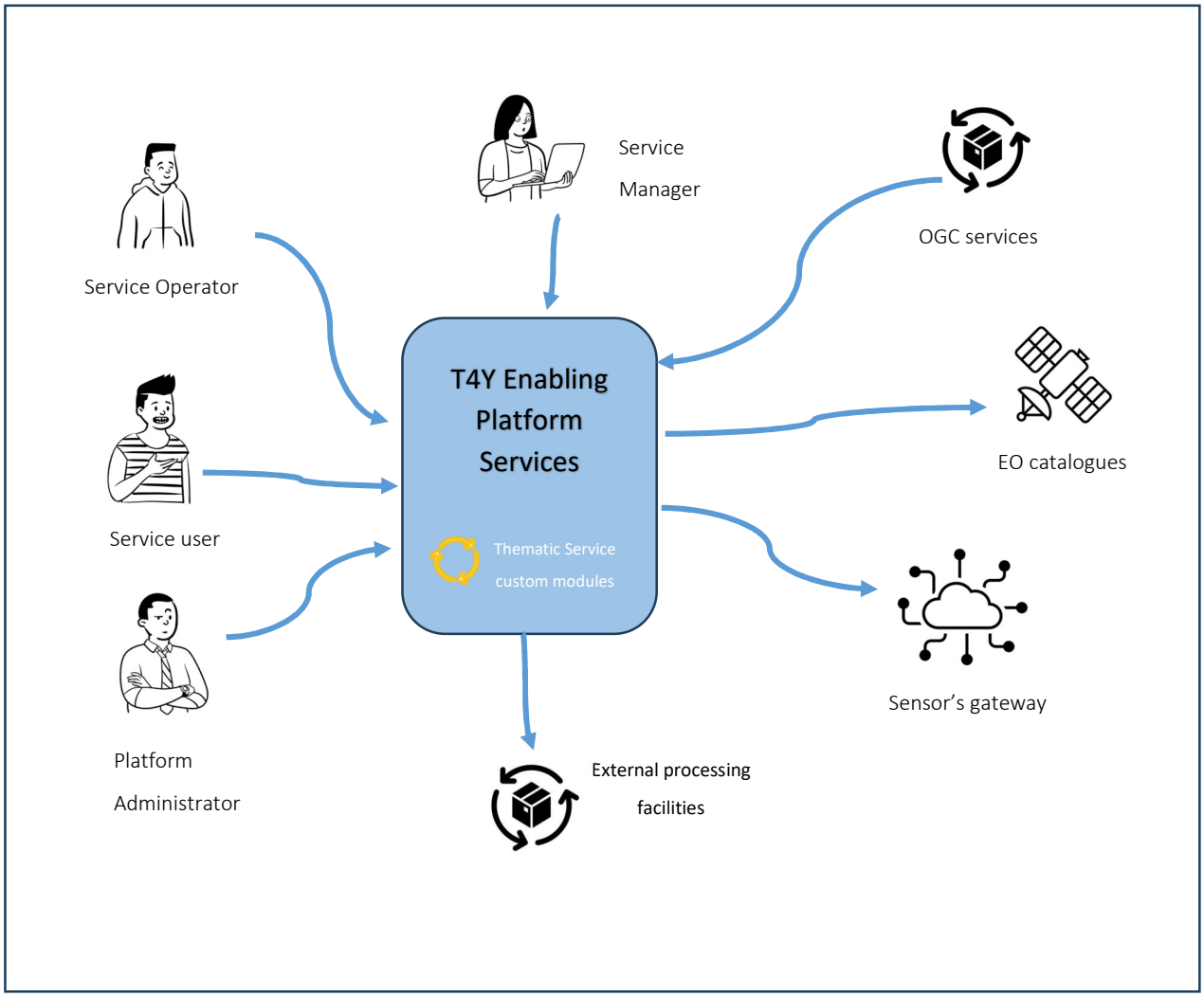


Figure 4 : Platform Context Diagram

6.2 MAIN ARCHITECTURE

At the bottom, the Data Collection Layer includes IoT gateways, connecting to satellites and in-situ gauging stations, managed by a device manager and IoT Agent.

Above it, the Data Management Layer consists of spatial and non-spatial databases, geospatial servers, and a context broker that handles historical context data, with an interoperability and data harmonization layer to ensure consistent data formatting.

The Data Processing Layer include the following modules:

- Processing modules
- Service registry
- Service orchestrator
- Service log

The topmost Application Layer includes user-facing elements such as management & control, field support, and citizen signalling modules, with a T4Y Northbound API (Open-API) for external interfacing.

Security and privacy are emphasized, as indicated by the DEV API section on the right, suggesting development and integration APIs are managed with these concerns in mind.

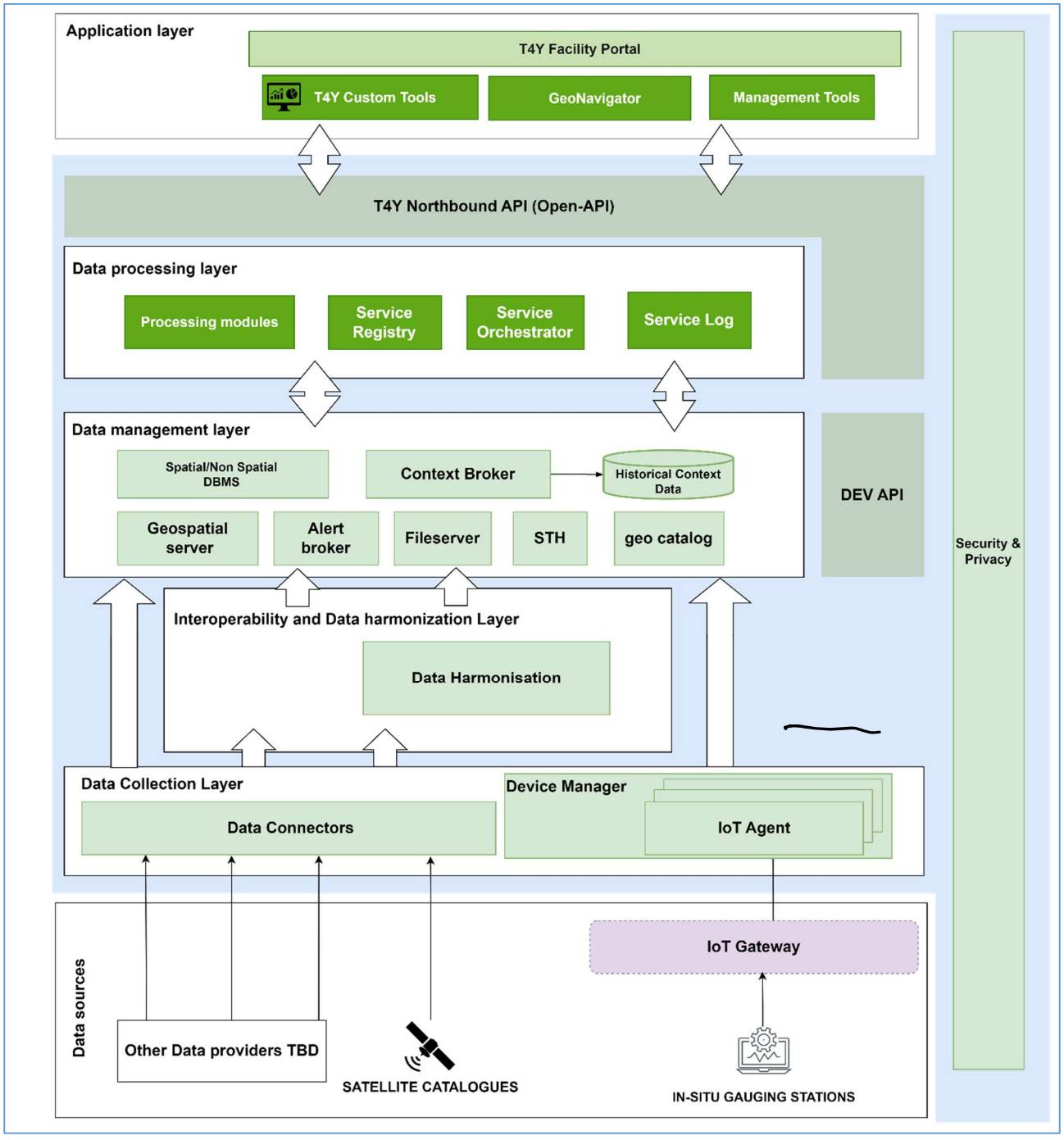


Figure 5 : Main architecture diagram

6.3 APPLICATION LAYER

At application layer level there are front-end and back-office features for platform actors. An approach will be provided that allows two reference figures for each service, the user, and the operator.

6.3.1 T4Y Geoportal (all)

The T4Y GeoPortal is the access point to the platform and to the tools for using the project's resources.

It provides general information on services available to anonymous users and vertical services for registered users. After sign-up on the platform and on a specific service, users will be able to access the specific features.

The geoportal can be seen as a content management tool, for the intended functionality and project pages. Anonymous users will be able to access to public contents. Registered users will be able to access management areas, for services, workflows, jobs.

The diagram below represents the page structure of the envisaged geoportal.

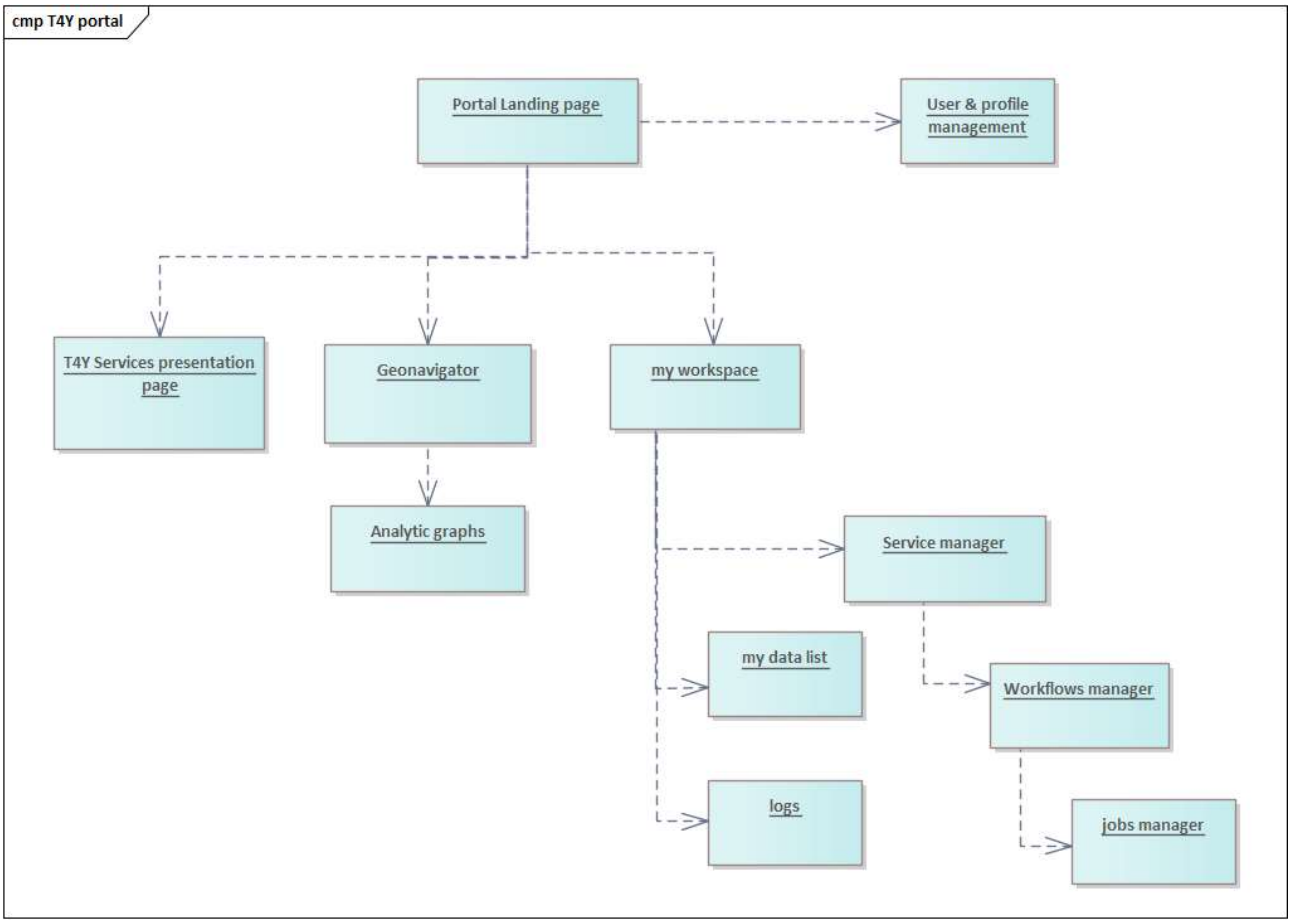


Figure 6 : T4Y Geoportal structure

There are many functionalities including presentation of services, cartographic services and data analytical consultation.

For registered users, pages are available such as a personal area for consulting their data. For service managers, service management pages, workflow management, job status consultation will be available. For the administrator there will be user management pages, profiling, and main master data.

In the personal area, the registered user will see what data is available to him and the progress of the processing.

Figure 2 - Artistic representation of the landing page of the T4Y geoportal

6.3.2 Services and workflow management (SM, SO)

As shown in Figure 7, the platform is a set of thematic service. Each of them is composed of one or more workflows. Every workflow is then associated with one job, the workflow process execution.

The thematic service is also composed of a set of geospatial layers to be shown as ancillary data.

At application level, a new service can be created by the platform upon the request of the external provider and "authoring" access assigned to the service manager. The service manager is then able to update details for the new service (title, description, request data, generated products, etc.).

Therefore, it must describe the service, link the associated workflows, and base layers to be integrated into the geo-navigator TOC. For this activity, they will use a UI or the T4Y REST services. Here follows a schematic description of the model.

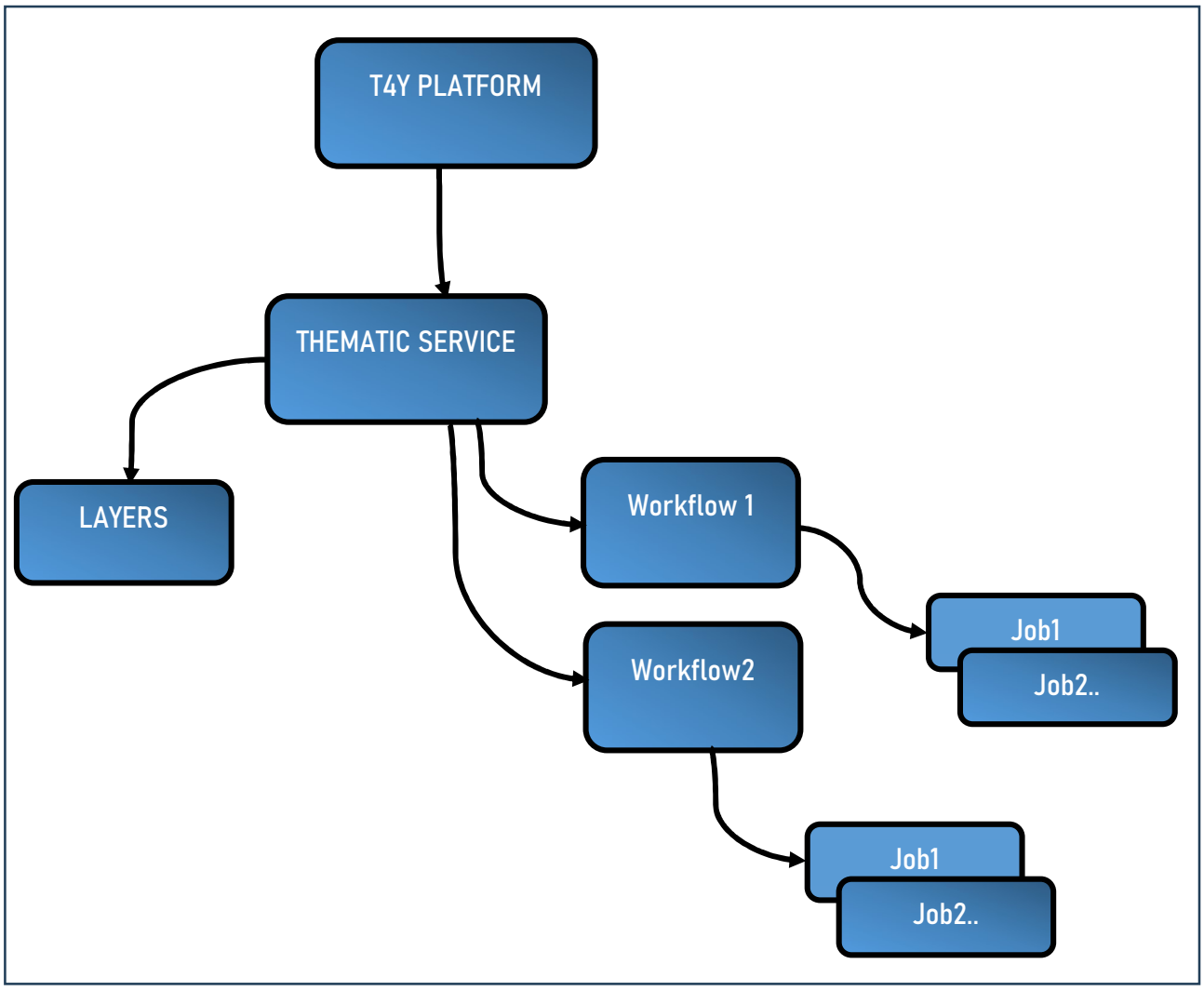


Figure 7 : The service domain model

In Figure 9 is shown a mock-up of the management page. As we previously seen in this section, the service manager is enabled to create, edit, or delete a Thematic service. He can also create workflows associating them with a thematic service. The concrete implementation of the workflow is made using Airflow workflow manager.

Both Service Manager and Service Operator, can visualize service, workflows, and their status. Furthermore, they are able to manually launch a workflow, and stop it.

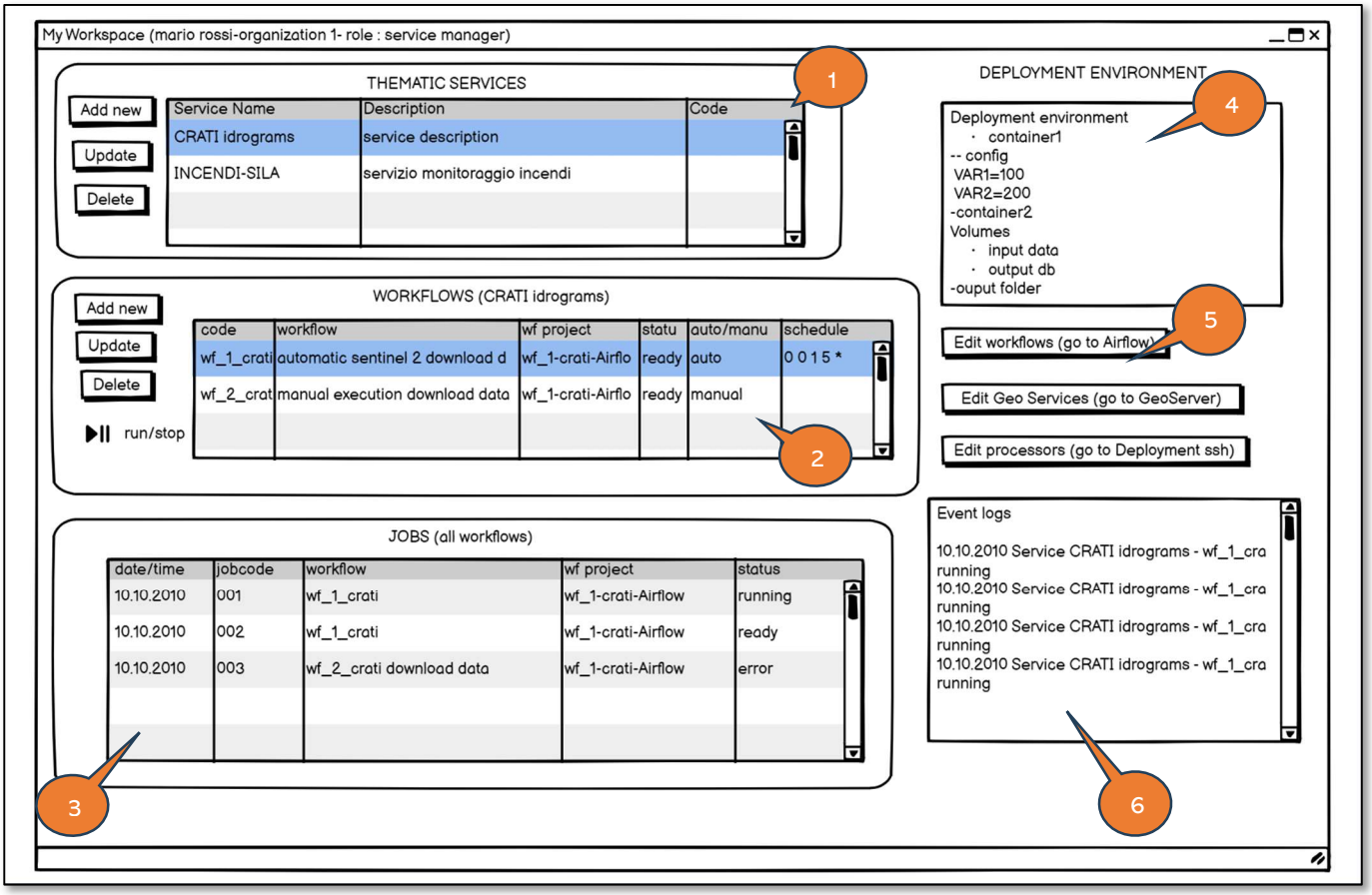


Figure 9 : T4Y Service & workflow management page

The SM and SO can launch or schedule an automatic processing workflow, to check the execution status or to stop it. They are, furthermore, able to see processing for an ongoing and historical service-workflow.

In the previous image a firewire presentation is shown, describing all the main functionalities of the management console (myWorkspace).

The page includes the following tools (see callout reference):

1. Thematic service tool, able to create, modify and delete a thematic service.
2. The workflow tool, able to create, modify or delete a workflow within a thematic service.
3. Jobs status tool
4. Deployment environment tool
5. Access console to other platform tools (Airflow, geoserver, linux ssh environment)
6. Log events window

6.3.3 Geo-Navigator (SU)

The geo-navigator is a one-page web app, that allows a registered user to navigate cartographic datasets of the service he subscribes to, and view and the analytics of the output data. It is divided in two physical components:

- The UI front-end: graphical user interface (UI) functionality
- The UI back-end: features and libraries to support the front-end (e.g. download, data upload, etc.).

A typical geo-navigator layout is show in the following image (Figure 10. The main parts are highlighted:

- Central map area
- navigation toolbox
- time slider
- general menu
- layer TOC
- query tool
- processing panel (for enabled users)

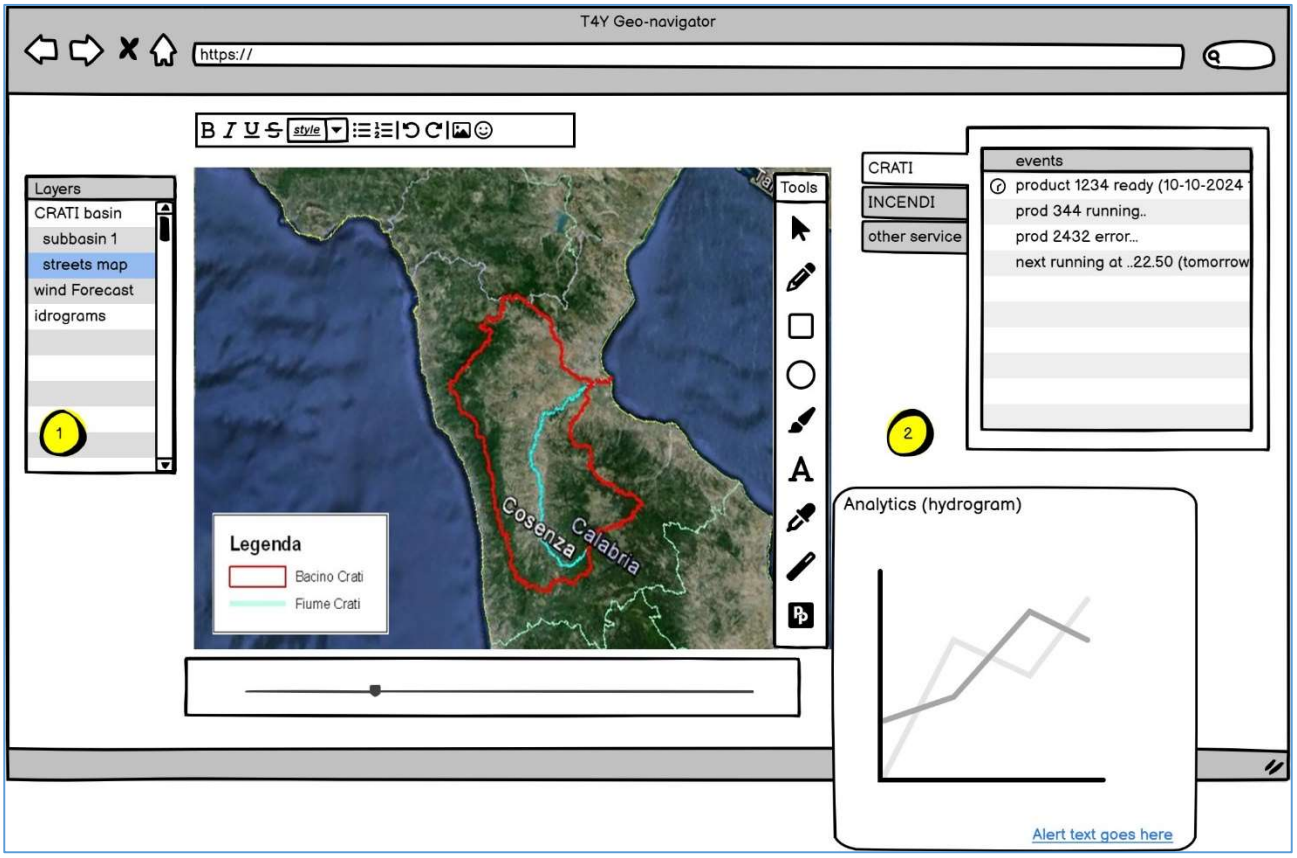


Figure 10 : Geo-navigator UI

The main functional features of the geo-navigator are the following:

1. Change map extent (zoom/pan, zoom extent, full extent, etc.)
2. Search for a location/POI
3. Searching for a point using coordinates
4. Measurement of distances and areas
5. TOC - Table of Contents (activation of layers, transparency, legend, metadata).
6. Queries with geographic and alphanumeric filters on the layers
7. Alphanumeric information (identify) on objects.
8. Print map
9. Export/download data in standard format (e.g. Shapefile, geojson)
10. Import temporary backup data (shapefile, raster data)
11. Time series consultation slider
12. Viewing analytical information (smart analytics)
13. Notification area

6.3.4 Analytical tools (SU)

Tools for the interactive consultation of analytical data from cartography. These are graphs of the temporal trend of physical quantities, being monitored and which will be used in the reporting and decision-making phase.

An example is described in the image below (by selecting a point, the user can view a graph that highlights the movements of PS (Permanent Scatterers) distributed over time.

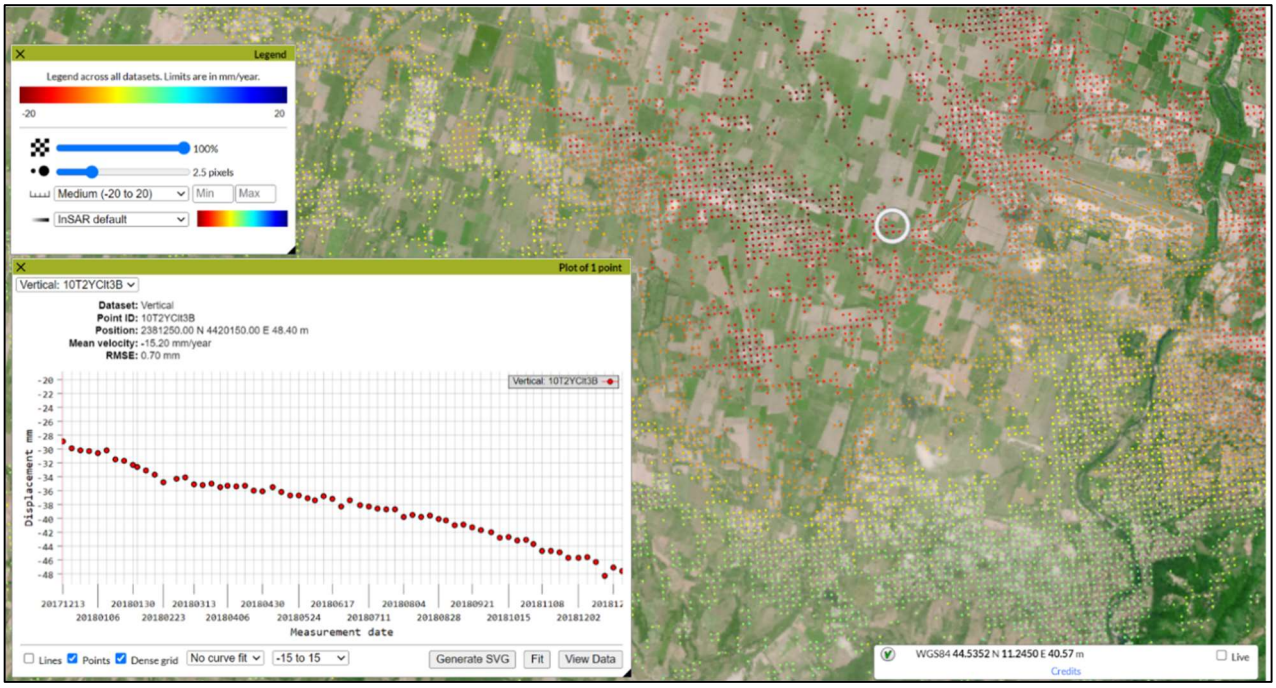


Figure 11 : Analytics shown in the geo-navigation.

Note that these the implementation of a user interfaces can be partially customized by the platform.

6.3.5 User management (ADM)

The administrator (ADM), in addition to being enabled to manage users and their profiles, is enabled to create a new service on the platform, through the appropriate user interface.

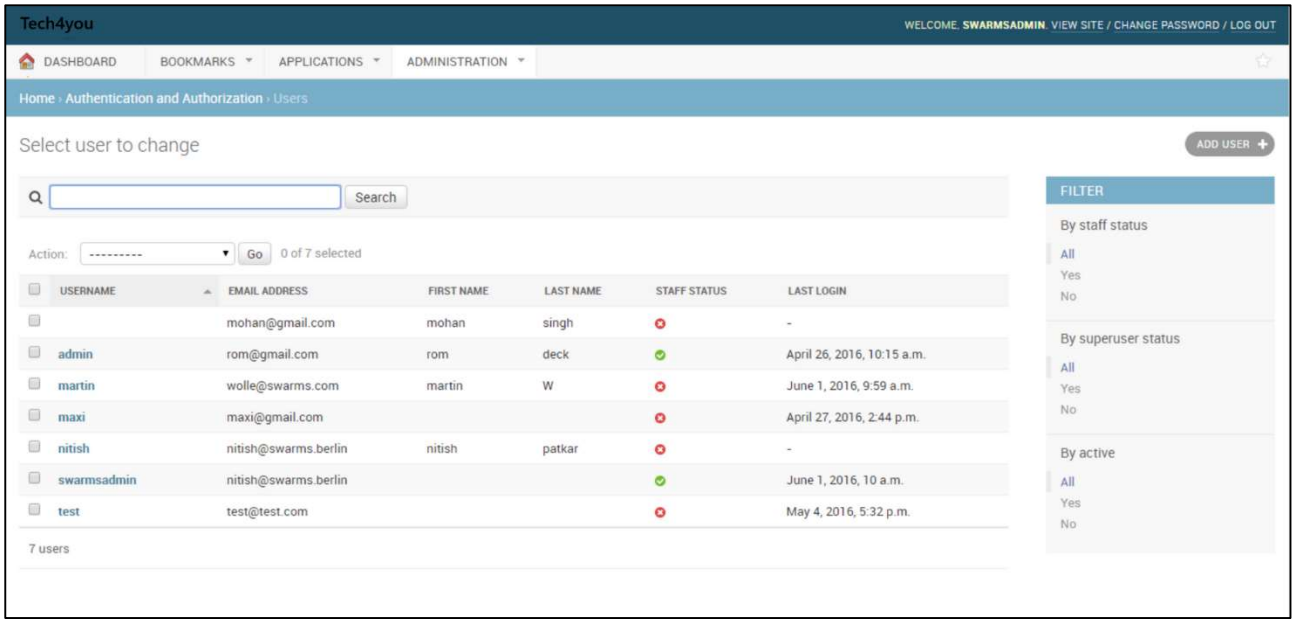


Figure 12 : User Management page

6.3.6 User profiling and permissions (ADM)

The following matric shows who are enabled to use the platform functionalities.

User profile/ Functionality	Geoportal public sections	Geo-navigator	Thematic Services manager	Workflow server manager	Run/stop processing	Analytics	User mgmt.	API REST services
Anonymous	x	x						
Registered user	x	x				x		
Service manager	x	x	x	x	x			
Service operator	x	x			x			
Administrator	x	x					x	
External client	x	x						x

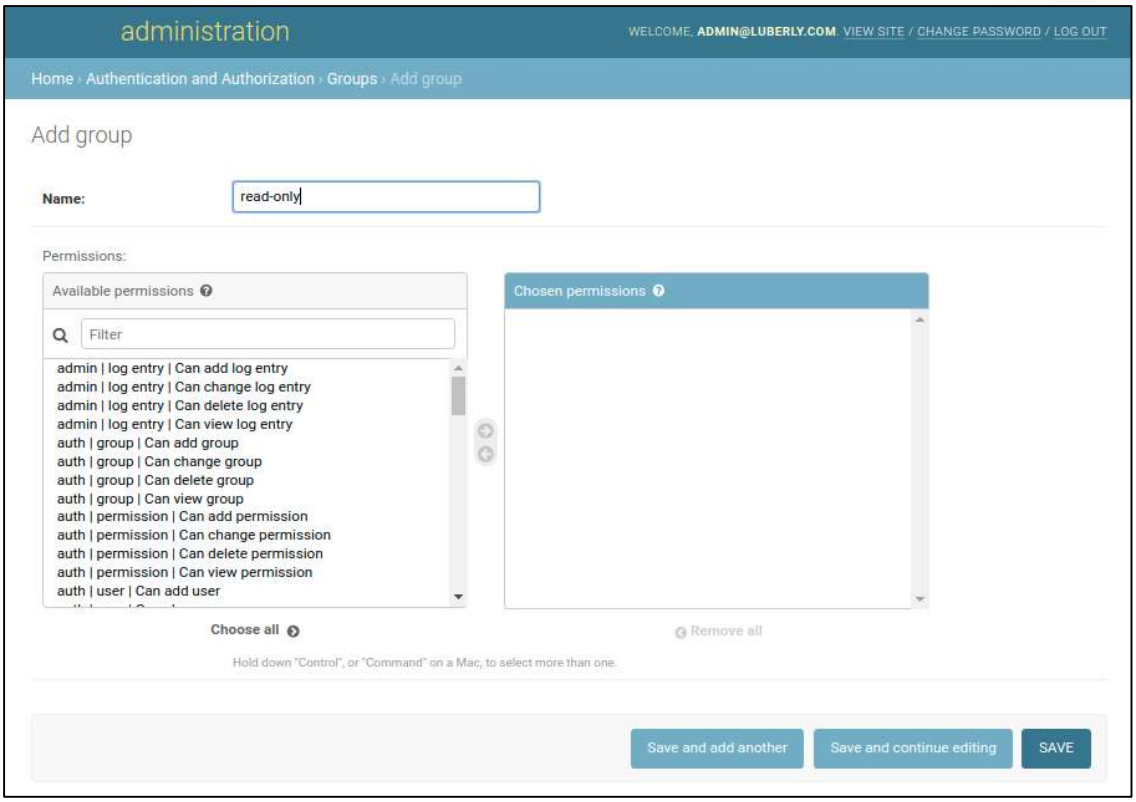


Figure 13 : User Profiling page (example)

6.4 DATA PROCESSING LAYER

The data processing layer is the core component that provides core processing services to the user. The following diagram represents the main architecture components of the Processing Layer:

- Back-end core processing services
- Orchestrator of processing workflows.
- Processing functionality containers

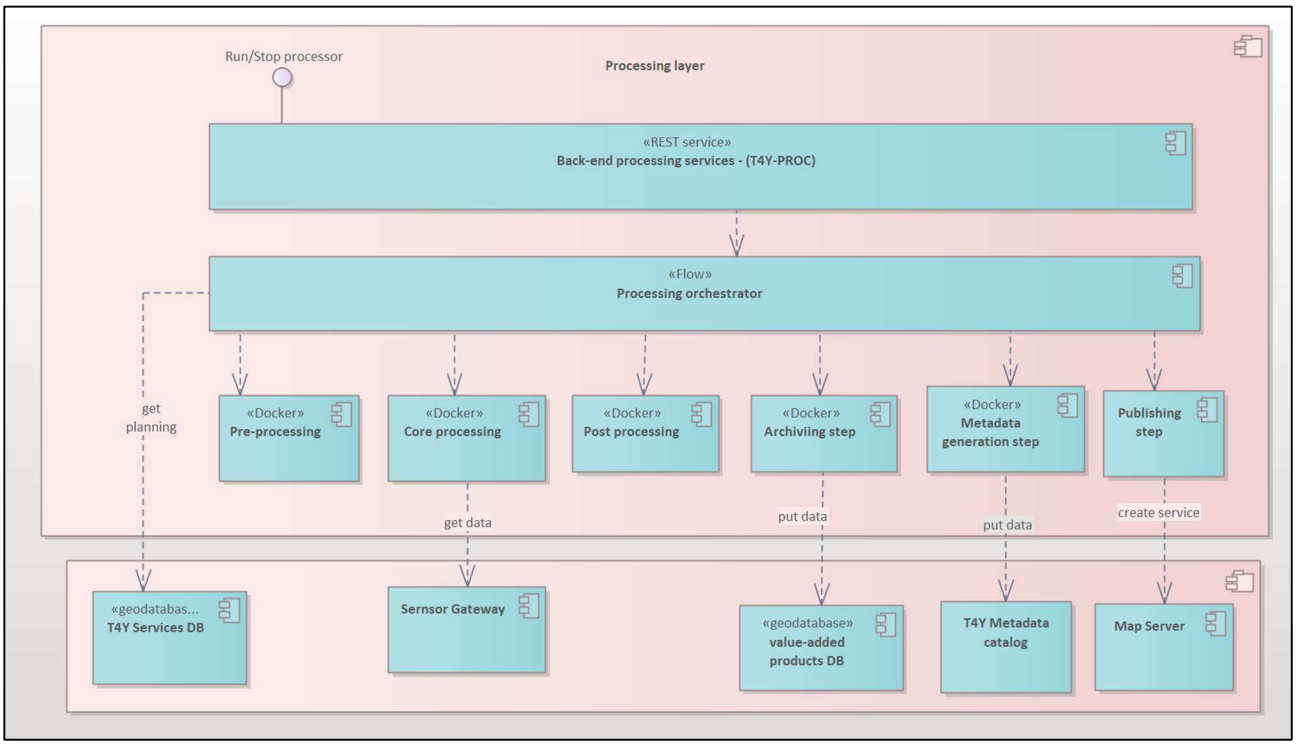


Figure 14 : Processing orchestrator architecture

6.4.1 T4Y core services

The back-end core services aim to provide high-level general functionality for the management of data processing phases. These are stateless RESTful services, which guarantee the geoportal access to processing services. They will be used by the Application layer component or by external components where enabled.

The services provided will be the following application interfaces.

6.4.1.1 Thematic Services

CRUD for Services

Method	Path	Description
GET	/services	List all services
GET	/services/{id}	Get service with ID
POST	/services	Add service
DELETE	/services/{id}	Delete service with ID
PUT	/services/{id}	Update a service with ID

6.4.1.2 Workflows

CRUD for workflows

Method	Path	Description
GET	/ services/{id}/Workflows	List all workflows
GET	/services/{id}/workflows/{id}	Get workflows with ID
POST	/services/{id}/ Workflows	Add workflow
DELETE	/services/{id}/workflows /{id}	Delete workflows with ID
PUT	/services/{id}	Update a service with ID
POST	/services/{id}/workflows/{id}/run	Run workflow
POST	/services/{id}/workflows/{id}/stop	Stop workflow

6.4.1.3 View of jobs

Method	Path	Description
GET	/services/workflows/jobs	List all jobs
GET	services/workflows/jobs/{id}	Get jobs status with ID

6.4.1.4 View of logs

Method	Path	Description
GET	/services/logs	List all logs
GET	services/{id}/workflows/{id}/logs/{id}	Get logs with ID

6.4.1.5 CRUD of service configs

CRUD for configs

Method	Path	Description
GET	/ services/{id}/workflows/{id}/configs	List all configs
GET	/ services/{id}/workflows/{id}/configs/	Get configs with ID

6.4.1.6 CRUD of workflows schedules

CRUD for schedules

Method	Path	Description
GET	/services/workflows/schedules	List all schedules
GET	services/workflows/ schedules /{id}	Get schedule with ID
POST	services/workflows/ schedules	Add schedules
DELETE	services/workflows/ schedules /{id}	Delete schedules with ID
PUT	services/workflows/ schedules /{id}	Update a schedule with ID

6.4.1.7 CRUD layers

Method	Path	Description
GET	/services/ layers/{id }/layers	List all layers
GET	/services/ layers/{id }/layers/{id}	Get layer with ID
POST	services/ layers/schedules	Add layers
DELETE	services/ layers/schedules /{id}	Delete layer with ID
PUT	services/ layers/ schedules /{id}	Update a layer with ID

6.4.2 Processing Orchestrator

Main features

The orchestrator has the objective of executing and managing the execution of processing steps in a coordinated manner following the logic and control flow envisaged by the implemented processing algorithms. You will also be able to run steps in parallel where possible or choose between two branches of the flow based on specific conditions. In the image below an example of the processor and the associated steps.

The planned technological platform is the Apache AirFlow tool. More details will be described in the deployment and technology description section.

Structure of a workflow

Given that all processing activities can be incorporated into a single container, in theory, it must be considered that the execution activities of a processor are preferably organized by dividing **them into** steps that are as independent and specialized as possible according to a specific execution order and can provide some categories including:

- Data acquisition
- Data pre-processing
- core processing step
- post processing step
- archiving and metadata creation steps
- publication step

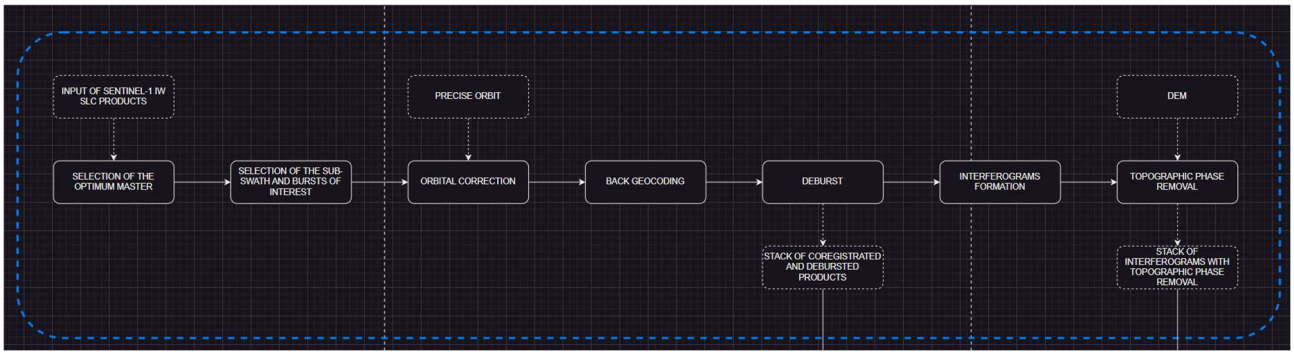


Figure 3: Example of a workflow

The processing steps will be implemented as individual docker containers. This will guarantee that the single step is independent from the context. It will obviously be necessary to: ensure the passage

of the input parameters to the container, in the call, access external data via T4Y platform services, adequately configure the memory and processing cores, carry out a mapping via a volume between the disk space of the container and that of the host.

Interaction with other components

Services provided: The orchestrator provides workflow processing services to the processor back-end (T4Y-PROC).

Required services:

- Synchronous/asynchronous services for accessing input data from external sources (sensors, EO catalogues, etc.)
- Storage services for generated raster or vector data
- T4Y General/Local Metadata Catalog Services (OGC-CSW Protocol)
- T4Y service database

Commit management and resource use possible issues:

Processing activities can require execution times from a few minutes to many days, for this reason there are two primary problems:

- Commit/Roll back: Ensure that in the event of an error you can return to an initial situation without "dirtying" the platform with partial or incorrect data.
- Memory and processing resources Allow an adequate distribution of memory and processor resources to ensure that processes running at the same time, but with different resources are dependent on each other, thus limiting the least expensive processing, compared to the most expensive one.

To resolve the first problem, it will be necessary to guarantee the execution of the processing in a temporary support area, until the actual completion of the processing. At the end of the processing, it will then be possible to transfer the data (raster or vector) to the right location to be published.

In the event of an error during the processing phase, it will therefore be possible to interrupt the process, check and debug by analysing the partially generated data, all without having operated on the "production" platform. These temporary areas (tables or folders) will then be deleted when running again.

Regarding the second problem on the management of machine resources, it will be necessary to carry out a technical budget calculation in advance to then guarantee the optimal space and power for execution. In any case, shared configuration management will be useful to maintain the right balance.

Multiusers/tenancy

Considering that the processing services running will be different and parallel, and therefore the resources could be very complex to manage in a single context.

It makes sense to manage this in multiple tenants, one per service. This could mean creating copies of specific data databases, specific raster data repositories (folders), and even a specific map server.

The catalog could instead be centralized.

6.4.3 Integration and processing workflow design steps

When integrating a specific processor, it will be necessary to proceed with the following steps:

- Estimate the technical budget useful for execution (RAM and core) and its configuration.
- Creation of a Docker container that includes the input and output data access functionalities as provided by the platform. E.g. a python script is inserted into a container via docker file.
- Deployment of the container in the platform
- Definition of volume
- Creation of a new workflow project to build the sequence of planned steps.
- Enter the new workflow in the service registry.

6.5 DATA MANAGEMENT LAYER

6.5.1 Context Broker

IoT Device Manager allows the centralized management of IoT devices, providing the possibility for a decision-maker to increase the situational awareness of a territory, exploiting all the sensors available in a real environment.

It is the service that allows users to manage the *sensors* distributed throughout the territory of interest. The territory of interest is covered by areas which constitute the territorial areas on which the various sensors insist.

For each *area* functional areas are defined, called services, which constitute the subdivision classes of all sensors set up in the "area" of interest.

Finally, each service includes the different sensors, managed directly by the IoT Device Manager, which perform the same common purpose as the "service" offered for that "area". The set of sensors, therefore, is divided into classes of sensors that perform a particular task, the service.

Module Name	**Context Broker**
Module Description	Context Broker (CB) enables discovering gathering and publishing of context information through APIs. CB, through its standard interface, makes available the context information regardless data source and using different types of interactions.
Main Functionalities	• **Context Availability:** represents the operations to identify which context data sources are connected to the platform. • **Query and Subscription** represents the synchronous and asynchronous interactions with context data source. Synchronous interactions are performed using a query mechanism to obtain context information; the component, allows building powerful queries using different types of filters to retrieve information with high level of precision. The asynchronous interaction is performed by publish/subscribe mechanism: a notification is generated when published data meets the subscription conditions this feature is useful to avoid the implementation of a polling process on data sources of interest, allowing to be notified when the context information changes.

	• **Command Dispatcher:** through this function the Context Broker acts as an input channel for an IoT device that can receive commands from external systems.
Covered user stories	UC-10-10: Thematic Service creation
Interaction with other Modules	This module interacts with **STH** and **Alert Broker**

Table 1: Context Broker Description

6.5.2 Alert Broker

Module Name	**Alert Broker**
Module Description	Alert Broker implements an alert system useful to automatically notify **Application Layer** components whenever a new event occurs. This allows to keep the responders up-to-date on the phenomenon, providing them with a comprehensive picture of what happened near real time.
Main Functionalities	Alert system
Covered user stories	UC-10-40: Publishing of a new generated Thematic product
Interaction with other Modules	This module interacts with **STH** and **Context Broker**

Table 2 - Alert broker

6.5.3 Geospatial Server

Module Name	**Geospatial Server**
Module Description	Geospatial Server is responsible for sharing geospatial data. It is designed for interoperability; it publishes data from any major spatial data source using open standards.
Main Functionalities	It implements industry standard OGC protocols such as Web Feature Service (WFS), Web Map Service (WMS), and Web Coverage Service (WCS). Additional formats and publication options are available as extensions including Web Processing Service (WPS), and Web Map Tile Service (WMTS).
Covered user stories	UC-10-40: Publishing of a new generated Thematic product

Interaction with other Modules	This module interacts with **Spatial/Non spatial DBMS** in order to retrieve all the data provided by SATELLITE Connector, and with the **Geo-Navigator** as OGC Client.

Table 3: Geospatial Server description

6.5.4 STH

Module Name	**STH**
Module Description	Time Series Query is a component in charge of managing (storing and retrieving) historical data and aggregated time series information about the evolution in time of context data (i.e., entity attribute values) registered in a Context Broker
Main Functionalities	Time Series Query module exposes an HTTP(s) REST API to let external clients query the stored historical aggregated time series context information. The requests for aggregated time series context information can use the following query parameters: • **aggrMethod**: The aggregation method. This module supports the following aggregation methods: max (maximum value), min (minimum value), sum (sum of all the samples) and sum2 (sum of the square value of all the samples) for numeric attribute values and occur for attributes values of type string. Combining the information provided by these aggregated methods with the number of samples, it is possible to calculate probabilistic values such as the average value, the variance as well as the standard deviation. It is a mandatory parameter. • **aggrPeriod**: Aggregation period or resolution. A fixed resolution determines the origin time format and the possible offsets. It is a mandatory parameter. Possible valid resolution values supported by Time Series Query are: month, day, hour, minute and second. • **dateFrom**: The starting date and time from which the aggregated time series information is desired. It is an optional parameter. • **dateTo**: The final date and time until which the aggregated time series information is desired. It is an optional parameter.
Covered user stories	UC-30-10: Data Geo-Navigation
Interaction with other Modules	This module interacts with **Context Broker**

Table 4: non-Spatial DBMS Description

6.5.5 File server

Module Name	File Server
Module Description	This module is responsible for providing file such as image and video to the other modules.
Main Functionalities	The main functionalities of this module are saving, reading, encrypting, creating centralized files and folders, shared by everyone or accessible according to rules or permissions defined using Security and Privacy module.
Covered user stories	UC-20-10: Run processing workflow (manual mode) UC-30: DATA NAVIGATION UC-10-40: Publishing of a new generated Thematic product
Interaction with other Modules	This module interacts with **Geospatial Server**

Table 5: Fileserver description

6.5.6 Spatial/Non spatial DBMS

Module Name	Spatial/Non spatial DBMS
Module Description	This module implements a DBMS with GIS plugin, in order to perform geographic queries
Main Functionalities	Spatial/Non spatial DBMS performs several important functions that guarantee the integrity and consistency of the data in the database. The most important functions are: • Data Dictionary Management • Data Storage Management • Data Transformation and Presentation • Security Management • Multiuser Access Control • Data Integrity Management • Database Access Languages • Database Communication interfaces • Geographic queries
Covered user stories	UC-20-10: Run processing workflow (manual mode) UC-30: DATA NAVIGATION UC-10-40: Publishing of a new generated Thematic product
Interaction with other Modules	This module interacts with **Geospatial Server**

Table 6: Spatial/Non spatial DBMS

6.6 DATA HARMONIZATION

6.6.1 Data Harmonization

Module Name	Data Harmonization
Module Description	This is the module in charge of adapting heterogeneous context data to a specific data model belonging to specific set classified by application domains. The approach is to define a group of harmonized data model that cover the typical application domains of smart water applications. The module gets information stored in a specific context of Context Data Broker and creates a new copy based on a harmonized format. In the Context Data Broker will reside both the original entity and its real-time updated harmonized copy. This module plays an important role in terms of interoperability aspects, providing a way for application providers, to define their applications using these models and ensuring a greater level of reusability of them in T4Y scenarios.
Main Functionalities	Data Harmonization allows defining the structure of the new data model to use in the mapping phase. Data Harmonization allows manipulating information coming from Context Data Broker, by applying transformation, elaboration, filtering, merging on these data in order to adapt them to the specific data model requested. Finally, it allows publishing on specific context of the Context Data Broker the result of harmonisation of an entity with a supported data model.
Covered user stories	UC-10-10: Thematic Service creation UC-10-20: Workflow creation (internal processor) UC-10-30: Workflow creation (external processor) UC-20-10: Run processing workflow (manual mode) UC-20-20: Running a processing workflow (scheduled mode)
Interaction with other Modules	This module interacts with all the **connector** and with the **Context Broker**

Table 7: Data Cleaning and Harmonization description

\

6.7 DATA COLLECTION

6.7.1 Platform domain model

As shown in the diagram below, the domain model includes a *thematic service* entity (t4y_service), that has composing relationships to the *processing workflow* entity. Every workflow entity is then associated with a single *processing job* (the workflow process execution).

The thematic service is also associated to a set of geospatial layers defined in the Toc_dataset entity.

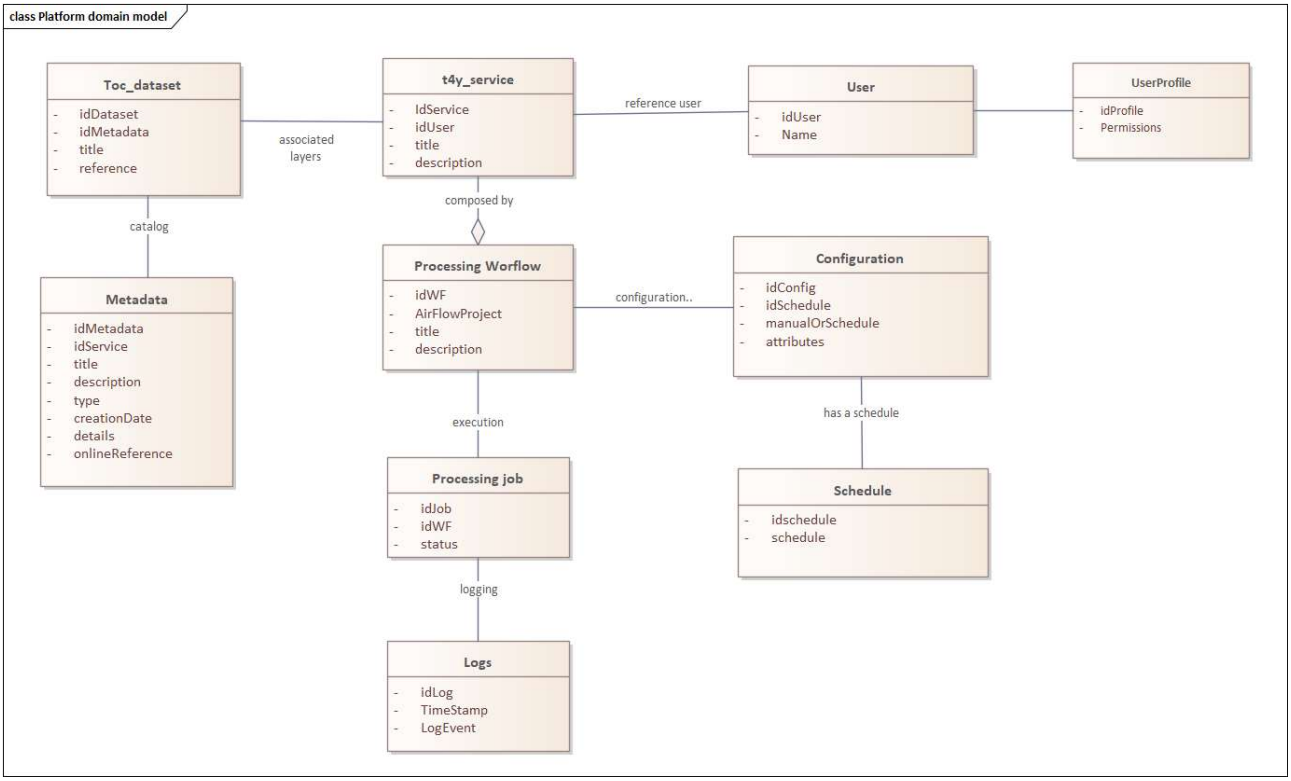


Figure 15 : T4Y core services domain diagram

The following sections detail the domain classes defined in the domain diagram.

6.7.2 T4Y_service

It includes information relating to a T4Y thematic services. So, there will be stored the following information attributes:

- A unique identifier of the service
- title of the service
- service category (statically defined among those provided by ISO)
- The description of the service
- The identifier of the associated tenant/user

- The list of associated monitoring workflows to be performed.
- The geographic area where the service is available.
- The distribution rights of the products
- The service responsible organization
- A reference mail to the help support.

other information will be the name of the external contact person for the hosted service.

6.7.3 Processing Workflow

Below is the list of all the workflows that can be executed by the orchestrator.

Every workflow has the following attributes:

- workflow identifier
- workflow title
- description of the workflow
- name of the orchestration service to launch.
- type of execution (manual or scheduled)
- scheduling (if scheduled with associated crontab)
- list of input parameters
- output list
- The geographic area where the service is available.
- Configuration parameters
- The workflow airflow project

Each workflow that is executed by the orchestrator, is composed of atomic steps that must be executed as docker containers or run external if required.

6.7.4 Processes jobs

It includes every process that is running, or executed, and its execution status. The reference attributes are as follows:

- Job identifier
- workflow identifier
- Process status (running, error, completed, deleted)
- Process launch date/time
- Process end date/time
- Message: any error message, warning, etc.

6.7.5 Event Logs

Includes processing event logs of the platform.

- Process identifier
- Workflow identifier
- User responsibility
- Event timestamp
- Log code
- Log message

Below an example of coding table for the event logs.

Log Code	description
Process_run	The processing workflow is running
Process_completed	The processing workflow is ended successfully
Process_error	The processing workflow is stopped with errors

6.7.6 Metadata structure schema

For every generated product a metadata will be created. A metadata fields in the "Dublin Core" standard schema, to be used in the cataloguing of the Thematic products.

Field	Description	type
Title	The name given to the resource.	String
Creator	The entity responsible for creating the resource.	String
Subject	Keywords or terms describing the content of the resource.	String
Description	A summary of the resource content.	String
Publisher	The entity responsible for making the resource available.	String
Contributor	Other entities that contributed to the resource.	String
Date	A date associated with the resource event (creation, publication).	DateTime
Type	The nature or genre of the resource content.	String
Format	The file format or resource medium (PDF, JPEG, etc.).	String

Identifier	A unique reference to the resource.	String
Source	The resource from which the current object is derived.	String
Language	The language of the resource.	String
Relation	A reference to a related resource.	String
Coverage	The extent or scope of the resource (spatial, temporal).	String
Rights	Copyright and other rights information.	String

These fields provide a comprehensive framework for resource description, facilitating their searchability and interoperability across different platforms and services.

6.8 DATA SOURCES

6.8.1 T4y Platform Data Sources

A list of possible external data sources is shown below. The platform will support an access to them.

Source	Description	Protocols
EO catalogues	EO catalogues are the main source of imaging from Earth. Examples of catalog services are: WekEO, Sentinel Hub, OCRE, CREO Dias, ONDA-Dias	CS-W, OpenEO, STAC
External geospatial services	Geospatial services are public geospatial services provided by public entities, able to be integrated in maps or used in the processing of a product.	WMS, WFS, WCS, WMTS
IoT Sensors services	Sensors IoT services are data provided by REST services	IoT protocols

6.9 SECURITY AND PRIVACY LAYER

6.9.1 Identity Management

Module Name	Identity Management
Module Description	The Identity Management module is the first step for accessing data, services, and applications, by providing secure and private identification and authentication of users, trust management, and Single Sign On (SSO) to service domains and Identity Federation towards applications.
Main Functionalities	This module offers functionalities for: • Management of user life cycle functions by providing account creation and management and enforcement of policies and procedures for user registration, Identification, and authentication. • Support to different authentication providers and several applications can be linked to this module, thus enabling Single Sign On (SSO) to all these applications.
Covered user stories	All US
Interaction with other Modules	This module interacts with all the modules. On the one hand, it verifies that the authentication token exchanged between the software modules that communicate with each other is valid, and on the other hand, it verifies that the logged-in user has access rights to the data.
Reference Data Model	No smart data model

Table 8: Identity Management Description

6.9.2 Authorization and Accounting

Module Name	Authorization and Accounting
Module Description	This module provides authorization and accounting capabilities. It enforces a set of conditions defining whether users have access granted to a particular resource while also storing information regarding access for audit purposes.
Main Functionalities	Authorization: provides a Policy Enforcement Point (PEP) which intercepts resource access requests makes access control decision requests and enforces access control decisions. Moreover a Policy Decision Point (PDP) evaluates access request by checking authorization policies for rendering an access control decision.

	It provides also a Policy Administration Point in order to manage policies that allows to create/read/update/delete multiple policies and also supports the policy versioning. It provides also a Policy Information Point (PIP) to obtain applicable authorization policies according to an access control decision request and attributes that are needed for evaluating authorization policies for example the IP address of the requester creation time of the resource current time or location information of the requester. This information is then combined in order to get a finial access control decision. Accounting: measures resource consumption performed by users during access. This includes the amount of system time or the amount of data a user has sent and/or received during a session. Accounting is carried out by logging of session statistics and usage information. It is used also for authorization control and resource utilization.
Interaction with other Modules	This module interacts with the Identity Management module, and with every module requesting access to protected resources.

7 BEHAVIOURAL VIEW

This section describes the main application scenarios envisioned on the platform.

7.1 Running of a processing workflow

This section allows the reader to understand the sequence of behavioural relationships between the processing container and all the related components in the platform. Let's look to the following diagram, as a simplified scenario.

When the scheduler is ready, it requires the running details to the Platform Service manager, and then allow the Orchestrator to proceed to the running of the processor (for instance called CRATI processor).

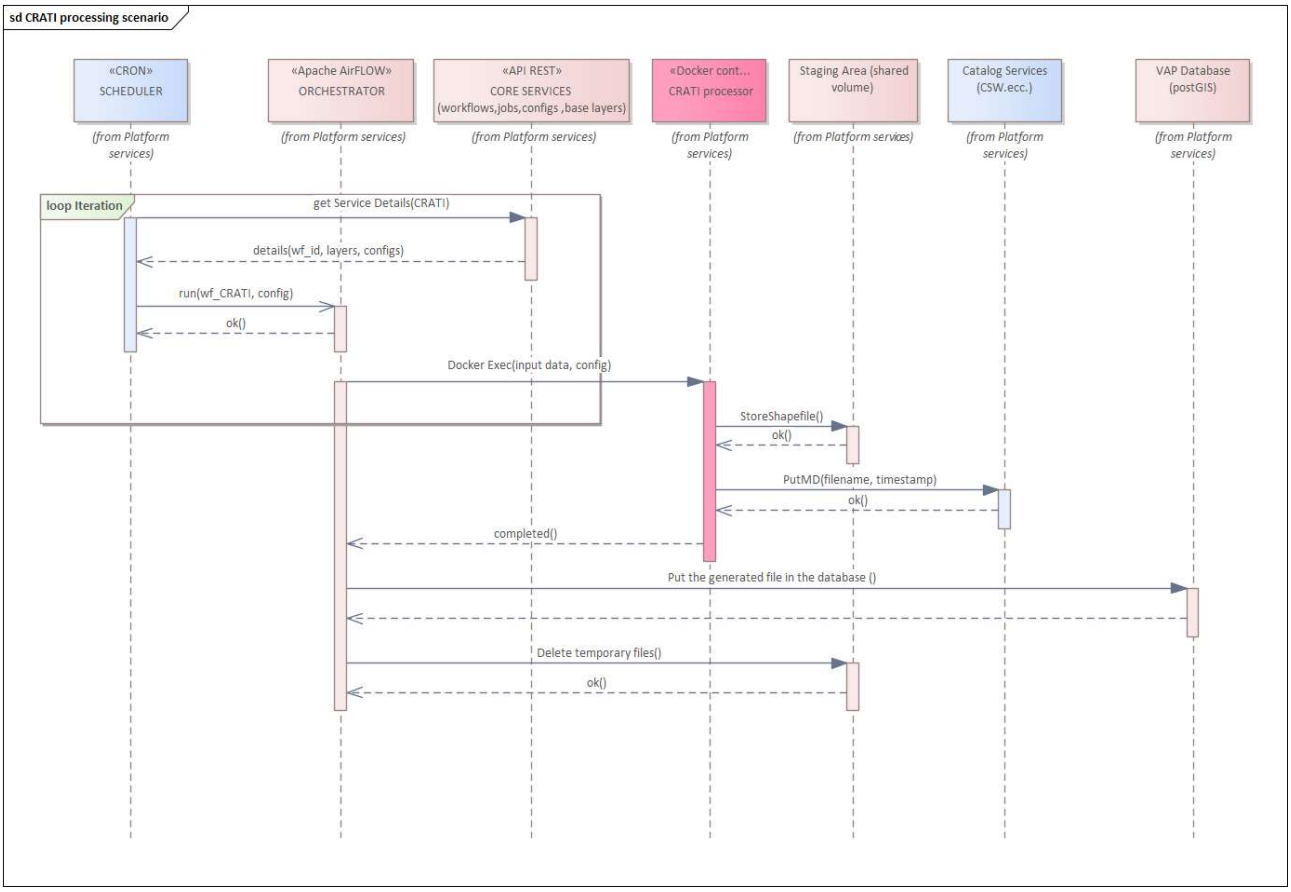


Figure 16 : Processing scenario (simplified)

The Orchestrator executes the workflow, and as first step runs the processor (docker container).

When the container has completed the generation of the output (a shapefile) then it saves the product metadata (calling the catalog back-end service).

Afterwords it saves the output in a staging area and confirms the Orchestrator that the processing step is completed.

The Orchestrator then loads the output in the Platform database and deletes the staging area. At the same time, it can inform the other components that the processing is then completed.

7.2 Services Integration

The section wants to describe the integration architecture of the two kind of services in the T4Y enabling platform.

The Figure 17, highlights the interactions between the platform and CRATI and SILA-INCENDI services.

The shown architecture features two types of processors (the green boxes) :

1. An internal processor embedded within the platform (the CRATI processor).
2. An external processing facility that interfaces with the platform (the SILA-INCENDI processing facility).

At the top level, the **Platform service** serve as the core category for the platform's offerings. The external processor acts as an operator/actor responsible for running, configuring, and scheduling tasks. It interfaces with the platform via an External processor interface, which likely facilitates data querying and indexing operations.

Central to the system is the Processing core, where the primary data processing activities occur. This core is interconnected with multiple components. Processing Services (WPS or similar) refers to services that execute processing tasks, potentially through Web Processing Services.

The Catalog Services manages data cataloguing, utilizing standards protocols like the Catalog Service for the Web (CSW). The Logs Services are used for logging operations, potentially employing a standard platform like Elasticsearch, Logstash, and Kibana (ELK) stack.

The Repository Services represent repository services for file storage management, with tools like a file access browser for file navigation (AWS S3, bucket, WebDav, etc. protocols).

The Schedule Facility (crontab, etc.) connects to the Processing Core, providing a mechanism for scheduling tasks, perhaps using a scheduling utility like crontab.

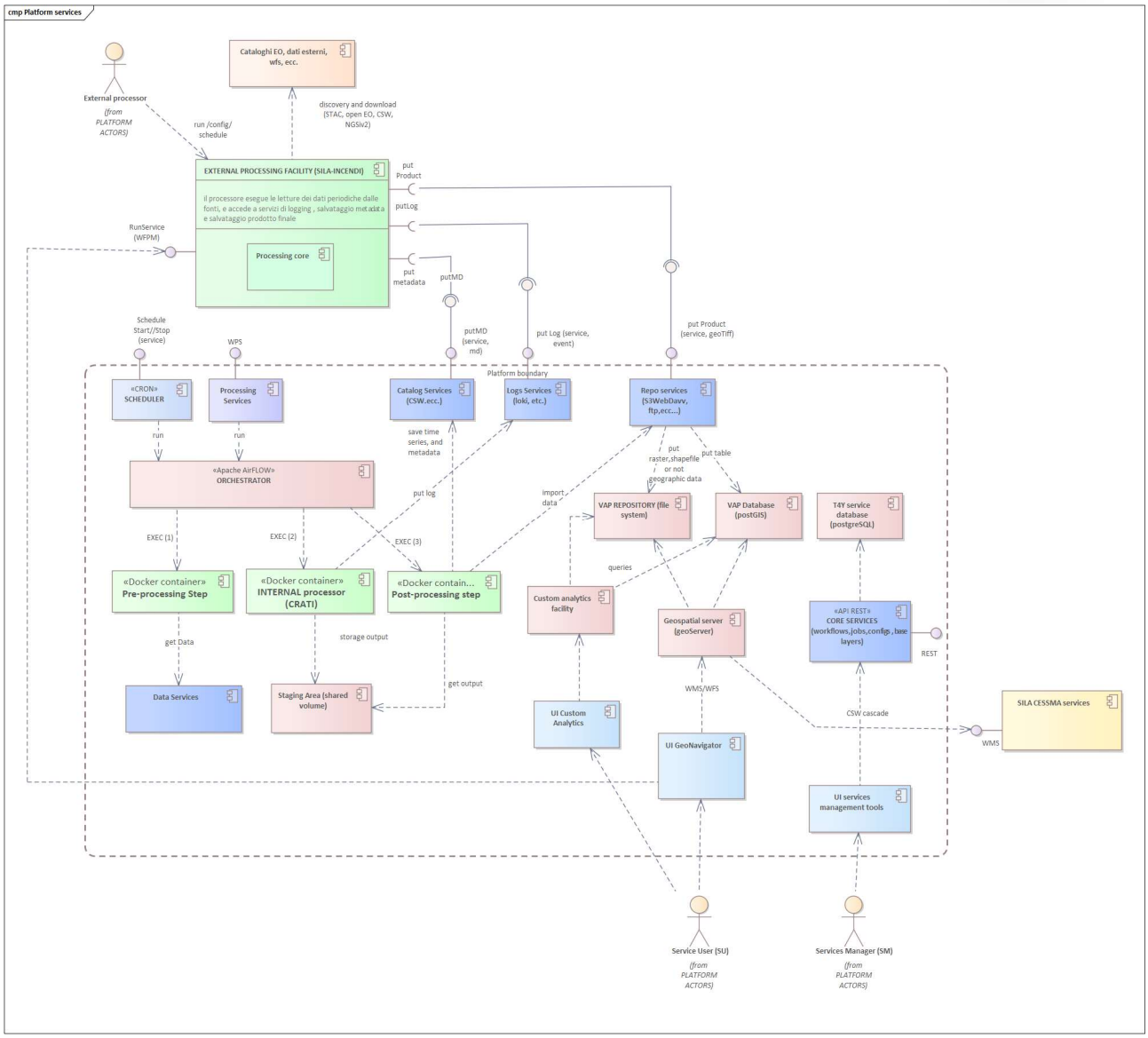


Figure 17 – Internal and external processing integration diagram

The Orchestration Facility indicates the utilization of a workflow orchestration tool, such as Apache Airflow, to manage and coordinate tasks within the system.

The Staging Area is designated for preparing or holding data or tasks before they are processed. Within a virtual environment, the Processing scheduler described a contained processing environment, which may be for statistical processing or a related application. Additional components include Custom analytics facility, a Geospatial server (GeoServer) and T4Y core services for handling specialized analytics, managing geospatial data, and thematic information exchange services, respectively.

User interfaces, such as Custom Analytics and Geo-Navigator, provide interactive access for custom analytics and geospatial navigation. Finally, the Services Manager and Service user are roles that

interact with the UI services management tools to oversee and utilize the services offered by the platform.

This platform is thus structured to facilitate the integration and processing of geospatial data, culminating in the generation of valuable, enhanced data products.

The two services will be part of the database of the services provided by the platform and with them will be present all the processing workflows and information on the processing status, and on the support data in view.

8 COMPONENTS VIEW

8.1 INTRODUCTION

8.2 TECNOLOGY COMPONENTS

The present section wants to detail the technology used to implement the architecture of the platform and how them will be physically deployed.

The diagram shown below clarify the technological components in the platform. Note please that the technical choices are the outcome of a preliminary trade-off analysis, after a survey of the IT sector.

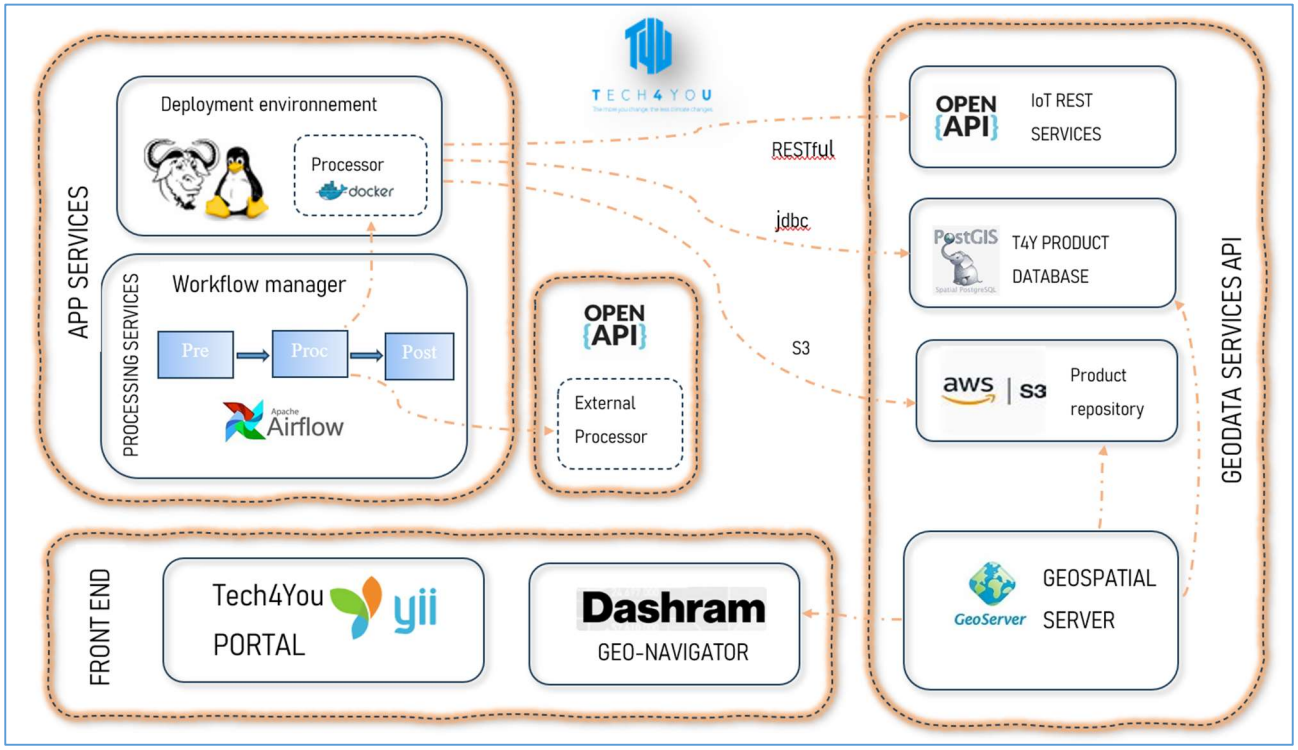


Figure 18 : Physical diagram

8.2.1 List of provided COTS

Hereafter a list of all the tools and related technologies is shown in the table below.

#	Software COTS	Description	Reference	Licence	Ver.
1	GeoServer	Geospatial server	geoserver.org	GNU GPL v2	2.24.2
2	Apache AirFlow	Workflow manager	airflow.apache.org	Apache 2.0	2.8.2
3	PostGIS	Geodatabase	postgis.net	GNU GPL	3.4.0
4	Docker	Container Framework	docker.com	Apache 2.0	25.0.2
5	Digital Enabler	IoT Enabling Platform	digital-enabler.eng.it	ENG-FOSS	-
6	YII	Content Manager	yiiframework.com	New BSD	2.0
7	Geonetwork	Catalog Manager	geonetwork-opensource.org	GNU GPL v2	4.4.2
8	Dashram	Geo-navigation and analytics tool	Dashram: a new tool for data analysis (eng.it)	ENG-FOSS	-
9	keyCloack	Identity and Access management	keycloak.org/	Apache 2.0	23.0.6
10	Perseo	Alert Broker	iware-perseo-fe.readthedocs.io	FOSS	-
11	Orion	Context Broker	fiware-orion.readthedocs.io	FOSS	-
12	Flask	REST back-end services	flask.palletsprojects.com	BSD Licence	3.0.1
13	PostREST	PostGIS Plugin	postgrest.org	GNU GPL	12.0

8.2.2 APPLICATION LAYER

8.2.2.1 T4Y GeoPortal (YII)

The GeoPortal is the main front-end access mean to the platform. It is the container of all the data and functionalities provided by the platform. The making of the Geoportal will be made using the content manager YII framework.

The Yii framework is a PHP tool for web application development. It employs the MVC (Model-View-Controller) pattern to structure code. It provides features for database management, caching, authentication, and more. It is extendable with custom components.

Yii supports an active record pattern, making database operations simpler and more intuitive. This framework includes a set of AJAX-enabled widgets, which helps in creating highly interactive user interfaces. Its security measures include input validation, output filtering, and SQL injection prevention. Yii is optimized for performance, offering powerful caching support and tools to minimize script execution time. Its design encourages rapid development, helping developers to get their applications up and running quickly.

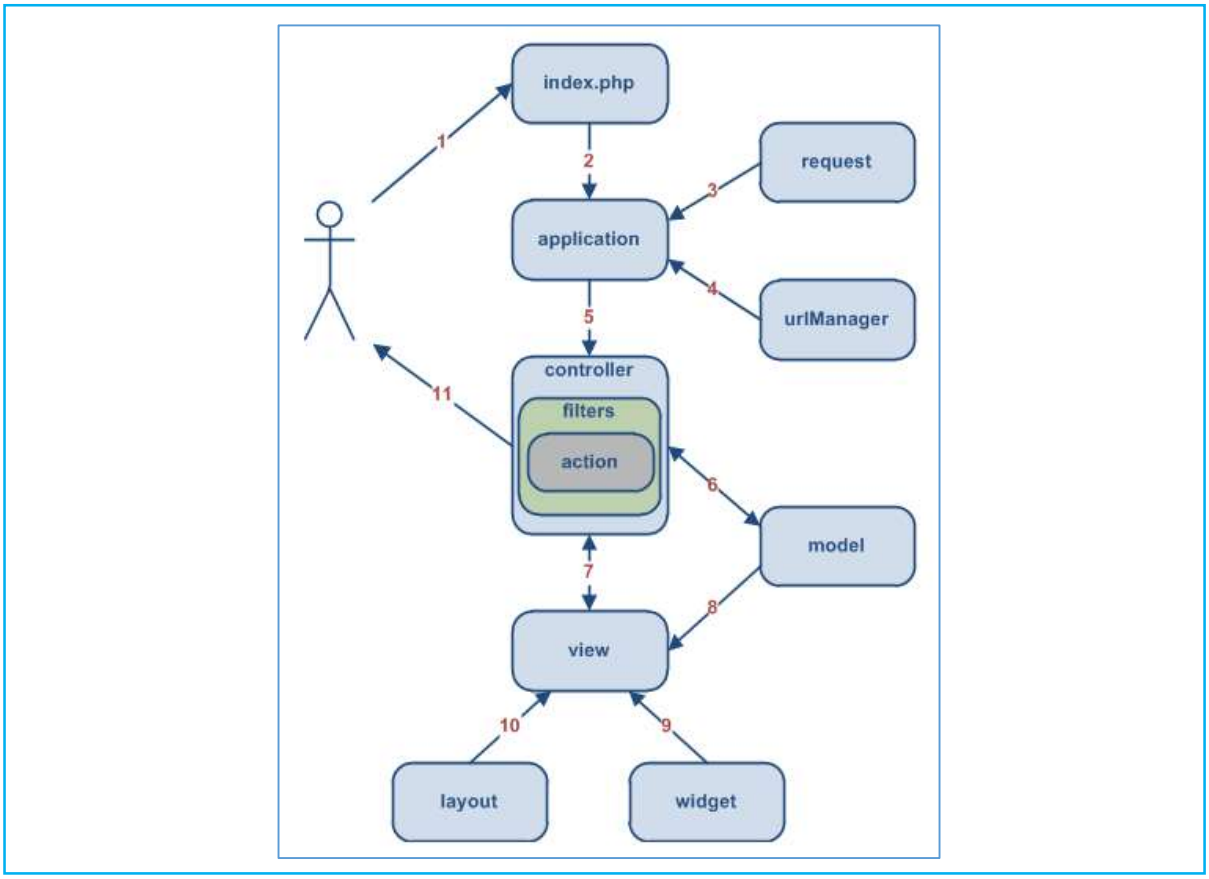


Figure 19 : The architecture of YII

8.2.2.2 Geo-Navigator (Dashram)

The geo-navigator will be built using DASHRAM tool.

Dashram is a tool provided by ENG and provides all the tools necessary to visualize data, the display of numerous charts, and the composition and sharing of dashboards. To do this, Dashram implements a layered architecture composed of different components that allow:

- to connect to heterogeneous data sources through the Digital Enabler (powered by FIWARE Platform)
- to interface with Orion Context Broker
- to create and manage different types of graphs, including 3D urban and timeline graphs
- to create, manage and share dashboards made up of multiple charts and other graphic and text components.

Figure 20 : DashRAM

Dashram technology has been used in the past to navigate data related to the COVID-19 crisis.

The Idra Generic Enabler and Digital Enabler are responsible for the federation of open data portals and resources (via APIs or web scraping) that include data about COVID-19 (e.g. Italian Civil Protection GitHub repository) and the harmonization of the datasets in compliance with European standard (i.e DCAT-AP meta-model). Once the open datasets are harmonized, they are mediated by the FIWARE Orion Context Broker, which provides access to the visualization component.

8.2.3 DATA PROCESSING LAYER

The data processing layer includes all the components that are involved in the data processing chains. Hereafter the description.

8.2.3.1 Processing facility (Docker)

The processing facility is the processing environment where the third-party can deploy the value-added product processor, in the platform.
Every processing environment can be seen as an independent virtual machine. The processing facility must be supplied as a dockerized container.
The processing container will use the platform storage components access methods to these components (DB connection strings, etc.) may be provided during the integration phase.

Docker is an open-source platform that automates the deployment of applications inside lightweight and portable containers. Containers allow a developer to package up an application with all of the parts it needs, such as libraries and other dependencies, and ship it all out as one package. This ensures that the application will run on any other Linux machine regardless of any customized settings that machine might have that could differ from the machine used for writing and testing the code.

In essence, Docker provides an additional layer of abstraction and automation of virtualization on top of the Linux operating system. Using Docker, you can quickly scale applications by adding new containers to handle additional load. Docker containers are isolated from each other and bundle their own software, libraries, and configuration files; they can communicate with each other through well-defined channels.

All containers are run by a single operating system kernel and are thus more lightweight than virtual machines. Containers are created from "Docker images" which can be built from a Dockerfile, a plaintext file that specifies all the commands that need to be run to build a given image. Docker Hub and Docker Cloud are public registries that anyone can use, and Docker is configured to look for images on Docker Hub by default. Docker also provides a service called Docker Swarm that helps with orchestrating multiple containers to work together in a microservices architecture.

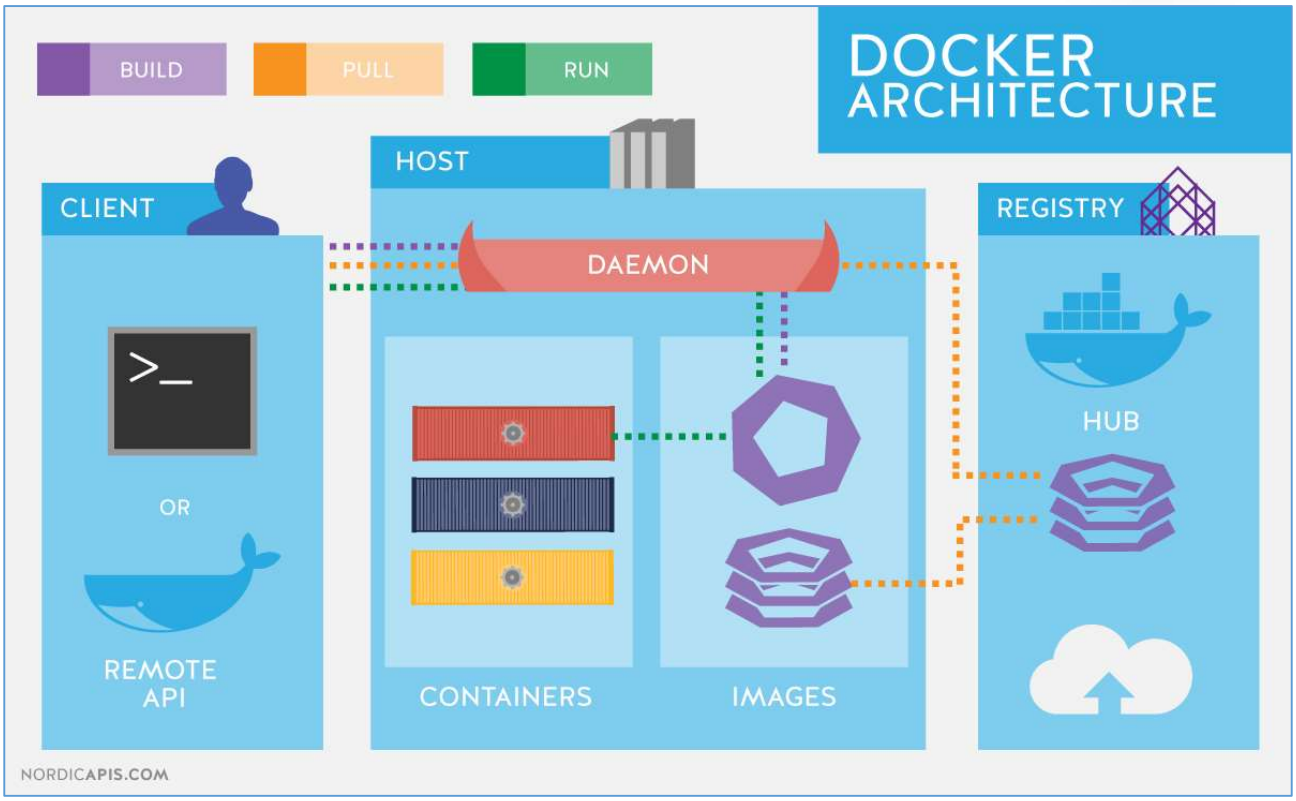
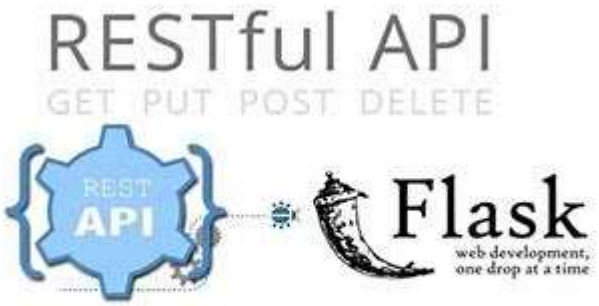


Figure 21 : Docker architecture

8.2.3.2 Service registry (Open API)

A registry containing information about processing modules. There will be general information on the service, workflows, ongoing processes, logs, and on configurations and execution schedules. This data will be made available via a backend service.

Flask will be described as a lightweight WSGI (Web Server Gateway Interface) web application framework in Python, designed to make getting started with a web application quick and easy, without having to deal with much initial setup for development. It provides the simplicity, flexibility, and fine-grained control needed to build scalable and maintainable web applications.



In the context of developing a backend component for a REST service, Flask will be utilized for its ability to easily set up RESTful APIs. It will allow the definition of resource-based routes and handlers, facilitating the creation, retrieval, updating, and deletion of resources through HTTP methods GET, POST, PUT, and DELETE, respectively.

The Flask framework will enable the backend to efficiently handle requests and responses, converting incoming HTTP requests from the client into Python objects, and then converting Python objects back into HTTP responses to be sent back to the client. Flask will also support extensions that can add application features as if they were implemented in Flask itself, including but not limited to, object-relational mappers (ORMs), form validation, and authentication mechanisms.

For the implementation of the REST service backend, Flask will be employed to:

- Initialize and configure the application and its environment.
- Define route URLs for each API endpoint, mapping them to Python functions.

- Parse request data (JSON, form data, query parameters) and validate it against predefined schemas.
- Implement business logic within each endpoint handler to process requests and prepare responses.
- Interact with databases or other services to retrieve or store data needed for the application's functionality.
- Utilize Flask's response objects to customize headers, status codes, and the body of the HTTP response to adhere to RESTful standards.
- Apply middleware for cross-cutting concerns like logging, security headers, and CORS (Cross-Origin Resource Sharing) handling.

Integrating Flask into the platform design will ensure a modular, easy-to-understand, and scalable backend component, leveraging Python's expressiveness and Flask's minimalistic but powerful framework to build a robust REST service.

8.2.3.3 Workflow Orchestrator (Apache AirFlow)

The workflow orchestrator is an application service able to perform the running of general-purpose workflows. It can be easily used also for the execution of processing modules.

The workflow orchestrator use in the platform is FOSS *Apache airFlow*.

Apache Airflow is an open-source platform designed for orchestrating complex computational workflows and data processing pipelines. Developed by Airbnb and later contributed to the Apache Software Foundation, Airflow allows users to programmatically author, schedule, and monitor workflows. It provides a robust framework for managing the execution of tasks, ensuring that they are run at the correct times and in the right order, while also handling dependencies between tasks efficiently.

Below a simple diagram describes the main architecture of Apache Airflow.

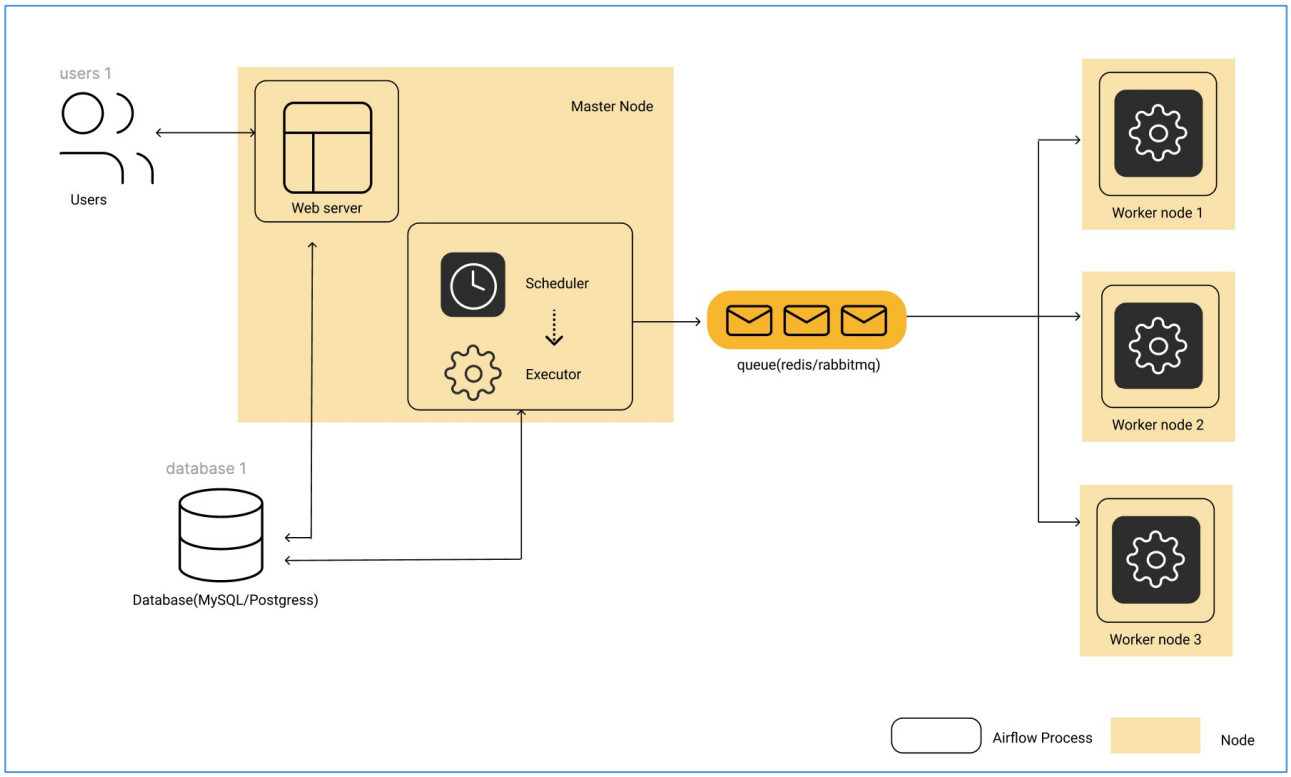


Figure 22 : Airflow architecture

At its core, Airflow uses directed acyclic graphs (DAGs) to define workflows in Python, allowing for dynamic generation and scalability. This approach enables data engineers and scientists to script complex data pipelines with ease, leveraging the full power and flexibility of Python. The platform supports integrating with numerous external systems, from traditional databases to modern cloud-based services, making it highly versatile for various data engineering needs.

Airflow's user interface offers visibility into the system, allowing users to monitor and troubleshoot their workflows in real-time. It provides detailed views of DAGs, including their dependencies, progress, logging, and the outcome of executed tasks. This level of transparency and control makes it easier for teams to manage their data workflows and ensure data quality and reliability.

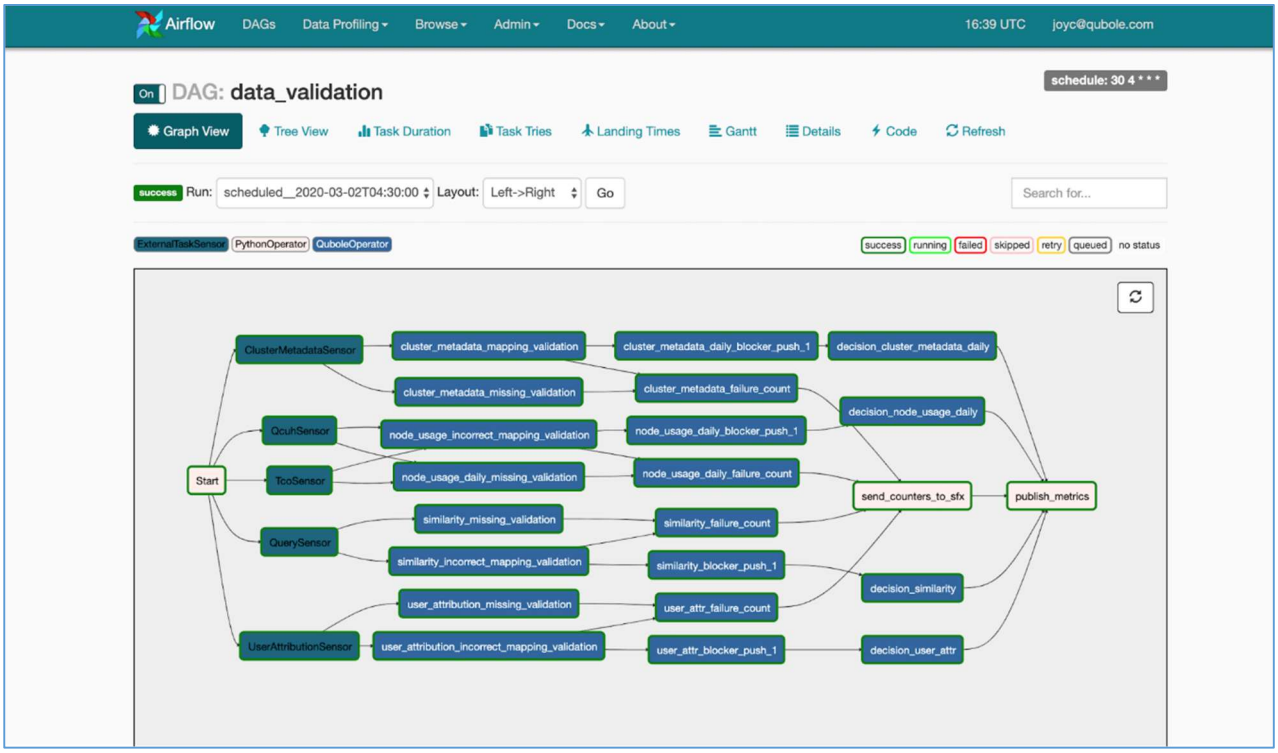


Figure 23 : Airflow authoring module.

The platform is designed to be extensible, supporting custom operators, hooks, and executors, enabling users to extend its functionality to meet their specific requirements.

Apache Airflow has become a vital tool in the field of data engineering, widely adopted for its efficiency in automating and managing complex data pipelines across various industries.

8.2.3.4 Service Log (Apache Logging service)

The Loggin service will support the logging on the execution of the processing modules. Basically, the platform will use the Apache Airflow logging services.

To decouple the processing to the orchestration tools it's proposed to use a specific class and table in the service database, to save the main log that will be produced. A single processing module (an instance) can send information about its execution state to the Context Broker.

8.2.4 DATA MANAGEMENT LAYER

The data management layer is the set of components that has in charge the storage and publishing of the geospatial (or not) data.

8.2.4.1 Context Broker (Orion)

The context broker is a software broker for managing context information. In the architecture the *Orion Context Broker* will be used. As the persistence component of the Context Broker, we will use Cygnus and MongoDB.

Orion Context Broker (see [RD9] for details) is a C++ implementation of the NGSIv2 REST API binding developed as a part of the FIWARE platform.

The broker allows to manage the entire lifecycle of context information including updates, queries, registrations, and subscriptions. It is an NGSIv2 server implementation to manage context information and its availability. The following image describes the Orion architecture.

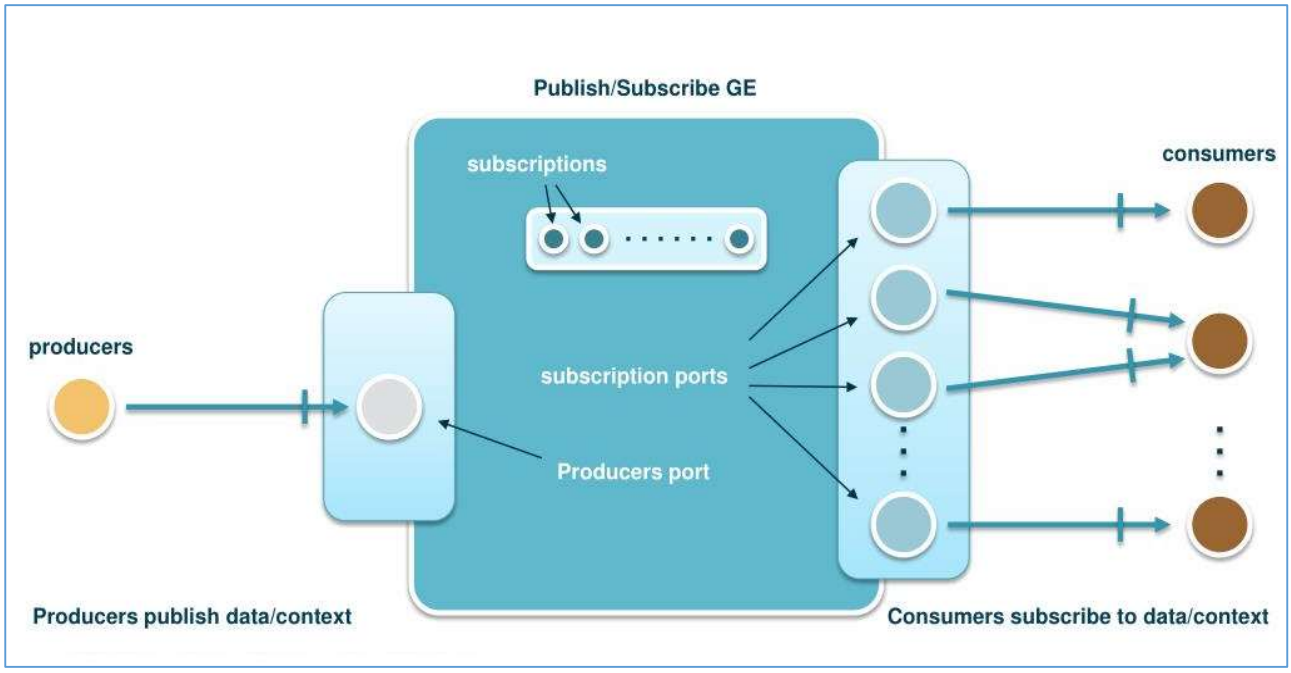


Figure 24 : Orion Context broker architecture

Using the Orion Context Broker, the platform is allowed to create context elements and manage them through updates and queries.

In addition is possible to subscribe to context information so when some condition occurs (e.g. the context elements have changed) the user can receive a notification.

The Cygnus agent (whose architecture is shown in the following diagram) plays the role of a connector between Orion Context Broker (which is a NGSI source of data) and many FIWARE storages such as CKAN, Cosmos Big Data (Hadoop) and STH Comet, it allows to add PosGIS, Kafka, Carto, etc as other non FIWARE storages to the FIWARE architecture.

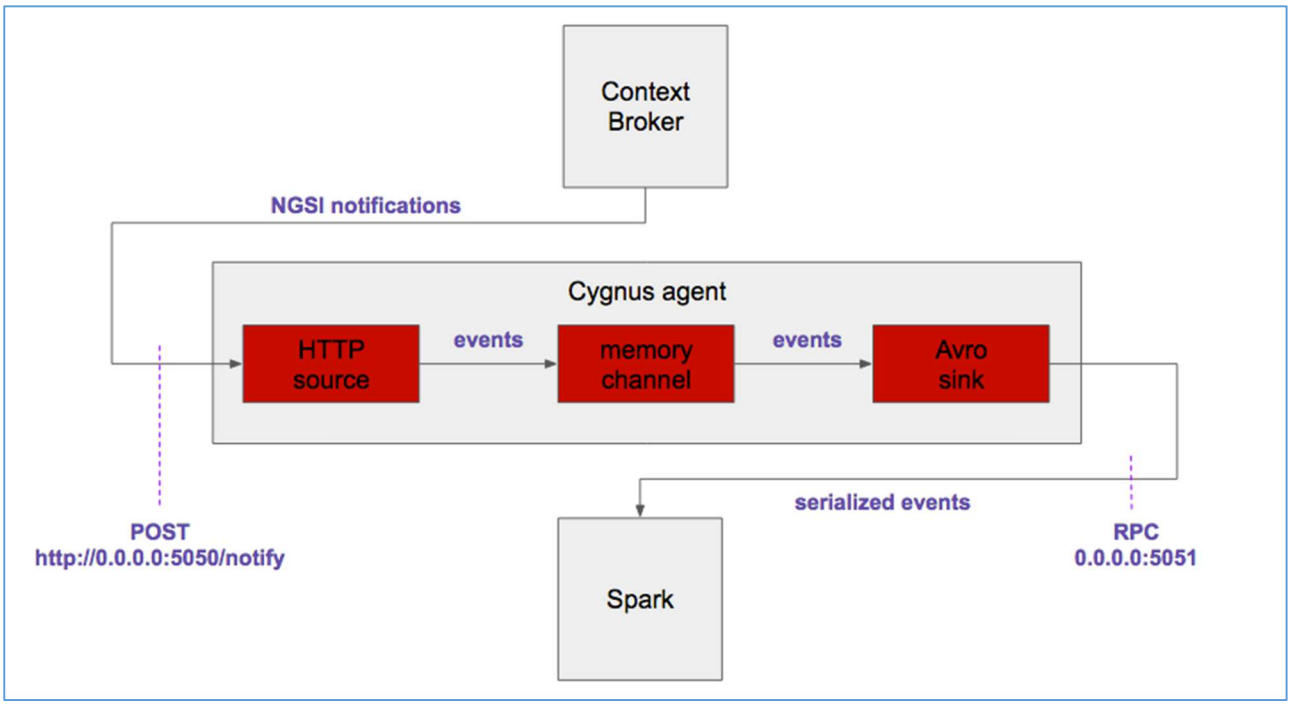


Figure 25 : Architecture of cygnus as part of the context broker.

8.2.4.2 Geospatial Services (GeoServer)

A geospatial server is a type of server designed to store, manage, and deliver geospatial data, such as maps and satellite images, over the web. It provides services that enable users to access and interact with spatial data through various applications. These servers handle queries for geospatial information, perform spatial analysis, and support the visualization of geographic information systems (GIS) data. Geospatial servers are crucial for applications requiring real-time location-based data analysis and mapping functionalities. Their primary function is to facilitate the seamless integration and sharing of geospatial data across different platforms and users.

The Geospatial server, used in the platform is *GeoServer* (see [RD7] for more details). This application server is one of the most used and stable Map Servers in GIS field, and is widely used in the great geospatial platforms to publish geospatial datasets. A brief description of Geoserver follows.

Geoserver is an open-source server designed to share and edit geospatial data. Designed for interoperability, it publishes data from any major spatial data source using open standards. Geoserver is a key component of the Geospatial Web, allowing users to view and edit geospatial data online. It supports a wide range of data formats, including GML, GeoJSON, and KML, and can serve as a backend for a variety of GIS (Geographic Information System) software.
GeoServer's capabilities are extensive. It allows for the sharing of geospatial data through web services following standards set by the Open Geospatial Consortium (OGC), such as Web Feature Service (WFS), Web Map Service (WMS), and Web Coverage Service (WCS). These services enable users to query, display, and manipulate geographic information in numerous ways.

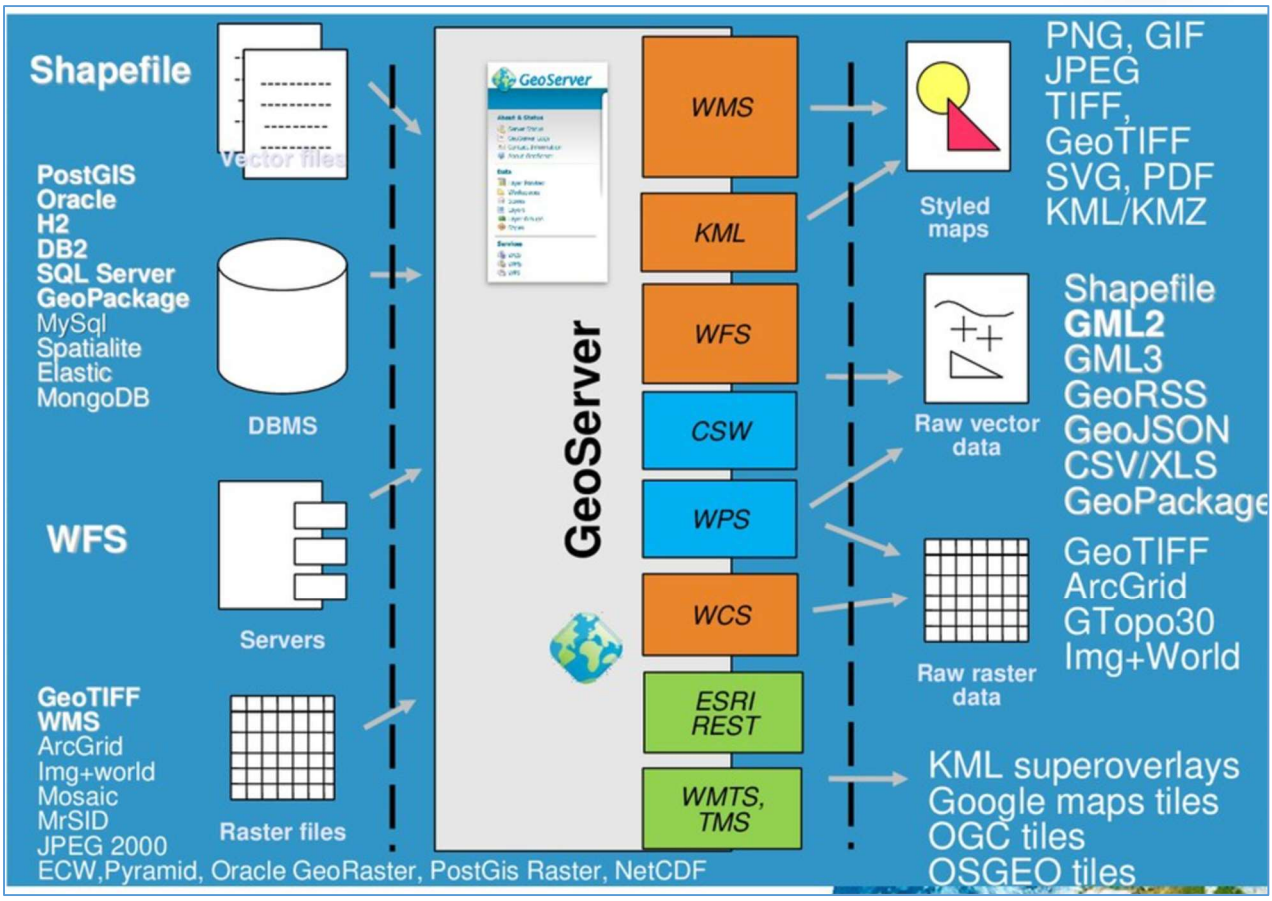


Figure 26 : Geoserver architecture

The platform is built on *GeoTools*, a Java library that provides tools for geospatial data. *Geoserver* can run as a standalone application or be integrated into existing web applications. It offers a web-based administrative interface, making it accessible to users with varying levels of technical expertise. Through this interface, users can manage layers, styles, and settings, as well as monitor server performance.

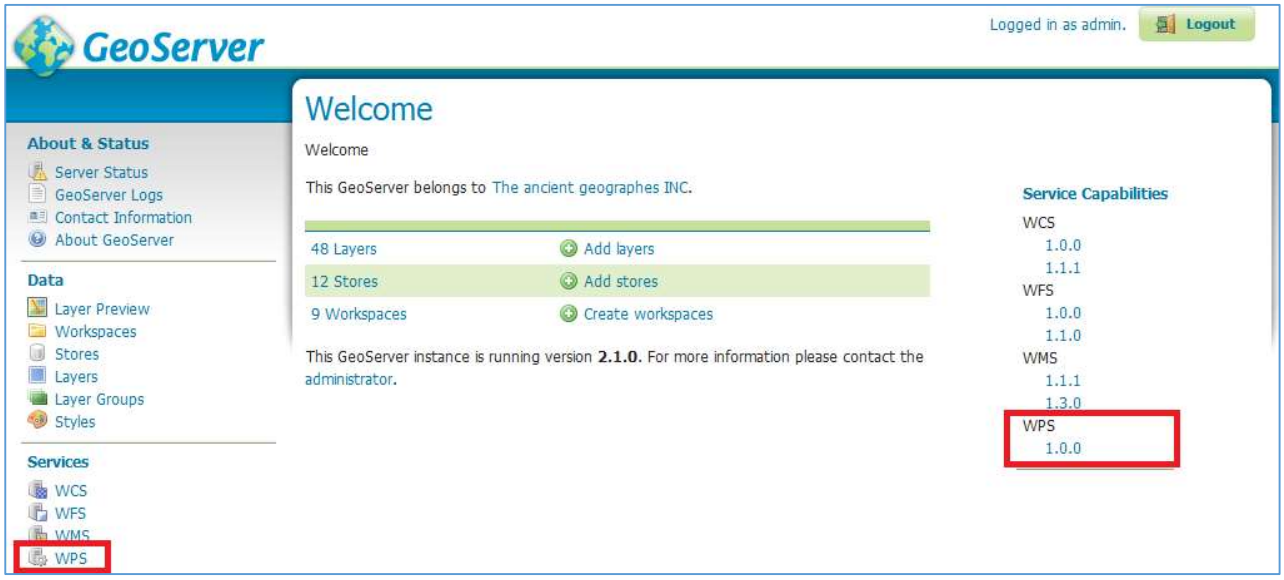


Figure 27 : Geoserver user interface

8.2.4.3 Geodatabase (PostGIS)

The Spatial/Non-Spatial DBMS used in the platform is PostgreSQL/ PostGIS. A brief description of it, follows. PostGIS is an open-source software program that adds support for geographic objects to the PostgreSQL object-relational database. Essentially, it "spatially enables" a PostgreSQL database, allowing it to store geographic data (geodata) and perform spatial queries and analysis. *PostGIS* complies with the Open Geospatial Consortium's (OGC) standards, ensuring compatibility with a wide range of GIS software and geospatial data formats.

PostGIS extends PostgreSQL by adding spatial types, indexes, and functions that are essential for processing geospatial data. The spatial types allow for the representation of point, line, polygon, multipoint, multiline, multipolygon, and geometry collections. These data types are used to represent locations, areas, and other spatial entities in a database.

PostGIS uses the R-tree-based GiST (Generalized Search Tree) indexing to efficiently query and manage spatial data. This allows for fast retrieval of spatial objects based on their location and geometric properties.

It provides a comprehensive set of functions for performing operations on spatial data, including spatial relationships (e.g., intersects, touches, contains), spatial measurements (e.g., area, distance, length), and transformations (e.g., reprojection, buffering).

PostGIS supports both geometry and geography data types. The geometry type is used for planar spatial data, while the geography type is designed for large-scale and spherical spatial data, such as coordinates in the WGS84 system.

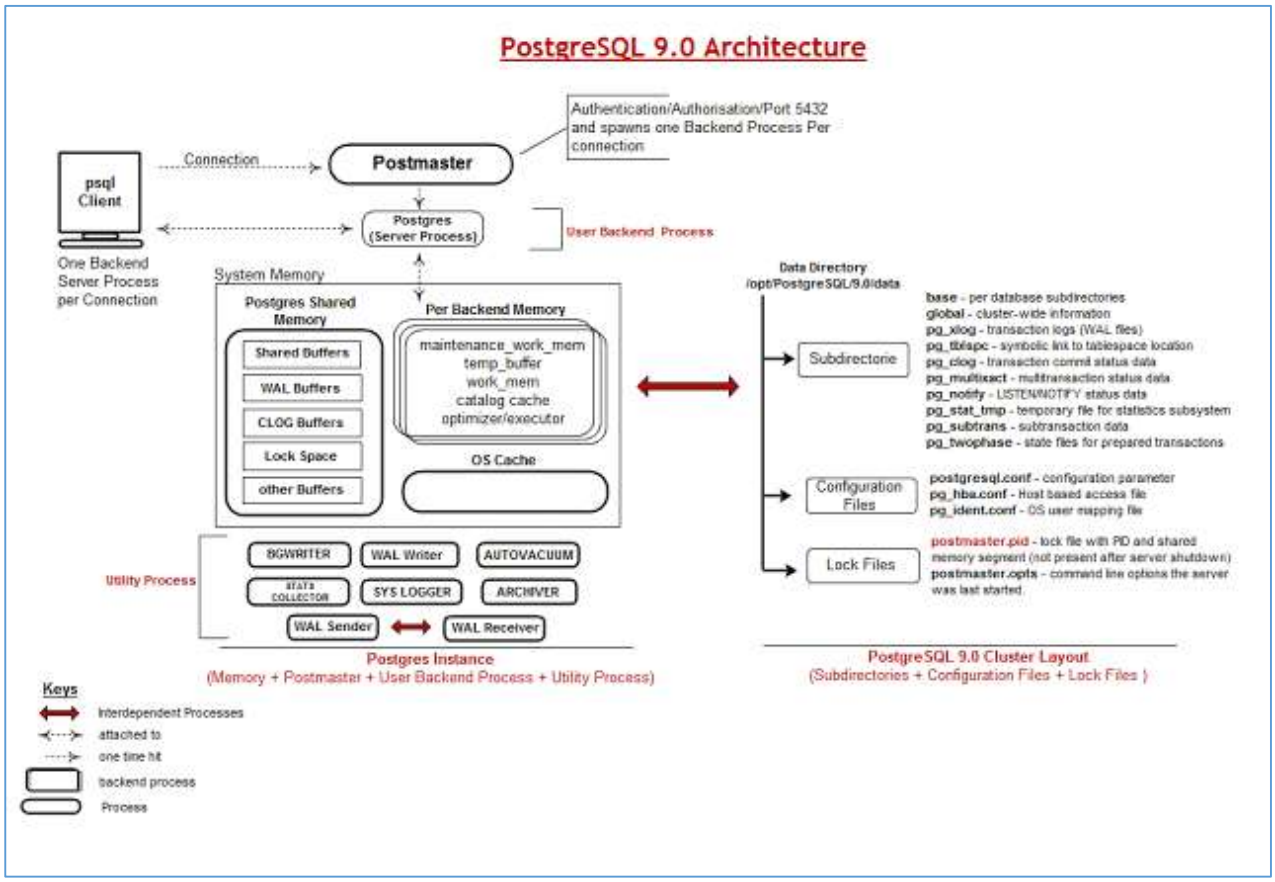


Figure 28: PostgreSQL architecture

PostGIS includes support for raster data, allowing for the storage, analysis, and manipulation of pixel-based data such as satellite imagery and digital elevation models.

PostGIS can be integrated with a variety of GIS software, both open-source (e.g., QGIS, GeoServer) and proprietary (e.g., ESRI ArcGIS), for advanced geospatial data analysis and visualization. Through its compatibility with web mapping technologies and standards, PostGIS serves as a powerful backend for web GIS applications, enabling dynamic spatial data queries and map rendering.

PostGIS allows for the extension of its capabilities through additional PostGIS modules, such as PostGIS Topology for managing topological data and PostGIS Raster for advanced raster operations. PostGIS has become the de facto standard for spatial databases in the open-source GIS community, widely recognized for its robustness, performance, and compliance with OGC standards.

It is extensively used in various domains, including environmental monitoring, urban planning, logistics, and more, for spatial data management, analysis, and visualization.

8.2.4.4 Alert Broker (Perseo)

An *Alert Broker* is a component that is listening to events coming from context information to identify patterns described by rules, in order to immediately react upon them by triggering actions.
As the Alert Broker the platform will use *Perseo*. This component is currently under evaluation.

Perseo is an Esper-based Complex Event Processing (CEP) software designed to be fully NGSI-v2-compliant. It uses NGSI-v2 as the communication protocol for events, and thus, Perseo can seamless and jointly work with context brokers. The context broker tested with Perseo and officially supported is Orion Context Broker.
STH is the module that allows queries to be executed on the persistence component, it will be developed ad-hoc.

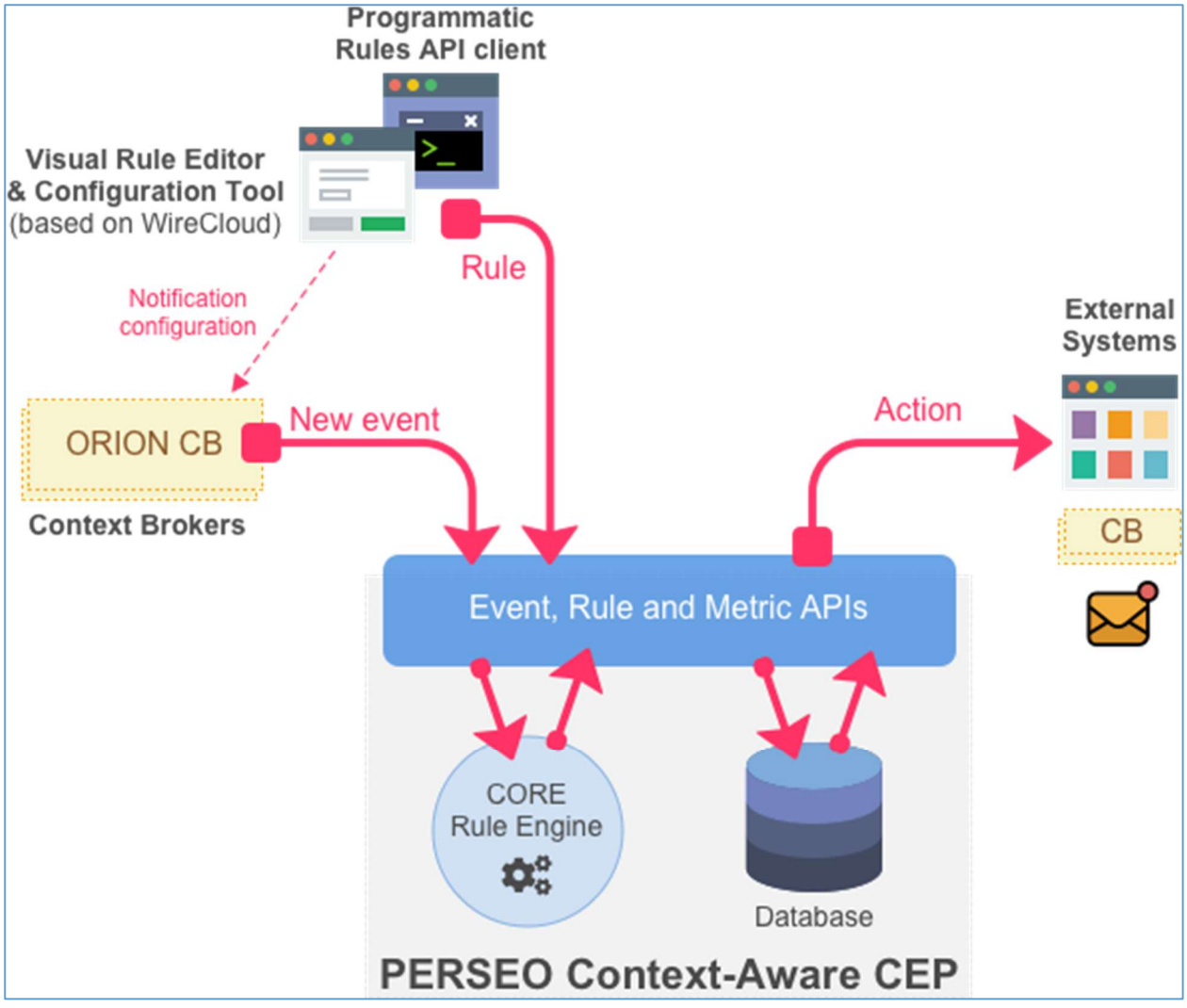


Figure 29 : Perseo architecture

8.2.4.5 Geo Catalog (Geonetwork)

Geographic data catalogues are information systems that organize, manage, and search metadata related to geospatial resources, such as maps, geographic datasets, and services. These catalogues facilitate the sharing and discovery of geospatial information among users and systems.

Protocol standards like ISO 19119 are crucial for ensuring interoperability among different geospatial systems. ISO 19119 specifies a framework for describing geographic services, including models and methods for their publication, discovery, and invocation. This standard facilitates the exchange of geographic information and the integration of services into broader applications.

An example of catalog is GeoNetwork, an open-source software for managing geographic data catalogues that implements OGC (Open Geospatial Consortium) and ISO standards for geospatial metadata. The main functionalities of a catalog include:

- Advanced management and search of geospatial metadata, allowing users to easily find geospatial resources based on various criteria. The ability to integrate metadata from different sources, promoting the aggregation of geospatial data.
- An interface for publishing and sharing metadata, which facilitates the distribution of geospatial information. Support for various metadata standards, ensuring interoperability with other systems and geospatial platforms.
- A catalog can integrats into a geospatial platform by acting as a central component for metadata management, thereby enabling the discovery and access to geospatial data and services. This integration supports the construction of coherent and interoperable Spatial Data Infrastructures (SDI), enhancing access to and use of geospatial information.

The OGC CSW (Catalog Service for the Web) is a protocol standardized by the Open Geospatial Consortium that facilitates the search, retrieval, and management of geospatial metadata through web services. This protocol enables interoperability between GeoNetwork and other platforms, allowing efficient access to and sharing of geospatial metadata. CSW protocols are useful for metadata archiving as they standardize the methods of data publication and search, thus improving the discovery and utilization of geospatial resources.

8.2.5 DATA COLLECTION LAYER

The data collection layer includes the group of components that manage external devices as collections. In the next sections we can present the technology usefully involved.

8.2.5.1 Device Manager (DEMA)

The Device Manager (DEMA) is a software component that allows a user, to a visual management of sensors located on a devices network.

As shown in the following images the manager can collect a set of sensors, easily accessible and associate them to a geographic position.

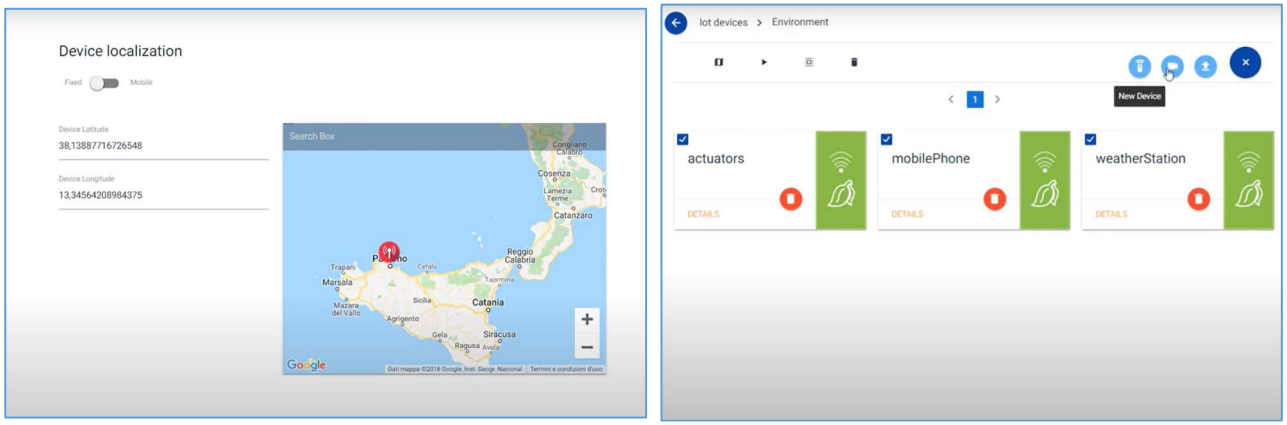


Figure 30 : DEMA UI Device Localization and configuration

8.2.5.2 IoT Agents

The FIWARE IoT Agents (see [RD5] for details) used for IoT data management. The representation of the data must be made following the NGSIv2 standard. The following image clarify the role of the IoT agents in the framework of the data acquisition.

The following diagram illustrates an Internet of Things (IoT) architecture. This schema represents the various components of an IoT system, showing how different agents using various protocols interact

with a central Context Broker and are managed by an IoT Agent Manager, all underpinned by a backend device management system.

At the top, there is the Context Broker component which interacts with the OMANGSI API (northbound interface). Below the Context Broker, there's a layer that includes multiple "IoT Agent-1" blocks, indicating there could be multiple instances of this agent.

These agents communicate with the *Context Broker* through dashed lines, signifying data flow or interactions, and are associated with different protocols:

- Ultralight 2.0 HTTP
- MQTT
- LWM2M/CoAP

To the right side of the diagram, there is the IoT Agent Manager which is responsible for creating and monitoring IoT Agents. Below the agents, there is an IoT Backend Device Management section.

Indicating the backend systems for managing IoT devices. This is labelled as the (southbound interfaces), as these are interfaces for internal or device-facing communications.

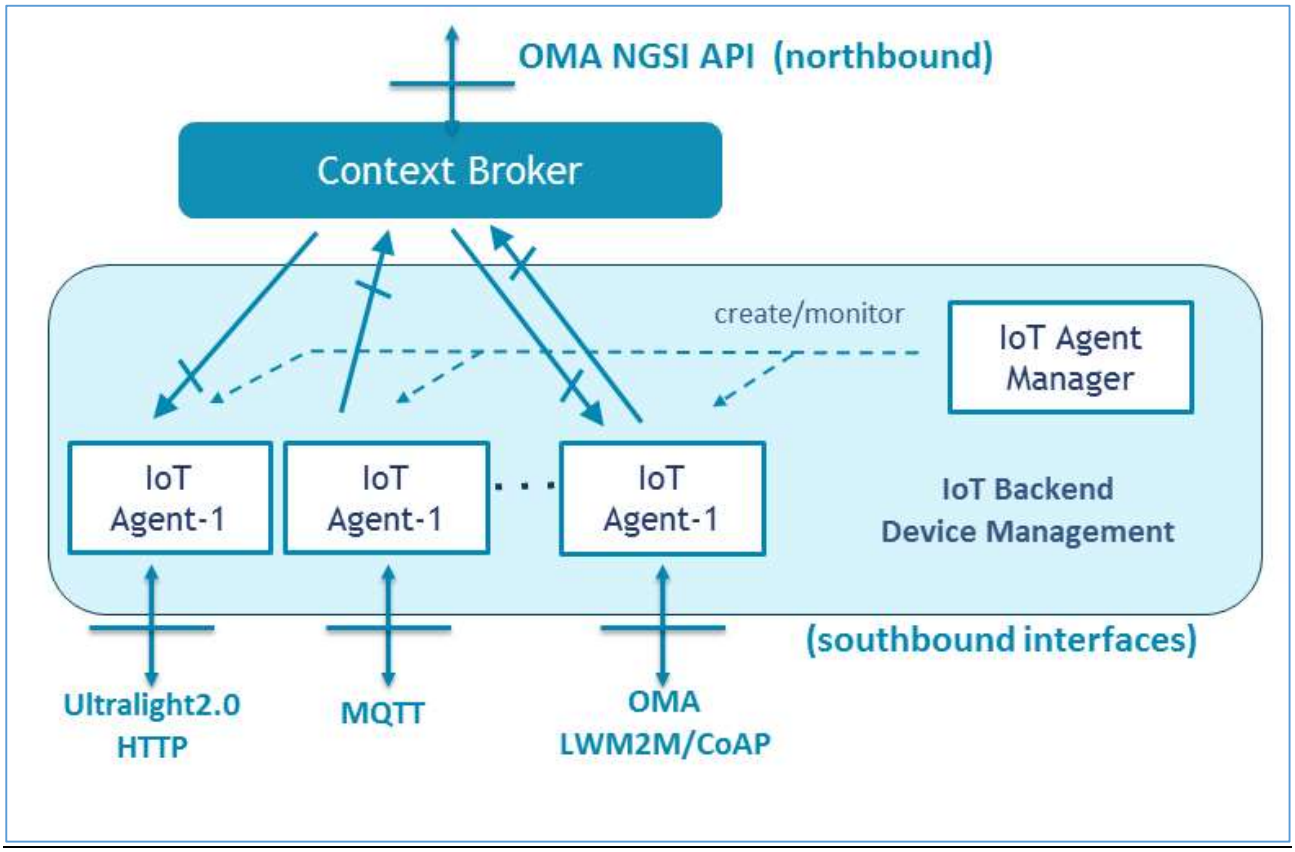


Figure 31: Context Boker architecture

For this purpose, there will be used NGSIv2 models useful for sensors and part of the FIWARE data models. A candidate library is the Smart Data Models *DeviceMeasurement* open-source library (see [RD6]). The choice to use a FIWARE data model is not mandatory. What is important is to comply with the NGSIv2 standard. The IoT Agent will be deployed as a docker containers.

8.2.5.3 Data connector

An alternative to the use of FIWARE IoT Agents, for IoT data management, is to develop a custom Data Connector (see next point) that sends the data directly to the Context Broker, the data format must still be in NGSIv2. The data connector must be dockized.

Data Connector. It is a custom connector for sending data. It can be developed using FastAPI. The Data Connector must be dockerized.

If data connectors want to send data to the Context Broker, the NGSIv2 protocol must be used. For connectors used to send large files (e.g., satellite imagery), a shared directory will be provided through which partners (via the scp protocol) can send files to the folder.

8.2.6 INTEROPERABILITY AND DATA HARMONIZATION LAYER

The interoperability and harmonization layer includes the set of components that allow harmonization of the data coming from the data collection layer. Its use is optional and application dependent. The framework will provide the software *Apache Airflow* for data harmonization.

8.3 DEPLOYMENT VIEW

A physical architecture refers to the tangible components of a computing system and their interconnections. This encompasses the hardware elements such as servers, storage devices, network routers, and the physical layout of these components within a data centre or across multiple locations. Physical architecture also extends to the software and operating systems installed on this hardware, focusing on how they are deployed to work together in a cohesive environment.

The design of physical architecture is crucial for ensuring that the system meets performance, reliability, and scalability requirements. It considers the physical connections between devices, the distribution of processing tasks, and the infrastructure needed to support data flow and access. In essence, physical architecture is about translating the conceptual and logical design of a system into a detailed blueprint that outlines the physical reality of how the system will operate, including considerations for maintenance, disaster recovery, and future expansion. Understanding the physical architecture of a system is essential for IT professionals to manage and optimize the infrastructure effectively, ensuring that it can support the organization's applications and services reliably.

A deployment view (described below in the diagram) illustrates the physical deployment of artifacts on nodes.

Artifacts represent tangible elements of a system, such as executables or database schemas, while nodes are typically physical entities where the artifacts are deployed, like servers or devices. The diagram shows how software is distributed across different hardware components, detailing the physical relationships and connections.

It is particularly useful for understanding the hardware topology on which a system will run, the distribution of components across various nodes, and the physical interconnections between them.

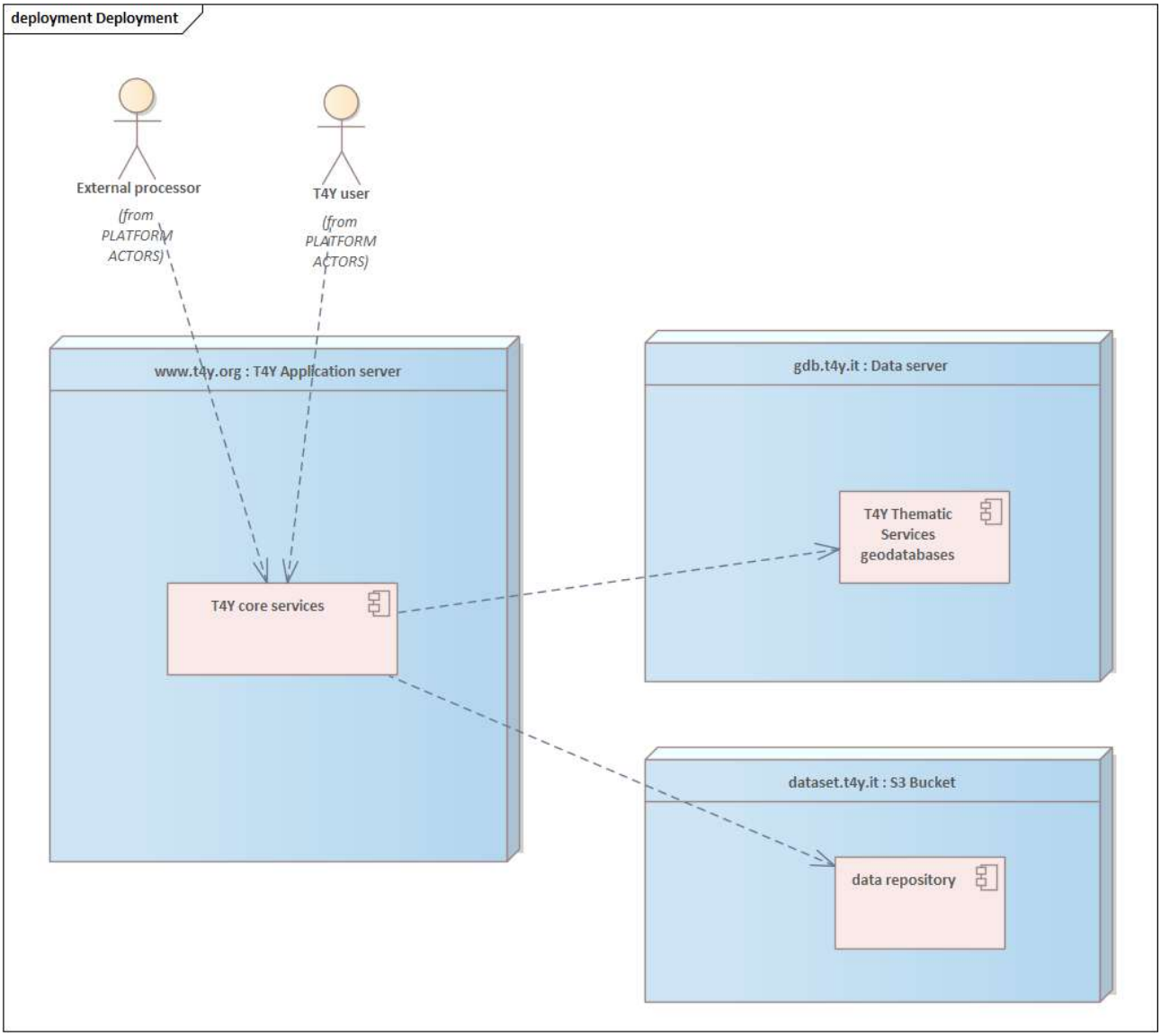


Figure 32 : Tech4You deployment diagram

8.3.1 Nodes details

#	Hostname	Description	Specs
1	www.t4y.it	T4Y application server	OS : Ubuntu Linux Server Core: - RAM: - Disk: -
2	gdb.t4y.it	geodatabase server (for vector products)	PostGIS geoatabase server Core: - RAM: - Disk: -
3	dataset.T4y.it	Data server of T4Y Raster Thematic products	AWS S3 Bucket server Core: - RAM: - Disk: -

9 ANNEX A - PLATFORM REQUIREMENTS

Enabling platform requirements are herein summarised and listed according to ISO/IEC/IEEE 29148.

The naming conventions are listed as follows:

RQ-PLT-<type>-<sequence>

where:

- <type> is a group of uniform functions.
- <sequence> is a sequence number (0nn x10)

Validation: T | D | I | A

9.1 RQ-PLT-GEN-010 Thematic services integration

Description

The platform shall provide facilities to support external providers to integrate their monitoring thematic services.

Validation: T

9.2 RQ-PLT-GEN-020 Geospatial navigation tool

Description

The platform shall enable user to integrate new layers in the geo-spatial navigation tool

Validation: T

9.3 RQ-PLT-GEN-030 Data analytics

Description

The platform shall allow third parties to integrate analytics to be visualized by platform users.

Validation: T

9.4 RQ-PLT-GEN-040 T4Y Core Services

Description

The platform shall allow third parties to integrate a new thematic service by acquiring, processing, storing, cataloguing, and publishing geospatial information.

Validation: T

9.5 RQ-PLT-GEN-050 New thematic services processing orchestration

Description

The platform shall enable the processing of new algorithms by means of processing and orchestration facilities.

Validation: T

9.6 RQ-PLT-GEN-060 External sources accesses

Description

The platform shall allow to access data from various external sources such as sensors (proprietary and standard protocols), EO data catalogues, and external data sources.

Validation: T

9.7 RQ-PLT-GEN-070 Manual or scheduled processing facilities

Description

The platform must allow the manual execution of data processing, or a scheduled timing of the processing.

Validation: T

10 ANNEX B - REQUIREMENTS vs DESIGN TRACEABILITY MATRIX

The hereafter table shows the relationships between requirements and Design components.

#	Requirements	Design references
1	RQ-PLT-GEN-010 Thematic services integration	§5.1
2	RQ-PLT-GEN-020 Geospatial navigation tool	§6.3.3
3	RQ-PLT-GEN-030 Data analytics	§6.3.4
4	RQ-PLT-GEN-040 T4Y core services	§6.4.1
5	RQ-PLT-GEN-050 Thematic services processing orchestration	§6.4.2
6	RQ-PLT-GEN-060 External sources accesses	§8.2.6
7	RQ-PLT-GEN-070 Manual or scheduled processing facilities	§8.2.3.1

11 ANNEX C - PLATFORM SERVICES

In the present annex a detailed list of the core and thematic services is available. The purpose of this section is to support third parties to a better understanding of the interaction and integration of the platform.

11.1 CORE SERVICES

Here, below the detailed definition of the core services provided by the T4Y platform is described.

11.1.1 PROFILING AND SECURITY SERVICES

This section lists the security services. They aim to manage all aspects of access and user data management, as well as their profiling. It allows a unique single sign-on access to every tool made available by the platform.

#	Resource	Parameters	Note	Protocols	Format
1	Creating, modifying, and deleting a user	About the Service Manager	It provides for the issuance of a unique management key to access the service's authoring services.	REST Web	JSON
2	Login, Logout, User Profiling	User credentials	Password sign-in functionality to the Service Management Portal.	REST Web	JSON
3	Change of password and profile data management	Profile Data & Password	Profile editing and Password Change Functionality.	REST Web	JSON

11.1.2 THEMATIC SERVICES MANAGEMENT

The present section lists the thematic services management functionalities.

#	Resource	Parameters	Notes	Protocols	Format
1	Creating, modify and deleting a *thematic service*.	Primary information about the service - Data Sources - Processing Steps - Archiving results - Cataloguing - Publication - Ancillary data	The service provides for the setting of the data sources, and if they are not available, the request with the data structure in JSON format. The external provider will need to detail the specifications of the expected data.	REST Web	JSON HTTP
2	Creating, modify and deleting an external processing module	Processing Module: - Input Parameters - Output parameters - Security key	Each external module must have an end point exposed and provide a standard application interface that is defined in this section.	REST	JSON
3	Creating, modify and deleting a *processing workflow* related to a service.	Workflow details	Workflow Module associates the orchestration project with a specific workflow.	REST	JSON
4	View of running processes (*jobs*)	Jobs details	Provides a detail object for each run with the execution status, and other reference information.	REST	JSON
5	View processing *logs*	job id	Provides logs for each execution of the running processes.	REST	JSON

11.1.3 ORCHESTRATION SERVICES

Orchestration functionality of the processing chain.

#	Resource	Parameters	Note	Protocol	Format
1	Creating, modify and deleting an Airflow workflow project	Workflow design, processor chaining	Apache AirFlow technology is used to design the processing chain. It can be used both via the user interface and via the REST protocol in standard OpenAPI format.	OpenAPI (AirFlow) Web interface	-
2	Running, deleting a processing workflow	workflow	It allows you to access the management of workflows, checking their status and completion.	OpenAPI (AirFlow) Web interface	-
3	View Logs	Running Jobs	Allows access to logs	OpenAPI (AirFlow) Web interface	-

11.1.4 DATA ACQUISITION AND EVENT BROKER SERVICES

These services are functionalities for accessing measurement data from sensors and generic contextual data.

#	Resource	Parameters	Note	Protocol	Format
1	Data Source definition	Data source Structure	Data source structure creation services. A JSON file is created that serves as a mapping to the measurement data.	REST	JSON
2	NRT Measurement Reading	Data source	Reading of the latest measurement data	REST	JSON
3	Reading historical measurements	Data source	Reading of historical measurement data	REST	JSON
4	Context Availability	Entity NGSIv2	Represents the operations to identify which context data sources and entities that are connected to the platform	REST	JSON/NGSIv2

#	Resource	Parameters	Note	Protocol	Format
5	Query and Subscription	Entity NGSIv2	Represents the synchronous and asynchronous interactions with context data source.	REST	JSON/NGSIv2

11.1.5 IoT AGENTS SERVICES

The present section shows the IoT Agents services specifications. An IoT Agent is a software module that enables sensors and actuators to send their data to and be managed from a Context Data Broker using their own native protocols.

IoT Agents are also able to deal with security aspects (authentication and authorization of the channel) and provide other common services to the device programmer.

#	Resources	Parameters	Note	Protocol	Format
1	Managing of information exchanged with devices	Device	e.g. measure reading	REST, device native protocol	JSON
2	Execution of commands or actions by an actuator device.	Device command	-	REST, device native protocol	JSON

11.1.6 IoT DEVICE MANAGER SERVICES

Hereafter the specifications of the IoT devices management system are listed.

IoT Device Manager allows the centralized management of IoT devices, providing the possibility for a decision-maker to increase the situational awareness of a territory, exploiting all the sensors available in a real environment.

#	Resource	Params	Note	Protocol	Format
1	Area Management	-	• *Area's list. Displays the list of areas monitored by the system.* • *Adding an Area. It allows the user to add a new area by specifying its name.* • *Geographic Position of an Area. The IoT Device Manager module, based on the name of the area, obtains the information of its geographical position (latitude and longitude).* • *Deleting areas. It allows the user to delete a particular area or a set of areas from the list of managed areas.* • *Viewing and Editing area Details. It allows the user to view the details of a particular area, listing its services. For each of them it provides the tools for their modification*	Web/HTTP	-
2	Service Management	-	• *Services list*. Display the list of services of a particular area previously selected. • *Adding a service*. It allows the user to add a new service by selecting its category. • *Deleting services*. It allows the user to delete a particular service or a set of services from the list of services of the particular area selected. • *Viewing and Editing service Details*. It allows the user to view the details of a particular service, listing its sensors. For each of them it provides the tools for their modification	Web/HTTP	-
3	Sensor Management		• *Sensor's list. Displays the list of sensors present in the previously selected service of the particular area, also previously selected. The list of sensors is of two types.* • *Adding a sensor. It allows the user to add a new sensor or a block of sensors (massive addition mode), using the relative commands.* • *Deleting sensor. It allows users to delete a particular sensor (single selection) or a set of sensors (multiple selection) from the list of sensors of a particular service selected.* • *Viewing and Editing sensor Details. It allows users to view the details of a particular sensor (both "device" and "webcam"), listing its characteristics. For each of them it provides the tools to:* ▪ modify: modification of the specific attributes of the sensor. ▪ duplicate: copy of the sensor with all the attributes, for a quick deployment	Web/HTTP	-

#	Resource	Params	Note	Protocol	Format
			of sensors with the same characteristics. ▪ view info: synthetic view of the attributes. ▪ activate / deactivate remote control and command of the sensor status. ▪ send commands: remote control of the sensor activities.		

11.1.7 ALERT BROKER SERVICES

This service analyses the results provided by the analysis layer and field sensors in order to generate alerts on possible critical situations emerging from the analyses themselves. Alerts are notified via email or SMS as well as to any external modules or applications via REST API.

#	Resource	Parameters	Note	Protocol	Format
1	Sending Mail	Recipients specified in a configuration file	-	REST	NGSIv2
2	Sending SMS	Recipients specified in a configuration file	-	REST	NGSIv2
3	Sending HTTP Request	URL specified in a configuration file	-	REST	NGSIv2

11.1.8 DATA HARMONIZATION SERVICES

The present section describes the harmonization services.

#	Resource	Parameters	Note	Protocol	Format
1	Harmonizer of data formats.	data structures	Open AirFlow technology is used to design the harmonization workflow. It can be used both via the user interface and via the REST protocol in standard OpenAPI format.	OpenAPI (AirFlow) Web interface	-

11.1.9 DATA CONNECTORS

Hereafter the data connection specifications details, as required in the platform.

Custom data connectors can be developed (which must be provided to the platform in a dockerized form), that interface with the data sources and supply data to the harmonization layer for further processing, or to the context broker using the NGSIv2 protocol, or to the data storage service using the S3 protocol (or equivalent).

#	Resource	Parameters	Note	Protocol	Format
1	Custom data connectors	Data sources	Provided to the platform in a dockerized form	NGSIv2, S3	-

11.1.10 DATA STORAGE SERVICES

Hereafter the data storage services specifications are shown in the table below.

#	Resource	Parameters	Note	Protocol	Format
1	Products geodatabase access	Database, schema, table	Internal database access functionality for saving vector products. Accessible only from a development environment.	JDBC or equivalent	-
2	Products Repository storage	Path repo	Functionality to access the internal folders for saving the files produced. Accessible only from a development environment.	AWS S3	-

11.1.11 GEOSPATIAL DATA PUBLISHING SERVICES

#	Resource	Parameters	Note	Protocol	Format
1	Access to Web Map Services (WMS)	Extent, layer	OGC Standard	OGC WMS	XML
2	Access to Tiled Web Map Services (WMTS)	Extent, layer	OGC Standard	OGC WMTS	XML
3	Access to Web Feature Service (WFS)	Extent, layer	OGC Standard	OGC WFS	XML
4	Editing a GS geospatial services	datastore	Accessing Geoserver	HTTP/Web	-
5	Editing a GS data store	External source	Accessing Geoserver	HTTP/web	-
6	Editing a GS Workspace	-	Accessing Geoserver	HTTP/web	-

11.1.12 CATALOG SERVICES

#	Resource	Parameters	Note	Protocol	Format
1	Metadata generation	-	Create a metadata file	OGC CS-W	XML
2	Metadata archive	-	Archiving a new metadata in the theme service catalog	OGC CS-W	XML
3	Metadata discovery	-	Queries to the catalog	OGC CS-W	XML

11.1.13 DATA PRESENTATION

Hereafter the data presentation services.

#	Resource	Parameters	Note	Protocol	Format
1	Map navigator	-	Geo-navigation app, including Front-end: graphical user interface (UI) functionality, Back-end: features and libraries to support the front-end (e.g. download, data upload, etc.). A typical geo-navigator layout is show in the following image. The main parts are highlighted: • Central map area • navigation toolbox • time slider • general menu • layer TOC • query tool • processing panel (for enabled users)	Web, OGC protocols	XML, JSON
2	Analytics tool	Queries	• Tool for the extraction of analytics. Representation of the processed products. • Representation of diagrams, pies, and all the graphs.	Web, OGC protocols, REST	JSON XML

11.1.14 EXTERNAL PROCESSING SERVICES

Hereafter the external processing services

#	Resource	Parameters	Note	Protocol	Format
1	Run external processing facility	Processor id Processing parameters	Executes a processing on the external processing facility.	REST	JSON
2	Completion call-back	Process id	Allow to confirm the processing completion.	REST	JSON

11.2 VALUE-ADDED CANDIDATE SERVICES

Hereafter there will be detailed the thematic services, as possible candidate to be available as value added service in the platform.

11.2.1 SILA-FIRES

Area of interest: Sila region (Calabria-Italy).

#	Resource	Parameters	Note	Protocol	Format
1	Estimating Flame Front	coordinates	Generates one or more polylines that describe a flame front, starting at a defined point.	REST	JSON

11.2.2 CRATI

Area of interest: CRATI river basin (Calabria-Italy)

#	Resource	Parameters	Note	Protocol	Format
1	Flow Hydrograph	coordinates	Generates a table that contains a flow hydrograph.	REST	JSON
2	Flow Hydrograph graph representation	coordinates	It integrates a graphical representation of the flow rate.	REST	JSON

12 ANNEX D - SERVICES API

The present section describes the core API RESTful interface.

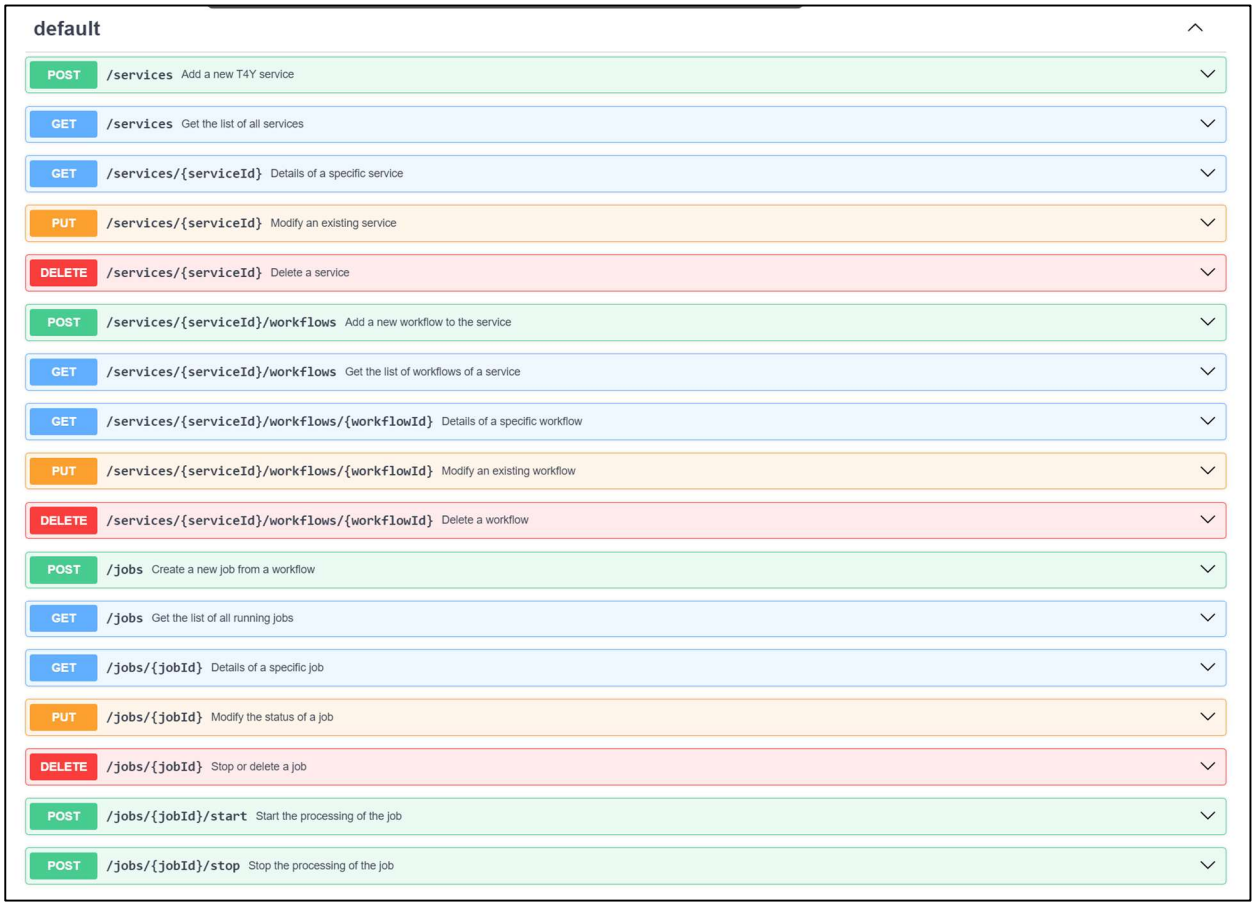


Figure 33 : T4Y Api swagger

13 ANNEX E – TECH4YOU ARCHITECTURE (SLIDE)