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

multi-Risk sciEnce for resilienT commUnities undeR a changiNgclimate

Codice progetto MUR: **PE00000005** – CUP LEAD PARTNER C93C22005160002

Deliverable title: Definition of a high-level reference architecture able to conceptually integrate models and input/output data, adopting standard data models

Deliverable ID: 1.5.1

Due date: June 1st, 2025

Submission date: June 1st, 2025

AUTHORS

Tasso Alberto (Tasso A.); Paolo Campanella (Campanella P.); Menapace Marco (Menapace M.): Pintus Fabio (Pintus F.)

1. Technical references

Project Acronym	RETURN
Project Title	multi-Risk sciEnce for resilienT commUnities undeR a changiNg climate
Project Coordinator	Domenico Calcaterra UNIVERSITA DEGLI STUDI DI NAPOLI FEDERICO II domcalca@unina.it
Project Duration	December 2022 – November 2025 (36 months)

Deliverable No.	DV#.1.5. 1 Definition of a high-level reference architecture able to conceptually integrate models and input/output data, adopting standard data models
Dissemination level*	
Work Package	WP 1.5.
Task	T#.# - Task Title
Lead beneficiary	POLIMI
Contributing beneficiary/ies	CIMA, POLIMI

* PU = Public

PP = Restricted to other programme participants (including the Commission Services)

RE = Restricted to a group specified by the consortium (including the Commission Services)

CO = Confidential, only for members of the consortium (including the Commission Services)

Document history

Version	Date	Lead contributor	Description
0.1	02/05/2025	Alberto Tasso (CIMA)	First draft
0.2	15/05/2025	Francesco Ballio (POLIMI)	Critical review and proofreading
0.3	20/05/2025	Fabio Pintus (CIMA)	Edits for approval
1.0	30/05/2025	Francesco Ballio (POLIMI)	Final version

2. Abstract

This document outlines the high-level reference architecture for the Digital Twin platform for the Vertical Spoke VS 1 within the "RETURN" project, which focuses on multi-risk science for resilient communities under a changing climate.

The document describes the integration of models and input/output data using standard data models. Key themes include:

- **Execution Platform (WASDI):** This is a cloud-based analytical platform designed for the development and deployment of Earth Observation (EO) and general-purpose scientific applications. It provides user interfaces and APIs for running applications, exploring data products, and managing workspaces. WASDI's architecture includes components like a web server, database, scheduler, launcher, and supports auto-deployment of user-created processors. It also features data provider abstraction and libraries for various programming languages to facilitate application development.
- **Digital Ecosystem:** This component aims to provide an accessible interface for non-scientific users to explore and utilize research results from the "RETURN" project. Built on top of WASDI, it offers customized user interfaces and manages workspaces for users to access and work with data and services. The Digital Ecosystem includes identity management and mechanisms to control resource access, ensuring a user-friendly experience for stakeholders like public administration and professionals.

The document sets the stage for the solution developement, highlighting its goals of delivering innovative scientific results and making them available to a broad audience. It distinguishes between the needs of the scientific community, who will use the WASDI platform, and other stakeholders, who will interact with the Digital Ecosystem. The architecture is designed to accommodate these different user groups and their specific requirements.

3. Table of contents

List of Figures

4. Introduction

The purpose of this document is to describe the overall architecture of the Digital Twin platform to be used in the context of the Spoke VS1 in the "Return" project.

The project is carrying out a number of research activities that will lead to innovative scientific results and that could be useful to stakeholders outside the scientific community.

In the context of this project, we have identified two well-defined classes of users: members of the scientific community, that will bring innovative services and other research results, and a potentially broader audience of stakeholders like public administration and professionals that can leverage the research results.

Since different types of users have different needs, the solution provides two different user interfaces: the scientific community can work directly on the WASDI platform, while the stakeholders will use the Digital Ecosystem UI.

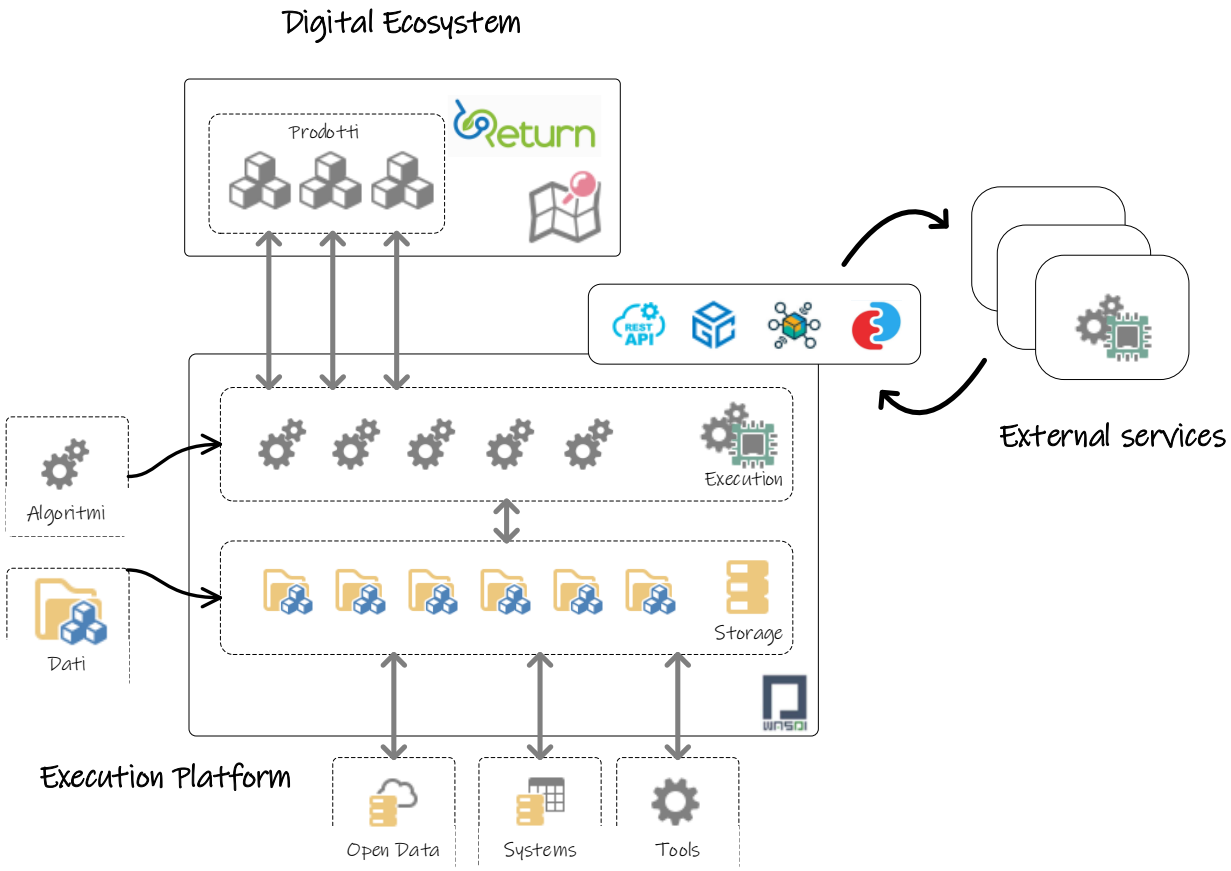

Figure 1 - High level architecture

The two main elements composing the solution, the Execution Platform and the Digital Ecosystem, will be described in detail in the next sections.

5 Execution Platform – WASDI

The execution platform is based on WASDI, a fully scalable cloud-based analytical platform that allows development and deployment on the cloud of EO and general-purpose scientific applications, without the need for any specific IT/ Cloud skills.

The platform offers end users the opportunity to run applications from both a dedicated user-friendly interface and an API based software interface. Users can also explore a catalogue of available data products and connect other data sources to the platform.

Scientists and researchers can develop new processors in their own environment using their own language, integrating the WASDI Library; once the processor is finished, it can be deployed to the cloud with a drag and drop of the source code. Additionally, they can also develop directly on-line using notebooks in their WASDI workspaces.

From the web interface, it is also possible to manage the users' workspaces, to search the linked data catalogues and to explore the results of the applications in the embedded web-GIS.

WASDI is an open-source project that can be installed and hosted on private and public clouds, a managed version is also available through a commercial subscription as SaaS.

5.1 WASDI platform architecture

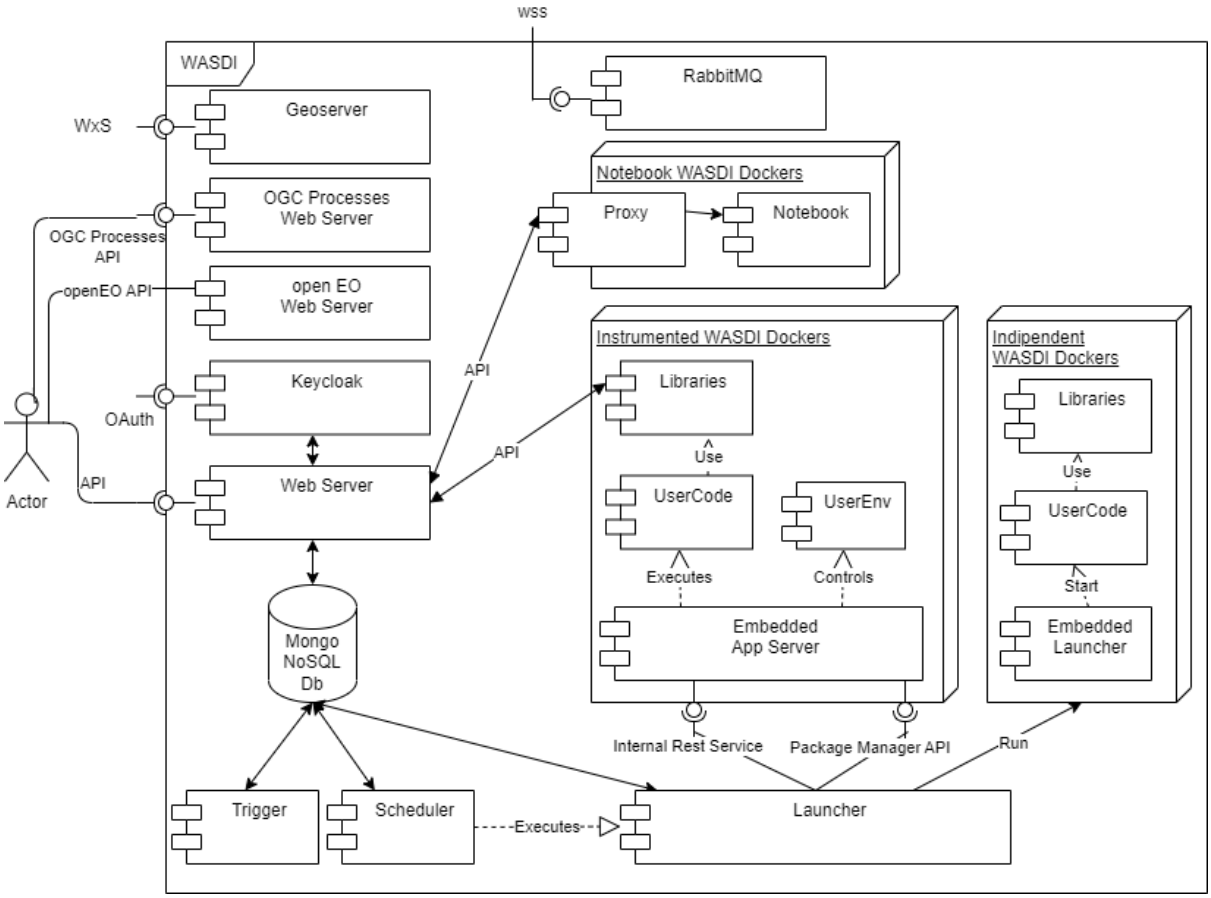

Figure 2 - WASDI platform components

The main entry point is the **Web Server**. The server exposes the WASDI REST API. Can be called both by the client user interface and directly by API.

The WASDI processing services are exposed also using the OGC Processes API standard: a user or a third-party system can decide to use the native WASDI API or the OGC standard to get the list of available processors, start a Job and get the results.

WASDI implements also an openEO Backend, that can be used by the client using the official openEO Client library.

The Web Server handles the authentication and authorization using the internal **Keycloak** server [https://www.keycloak.org/]. This is federated also with external networks like COIH and EduGain.

The WASDI data is hosted on a **NoSQL Mongo database**. The Web server is connected to the database to read and write data.

All the work done by users in WASDI is linked to a workspace: the workspace is a space where the user can ingest EO images and other data, run processes, create results. Each user can create multiple workspaces.

When a user starts an operation, the request is saved on the database.

WASDI supports different operations. Each operation is called Process Workspace, since it is a process that is executed in a Workspace.

Each WASDI process has a state. The processes states are:

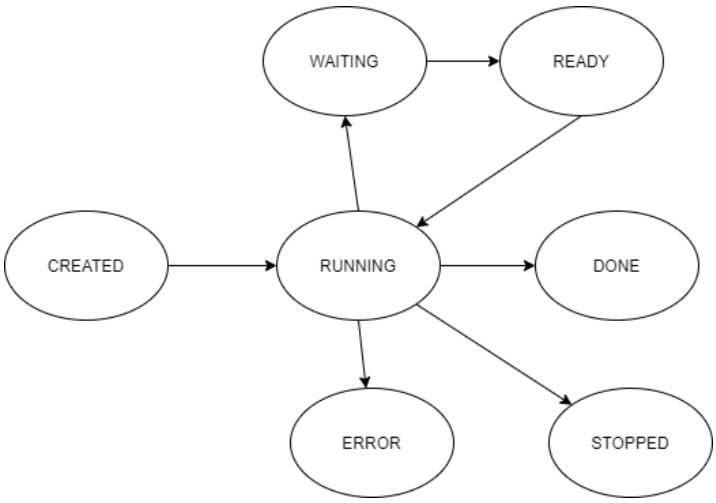

Figure 3 - Process Workspace State Diagram

When the user starts a process, a ProcessWorkspace is written in the database with the state CREATED.

The **Scheduler** is a process that runs in the server and polls the database to find new processes to start. When there is space in the queue, the scheduler executes the Launcher.

The **Launcher** is the wrapper of every WASDI operation. It can handle some operations itself (fetch data, execute snap workflow, mosaic, subset…) and it is responsible for triggering the execution of all the WASDI Apps.

To create a WASDI App a user must include the specific WASDI **Library**: the libraries are written for different programming languages. The goal of the libraries is to authenticate in WASDI and abstract the access to the resources in a workspace. The libraries are designed to work both on the User PC and on the server.

When the processor is deployed in WASDI it runs in a dedicated docker. The main application docker is instrumented: the docker starts an **embedded web server** that exposes its API only on the internal network and can be called by the Launcher.

To start an application the Launcher makes a run request to the docker server. This docker server executes inside the docker the user code; the libraries recognize it to be on the server and can communicate directly with the WASDI Web server to perform the required applications.

The internal server exposes also a subset of APIs that can be used internally by the Launcher to interact with the Package Manager of the specific language inside the docker (i.e., pip, conda, nuGet..).

Another Docker can be hosted in WASDI to expose a Jupiter Notebook and let the user work directly online in a specified workspace. The Docker hosts the Notebook server and also a Telerik dynamic firewall that is chained with ngnix to let the user reach the notebook in a secure way.

The Independent WASDI Docker is another template that can be used with the platform: this docker is very similar to the instrumented WASDI one but is designed to be able to run also one-shot and also outside a computational node (ie in any other cloud environment or local computer able to run a docker image). In this case, since the instance can also be outside the WASDI private network, the internal web server is not active and the Docker shutdown automatically once the application is ended.

A **GeoServer** instance runs is linked to WASDI to add a WxS interface, both to see maps and to support WPS to start services.

A **RabbitMQ** server is also installed to provide async communications between all the components.

5.2 Computational Nodes

The schema presented in Figure 2 - WASDI platform components refers to a single node, while WASDI is designed to support different nodes also in different cloud environments.

The main node is the one hosting the main database and API. From the technical point of view, it can be used also as computational itself or not, depending on the available resources.

Computing Nodes share the same components of the main node; the difference is in the configuration that instructs the server if it is the main one or a computational one.

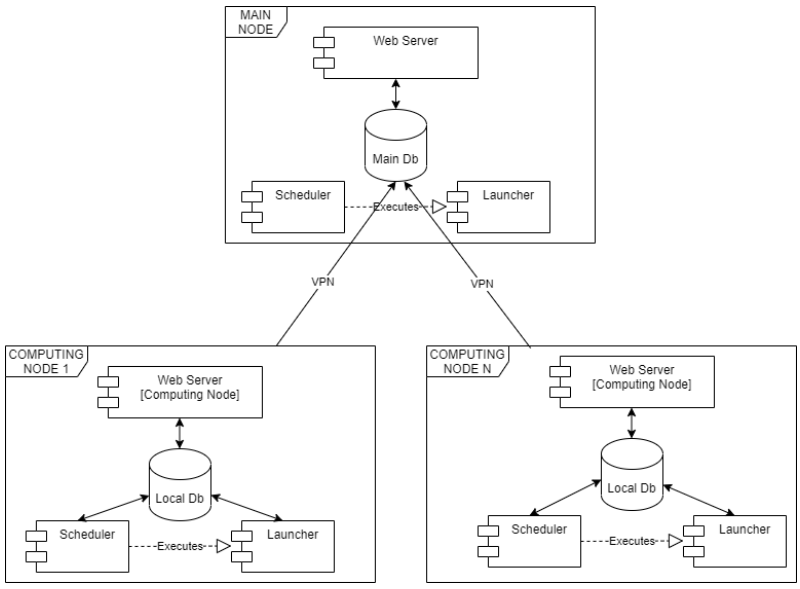

Figure 4 - Connections between main and computing nodes

Each computing node has a local Mongo Database and exposes only a subset of the main server API. All the nodes are connected to the main database using a VPN (or SSH tunnel) connection.

The local database is used only for the local process workspaces (and processor logs) while all the other information is stored in the main one and not replicated.

Libraries and clients are designed to recognize where the workspace is where the process is running and they route the requests to start and stop processes, to read or update the state, to read the logs directly to the involved node.

This lets the system distribute the overall load of requests and lets a local scheduler and launcher to work in any node, assuring a more robust solution.

To add a computational node, once installed, is sufficient to add it to the node list of the main server and can be used without any discontinuity of the services.

5.3 Auto-deploy

Auto deployment of Users' processors is one of the main features of WASDI. The paradigm to move processors to the data is well known in the EO Community. Available platforms proposed different solutions:

- Develop directly online.
- Activate Virtual Machines in the cloud.
- Host third party dockers.

While all these solutions work and will be supported also in WASDI, a huge community of EO Experts does not have the necessary IT skills needed.

The solution implemented in WASDI is to let them work exactly has they do now and deploy the service with a simple drag and drop of the code.

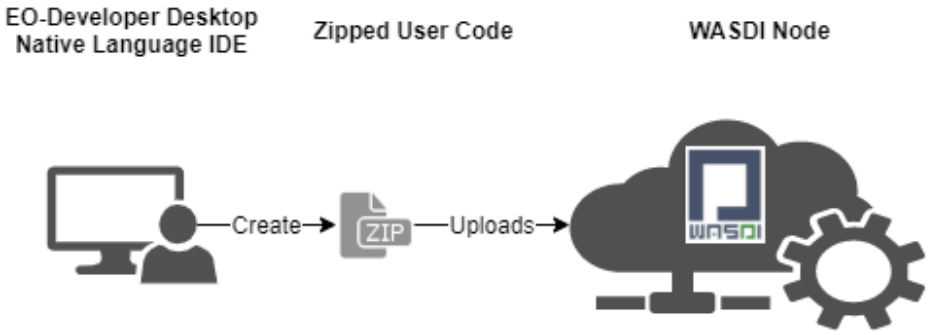

Figure 5 - Processors Drag and Drop

To achieve this result, the following steps are required:

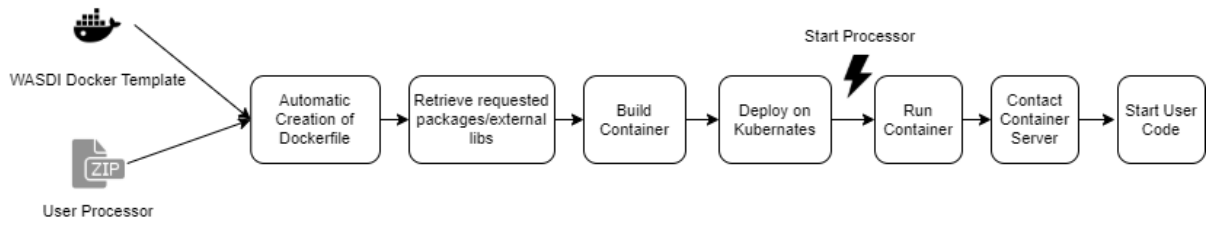

Figure 6 - Automatic deploy

Users are requested only to include the WASDI Libraries to authenticate and to abstract the access to the resources available in any workspace.

Once the code is ready, a zip file with the source is uploaded. This file is extracted and combined with a Docker Template that is hosted on the server for each different type of application. From this template a real Dockerfile is created that includes the user code and all the external packages or libraries required. This container is built on the server and deployed.

Each container hosts its own internal web server that is listening to a specific http internal port. When the process starts, the launcher communicates with the internal server to physically start the user code and to control its execution.

Each WASDI Computing Node can download from the main server the original user application and deploy it locally when it is needed, on demand.

5.4 Libraries

The libraries are used by developers to create WASDI Applications.

From the IT point of view the libraries are a client of the WASDI API written in a specific language. The main service of the lib is very simple: the user can code whatever he wants given two constrains:

- Do not use any user interface, but use a JSON file to get the inputs that can be accessed in the code with the method getParameter
- Do not use any absolute path, but ask the path of files that must be open or created to the lib with the method getPath

Libraries are available for the following languages:

- Python (https://wasdi.readthedocs.io/en/latest/python/waspy.html)
- Octave/Matalb (https://wasdi.readthedocs.io/en/latest/octave/octave.html)
- IDL (https://wasdi.readthedocs.io/en/latest/java/WasdiLib.html)
- C#.Net (https://wasdi.readthedocs.io/en/latest/c%23/WasdiLib.html)
- JS (https://wasdi.readthedocs.io/en/latest/typescript/wasdi.html)

The detailed documentation of each library is maintained online in the addresses indicated, where it is possible to find all the methods, the inputs and the outputs.

The main functionalities of the libraries are the followings:

- Login in WASDI
- Abstract the access to the file system
- Search EO Images
- Get Images in a workspace
- Run Workflows
- Run other WASDI Applications

Almost all the operations can be done synchronously or asynchronously; the libraries also have methods to get the state of other processes and to wait for one or more processes to finish.

5.4 Data Providers

The paradigm to move the processors to the data is well received on WASDI. Indeed, this is not always possible in the real world: it can happen that some processors use different data sources or that some providers have a collection of images but may not cover the full time or space resolution.

The Data Provider Abstraction Layer has the objective to move this logic inside the WASDI Platform, that will be able to automatically decide the best configuration for that specific application in that specific moment. The block diagram of this solution is represented in the following figure:

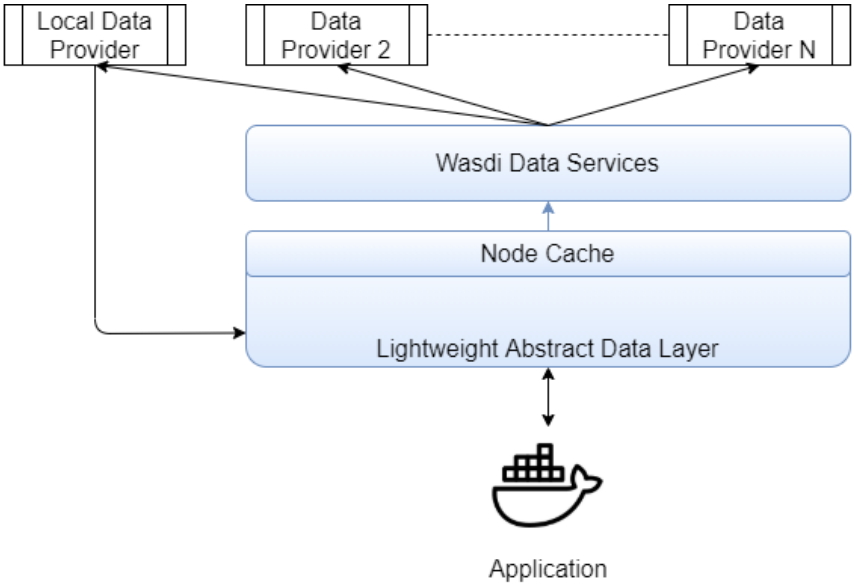

Figure 7 - Data Provider Abstraction Layer

Any application, or user, makes a request for data to the Lightweight Abstract Data Layer, embedded both in the libraries and in the APIs. This layer will first check for the local availability in the node cache. If the data is not present, the request is forwarded to the WASDI services that can search all the data providers, trying to localise "the nearest" to the node where the application is running; once the Provider has been individuated, the data can be fetched and returned to the application.

This mechanism is better described in the following sequence diagram:

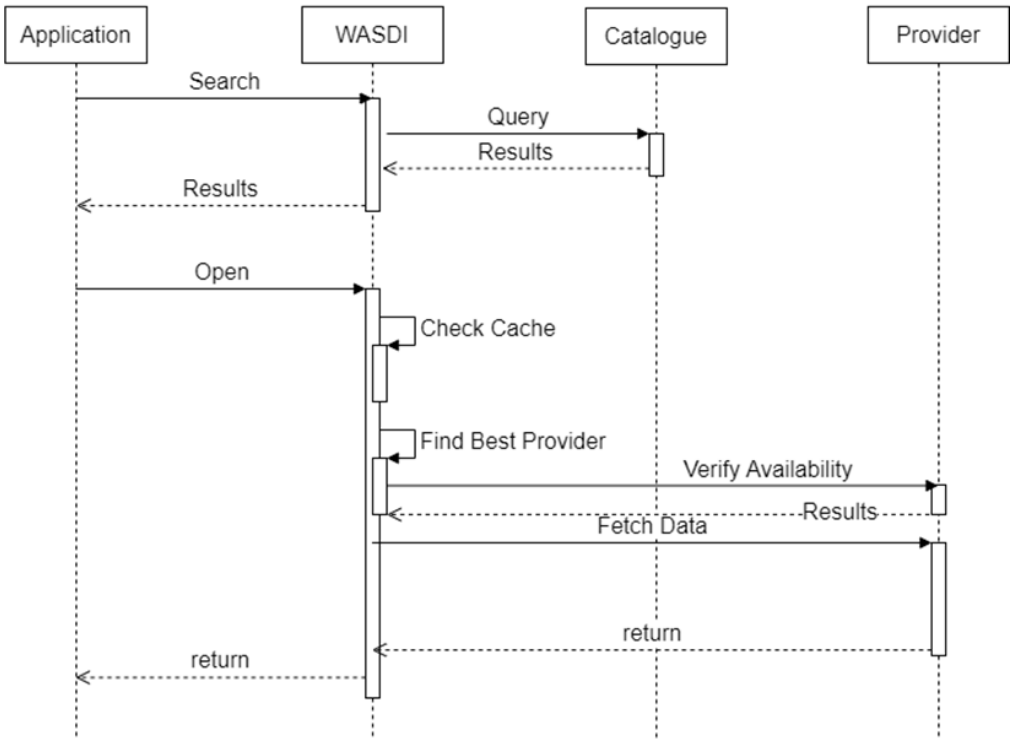

Figure 8 - Data Provider Abstraction Sequence

The WASDI Platforms implements many data provider for Earth Observation data and products but also offers the opportunity to the users to federate a custom Data Provider in order to connect specific data sources to the platform.

To be available to the WASDI platform a custom data provider must expose a specific REST API, with the following interfaces:

- QueryExecutor: used to make a query to a specific provider
- ProviderAdapter: used to fetch (copy, download, ingest..) one instance of data from a specific provider

While the data providers implemented in the WASDI platform are "public", i.e. are available to any registered user of the platform, for custom data providers an access control mechanism is in place, each provider could be:

- Public, like the standard WASDI data providers
- Private, i.e. available only to a single user
- Private, but shared with a limited number of specific users.

6. Digital Ecosystem

The aim of the Digital Ecosystem is to provide an easy-to-use exploitation interface for the research results provided by the scientific community, allowing non-scientific users to explore and use the innovative services and demonstrations developed in the project.

6.1 Digital Ecosystem Architecture

The Digital Ecosystem is built on top of the WASDI platform, that is used as development and execution platform by researchers, leveraging the platform features with easy to use and customized user interfaces.

All the scientific community members involved in the development of services and/or demonstrators/case studies can have access to the WASDI platform, where a subscription dedicated to the project is active. Moreover, a dedicated WASDI node has been deployed in order to host data and services for the project.

Impacts products in the Spoke VS1 can provide different artifacts to be included in the system:

- Data products: depending on the volume of the data and the requested usage, data products can be hosted as collections of files in specific WASDI or integrated as custom data providers.
- Services: these are usually implemented as WASDI processors.
- Demonstrators/Case studies: depending on the type of the results could be a set of data files hosted in workspaces od processors working on a limited set of data files.

The connection between the WASDI Platform and the Digital Ecosystem is implemented through a service component, that integrates the WASDI libraries and acts as a proxy to orchestrate the execution of services and the data management for the ecosystem.

All the services and custom data providers provided by the researchers are accessed by the Digital Ecosystem through WASDI, while specific features such as basin identification and statistical analysis are provided by an internal dedicated service working on a shared storage.

A shared PostgreSQL database, with PostGIS spatial extensions, is shared between the Digital Ecosystem and the data providers requiring spatial data services.

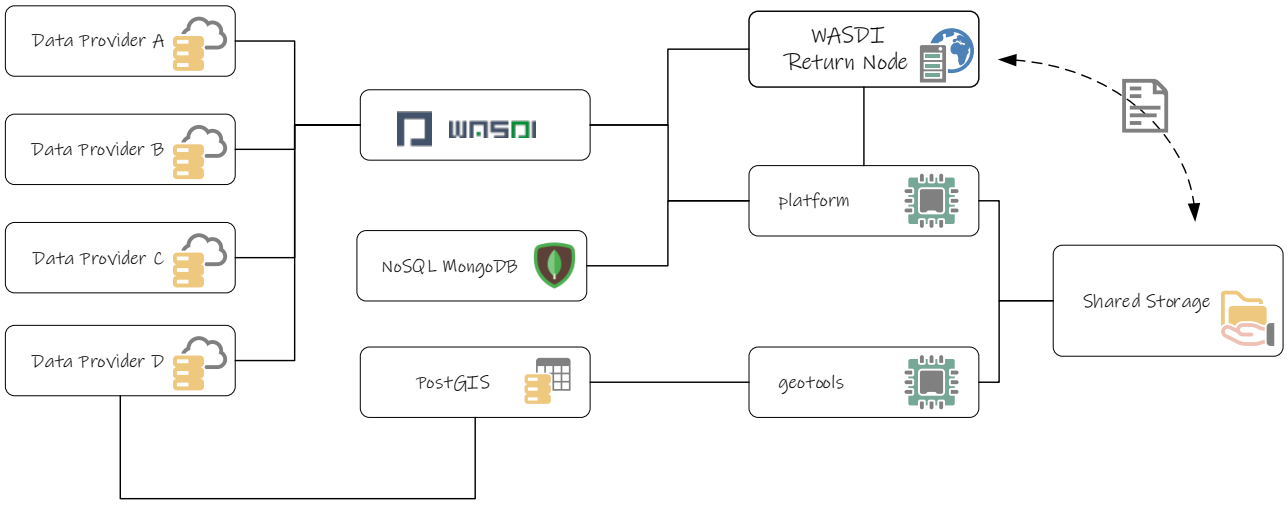

Figure 9 - Digital Ecosystem Architecture

The User Interface is implemented as a Web Application, communicating through a set of REST Api with the backend services, and provides basic WebGIS features allowing user to visualize and analyze spatial data.

All the impact products that are included the ecosystem will have a dedicated UI in order to get access to the resources or the services hosted on the WASDI platform.

6.2 Identity Management

Researchers and scientists working on WASDI have their own WASDI accounts, while the Digital Ecosystem has a light authentication and authorisation mechanism: registered users, once approved by the administrator, have access to all the resources exposed within the Digital Ecosystem, which works towards the WASDI platform with a dedicated system user who has access to all the necessary services and data providers.

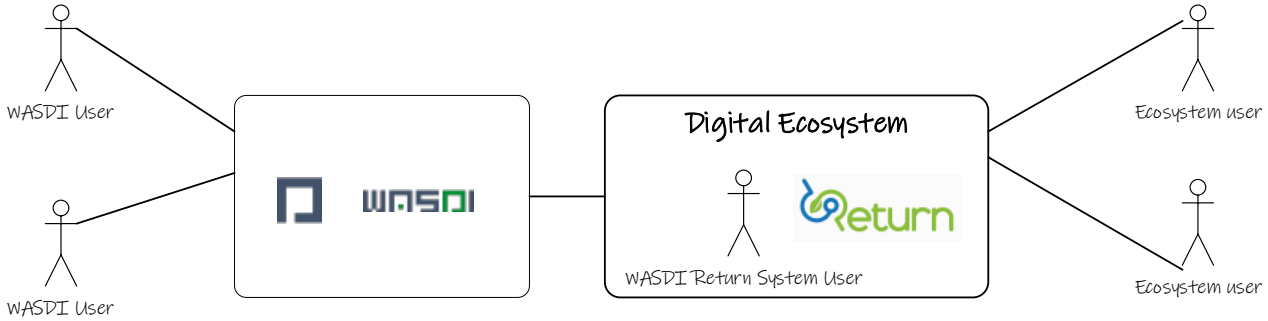

Figure 10 - Identity Management

The features available in the single resource of the Digital Ecosystem, and thus the data and services deployed on the WASDI platform exposed, depends on the design of the resource itself.

6.3 Workspaces

While WASDI users can create and work with multiple workspaces within WASDI, the Digital Ecosystem manages two kinds of working areas for each user:

- a hidden WASDI workspace, owned by the WASDI Return System User, holding the processing results and the downloaded data.
- a personal working area on the Digital Ecosystem in which each user can copy data from the WASDI workspace and perform additional operations provided by the platform.

Digital Ecosystem users haven't direct access to the WASDI workspace, in order to be able to control the resources access through the constraints implements in the Digital Ecosystem resources.

5. Conclusions

The high-level reference architecture for the Digital Twin platform within the RETURN project is designed to support the scientific community via the WASDI execution platform, as well as a broader range of stakeholders via the Digital Ecosystem. By providing distinct user interfaces and leveraging cloud-based technologies, the platform aims to deliver and utilise innovative research results related to multi-risk science for resilient communities in a changing climate effectively. Integrating models, data and services, and designing the WASDI platform robustly and making the Digital Ecosystem accessible ensures that the project's outputs are both scientifically sound and broadly applicable.

6. References

Schumann GJ, Campanella P, Tasso A, Giustarini L, Matgen P, Chini M, Hoffmann L. An online platform for fully-automated eo processing workflows for developers and end-users alike. In 2021 IEEE International Geoscience and Remote Sensing Symposium IGARSS 2021 Jul 11 (pp. 8656-8659). IEEE. https://doi.org/10.1109/IGARSS47720.2021.9554498.

Living Planet Symposium, Bonn 2022 - Democratizing Earth Observation with the WASDI EO analytics platform

Corban, C., Maffenini, L. and Campanella, P., Next Generation Mapping of Human Settlements from Copernicus Sentinel-2 data, EUR 30344 EN, Publications Office of the European Union, Luxembourg, 2020, ISBN 978-92-76-21441-0, https://doi.org/10.2760/54360 ,JRC121560