



SEBASTIEN

## D6.2 - Document describing Portal's architecture

Deliverable Lead	CINECA
Deliverable due date	2023/04/30
Status	FINAL
Version	V2.0
Project	SEBASTIEN



## DOCUMENT INFORMATION

---

Title	D6.2 - Document describing Portal's architecture
Agreement	INEA/CEF/ICT/A2020/2373580
Action	2020-IT-IA-0234
Creator	Cineca
Deliverable Description	Document describing Portal's architecture
Contributors	Giuseppe Trotta (Cineca), Cinzia Caroli (Cineca)
Requested deadline	M16
Reviewer	Alessandro D'Anca (CMCC), Paola Nassisi (CMCC)

# Table of contents

---

1. Executive summary.....	4
2. Data and Services Exploitation Portal.....	5
2.1. Web Portal Architecture.....	5
2.2. Web Portal Architecture Components.....	7
2.2.1. Load Balancer.....	7
2.2.2. Web App Servers.....	8
2.2.3. Databases.....	8
2.2.4. Caching Service.....	8
2.2.5. Job Queue.....	8
2.2.6. Services.....	8
2.2.7. Data Lake.....	9
2.3. Data Lake Connection.....	9
2.4. Technology stack.....	11
3. Services Wireframes.....	13
3.1. Service 1: Coping with environmental stressors for breeds.....	14
3.1.1. Service 1A: Estimation of production loss as a function of climatic variables.....	14
3.1.2. Service 1B: Adaptability of species/breeds to stress due to climate change.....	16
3.2. Service 2: Intensive farming risk management under climate extremes.....	17
3.3. Service 3: Extensive farming management and feed availability.....	18
3.4. Service 4: Livestock farming under risks from combined abiotic and biotic factors.....	19
3.5. Dashboard for sensors on animals.....	20
4. Data Catalog Wireframes.....	21
5. Conclusions.....	23

# 1. Executive summary

The aim of the document is to describe the architecture of the web portal for data and services exploitation and to show the macro components it is made up of as well as the interactions between them. This work is the result of the requirement analysis of the application services coming from a continuous interaction with partners and stakeholders to identify the end-user needs and the way in which the data must be accessed and visualized. The web portal will be created “on top” of the services prediction models and will query and/or retrieve the forecast indicators periodically produced. To provide unified access to datasets and services, the web portal needs tight integration with the SEBASTIEN data lake. For this reason, this document has strong dependencies on what is described in deliverable D3.1 - *“Report on HIGHLANDER data platform extension implementing SEBASTIEN data lake”*.

This document is divided into two parts: a first part which describes the architecture up to the technological solutions to be adopted; the second part will instead deal with a UX/UI web design where it is shown how the portal will have to present the outcomes to the end user.

## 2. Data and Services Exploitation Portal

The Sebastien Data and Services Exploitation Portal aims to be a web portal as the main access point to the data and downstream applications envisaged in the project.

Leveraging the work done in the past years, the SEBASTIEN web portal adapts and integrates the new services in the main components of the HIGHLANDER project architecture.

Figure 1 summarizes the scope of the project and provides a high-level architectural overview. The data coming from different sources and produced by predictive models feed the data lake which has the task of organizing them and providing a first level of access. The actual access level will be implemented through restful APIs that will expose the data and the services functionalities to the portal, the SEBASTIEN mobile app and other downstream applications.. The SEBASTIEN web portal implements the UI components for data navigation and interaction with the various downstream services.

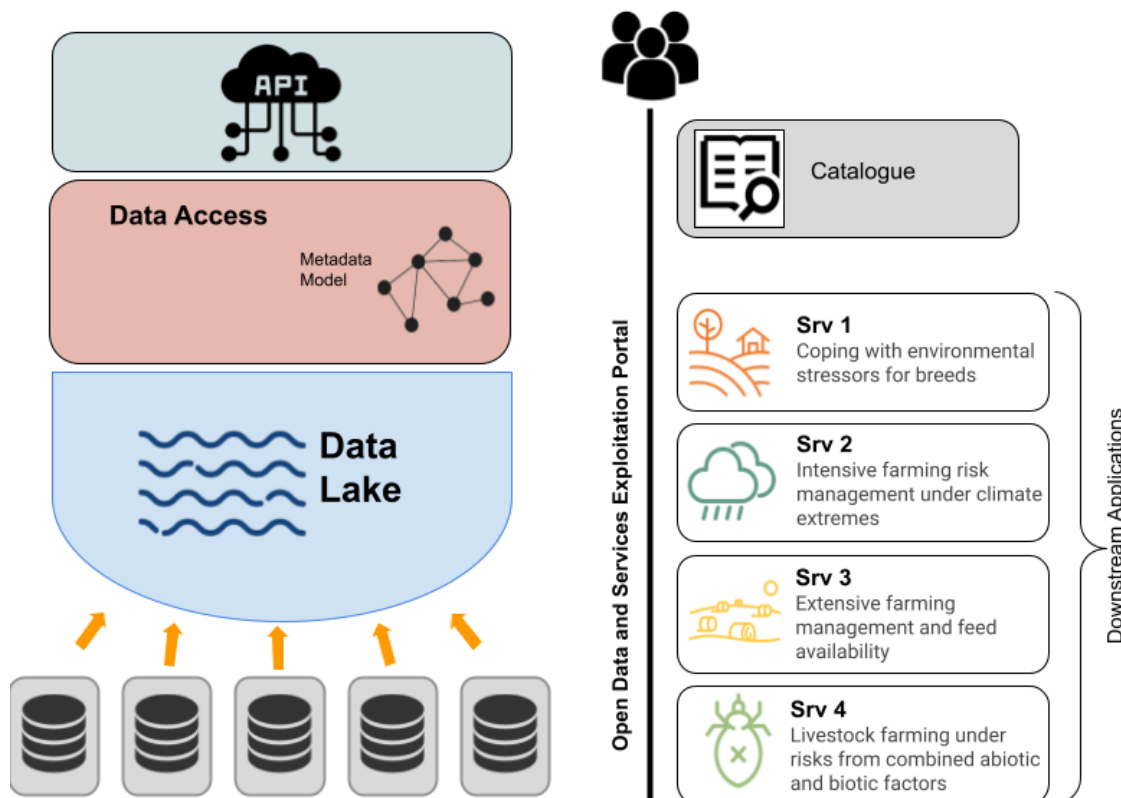


Figure 1 – Data and Service Exploitation Portal Overview

## 2.1. Web Portal Architecture

Before going into the details of the components of the web portal, let's introduce the basic characteristics of a web application and describe its well-established and commonly used architecture. When thinking of a web portal, we are not limited to front-end components that usually run in a client-side browser, but it also involves components that execute business logic typically deployed on a remote server. In SEBASTIEN those logics can be identified in the on-demand calculation of prediction indicators as well as in the user application data management tools such as data access request management, search data, user profile, user stables and features etc.

The web application architecture describes the interactions between application components, middleware systems, user interfaces and databases. Once a user opens a webpage and asks for some data, the server sends back specific data to the browser as a response to the user's request. The following Figure 2 shows such an interaction and identifies the main high-level components.

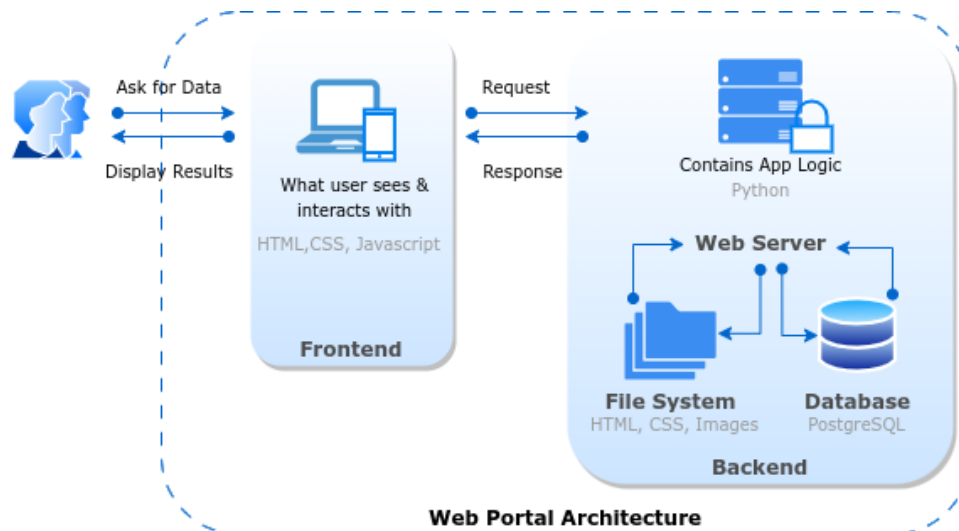


Figure 2 – Web Portal Architecture

Typically, there are two different codes running side-by-side:

- **Client-side Code** - The code that is in the browser and responds to some user input
- **Server-side Code** - The code that is on the server and responds to the HTTP requests

The client components will be developed using traditional web technologies such as HTML and CSS (more details will be presented later in the document). The client component is a representation of a web application functionality that the end-user interacts with. Server components instead implement the actual business logic that deals with data or asset contents stored in a data system such as a relational database or a file system.

## 2.2. Web Portal Architecture Components

After presenting an overview of a typical web application, a scheme of the specific SEBASTIEN user-server process is shown in Figure 3.

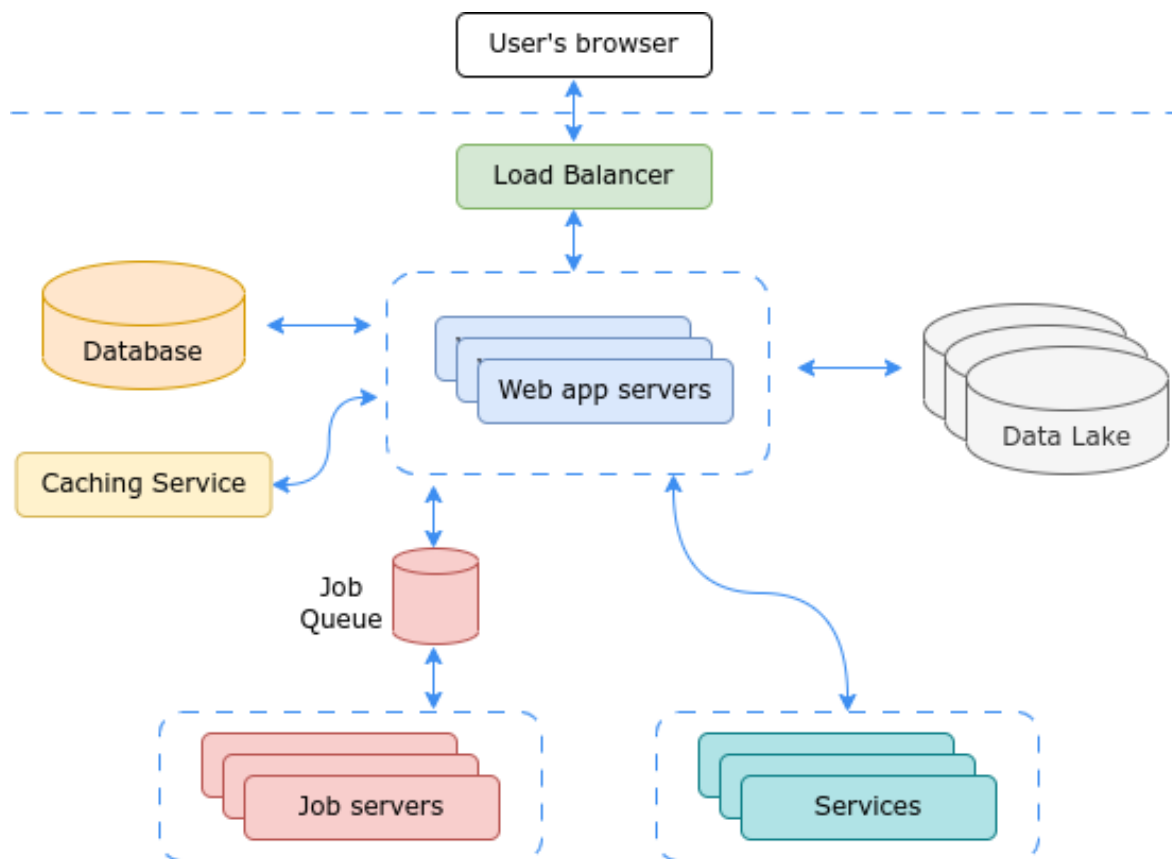


Figure 3 – User-Server Process

The scope of each single component is explained in the following sub-paragraphs.

### 2.2.1. Load Balancer

Load Balancer primarily deals with horizontal scaling. By directing incoming requests to one of the multiple servers, the load balancer sends an answer to a user. Usually, web application servers exist in the form of multiple copies mirroring each other. Hence, any server processes requests in the same manner, and the load balancer distributes tasks among them so they will not be overcharged.



### **2.2.2. Web App Servers**

This component processes a user's request and sends data (e.g., JSON) back to a browser. To perform this task, it usually refers to back-end infrastructures such as database, cache server, job queue, and others. Besides, at least two servers, connected to the load balancer, manage to process the user's requests.

### **2.2.3. Databases**

The database provides the tools to organize, add, search, update, delete data. In some other cases, web application servers directly interact with the job servers.

### **2.2.4. Caching Service**

Caching service provides storage for data, which allows storing and searching data. Whenever a user gets some information from the server, the results of this operation go to cache. So, future requests return faster. In one word, caching allows you to refer to the previous result to make a computation much faster.

### **2.2.5. Job Queue**

A job queue consists of two components: the job queue itself and servers. These servers process jobs in the queue. It happens that most of the web servers need to operate a vast number of jobs that are not of primary importance. Therefore, when a job needs to be fulfilled, it goes to the job queue and is operated due to a schedule.

### **2.2.6. Services**

When a web application reaches a certain level of complexity, services are created in the form of separate apps. They are not visible among other web application components, but the web application and other services interact with them.

### **2.2.7. Data Lake**

Large amounts of data cannot always be stored locally. Very often the data comes from different sources, and it is not appropriate to transfer them all because they would occupy large volumes. The data lake provides transparent access to a multitude of heterogeneous data and allows local transfer only if required.

## 2.3. Data Lake Connection

In the design presented so far it is now clear that we need a middleware between the client applications and the data repository. The goal is to expose the functionality of different services "on top" of those provided directly by the data lake. The APIs will be designed in such a way as to be easily extensible to allow easy integration of new services and allow access to new future datasets. Figure 4 below shows this integration.

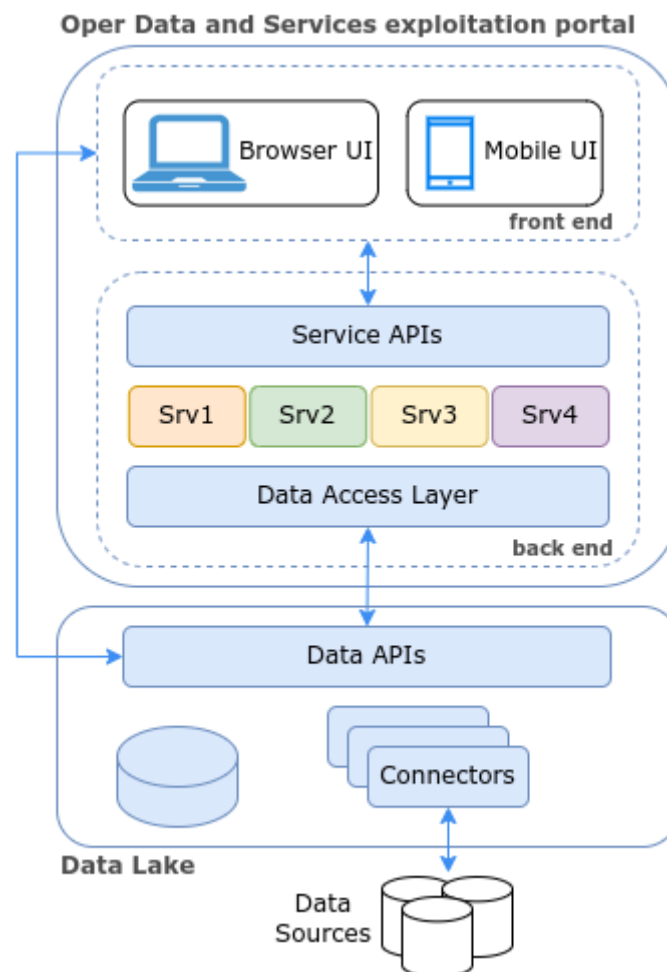


Figure 4 – Data Lake integration

Furthermore, direct front-end access with the data lake APIs could be possible or cannot be excluded. It could be necessary if we want to expose the data also through OGC<sup>1</sup> compliant services such as the web map service (WMS) or web feature service (WFS). In this case the application will access the geographical maps pointing directly to the data lake. The same could

<sup>1</sup> <https://www.ogc.org/>

also happen for the realization of the catalog view of the project datasets with the possibility of searching and filtering.

For the sake of clarity, at the time of writing this document, a first prototype of the connection between the data lake and the portal back-end has already been created: it implements the needed data exchange mechanism and creates a first version of the Service 2 "Intensive farming risk management under climate extremes". The following figure shows how the UI component interacts with the Service API to evaluate the THI of a specific geographical location selected by the user on a map, and how the data lake provides the data in JSON format then presented to the user in the form of a table.

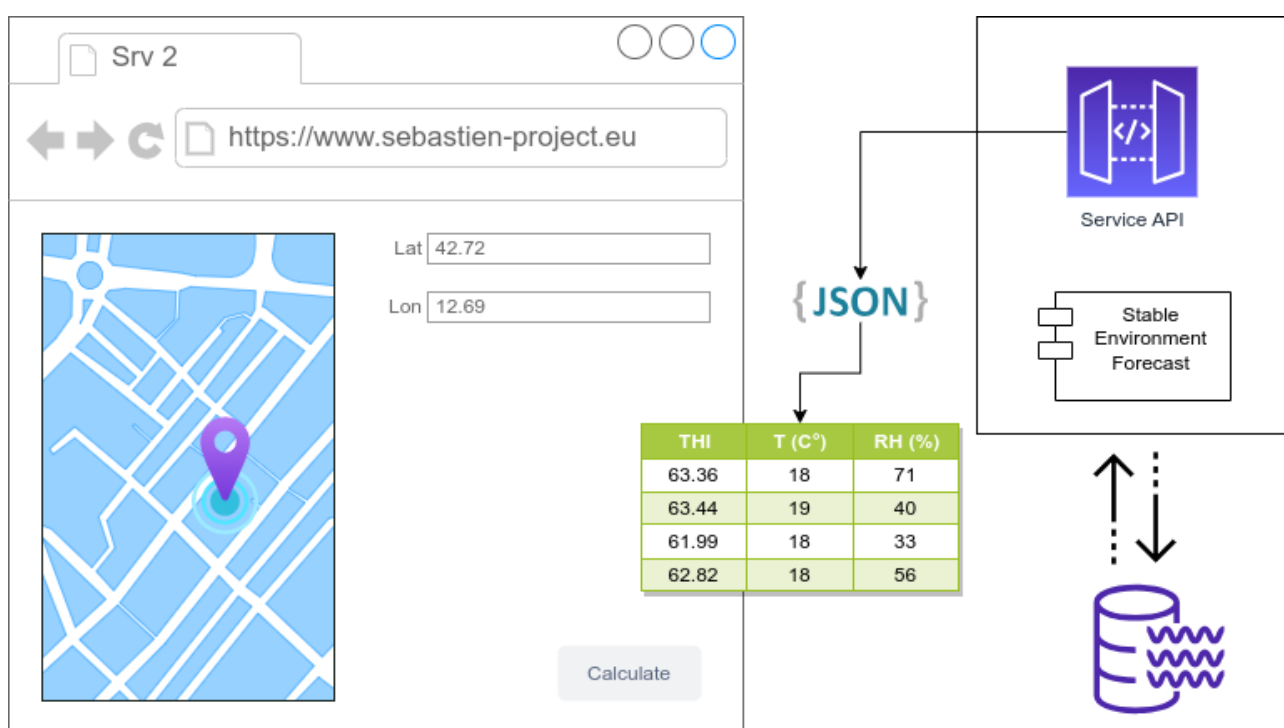


Figure 5 – Interaction between front-end and back-end to implement the Sebastien Service 2

## 2.4. Technology stack

To conclude this part on the web portal architecture we briefly describe the technologies chosen to implement the components shown up to now. These choices are also a consequence of the requirement to reuse what has already been done in the HIGHLANDER project to leverage its results. The solutions are exclusively open source and easy to use, always supported by a large community behind them, thus ensuring a certain stability and robustness. Figure 6 shows the main

technological solutions for each of the macro components previously introduced; each box represents the “container” within which the applications are executed as explained below.

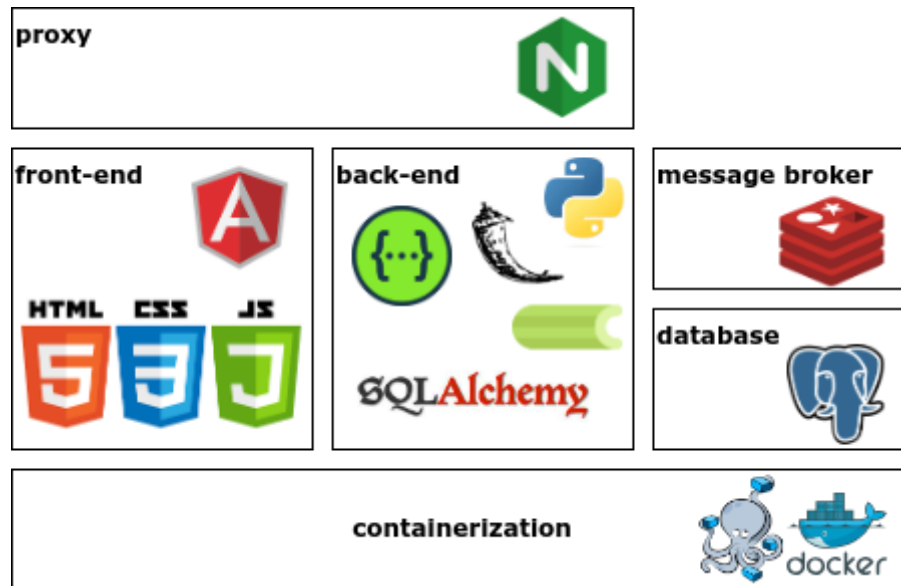


Figure 6 – Technology stack

## Proxy

User requests pass through a **Nginx**<sup>2</sup> proxy server. It sends the request to a specified proxied server, fetches the response, and sends it back to the client. At the moment, the only server behind the scenes is the backend, while the frontend is served as assets and pre-bundled packages. Nginx also implements a secure HTTP over TLS connection and a set of security policies in front of the other services.

## Front-end

Implements the client-side components using technologies for the web. The front-end is built up using **Angular**<sup>3</sup>: a platform and framework for building single-page client applications. Angular components and services are written in Typescript while template files in HTML and styles with **Bootstrap**<sup>4</sup> and SCSS (a CSS syntax extension).

## Back-end

The HTTP APIs, written in **Python**<sup>5</sup> by using the **Flask**<sup>6</sup> framework, are used through the web interface but are also directly accessible by users to implement programmatic access to the data.

<sup>2</sup> <https://www.nginx.com/>

<sup>3</sup> <https://angular.io/>

<sup>4</sup> <https://getbootstrap.com/>

<sup>5</sup> <https://www.python.org/>

<sup>6</sup> <https://flask.palletsprojects.com>



All endpoints are described as OpenAPI specifications by adopting the **Swagger**<sup>7</sup> framework. To implement data-intensive operations the endpoints dispatch the requests to a task queue based on Redis to allow **Celery**<sup>8</sup>, an asynchronous task manager based on distributed message passing, to process the request.

### Message Broker

Message Broker is what makes asynchronous task management possible. **Redis**<sup>9</sup> is a super fast K/V store, making the fetching of the results of a task call very efficient. It works well for rapid transport of small messages.

### Database

Some application data may require to be stored in a **PostgreSQL**<sup>10</sup> database. At the moment the persistent data model is limited and includes models for managing registered users, as well as roles and groups, to support access control functions. The domain model also allows saving user requests for access and data download by tracking information such as request status; start and end data; filtering details and so on. In addition, application services may require the storage of additional data to facilitate their use. In particular, registered users will be able to save and manage their own stables by specifying their position via geographical coordinates. The data model described here is represented in the following diagram:

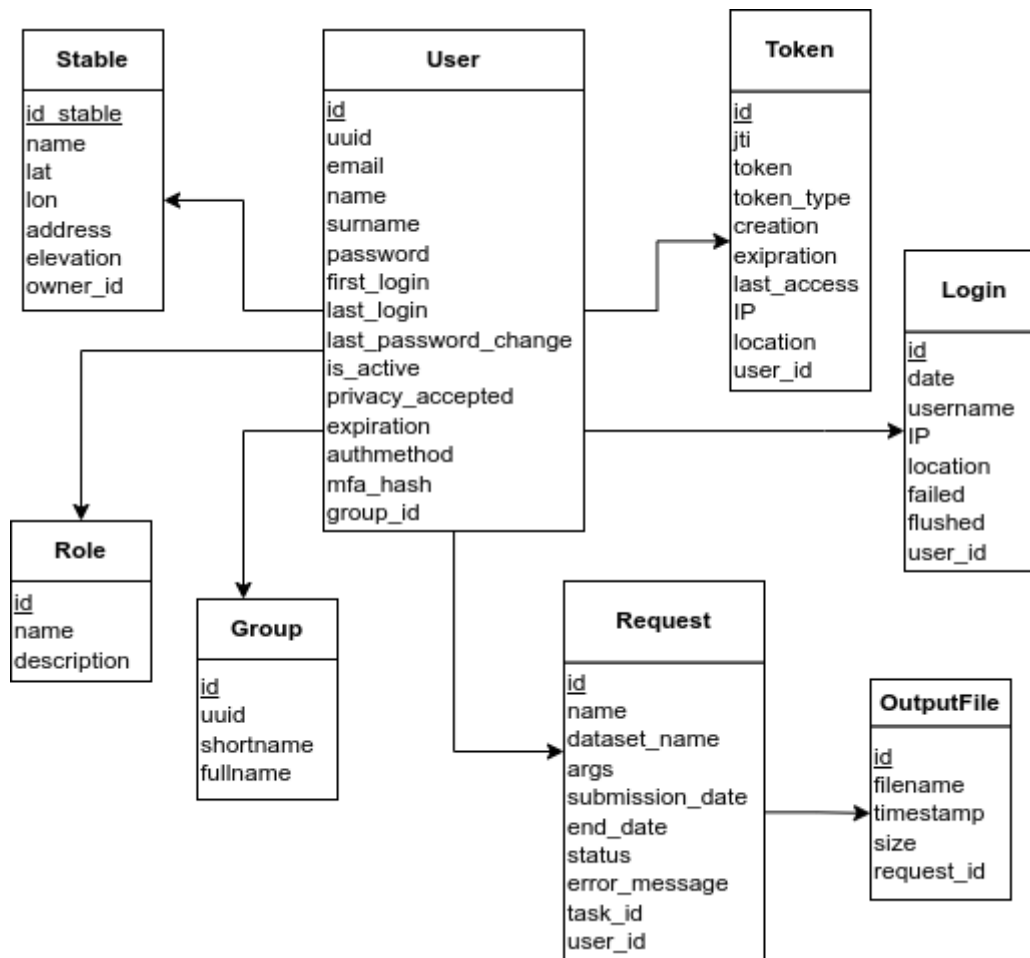
---

<sup>7</sup> <https://swagger.io/>

<sup>8</sup> <https://docs.celeryq.dev>

<sup>9</sup> <https://redis.io/>

<sup>10</sup> <https://www.postgresql.org/>



### Containerization

To make these services more easily manageable and deployable a container-based solution has been adopted. Containerization, using **Docker**<sup>11</sup>, allows to package the application software with all of its dependencies into a standardized unit and creates an environment that is isolated from the rest of the applications and can be run anywhere. This allows us to create the components as building blocks of a larger system and makes services easier to manage and maintain. **Compose** instead allows to define and run services together in the multi-container environment.

## 3. Services Wireframes

The following wireframes describe what the web interface that will allow end users to interact with services on the Sebastien portal should look like. They are the result of UX/UI design which had as

<sup>11</sup> <https://www.docker.com/>

its requirements the output of the analysis conducted with partners and stakeholders to identify the needs of end users. Milestone 4 “Report on data synthesis methodologies” extensively reports, instead, the relationship between the input data stored in the data lake and how the ML models interact with them in order to carry out the needed computations to reach the final output. The results are then represented in the web interface upon user requests.

### 3.1. Service 1: Coping with environmental stressors for breeds

Service 1 has been split into two separate services, for simplicity designated 1A and 1B.

#### 3.1.1. Service 1A: Estimation of production loss as a function of climatic variables

Exploiting climate data, the service will provide information on the risk of production loss in terms of milk quantity, proteins and fat, due to heat stress in dairy cattle.

After choosing the phenotype and the breeding method, users will see the assessment of the short (2 days) and long (until 2050) term risk increase on the map.

By selecting a location on the map, the service will open a new section on the page to show the area around the selected location in higher spatial resolution.

The required input data of the service and its output are summarized in Table 1.

<b>INPUT</b>	<p><i>User:</i></p> <p><b>Phenotype:</b> amount of milk, amount of fat, amount of protein</p> <p><b>Breeding method:</b> in the stable, outdoors</p> <p><b>Period:</b> short term (2 days), long term (2050)</p> <p><i>Data lake:</i> weather forecast and climate data</p>
<b>OUTPUT</b>	Map of the estimation of production loss

Table 1 - Input and output of the Service 1A

Figure 7 shows the wireframe of the web interface for Service 1A on the Sebastien Portal.

SEBASTIEN SERVICES DATASETS DOCUMENTATION SIGN IN

### Stima del calo produttivo in funzione delle variabili climatiche

**FENOTIPO**

Quantità di latte

Grasso

Proteine

**MODALITÀ DI ALLEVAMENTO**

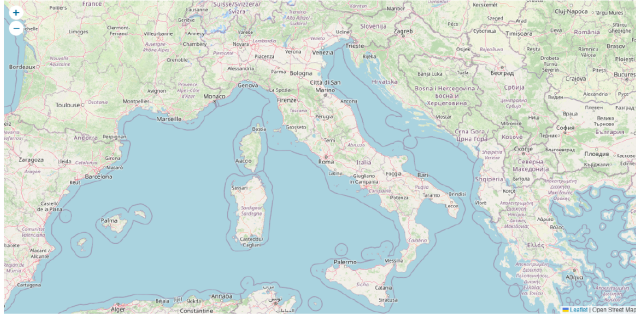
In stalla

**PERIODO**

Breve termine (prossimi 2 giorni)

Lungo termine (fino 2024)

**CALCOLA**



Sebastien DGS 0.1 Co-financed by the Connecting Europe Facility of the European Union

SEBASTIEN SERVICES DATASETS DOCUMENTATION SIGN IN

### Stima del calo produttivo in funzione delle variabili climatiche

**FENOTIPO**

Quantità di latte

**MODALITÀ DI ALLEVAMENTO**

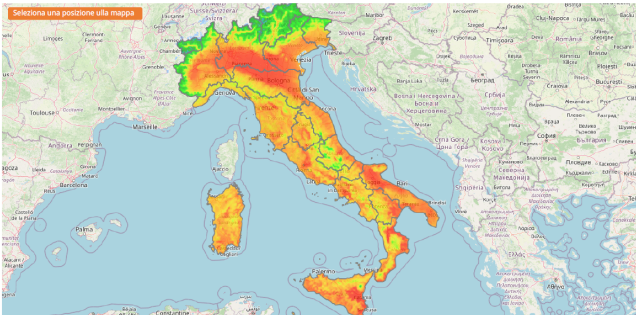
In stalla

**PERIODO**

Breve termine (prossimi 2 giorni)

**CALCOLA**

Seleziona una posizione sulla mappa



Sebastien DGS 0.1 Co-financed by the Connecting Europe Facility of the European Union

SEBASTIEN SERVICES DATASETS DOCUMENTATION SIGN IN

### Stima del calo produttivo in funzione delle variabili climatiche

**FENOTIPO**

Quantità di latte


**MODALITÀ DI ALLEVAMENTO**

In stalla

**PERIODO**

Breve termine (prossimi 2 giorni)

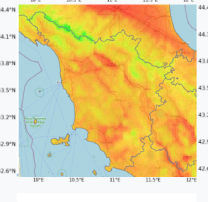
**CALCOLA**



**DETTAGLI**

Quantità di latte - Prossimi due giorni

Località selezionata: Nome località



Sebastien DGS 0.1 Co-financed by the Connecting Europe Facility of the European Union

Figure 7 - Wireframe of the Service 1A



### 3.1.2. Service 1B: Adaptability of species/breeds to stress due to climate change

The service will provide information on the adaptability of species and breeds to stress due to climate change.

The service will open by presenting a map of Italy on which the estimated THI values are shown.

Users will be able to select a different species or THI and consequently the map will update with the corresponding THI values.

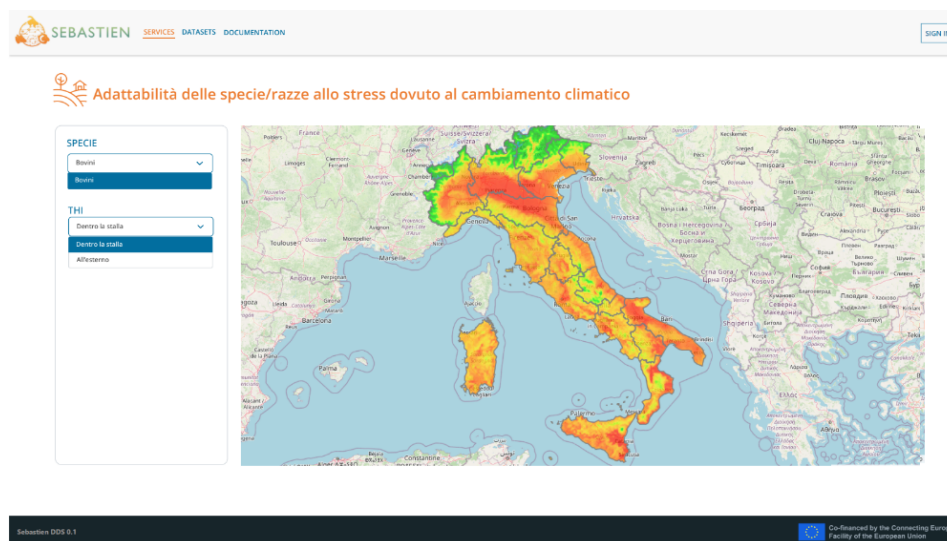
By selecting a location on the map, the service will open a new section on the page to show which breeds are compatible with the THI value in that location and which are not.

The required input data of the service and its output are summarized in Table 2.

<b>INPUT</b>	<p><i>User:</i></p> <p><b>Species:</b> bovine</p> <p><b>THI:</b> in the stable, outdoors</p> <p><b>Location</b> over Italy</p> <p><i>Data lake:</i> climate data</p>
<b>OUTPUT</b>	Breeds compatible with the THI foreseen for the future

Table 2 - Input and output of the Service 1B

Figure 8 shows the wireframe of the web interface for Service 1B on the Sebastien Portal.



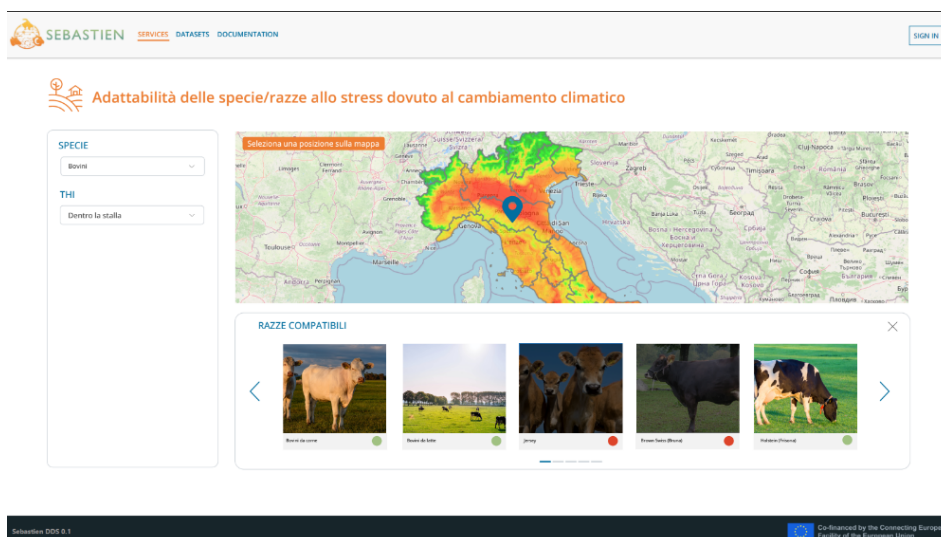


Figure 8 - Wireframe of the Service 1B

### 3.2. Service 2: Intensive farming risk management under climate extremes

The service will provide the forecast of the external temperature and humidity, of the THI (Temperature-Humidity Index) in stables and the associated stress value, upon the provision of the geographical position.

The required input data of the service and its output are summarized in Table 3.

<b>INPUT</b>	<p><i>User:</i> stable position (latitude, longitude)</p> <p><i>Data lake:</i> weather forecasts, climate datasets</p>
<b>OUTPUT</b>	<p>External temperature and humidity forecast, THI prediction inside the stable and related stress value, for the next 2 days at hourly resolution or related to a long term period (2050)</p>

Table 3 - Input and output of the Service 2

Figure 9 shows a screenshot of the web interface for Service 2 which has already been implemented on the Sebastien Portal.

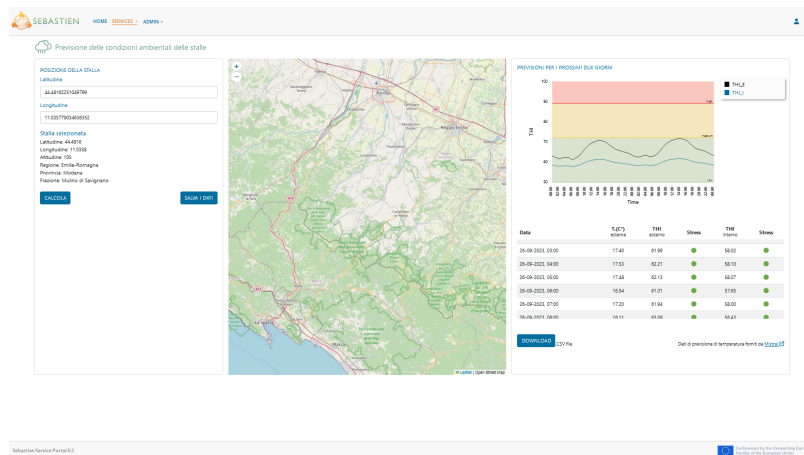


Figure 9 - Screenshot of the web interface for Service 2

### 3.3. Service 3: Extensive farming management and feed availability

Exploiting satellite data, the service will evaluate the zootechnical load of the pasture that the users will delimit using the map widget provided by the web interface.

The required input data of the service and its output are summarized in Table 4.

<b>INPUT</b>	<p><b>User:</b> pasture area</p> <p><b>User:</b> type of animals (bovine, ovine)</p> <p><b>User:</b> number of animals or days (optional)</p> <p><b>Data Lake:</b> satellite data</p>
<b>OUTPUT</b>	<p><b>Zootechnical load</b></p> <p>Evaluate the maximum number of animals per number of days for a pasture</p> <p><b>Alert on pasture availability</b></p> <p>Evaluate the number of days for which pasture is available, based on the zootechnical load</p>

Table 4 - Input and output of the Service 3

Figure 10 shows the wireframe of the web interface for Service 3 on the Sebastien Portal.

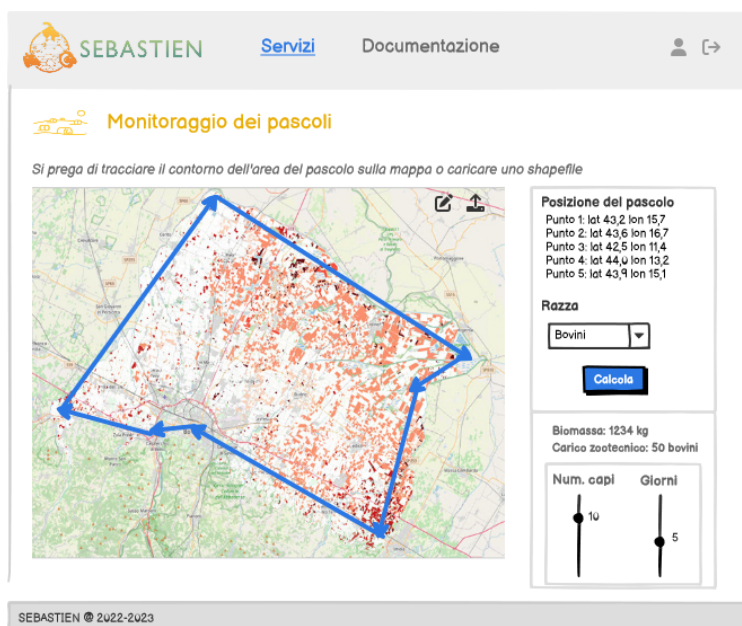


Figure 10 - Wireframe of the Service 3

### 3.4. Service 4: Livestock farming under risks from combined abiotic and biotic factors

Long term (up to 2050) risk maps of the spread of diseases will be produced on the basis of epidemiological, ecological and climatic data on some regions of Italy and, for certain diseases, on the whole Italian territory.

The required input data of the service and its output are summarized in Table 5.

<b>INPUT</b>	<p><b>User:</b> species (bovine, ovine), disease (mastitis, blue tongue)</p> <p><b>Data lake:</b> climate data</p>
<b>OUTPUT</b>	<p>Long term risk maps of the spread of diseases</p>

Table 5 - Input and output of the Service 4

Figure 11 shows the wireframe of the web interface for Service 4 on the Sebastien Portal.

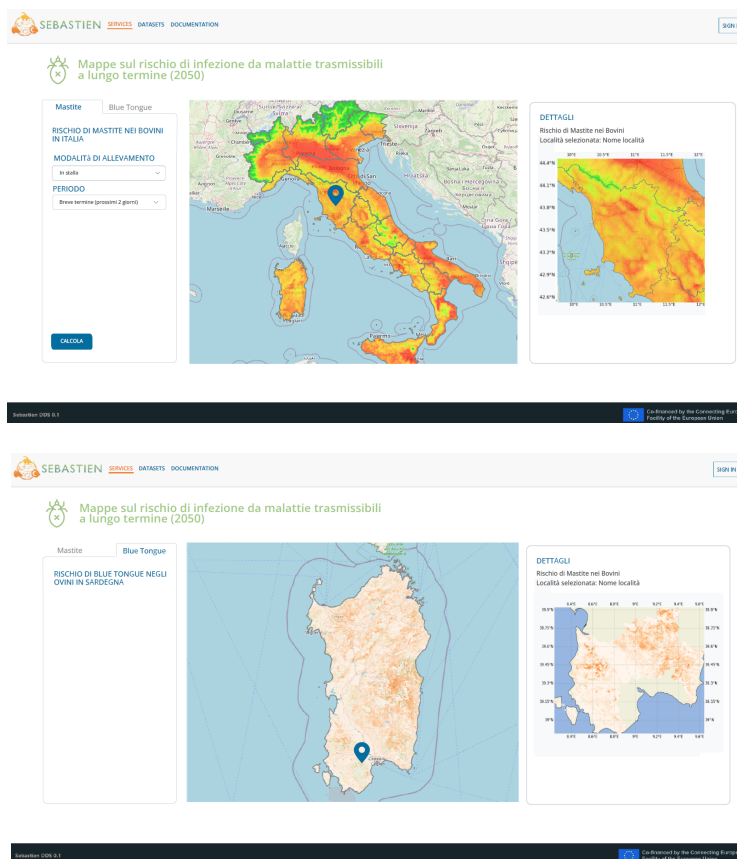


Figure 11 - Wireframe of the Service 4

### 3.5. Dashboard for sensors on animals

As part of the Sebastien project, Nature4.0 developed animal sensors (AnimalTalker) that are being used to collect physiological parameters in real-time related to a group of animals, in barns but also in pasture. Animal sensors can collect the information needed for assessing the animal welfare and the physiological stress and make them available remotely.

A dashboard, developed in the Sebastien Portal, allows the user to identify on a map the last position of each IoT animal sensor owned, to select one of them and to display some information and statistics related to the last 24 hours (i.e. animal temperature, environmental temperature and humidity, animal movements, heart rate, etc.).

Figure 11 shows the wireframe of the web interface for the dashboard with animal sensors data on the Sebastien Portal.

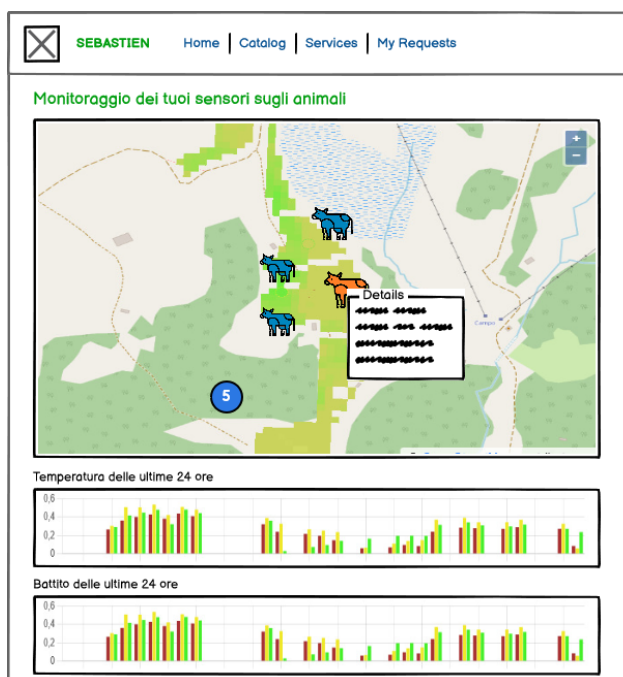


Figure 11 - Animal sensor dashboard wireframe

## 4. Data Catalog Wireframes

In addition to the services presented in the previous paragraphs, the Sebastien portal provides access to a catalog of data and indicators. Users are able to perform extraction queries on these datasets and download the resulting data to their local computers.

The following wireframes describe the web interface that allows end users to access the catalog on the Sebastien portal.

First, there will be a page with the list of all datasets available in the catalog, with a brief description for each dataset (Figure 12).



Figure 12 - List of the datasets of the catalog

From the list of datasets, clicking on the title of a dataset, users is taken to the page with the details of the chosen dataset (Figure 13).

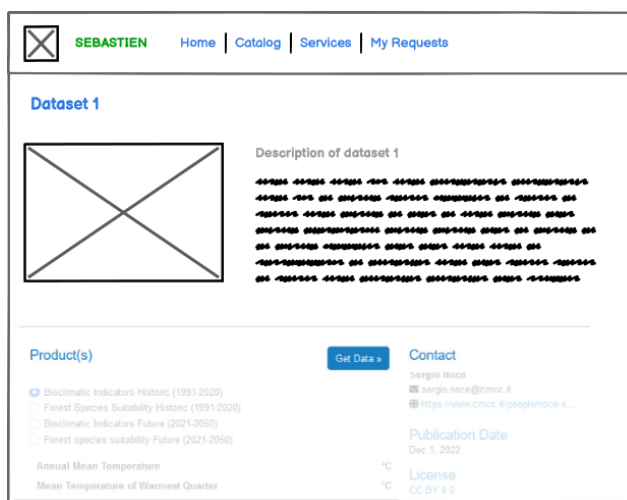
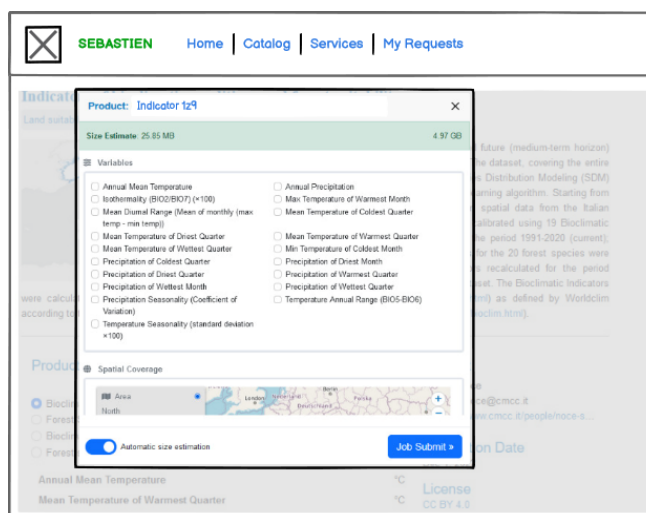


Figure 13 - Detail page of a dataset

From the dataset detail page, clicking the "Get Data" button a dialog window will open from which the user can build a query and then submit it to the system (Figure 14).



SEBASTIEN Home | Catalog | Services | My Requests

Product: Indicator tz9

Size Estimate: 25.85 MB 4.97 GB

Variables

- Annual Mean Temperature
- Isothermality (BIO/BIO7)(+100)
- Mean Diurnal Range (Mean of monthly (max temp - min temp))
- Mean Temperature of Driest Quarter
- Mean Temperature of Wettest Quarter
- Precipitation of Coldest Quarter
- Precipitation of Driest Month
- Precipitation of Wettest Month
- Precipitation Seasonality (Coefficient of Variation)
- Temperature Seasonality (standard deviation +100)
- Annual Precipitation
- Max Temperature of Warmest Month
- Mean Temperature of Coldest Quarter
- Mean Temperature of Warmest Quarter
- Min Temperature of Coldest Month
- Precipitation of Driest Quarter
- Precipitation of Warmest Quarter
- Temperature Annual Range (BIO5-BIO6)

Spatial Coverage

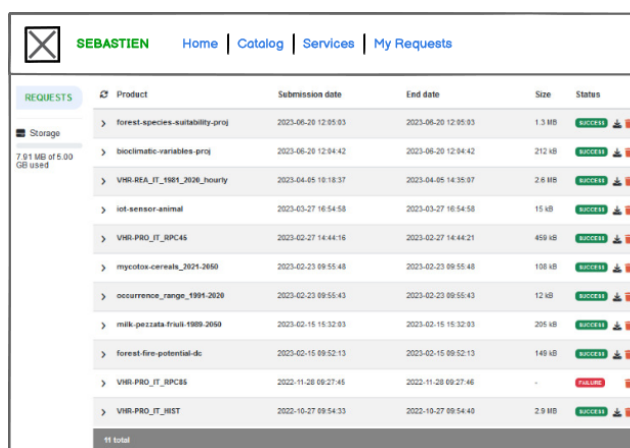
Area

Automatic size estimation

Job Submit

Figure 14 - Form to create the query

The result of the query is available for download by users in the "My requests" feature. On the "My requests" page, users can view the queries they submitted to the system and download the results (Figure 15).



Product	Submission date	End date	Size	Status
forest-species-suitability-proj	2023-06-20 12:05:03	2023-06-20 12:05:03	1.3 MB	Success
bioclimatic-variables-proj	2023-06-20 12:04:42	2023-06-20 12:04:42	212 kB	Success
VHR-REA_FT_1981_2026_hourly	2023-04-05 10:18:37	2023-04-05 14:35:07	2.6 MB	Success
iot-sensor-animal	2023-03-27 16:54:58	2023-03-27 16:54:58	15 kB	Success
VHR-FRO_FT_RPC45	2023-02-27 14:44:16	2023-02-27 14:44:21	459 kB	Success
mycolso-cereals_1921-2050	2023-02-23 09:55:48	2023-02-23 09:55:48	108 kB	Success
occurrence_range_1991-2020	2023-02-23 09:55:43	2023-02-23 09:55:43	12 kB	Success
milk-pezzata-fruit-1989-2050	2023-02-15 15:32:03	2023-02-15 15:32:03	205 kB	Success
forest-fire-potential-dc	2023-02-15 09:52:13	2023-02-15 09:52:13	149 kB	Success
VHR-FRO_FT_RPC36	2022-11-28 09:27:46	2022-11-28 09:27:46	-	Failure
VHR-FRO_FT_WEST	2022-10-27 09:54:33	2022-10-27 09:54:40	2.9 MB	Success

Figure 15 - List of the queries submitted by the user

## 5. Conclusions

In this document, the architectural design of the data and services exploitation portal has been provided, identifying the fundamental components that will be implemented or integrated. In particular, the connectivity with the data lake has been shown. The solutions described are inherited from the HIGHLANDERhlander platform as expected with the aim of reusing what has already been done.

A first version of the SEBASTIEN Open Data and Services Exploitation portal has already been released and is available at the following link:



<https://dds.sebastien-project.eu>

However, since the interaction with the stakeholders will continue until the end of the Action, an agile approach will be taken with a progressive refinement of what has been designed and described here.