



SEBASTIEN

## M6 – Report on Data Access Layer

Milestone Lead	Cineca
Milestone due date	2023/12/31
Status	FINAL
Version	V1.0
Project	SEBASTIEN



## DOCUMENT INFORMATION

---

Title	M6 Report on Data Access Layer
Agreement	INEA/CEF/ICT/A2020/2373580
Action	2020-IT-IA-0234
Creator	Giuseppe Trotta (Cineca), Marco Mancini (CMCC)
Milestone Description	Report on the released Data access layer
Means of verification	Report on the released Data access layer is shared with the Agency
Contributors	Giuseppe Trotta (Cineca), Marco Mancini (CMCC), Cinzia Caroli (CINECA), Valentina Scardigno (CMCC)
Requested deadline	M24
Reviewer	Paola Nassisi (CMCC), Alessandro D'Anca (CMCC)

## Table of content

1. Introduction.....	4
2. Data Access Layer.....	4
3. Requirements for downstream applications.....	5
3.1. Service 1: Coping with environmental stressors for breeds.....	5
3.1.1. Service 1.a: Estimation of production loss as a function of climatic variables.....	6
3.1.2. Service 1.b: Adaptability of species/breeds to stress due to climate change.....	7
3.2. Service 2: Intensive farming risk management under climate extremes.....	7
3.2.1. Service 2.a: Prediction of the environmental conditions of the stables.....	8
3.2.2. Service 2.b: Projection of environmental conditions of stables in the long term.....	8
3.3. Service 3: Extensive farming management and feed availability.....	9
3.4. Service 4: Livestock farming under risks from combined abiotic and biotic factors.....	10
3.5. Dashboard for sensors on animals.....	11
3.6. UI Data Portal.....	12
4. Data Lake APIs.....	14
5. References.....	22

# 1. Introduction

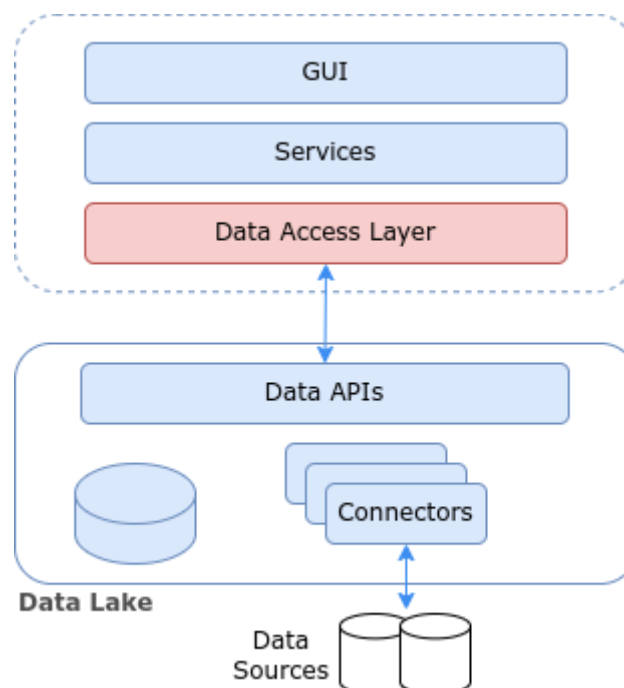
This milestone reports and shares with the Agency the methods of accessing data provided by the SEBASTIEN Data Access Layer that supports the decision system and the downstream applications foreseen by the project.

Datasets coming from different sources (short term forecasts of climate variability, climate projection data, IoT data and satellite observations) and new datasets of indicators/indices (produced by analytical processing procedures and ML/AI approaches) are integrated, harmonised and stored inside the SEBASTIEN data lake.

The Data Access Layer is the common interface that provides a uniform access to the datasets stored in the data lake, decoupling the analytics level and the indicators/indices computation from the access level. It relies on a set of drivers/adaptors available in the data lake, which enables the access to the different data sources and formats.

# 2. Data Access Layer

A Data Access Layer is a software abstraction that provides a consistent and efficient way to access data in storage systems. The primary purpose of a Data Access Layer is to abstract the underlying data storage details from the rest of the application, enabling service and application components to interact with the data using a simplified and uniform interface.



In order to create services for the end user, analyses were carried out supported by stakeholders to identify the data of interest and the most suitable representation. From the user experience design (UX), the requirements for the development of the data access layer and the data lake APIs emerged in terms of i) access to the dataset catalogue, ii) methods of access to the different results of the predictive models, iii) formats and methods of filtering and post-processing.

In the following sections, at first the different services will be described and for each of them the user requirements for data access and visualisation. Finally, a description of the developed API endpoints will be provided.

### 3. Requirements for downstream applications

The SEBASTIEN project involves the implementation of four services (downstream applications) available on the Open Data and Services Exploitation Portal:

**Service 1:** Coping with environmental stressors for breeds to support livestock farming towards breed adaptation to environmental conditions and production needs.

**Service 2:** Intensive farming risk management under climate extremes to alert about approaching or projected dangerous environmental circumstances for cattle.

**Service 3:** Extensive farming management and feed availability based on indicators/indices about the phenological stage and greening of the naturally vegetated or managed areas used to feed livestock heads when conducted outdoors.

**Service 4:** Livestock farming under risks from combined abiotic and biotic factors to provide updated risk maps of parasites and diseases spread.

In addition to the four services originally envisaged, the portal also features a **dashboard for sensors on animals**.

All these applications access data stored in Sebastien's data lake via the Data Access Layer.

In the following paragraphs, the functioning of the services and which datasets they are based on are briefly illustrated.

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

The general scope of this service is to support livestock farms to face climate change effects, in a short and long term frame. The service has been divided into two sub-services, with the aim of responding to the needs of stakeholders, designated 1.a and 1.b..

### 3.1.1. Service 1.a: 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.

The production loss risk indices are shown on the map of the Italian territory using a colour palette to represent the trend of the index in the various locations as can be seen from the mockup in Figure 1.



Figure 1 - Mockup of service 1.a

Users can choose to see on the map risk indices inside the stable or outdoors by selecting the parameter with the selection tools provided by the interface. Furthermore, risk estimates are provided over two different periods: in the short term (2 days) or long term (projection until 2050).

#### Requirements for the data lake

For this type of visualisation the data is represented as layers on a geographic map. This means that the data must be georeferenced with a projection system compatible with the visualisation technology. The preferable format here is raster which allows for easy loading and, with different resolutions possibly available, greater accuracy at different depth levels.

This service therefore requires a layer for phenotype and breeding condition (indoor or outdoor) as well as a time reference to indicate the short-term last prediction or that of a specific long-term period (2021-2050 median value).

### 3.1.2. Service 1.b: 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. Based on the THI values, the service allows users to obtain an estimate of the greater or lesser compatibility of certain cattle breeds on Italian territory.

As shown in the mockup in Figure 2, the user selects a location on Italian territory by clicking on the map and obtains the compatibility estimate of some cattle breeds in that specific location.



Figure 2 - Mockup of service 1.b

#### Requirements for the data lake

This service also requires a layered display on a geographic map. Therefore, the requirement of the previous service also applies in this case and as we will see in the others. Here the filter option depends on the species and breeding method while the data represents the estimate of the risk on Italian territory.

Furthermore, the user is allowed to click on a point on the map that represents a geographical position of application of the model, to load the "punctual" stress value so as to be able to compare it with a reference table of the breeds of a certain species and present their adaptability to local temperatures.

In addition to a raster image of the data, georeferenced numerical data will therefore be necessary, in GeoJSON format.

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

The aim of this service is to predict the approaching (on the short-term) or projections (on the long-term) of dangerous environmental circumstances for cattle in the farms, due to indoor but

also outdoor conditions, causing discomfort and loss of reproduction and/or production. The service has been divided into two sub-services, with the aim of responding to the needs of stakeholders, designated 2.a and 2.b.

### 3.2.1. Service 2.a: Prediction of the environmental conditions of the stables

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.

As shown in the mockup in Figure 3, the user selects a location by clicking on the map of Italy, then using the "Calcola" button asks the Sebastien system the forecast of the temperature, THI and stress inside and outside the stable in the selected location. The result is presented in a graph and in a table and the data can also be downloaded in CSV format.

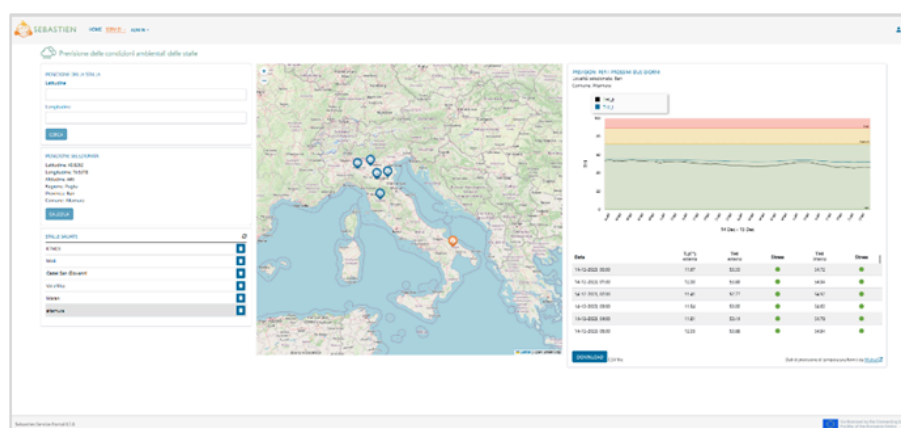


Figure 3 - Mockup of service 2.a

### Requirements for the data lake

To implement this service, the data lake must expose an API which has as input parameters the latitude and longitude coordinates of a location in Italy and the date, typically the current date time for the short-term prediction, and returns as output the prediction of the THI inside and outside the barn (and related stress values) in the input location. As a “punctual” request here we expect a set of values as predictions for the next 48 hours bound to a specific geographic location as output in GeoJSON format.

### 3.2.2. Service 2.b: Projection of environmental conditions of stables in the long term

Exploiting climate data, the service provides information about environmental conditions inside the stables in the long term, in terms of the following three indices:

- Internal THI variation
- Increase in days of stress



- Increase in high stress days

The information about environmental conditions inside the stables are shown on the map of the Italian territory using a colour palette to represent the trend of the selected index in the various locations.

Users can choose to see on the map the selected index in two different RPC scenarios (“4.5 - Reduced greenhouse gas emissions”, “8.5 - No emissions reduction”) and in four different reference periods (2030, 2040, 2050, 2060) by selecting the parameter with the selection tools provided by the interface.

### Requirements for the data lake

This service also requires a layered display on a geographic map. Therefore, the requirement of the previous service also applies in this case and as we will see in the others. Here the filter option depends on the variable, RPC scenario and reference period while the data represents the estimate of the risk on Italian territory.

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

By exploiting satellite data, the service will evaluate the available biomass and consequently the zootechnical load<sup>1</sup> of pasture areas chosen by users. On the Sebastien portal, a web interface allows users to identify the pasture areas of interest, view them on the map of the Italian territory and request the evaluation of the relative biomass currently available.

Figure 4 shows the mockup of the web interface of the service in the Sebastien portal.

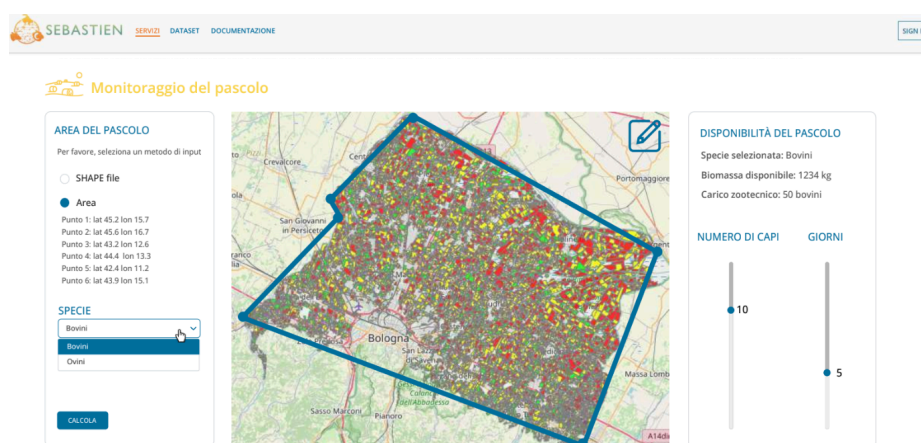


Figure 4 - Mockup of service 3

<sup>1</sup> Maximum number of animals per number of days for a pasture.

## Requirements for the data lake

This service requires the data lake to expose an API which has the species (e.g. bovine, ovine) and the pasture area (with geographical coordinates) as input parameters and returns the available biomass and consequently the zotechnical load of that area. The user should be able to draw a generic polygon on the area or at least a bounding box so that the data lake can apply the prediction model to the area only. This may require a long wait time and will therefore need to be handled asynchronously. A user could then register his pasture in the system and obtain the biomass calculation for his area in advance. This involves the activation of a processing schedule that will provide automatically updated data as soon as it is available. The expected output is a GeoJSON with the biomass prediction associated to the specified area.

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

Risk maps of the spread of diseases in the short term (next 2 days) or long term (projections up to 2050) are produced on the basis of epidemiological, ecological and climatic data.

Currently data for mastitis and blue tongue are available: for the mastitis on the whole Italian territory (short and long term) while for the blue tongue on the Sardegna region (only long term).

Figure 5 shows the mockup of the web interface of the service in the Sebastien portal.



Figure 5 - Mockup of service 4

## Requirements for the data lake

Similarly to service 1, the data visualisation involves applying a layer on a geographic map. This data represents the risk of the spread of disease in the Italian territory.

Currently two diseases are considered: mastitis for cattle and blue tongue for sheep.

In the case of mastitis, the filter option depends on the breeding method (outdoors or in stables) and on the temporal period (short or long term) while the data represents the estimate of the risk across the entire Italian territory.

In the case of blue tongue, the only option available is outdoors, long term and on the Sardegna region territory.

Furthermore, the user can click on a point on the map to obtain a further image with greater detail around the chosen point.

The format is a raster image covering the entire Italian territory or a limited area with higher resolution requested by clicking on the map.

### **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 the barns but also on 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 6 shows the mockup of the web interface for the dashboard with animal sensors data on the Sebastien Portal.

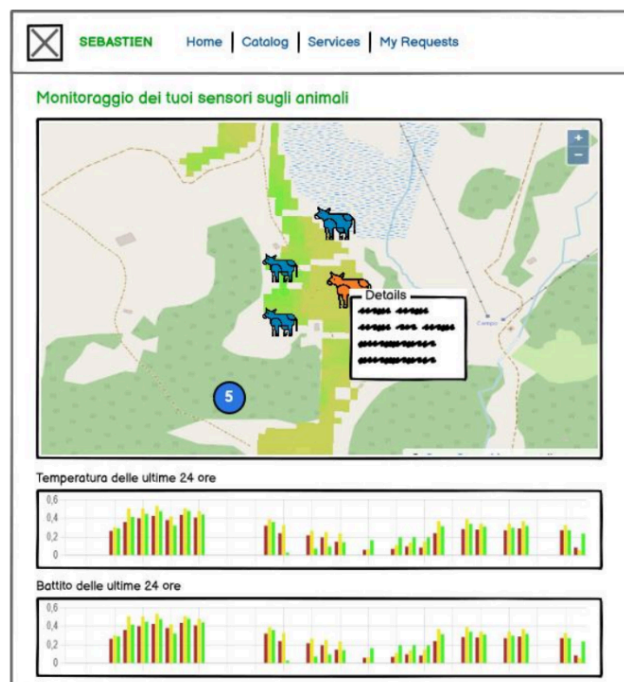


Figure 6 - Mockup of dashboard of animal sensor data

### Requirements for the data lake

This service requires the data lake to expose an API that returns:

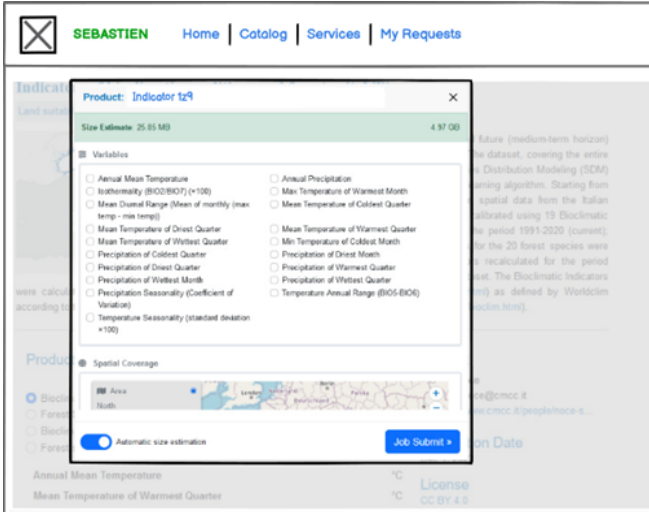
- the geographical position of each IoT animal sensor at a given time
- data collected by the selected sensor during a given period of time for the following variables:
  - Variation in animal movement, based on 3-axes (Y, X and Z) standard deviation (sd)
  - Air temperature (°C), air relative humidity (%) and THI (Temperature Humidity Index)
  - Animal Heart rate (bpm)
  - Animal skin temperature (°C)
  - Battery voltage (mV)

### 3.6. UI Data Portal

The Sebastien portal provides access to a catalogue of data and indicators. Users can perform extraction queries on these datasets and download the resulting data to their local computers.

First, the user is presented with the page with the list of all the datasets available in the catalogue, with a brief description for each dataset. From here, clicking on the title of a dataset, users are taken to the page with the details of the chosen dataset.

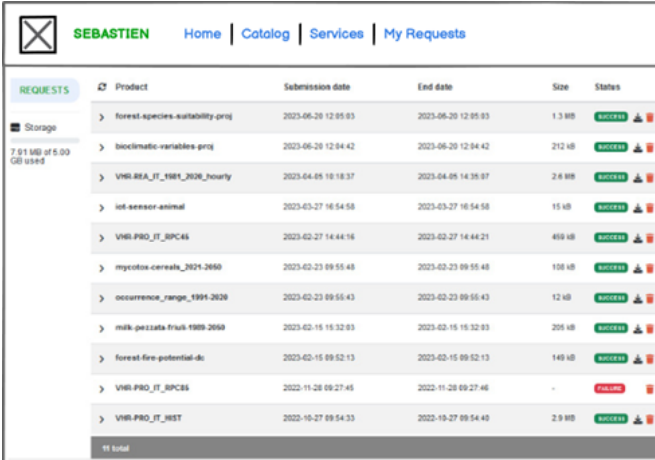
In the dataset detail page, clicking the "Get Data" button a dialog window opens from which the users can build a query and then submit it to the system (Figure 7).



The form is titled "Product: Indicator tz9" and includes a "Size Estimate" of 25.85 MB and a "4.97 GB" value. It features a "Variables" section with radio buttons for various climate indicators such as 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 Warmest Quarter, Mean Temperature of Coldest Quarter, Mean Temperature of Warmest Month, Mean Temperature of Coldest Month, Precipitation of Driest Month, Precipitation of Warmest Month, Precipitation of Coldest Month, Precipitation of Wettest Quarter, Precipitation of Driest Quarter, Precipitation of Warmest Quarter, Precipitation of Coldest Quarter, Precipitation Seasonality (Coefficient of Variation), and Temperature Seasonality (standard deviation \*100). There is also a "Spatial Coverage" section with a map and a "Submit" button.

Figure 7 - Mockup of the form to create the query

The result of the query will be 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 8).



Product	Submission date	End date	Size	Status
forest-species-suitability-prj1	2023-06-20 12:05:03	2023-06-20 12:05:03	1.3 MB	Success
bioclimatic-variables-prj1	2023-06-20 12:04:42	2023-06-20 12:04:42	212 kB	Success
VNI_RRA_FT_1981_2020_hourly	2023-04-05 10:18:37	2023-04-05 14:35:07	2.6 MB	Success
lot-sensor-animal	2023-03-27 16:54:58	2023-03-27 16:54:58	15 kB	Success
VNI_PBO_FT_RPC46	2023-02-27 14:44:16	2023-02-27 14:44:21	489 kB	Success
mycotos-cereals_2021-2020	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-1989-2020	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	140 kB	Success
VNI_PBO_FT_RPC65	2022-11-28 09:27:46	2022-11-28 09:27:46	-	Failure
VNI_PBO_FT_WST	2022-10-27 09:54:33	2022-10-27 09:54:40	2.9 MB	Success
11 total				

Figure 8 - Mockup of the list of the queries submitted by the user

### Requirements for the data lake

To implement the catalogue, the data lake must expose an API that provides:

- the list of the datasets of the catalogue
- the metadata of a dataset
- the possibility to submit a query for extracting data from a dataset
- the possibility to download the resulting data of the query

## 4. Data Lake APIs

This section presents the Application Programming Interface (API) exposed by the Data Lake to query datasets listed in the catalogue when they are requested from the Data Access Layer.

The Data Lake is the component that provides the SEBASTIEN Services and Data Portal with a unified way to access, process and store the datasets coming from different and heterogeneous data sources, as described in Deliverable 3.1.

The following table lists the input, output and action for each API endpoint provided by the Data Lake for data extraction and processing and accessible via <https://sebastien-datalake.cmcc.it/api/v2>.

ACTION	VERB	ENDPOINT	PATH PARAMETERS	QUERY BODY / PARAMETERS	RESPONSE	URL EXAMPLE
<b>ASYNCHRONOUS API USED BY DATA PORTAL</b>						
Get Datasets list	GET	/api/v2/datasets	None	None	Status Code 200 (OK) with the Datasets list object	/api/v2/datasets
Get first Product details	GET	/api/v2/datasets/{dataset_id}	{ "dataset_id": { "type": "string", "required": true } }	None	Status Code 200 (OK) with the Dataset object	/api/v2/datasets/pi
Get Product details	GET	/api/v2/datasets/{dataset_id}/{product_id}	{ "dataset_id": { "type": "string", "required": true }, "product_id": { "type": "string", "required": true } }	None	Status Code 200 (OK) with the Dataset object	/api/v2/datasets/pi/outdoors
Submit query	POST	/api/v2/datasets/{dataset_id}/{product_id}/execute	{ "dataset_id": { "type": "string", "required": true }, "product_id": { "type": "string", "required": true } }	JSON representation of the query	Status Code 200 (OK) with the request id	/api/v2/datasets/pi/outdoors/execute
Get all requests for the user	GET	/api/v2/requests	None	None	Status Code 200 (OK) with the requests list object	/api/v2/requests



Get request status	GET	/api/v2/requests/{request_id}/status	{ "request_id": { "type": "integer", "required": true }}	None	Status Code 200 (OK) with the request status	/api/v2/requests/6671/status
Get request resulting size	GET	/api/v2/requests/{request_id}/size	{ "request_id": { "type": "integer", "required": true }}	None	Status Code 200 (OK) with the request result size	/api/v2/requests/6671/size
Get request uri	GET	/api/v2/requests/{request_id}/uri	{ "request_id": { "type": "integer", "required": true }}	None	Status Code 200 (OK) with the request uri	/api/v2/requests/6671/uri
Downloaded the result of the request	GET	/api/v2/download/{request_id}	{ "request_id": { "type": "integer", "required": true }}	None	Status Code 200 (OK) with the result file	/api/v2/download/6671
<b>SYNCHRONOUS API USED BY SERVICES</b>						
Get Map	GET	/api/v2/datasets/{dataset_id}/{product_id}/map	{ "dataset_id": { "type": "string", "required": true }, "product_id": { "type": "string", "required": true }, "width": { "type": "integer", "required": true }, "height": { "type": "integer", "required": true }, "layers": { "type": "string", "required": true }, "time": { "type": "string", "format": date-time, "required": true }, "bbox": { "type": "string", "required": true }, "bgcolor": { "type": "string" }, "transparent": { "type": "boolean" }, "format": { "type": "string", }}}	None	Status Code 200 (OK) with the Map object	/api/v2/datasets/pi/outdoors/map?layers=fat&time=2023-11-20T00:00:00.000000000&bbox=14.7,44.7,15,45&width=100&height=100

Get Feature	GET	/api/v2/datasets/{dataset_id}/{product_id}/items/{feature_id}	<pre> {   "dataset_id": {     "type": "string",     "required": true   },   "product_id": {     "type": "string",     "required": true   },   "feature_id": {     "type": "string",     "required": true   },   "time": {     "type": "string",     "format": "date-time",     "required": true   },   "bbox": {     "type": "string",     "required": true   } } </pre>	None	Status Code 200 (OK) with GeoJSON object	/api/v2/datasets/pi/outdoors/items/fat?time=2023-11-20T00:00:00.000000000&bbox=14.7,44.7,15,45
-------------	-----	---	--	------	--	--

Table 1: Data Lake API

As shown in the table, the main endpoints used by SEBASTIEN Services are the Open Geospatial Consortium (OGC) API Map and Features. Since they are synchronous operations, the Data Access Layer submits the request and remains blocked until the request is completed.

In contrast, the interaction with the Data Portal occurs through the use of asynchronous calls. It means that the Data Access Layer submits the request and then, with the request identifier returned, it will poll the backend at regular intervals to check the request's status until it has been completed. The users can get information about data, submit a query on data of interest and download the result. In particular, the API offers the possibility of querying and retrieving data by giving the dataset id, the product id and the request body as a JSON file whose schema is the following:

```

{
  "$schema": "https://json-schema.org/draft/2020-12/schema",
  "$id": "sebastien_data_lake_api.json",
  "title": "Query",
  "description": "A query on a dataset",
  "type": "object",
  "properties": {
    "variable": {
      "description": "The list of variables for a dataset",
      "type": "array",
      "items": {
        "type": "string"
      }
    },
    "time": {
      "description": "The time period for a dataset",
      "anyOf": [ {

```





```
"type": "object",
"properties": {
  "year": {
    "type": "array",
    "items": {
      "type": ["string", "integer"]
    }
  },
  "month": {
    "type": "array",
    "items": {
      "type": ["string", "integer"]
    }
  },
  "day": {
    "type": "array",
    "items": {
      "type": ["string", "integer"]
    }
  },
  "hour": {
    "type": "array",
    "items": {
      "type": ["string", "integer"]
    }
  }
},
{
  "type": "object",
  "properties": {
    "start": {
      "type": "string",
      "format": date-time
    },
    "stop": {
      "type": "string",
      "format": date-time
    }
  }
} ]
},
"area": {
  "description": "The bounding box for a dataset",
  "type": "object",
  "properties": {
    "north": {
      "type": "number",
    },
  },
}
```

```
    "south": {
      "type": "number",
    },
    "west": {
      "type": "number",
    },
    "east": {
      "type": "number",
    }
  }
},
"location": {
  "description": "The location for a dataset",
  "type": "object",
  "properties": {
    "latitude": {
      "type": "number",
    },
    "longitude": {
      "type": "number",
    }
  }
},
"filters": {
  "description": "The filters for a dataset",
  "type": "object",
  "patternProperties": {
    "^.*$": {
      "type": "string",
    }
  },
  "additionalProperties": false
},
"format": {
  "description": "The format of the query result",
  "type": "string",
  "enum": ["netcdf", "geojson"]
}
}
```

The query allows to specify an area or location of interest, a time range or combo values for years, months, days and hours (depending on the temporal resolution of the dataset), the list of variables and other filters specific to the dataset itself. Once the request is successfully completed, the user can download the result.

The following tables report the detailed specifications on how the Data Lake API is used by each SEBASTIEN Service.

SERVICE 1a	
VARIABLES	<ul style="list-style-type: none"> <li>· fat</li> <li>· milk</li> <li>· proteins</li> </ul>
API EXAMPLE	<ul style="list-style-type: none"> <li>○ <b>MAP</b>  <b>Format:</b>            /api/map?layers=&lt;layers&gt;&amp;time=&lt;time&gt;&amp;bbox=&lt;bbox&gt;&amp;width=&lt;width&gt;&amp;height=&lt;height&gt;  <b>Example:</b> curl -X GET -H "Content-Type: application/png" -H "User-Token: &lt;api-key&gt;"            "https://sebastien-datalake.cmcc.it/api/v2/datasets/pi/outdoors/map?layers=fat&amp;time=2023-11-20T00:00:00.000000000&amp;bbox=14.7,44.7,15,45&amp;width=100&amp;height=100" -o test.png</li> <li>○ <b>GeoJSON</b>  <b>Format:</b> /api/items/&lt;layers&gt;?time=&lt;time&gt;&amp;bbox=&lt;bbox&gt;   <b>Example:</b> curl -X GET -H "Content-Type: application/json" -H "User-Token: &lt;api-key&gt;"            "https://sebastien-datalake.cmcc.it/api/v2/datasets/pi/outdoors/items/fat?time=2023-11-20T00:00:00.000000000&amp;bbox=14.7,44.7,15,45" -o test.json</li> </ul>
SERVICE 1b	
VARIABLES	<ul style="list-style-type: none"> <li>· external_thermohygrometric_index</li> <li>· internal_thermohygrometric_index</li> </ul>
API EXAMPLE	<ul style="list-style-type: none"> <li>○ <b>MAP</b>  <b>Format:</b>            /api/map?layers=&lt;layers&gt;&amp;time=&lt;time&gt;&amp;bbox=&lt;bbox&gt;&amp;width=&lt;width&gt;&amp;height=&lt;height&gt;  <b>Example:</b> curl -X GET -H "Content-Type: application/png" -H "User-Token: &lt;api-key&gt;"            "https://sebastien-datalake.cmcc.it/api/v2/datasets/thi/hourly/20230901/map?layers=external_thermohygrometric_index&amp;time=2023-09-01T12:00&amp;bbox=14.7,44.7,15,45&amp;width=100&amp;height=100" -o test.png</li> <li>○ <b>GeoJSON (for an area and a time)</b>  <b>Format:</b> /api/items/&lt;layers&gt;?time=&lt;time&gt;&amp;bbox=&lt;bbox&gt;  <b>Example:</b> curl -X GET -H "Content-Type: application/json" -H "User-Token: &lt;api-key&gt;"            "https://sebastien-datalake.cmcc.it/api/v2/datasets/thi/hourly/20230901/items/external_thermohygrometric_index?time=2023-09-01T12:00&amp;bbox=14.7,44.7,15,45" -o test.json</li> <li>○ <b>GeoJSON (for a location and a time series)</b>  <b>Format:</b> /api/items/&lt;layers&gt;?point=&lt;point&gt;  <b>Example:</b> curl -X GET -H "Content-Type: application/json" -H "User-Token: &lt;api-key&gt;"            "https://sebastien-datalake.cmcc.it/api/v2/datasets/thi/hourly/20230901/items/external_thermohygrometric_index?point=14.7,44.7" -o test.json</li> </ul>

Table 2: Service 1 API

SERVICE 2a	
VARIABLES	<ul style="list-style-type: none"> <li>external_thermohygrometric_index</li> <li>internal_thermohygrometric_index</li> </ul>
API EXAMPLE	<ul style="list-style-type: none"> <li>GeoJSON</li> </ul> <p><b>Format:</b> /api/items/&lt;layers&gt;?point=&lt;point&gt;</p> <p><b>Example:</b> curl -X GET -H "Content-Type: application/json" -H "User-Token: &lt;api-key&gt;"</p> <p>"https://sebastien-datalake.cmcc.it/api/v2/datasets/thi/hourly/20230901/items/external_thermohygrometric_index?point=14.7,44.7"-o test.json</p>
SERVICE 2b	
VARIABLES	<ul style="list-style-type: none"> <li>external_thermohygrometric_index</li> <li>internal_thermohygrometric_index</li> </ul>
API EXAMPLE	<ul style="list-style-type: none"> <li>GeoJSON</li> </ul> <p><b>Format:</b> /api/items/&lt;layers&gt;?point=&lt;point&gt;</p> <p><b>Example:</b> curl -X GET -H "Content-Type: application/json" -H "User-Token: &lt;api-key&gt;"</p> <p>"https://sebastien-datalake.cmcc.it/api/v2/datasets/thi/rcp4.5/20230901/items/external_thermohygrometric_index?point=14.7,44.7"-o test.json</p>

Table 3: Service 2 API

SERVICE 3	
VARIABLES	<ul style="list-style-type: none"> <li>pasture_tq</li> <li>pasture_ss</li> </ul>
API EXAMPLE	<ul style="list-style-type: none"> <li>MAP</li> </ul> <p><b>Format:</b></p> <p>/api/map?layers=&lt;layers&gt;&amp;time=&lt;time&gt;&amp;bbox=&lt;bbox&gt;&amp;width=&lt;width&gt;&amp;height=&lt;height&gt;</p> <p><b>Example:</b> curl -X GET -H "Content-Type: application/png" -H "User-Token: &lt;api-key&gt;"</p> <p>"https://sebastien-datalake.cmcc.it/api/v2/datasets/pasture/ss/Pasqualetti/map?layers=pasture_ss&amp;time=2023-09-01T12:00&amp;bbox=14.7,44.7,15,45&amp;width=100&amp;height=100" -o test.png</p> <ul style="list-style-type: none"> <li>GeoJSON</li> </ul> <p><b>Format:</b> /api/items/&lt;layers&gt;?time=&lt;time&gt;&amp;bbox=&lt;bbox&gt;</p> <p><b>Example:</b> curl -X GET -H "Content-Type: application/json" -H "User-Token: &lt;api-key&gt;"</p> <p>"https://sebastien-datalake.cmcc.it/api/v2/datasets/pasture/ss/Pasqualetti/items/pasture_ss?time=2023-11-20T00:00:00.000000000&amp;bbox=14.7,44.7,15,45" -o test.json</p>

Table 4: Service 3 API

SERVICE 4a	
VARIABLES	<ul style="list-style-type: none"> <li>• SCC</li> </ul>
API EXAMPLE	<ul style="list-style-type: none"> <li>○ <b>MAP</b>  <b>Format :</b>            /api/map?layers=&lt;layers&gt;&amp;time=&lt;time&gt;&amp;bbox=&lt;bbox&gt;&amp;width=&lt;width&gt;&amp;height=&lt;height&gt;  <b>Example :</b> curl -X GET -H "Content-Type: application/png" -H "User-Token: &lt;api-key&gt;"            "https://sebastien-datalake.cmcc.it/api/v2/datasets/pi/outdoors/map?layers=scc&amp;time=2023-11-20T00:00:00.000000000&amp;bbox=14.7,44.7,15,45&amp;width=100&amp;height=100" -o test.png</li> <li>○ <b>GeoJSON</b>  <b>Format :</b> /api/items/&lt;layers&gt;?time=&lt;time&gt;&amp;bbox=&lt;bbox&gt;  <b>Example :</b> curl -X GET -H "Content-Type: application/json" -H "User-Token: &lt;api-key&gt;"            "https://sebastien-datalake.cmcc.it/api/v2/datasets/pi/outdoors/items/scc?time=2023-11-20T00:00:00.000000000&amp;bbox=14.7,44.7,15,45" -o test.json</li> </ul>
SERVICE 4b	
VARIABLES	<ul style="list-style-type: none"> <li>• SCC</li> </ul>
API EXAMPLE	<ul style="list-style-type: none"> <li>○ <b>MAP</b>  <b>Format :</b>            /api/map?layers=&lt;layers&gt;&amp;time=&lt;time&gt;&amp;bbox=&lt;bbox&gt;&amp;width=&lt;width&gt;&amp;height=&lt;height&gt;  <b>Example :</b> curl -X GET -H "Content-Type: application/png" -H "User-Token: &lt;api-key&gt;"            "https://sebastien-datalake.cmcc.it/api/v2/datasets/pi/outdoors/map?layers=scc&amp;time=2023-11-20T00:00:00.000000000&amp;bbox=14.7,44.7,15,45&amp;width=100&amp;height=100" -o test.png</li> <li>○ <b>GeoJSON</b>  <b>Format :</b> /api/items/&lt;layers&gt;?time=&lt;time&gt;&amp;bbox=&lt;bbox&gt;  <b>Example :</b> curl -X GET -H "Content-Type: application/json" -H "User-Token: &lt;api-key&gt;"            "https://sebastien-datalake.cmcc.it/api/v2/datasets/pi/outdoors/items/scc?time=2023-11-20T00:00:00.000000000&amp;bbox=14.7,44.7,15,45" -o test.json</li> </ul>

Table 5: Service 4 API

## ANIMAL SENSOR DATA

<b>VARIABLES</b>	<ul style="list-style-type: none"> <li>· string_type</li> <li>· cycle_duration</li> <li>· acceleration</li> <li>· standardDeviationX</li> <li>· acceleration</li> <li>· standardDeviationY</li> <li>· accelerationZ</li> <li>· standardDeviationZ</li> <li>· AirT</li> <li>· AirH</li> <li>· HeartRate</li> <li>· surfaceTemp</li> <li>· battery</li> </ul>
<b>API EXAMPLE</b>	<ul style="list-style-type: none"> <li>○ <b>GeoJSON</b></li> <li><b>Format:</b> /api/items/&lt;layers&gt;?time=&lt;time&gt;</li> <li><b>Example:</b> curl -X GET -H "Content-Type: application/json" -H "User-Token: &lt;api-key&gt;"</li> <li>"https://sebastien-datalake.cmcc.it/api/v2/datasets/iot/animal/1123a005/items/AirT?time=NOW-1d/NOW" -o test.json</li> </ul>

Table 6: Animal sensor data API

## 5. References

- [1] GeoJSON format, website: <https://datatracker.ietf.org/doc/html/rfc7946>
- [2] OGC WMS, website: <https://www.ogc.org/standard/wms>
- [3] OGC WFS, website: <https://www.ogc.org/standard/wfs>
- [4] JSON, website: <https://www.json.org/json-en.html>