DELIVERABLE 2.4

AgriBIT Open APIs and interfaces (rev. 1)

Project Acronym	AgriBIT
Project Title	Artificial intelliGence applied to pRecision farmIng By the use of GNSS and Integrated Technologies
Grant Agreement number	101004259
Call	SU-SPACE-EGNSS-3
Funding Scheme	Innovation Action (IA)
Project duration	36 Months

Document Information			
Work Package:	WP2	**Task:**	T2.3
Due Date:	30/06/2022		
Version:	0.7	**Status:**	Final
Dissemination level:	PUBLIC		
Type	Report		
Lead Partner:	ENG		
Contributors:	RFSAT, INOV, AGENSO		
Keywords:	API, Open API, Interfaces, GNSS enabler services, Big Data Analytics		
Abstract:	The objective of this document is to describe the AgriBIT Open APIs and interfaces. Firstly, a technological context is given, then the list of AgriBIT APIs for third parties' users is presented. The APIs are splitted in three major macro areas: PA services, GNSS enablers services and Big Data Analytics (BDA) services. Each service is described using a summarization table containing all the information needed in order to invoke the service. This document provides both the guidelines for coordinating the activities development and also a general idea of the components and the software that will be delivered by the project.		

Document History			
Version	**Date**	**Contributor(s)**	**Description**
0.1	08/02/2022	ENG	Initial Version
0.4	13/04/2022	ENG	Intermediate Version
0.5	01/06/2022	ENG	Finalized Version
0.6	14/06/2022	ENG ,Agroapps	Almost finalized version after AGROAPPS internal review comments
0.7	20/06/2022	ENG	Final Version ready to be submitted

Document Authors	
Engineering	Giuseppe Vella Piero Scrima

Document Internal Reviewers	
AGROAPPS	Ifigeneia Tsioutsia

ACKNOWLEDGEMENT

This project has received funding from the European Union Agency for the Space Programme under the European Union's Horizon 2020 research and innovation programme under grant agreement No 101004259.

Executive Summary

This document addresses the description of the first version of AgriBIT Open APIs and interfaces. In the first place a technological context is given, describing the concept of Application Program interfaces (APIs) and the OpenAPI specification, and providing an overview on OGC WMS e WFS standards. These standards allow, by HyperText Transfer Protocol (HTTP) interface, to provide geospatial raster services and vector geospatial service. After these introductions on the main technologies and standards involved, the list of AgriBIT APIs for third parties' users is presented. The APIs are splitted in three major macro areas: Precision Agriculture (PA) services (GNSS enabled services), GNSS enablers services and Big Data Analytics (BDA) services. The first area addresses all the services related to target of the project itself, in fact with this information a third party's users can retrieve the main elaboration of the project to help and advice farmers in order to monitoring the crop growth, optimize activities of: irrigation, tillage and the fertilization; retrieve information and advices on the possible pests and diseases of the crops. GNSS enablers services enable the invoker to retrieve information on the parcels, as for PA services. The invoker should be the owner of the parcel or a user specifically authorized by the latter. GNSS enablers services also provides sensors data collected on the field, along with the coordinates in GeoJSON of the field border and geometry. Finally, the Big Data Analytics services (BDA) empower the third party's users by giving them the possibility to work with Big Data Analytics provided by ALIDA platform. BDA are designed using a graphical user interface (GUI). Once the workflow and the analytics are built the APIs allow invoker to manage it, and also to check the status and the associated resource, such as datasets, machine learning models, data sources and applications. In the conclusion paragraph the importance of this document for the coordination and the development of the AgriBIT components is stated.

Table of Contents

List of Tables

1. Introduction

AgriBIT is a project that aims to innovate the integrated farm management systems market by applying in synergy a set of Agriculture techniques and advanced technologies such as "On-site Sensing" and "Earth Observation" which in recent years have been subjects of major improvements [1]. The use of these techniques and technologies together in a unique software solution makes it possible to cover a broader set of use cases and involving various players in the sector such as operators who want to integrate services into their systems. The purpose of the Application Program Interfaces (APIs), object of this document, is precisely to allow external actors to join the project, expanding the market and including not only farmers and service advisors who want to use the platform but also users who wish to integrate AgriBIT features into different platforms and software products.

To include external actors in the project AgriBIT solutions need to be accessible to external systems. APIs (Application Common Interface) are the common software architectural solutions to provide data exchange among software modules and components, in other words they define the way in which software modules and components interact. APIs are also fundamentals to provide interoperability, the capability of a software system to connect to external systems. Also, AgriBIT project leverages APIs to provide a way to integrate and use the AgriBIT operational services by third parties' users. The most common and easier way to reach out a remote system or platform is by internet, AgriBIT implements web APIs in order to expose APIs over HTTP internet protocol and reach the broadest number of users.

This is the first version of the "AgriBIT Open APIs and interfaces" document in which a description of the services to the third users is provided, services are described in terms of architecture patterns and standard (es: REST) and in terms of data structure exchanged. Data structure outlines the information of the data exchanged in the APIs requests in terms of data attribute with description and type specification.

This deliverable integrates with other project activities and deliverables, in particular the API design takes into account the requirements collected during task T2.1 "Demonstrator User Requirements" that have been summarized into deliverable D2.1 "AgriBIT user requirements". The design of the APIs also took into consideration the technical requirements that emerged during task T2.2 "Technology Requirements". The APIs described in this deliverable and its subsequent versions will be developed in task T5.1 "Platform Infrastructure Development and collaborative services" and released with deliverable D5.2 "AgriBIT community platform" and the subsequent version. Furthermore, the APIs must be integrated with the rest of the system and with the upstream components that will be released by deliverables D4.2 "GNSS enablers and GNSS enabled services - 1st release" and D4.3 "GNSS enablers and GNSS enabled services - 2nd release". Finally, in task T5.4 "Continuous Integration: from prototype to commercial readiness" the APIs will be deployed along with the other components of the system.

In the next chapters firstly a technology context is given by a general introduction to the APIs and their features, along with the description of one of the most used APIs standards that is the OPEN API[1]. Apart from the Open API Specification (OAS) the chapter describes also an APIs standard to share map and Geospatial information: Web Map Service[2]WMS and Web feature service[3] WFS. Then a list of services is provided explaining and documenting their functionality and, as aforementioned, the related data structure.

[1] https://www.openapis.org/about
[2] https://www.ogc.org/standards/wms
[3] https://www.ogc.org/standards/wfs

1.1. APIs style

Software systems are often used to solve complex problems, a solution to deal with complexity is to divide the problem into simpler problems. For this reason software systems can be made up by subsystems and components that perform sub-parts of the general problem, as the maxim "divide and conquer" says. These components perform internally complex functions and expose only the functionalities needed by the other components hiding the code implementation. The solution to make components communicate is the so-called API. Initially, in software history, the systems were mostly isolated and monolithic, but with the progress and the consolidation of the communication infrastructures, the decomposition of the systems has gradually reached a larger level. Over time, the functionalities have been spread not just among internal components of a system but also among different systems in a network. APIs have also followed this trend, enabling software application to communicate over a network infrastructure and its most spread implementation: internet.

AgriBIT aims to reach all the actors and third parties that what to join the project to leverage its data and services and internet and APIs are the best solution to active this objective. The most spread internet application is the world wide web (web in short). Web is the name for the resources shared in internet using HTTP protocol, hence HTTP protocol is one of the most know and reliable standard in communication protocol, for this reason HTTP in fact is the most spread solution for internet APIs. APIs that are built on HTTP protocol and used over internet are called Web APIs.

A specific architectural style in which design Web APIs is REST [2], which stands for “Representational state transfer”.

REST APIs rely on common practice and principles. One of these principles says that the APIs should be stateless, meaning that all the required information for the server to be executed have to be included in the API call itself and there is no need to store context or session on the server between requests. That’s bring advantages such as horizontal scalability. In fact, in a configuration of multiple servers providing stateless APIs, all the servers can process the API request and scale because each server has already all the information needed in that request. In a configuration of multiple servers providing stateful APIs, instead the context is stored in a specific server which is the only one that can serve the request, scaling horizontally in this circumstance it is much more difficult. Another advantage of stateless is that the response is more predictable; in fact it depends just on the parameter of the request. Finally, the client that calls a stateless API is simpler, because it doesn’t need to store the state or the context to reflect the context of the server.

Another important feature of REST APIs is the resource orientation. This means that each request refers to a certain resource and that the operation directly addresses this resource. The fact that the request refers always to a resource allows the standardization of the operations. Four operations are enough to manipulate a resource: create, read, update, delete, the so called CRUD. The reduced set of operations helps to eliminate the complexity of the request and provides a simpler, easier and more understandable way of development.

To adopt these principles, REST APIs rely on the HTTP protocol. HTTP seems to be tailored for REST operations, due to the overlap of the operations: create, read, update, delete with the HTTPs methods: POST, GET, PUT and DELETE. HTTP moreover involves another important element that is perfectly suited with the resource orientation principle, which is the Uniform Resource Locator (URL). URL allows to identify and locate a resource over internet, so URL is essentially the way in which REST API refers to a resource.

So, thanks to the HTTP method and the URL specification, REST API can address the importance of the principles of resource orientation; stateless and limited in the set of operations. These ingredients make REST over HTTP a simpler and more effective API style.

Principles so far described are the general REST principles. However, there are other stricter rules that can also be considered in the design of REST APIs and if a set of APIs are compliant to all the REST principles, then the APIs are called RESTful. Most of the times, the specific problems and the project needs are addressed by a subset of these principles, also because REST can be combined with other project standards or constraints. In AgriBIT, the REST principles that have been taken into consideration have been combined with other APIs standards and principles, such as OpenAPI standard, Web Map Server (WMS) and Web Feature Server (WFS), which comprise the topics of the next chapters.

1.2. OpenAPI

Systems APIs are made up by long list of information, services, resources, endpoints and so on, and in order to be easily used, a well-prepared document describing all the possible APIs is needed. APIs can be documented in many different ways, such as simpler text documents or more structured referenced documents with tables and structured fields [3]. Over the years, different formats and standards have emerged to improve clarity and quality, such as Javadoc. One of the most used standards for documenting web APIs and in particular REST APIs is the OpenAPI Specification (OAS in brief). OAS is an API description format that was initially a proprietary project named Swagger Specification from the Swagger application by the SmartBear Software[4], and in the 2016 OAS became an open-source project under the Linux Foundation[5].

REST APIs can be implemented using different protocols and formats and web REST APIs rely on internet protocols and HTTP. OAS aims to standardize the specification of web REST APIs, introducing a format[6] that guides in the description of an HTTP request.

To define APIs set in OAS, a description document in YAML has to be created. The document in a nutshell aims to describe the most important features of an HTTP request, that as aforementioned, covers the most important parts of an API. The basic information that needs to be specified in the description is the name: in order to identify the API; the path: to locate the resource; operation and data.

Path is specified as the URL of the resource: it can be relative of absolute. The operation has to be one among the four HTTP method, GET, POST, PUT and DELETE. Data that need to be described, are the input and output data. Input data can be one between request parameters, or body parameters, and the model of the data can be described using the JavaScript Object Notation (JSON) schema[7]. Thanks to JSON schema, model can be described with a standard in terms of data structure and data types. The same description can be applied also to the output data. In the web APIs, output data are included in the HTTP response, and the HTTP response code is part of the output data, so each web API needs to specify the response code and the data structure in JSON schema.

[4] https://smartbear.com/
[5] https://www.linuxfoundation.org/
[6] https://swagger.io/specification/
[7] https://json-schema.org/ JSON Schema is a vocabulary that allows you to annotate and validate JSON documents.

Each of the element included in the API specification can be supported by a description text, which it's always recommended because helps the final user to understand the element and the context.

In this document, the REST APIs that are described, have taken into consideration all the features mentioned previously, and these data are presented in tables in order to be clearly readable. The project will also deliver APIs description document through OAS files, that will be accessible through the AgriBIT Community Platform.

An exhaustive and complete discussion of the OAS is beyond the scope of this document, but it is possible to find various resources and guides starting from the site of the consortium that manages and regulates the OAS.

1.3. WMS and WFS

The data collected by AgriBIT are data related to the territory, such as the soil data that are collected by the sensors or the different types of images representing parcels. Some territory data contain information with regards to their position or to the position of the data source, and these data are called geospatial data or georeferenced data accordingly.

There are two main approaches of data modelling that enable the representation of geospatial data: vector or raster data. Vector data use a set of discrete locations to build basic geometrical shapes, such as points, polylines, and polygons. Raster data use a regular tessellation, defining cells where one or more values are uniform. Usually, the cells are square; although, this is not a constraint. Raster data are generally used to represent models of the Earth's surface.

Georeferenced data are part of the information that need to be provided to third-party users, unlike to the non-georeferenced data, which these types of data are more suitable to be used and also to be integrated directly on maps. Open Geospatial Consortium[8] (OGC) is an independent standardization body made up by a committee that involves more than 500 organizations. This committee has defined some standards that have allowed the creation of services for the access spatial data directly on maps. In fact, there are numerous products that are compliant with these services that allow the integration of spatial information on platforms, libraries and GIS frameworks.

OGC has created two different standard that allow by HTTP interface to provide geospatial raster services and vector geospatial services, the WMS[9] and WFS[10] services.

By WMS (Web map service) GetMap method, users can consume services and obtain a map (raster). The required arguments of the method include the bounding box, the spatial reference system and the dimensions of the image (hight and width). The response of the service is usually a raster image such as PNG JPEG TIFF, and so on, and the list of possible result image formats depends on the configuration of the providing server. Table 1, shows the most important operation defined by WMS.

Operation	Description
GetCapabilities	Retrieves metadata about the service, including supported operations and parameters, and a list of the available layers
Exceptions	If an exception occurs
GetMap	Retrieves the map

[8] https://www.ogc.org/

[9] https://www.ogc.org/standards/wms

[10] https://www.ogc.org/standards/wfs

Table 1 - WMS operations

Alongside the WMS services, OGC also defines services to operate on vector data that are WFS which stands for Web Feature Service. WFS like WMS consists of an HTTP interface that defines the names of the services, methods and parameters to request vector data in the form of Geography Markup Language (GML), a subset of XML, from a compliant server with WFS.

In addition to allowing access and reading of vector data, unlike to WMS, WFS also provides services for modifying data using a simple system of transactions or locking.

Another feature that distinguishes WFS from WMS is direct download of data. WFS request can be set up to get the file with all the georeferenced data related to a specific layer.

Requests for access to georeferenced data must be made by setting the request parameter to "GetFeature", in addition to the type of request, the layer must also be specified using the typeNames parameter. Table 2 shows the most important operations of WFS.

Operation	Description
GetCapabilities	Generates a metadata document describing a WFS service provided by server as well as valid WFS operations and parameters
DescribeFeatureType	Returns a description of feature types supported by a WFS service
GetFeature	Returns a selection of features from a data source including geometry and attribute values
LockFeature	Prevents a feature from being edited through a persistent feature lock
Transaction	Edits existing feature types by creating, updating, and deleting

Table 2 - WFS operations

Some of AgriBIT services adopt the OGC WMS or WFS standard. All these services are also available using the Community platform as well their OAS, but in present document to avoid redundancies (all the WMS and WFS service implements the same method and parameters) of the information the WMS and WFS services are described just in terms of their results and path. The parameters will be described only if they are not included in the standard.

2. AgriBIT Services to third parties

AgriBIT's services to third-party users will be provided through the community platform, using a combination of the Web APIs standards described in the previous chapters (REST, WMS, WFS). The APIs will be described in the pages of the community platform from where it will also be possible to download the Open API specification. The APIs will be exposed with an authentication mechanism to limit the access to authorized users only. Third-party users who want to access specific APIs for specific parcels will have to apply through the community platform.

Through the APIs, third-party users will be able to:

- obtain information of the parcels imported, through the services linked to GNSS enablers data
- obtain results from the PA services for the imported parcels. run and manage big data analytics and workflow through REST APIs and a dedicated GUI.

The specific functionalities for each group are described in the following chapters.

The services are presented through the Table 3 - Service . The Table includes all the needed information to describe/ present the services. The "Service name" is the field in which the service is entitled. Furthermore, the table includes a categorization of the service illustrated in the "Type" field. Basically, the "Type" section summarizes the standard adopted by the API, such as WMS or WFS or the protocol and the method if it is a simple HTTP call (OPEN API standard).

Service name	<service name>		
Resource Path	</<resource_type/{data_model:ex_id}>		
Type	<service type: WMS…, HTTP GET, HTTP POST…>		
Data input (Paramenter/Body)			
Model name	**Format**	**Attribute**	**Example**
<name of the data model (geometry, crop type, crop data…)>	<format: json, GeoJSON, STRING…>	<attribute of the model, es: lon (string), lat (string),value (int)>	<an example of the data>
Data output			
Model name	**Format**	**Attribute**	**Example**
<name of the data model (geometry, crop type, crop data…)>	<format: json, GeoJSON, STRING…>	<attribute of the model, es: lon (string), lat (string),value (int)>	<an example of the data>

Table 3 - Service template

2.1. PA services

Precision agriculture services are farm management services, which by integrating advanced geolocation technologies, allow the management of the field at the sub-field level. Traditionally the field is considered as a single entity carrying out uniform operations. Specifically, the fields have areas with different properties that require different interventions in order to optimize the use of resources, improve their yield, and also reduce pollution. The precision agriculture services therefore make it possible to recognize and locate these different areas of the fields, monitoring their yield, provide advice for typical farm activities, including, for example, irrigation and tillage scheduling. However, AgriBIT does not offer only this. AgriBIT provides a set/ package of PA services that are delivered to the end-users through user interfaces, if it is applicable, in order to view results on a parcel-based level combining data from many available sources. The aforementioned information along with the output of the sensors and in-situ data will be offered to the end-users through the following services.

2.1.1. Crop Growth Monitoring

Crop Growth Monitoring service regards information on crops' health. The service provides information of the vegetation indices. Maps are processed periodically, so different maps are available based on date interval.

Apart from the standard WMS request parameters this service requires a date interval. Depending on the interval the last image available will be provided.

There are four different endpoints as much as the different index available. For each parcel a different WMS layer is provided, the layer parameter, which is a standard WMS parameter, is encoded by parcel text prefix + underscore + parcel id.

Crop density:

Is a multispectral TVI index characterizing vegetation intensity to detect uneven growth in zones within the parcel, with a frequency of receiving and updating satellite images 2 to 5 days.

Service name	Crop density		
Resource Path	/cropdensity/wms		
Type	WMS		
Data input			
Model name	**Format**	**Attributes**	**Example**
Layer	String	Layer= <parcel_parcel_id>	Layer=parcel_12
Request	String	Request=getmap	request=getmap
Data interval	ISO 8601	time	2002-09-01T00:00:00.0Z/2002-09-30T23:59:59.999Z
Standard WMS input (See WMS description chapter)			
Data output			
Model name	**Format**	**Attributes**	**Example**
TVI map	GeoTIFF	• Image: raster	<none>

Table 4 - Crop density service

Crop stress:

Is a CRI2 multispectral index characterizing the concentration of carotenoids that are indirect indicators of crop stress. It will assist in stress detection and stress pathogenesis caused by either biotic or abiotic agents, with a frequency of receiving and updating satellite images 2 to 5 days.

Service name	Crop stress		
Resource Path	/cropstress/wms		
Type	WMS		
Data input			
Model name	**Format**	**Attributes**	**Example**
Request	String	Request=getmap	request=getmap
Layer	String	Layer= <parcel_parcel_id>	Layer=parcel_12
Data interval	ISO 8601	time	2002-09-01T00:00:00.0Z/2002-09-30T23:59:59.999Z
Standard wms input (See wms description chapter)			
Data output			
Model name	**Format**	**Attributes**	**Example**
CRI2 map	GeoTIFF	• Image: raster	<none>

Table 5 - Crop stress service

Water availability index:

Multispectral NDWI index that captures the water content of the crop canopy. It will assist in identifying areas with water shortage, with a frequency of renewal of satellite images 2 to 5 days.

Service name	Water availability index		
Resource Path	/ wateravailabilityindex/wms		
Type	WMS		
Data input			
Model name	**Format**	**Attributes**	**Example**
Request	String	Request=getmap	request=getmap

Layer	String	Layer= <parcel_parcel_id>	Layer=parcel_12
Data interval	ISO 8601	time	2002-09-01T00:00:00.0Z/2002-09-30T23:59:59.999Z
Standard wms input (See wms description chapter)			
Data output			
Model name	**Format**	**Attributes**	**Example**
NDWI map	GeoTIFF	• Image: raster	<none>

Table 6 - Water availability index service

Crop vigor:

Multispectral NDVI index that is a measure of the state of plant health based on how the plant reflects light at certain frequencies and is directly related to the presence of chlorophyll, with a frequency of renewal of satellite images 2 to 5 days.

Service name	Crop vigor		
Resource Path	/ cropvigor/wms		
Type	WMS		
Data input			
Model name	**Format**	**Attributes**	**Example**
Request	String	Request=getmap	request=getmap
Layer	String	Layer= <parcel_parcel_id>	Layer=parcel_12
Data interval	ISO 8601	time	2002-09-01T00:00:00.0Z/2002-09-30T23:59:59.999Z
Standard WMS input (See WMS description chapter)			
Data output			
Model name	**Format**	**Attributes**	**Example**
NDVI map	GeoTIFF	• Image: raster	<none>

Table 7 - Crop vigor

2.1.2. Crop yield estimation

Provides an estimation of some important parameters regarding the crop yield.

Services provided are:

- ground biomass
- yield map
- seasonal yield prediction

For each parcel a different WMS layer is provided, the layer parameter, which is a standard WMS parameter, is encoded by parcel text prefix + underscore + parcel id.

Service name	Ground biomass		
Resource Path	/groundbiomass/wms		
Type	WMS		
Data input			
Model name	**Format**	**Attributes**	**Example**
Request	String	Request=getmap	request=getmap
Layer	String	Layer= <parcel_parcel_id>	Layer=parcel_12

Data interval	ISO 8601	time	2002-09-01T00:00:00.0Z/2002-09-30T23:59:59.999Z
Standard WMS input (See WMS description chapter)			
Data output			
Model name	**Format**	**Attributes**	**Example**
Biomass map	GeoTIFF	• Image: raster	<none>
Data	ISO 8601	time	2002-09-01T00:00:00.0Z/

Table 8 - Ground biomass service

Service name	Yield estimation		
Resource Path	/yieldestimation/wms		
Type	WMS		
Data input			
Model name	**Format**	**Attributes**	**Example**
Request	String	Request=getmap	request=getmap
Layer	String	Layer= <parcel_parcel_id>	Layer=parcel_12
Data interval	ISO 8601	time	2002-09-01T00:00:00.0Z/2002-09-30T23:59:59.999Z
Standard WMS input (See WMS description chapter)			
Data output			
Model name	**Format**	**Attributes**	**Example**
Yield estimation map	GeoTIFF	• Image: raster	<none>
Data	ISO 8601	• time	2002-09-01T00:00:00.0Z/

Table 9 - Yield estimation service

Service name	Seasonal yield estimation		
Resource Path	/seasonalyieldestimation/wms		
Type	WMS		
Data input			
Model name	**Format**	**Attributes**	**Example**
Layer	String	Layer= <parcel_parcel_id>	Layer=parcel_12
Request	String	Request=getmap	request=getmap
Data interval	ISO 8601	time	2002-09-01T00:00:00.0Z/2002-09-30T23:59:59.999Z
Standard WMS input (See WMS description chapter)			
Data output			
Model name	**Format**	**Attributes**	**Example**
Seasonal Yield Estimation map	GeoTIFF	• Image: raster	<none>
Data	ISO 8601	• time	2002-09-01T00:00:00.0Z/

Table 10 - Seasonal yield estimation

2.1.3. Irrigation scheduling

This service is design in order to provide to the farmer an optimum irrigation planning method. The service recommends the irrigation water requirements at sub-parcel combining satellite data and local weather information. The irrigation functionality includes also an irrigation status service that allows user to add the irrigation activities. The main service is the crop water requirements, which during the growing season, provide a map with a daily value of water requirements per pixel.

Service name	Crop water requirements		
Resource Path	/cropwaterrequirements/wms		
Type	WMS		
Data input			
Model name	**Format**	**Attributes**	**Example**
Layer	String	Layer= <parcel_parcel_id>	Layer=parcel_12
Request	String	Request=getmap	request=getmap
Standard WMS input (See WMS description chapter)			
Data output			
Model name	**Format**	**Attributes**	**Example**
Seasonal Yield Estimation map	GeoTIFF	• Image: raster	<none>
Data	ISO 8601	• time	2002-09-01T00:00:00.0Z/

Table 11 - Crop water requirements service

Service name	Add irrigation operation		
Resource Path	/Irrigation		
Type	HTTP POST		
Data input			
Model name	**Format**	**Attributes**	**Example**
Parcel id	String	Parcel_id	Parcel_id=12
Irrigation data	ISO 8601	Data	data=2002-09-01T00:00:00.0Z
Irrigation amount	String	Value	13
Standard WMS input (See WMS description chapter)			

Table 12 - Add irrigation operation service

2.1.4. Tillage scheduling

High yields are associated with an effective tillage schedule, which entails well-cultivated soil, providing a proper environment for seeds to germinate and roots to grow. It is important to correctly manage the scheduling in order to have an effective tillage specific soil conditions that are required, such as the right amount of humidity. The tillage service through the elaboration of the soil parameters provides an optimized tillage plan. Third-party users are provided with an endpoint which will provide a map of soil conditions for tillage, per pixel.

Service name	Tillage		
Resource Path	/tillage/wms		
Type	WMS		
Data input			
Model name	**Format**	**Attributes**	**Example**
Layer	String	Layer= <parcel_parcel_id>	Layer=parcel_12
Request	String	Request=getmap	request=getmap
Standard wms input (See wms description chapter)			

Data output			
Model name	**Format**	**Attributes**	**Example**
Tillage map	GeoTIFF	• Image: raster	<none>

Table 13 - Tillage service

2.1.5. Pest and Disease Early Warning System

AgriBIT also provides a pests and diseases warning service, that provides a decision tool for prophylactic intervention and timely applications of fungicides. The warning system considers various variables including those relating to meteorological conditions combined with the climatic analysis of the territory. The third-party user will be able to access these services through invoking an endpoint, and the service will return data containing the list of possible problems with the days in which the problem could occur.

Service name	Warning System		
Resource Path	/warning		
Type	HTTP GET		
Data input			
Model name	**Format**	**Attributes**	**Example**
Layer	String	Layer= <parcel_parcel_id>	Layer=parcel_12
Request	String	Request=getmap	request=getmap
Standard wms input (See wms description chapter)			
Data output			
Model name	**Format**	**Attributes**	**Example**
Warning list	JSON	• warnings	{Warnings:[<warning>]}
warning	JSON	• date: data iso • pest: string • description: string	{date:<date> Pest:<pest type> Description:<pest description> }

Table 14 - Warning system service

2.1.6. Prescription mapping

By invoking the prescription mapping service, a variable rate fertilization map is provided. The variable rate file, used on specific machines, allows farmers to apply the right amount of fertilizers for specific areas of land. The process employs three phases, in the first the Management zones service provides a map that indicates to the farmer the different zones of the land, the farmer for each of these zones will have to check the nitrogen value of the soil, through the update service of the managements zone (POST) the farmer will send this information to the server. The server will calculate the resulting data by analyzing the remote sensing data and combining them with the soil tests. At this point, by invoking the third service, the system returns the variable rate fertilization file.

Service name	Management Zone		
Resource Path	/zone/wms		
Type	WMS		
Data input			
Model name	**Format**	**Attributes**	**Example**
Layer	String	Layer= <parcel_parcel_id>	Layer=parcel_12
Request	String	Request=getmap	request=getmap
Standard wms input (See wms description chapter)			
Data output			

Model name	Format	Attributes	Example
Zone map	GeoTIFF	• Image: raster	<none>

Table 15 - Management zone service

Service name	Management zone update		
Resource Path	/zone		
Type	HTTP POST		
Data input			
Model name	**Format**	**Attributes**	**Example**
Parcel id	String	Parcel_id	Parcel_id=12
zones	JSON	Zones:list	{zones:[<zone>]}
zone	JSON	• Zone • value	{zone:1,value:3.4}

Table 16 - Management Zone Update service

Service name	Prescription map		
Resource Path	/prescriptionmap		
Type	HTTP GET		
Data input			
Model name	**Format**	**Attributes**	**Example**
Parcel id	String	Parcel_id	Parcel_id=12
Type	String	Type	Type=variableratefertilization
Data output			
Model name	**Format**	**Attributes**	**Example**
prescription map	Variable rate	-	-

Table 17 - Prescription map service

2.2. GNSS enabler services

These APIs provide the information on the details of the parcel and on the sensory data collected on the field. Third-party users can get all the parcels related to a specific owner, with prior authorization, third-party users can also request parcel details in terms of boundaries and sensors installed. Finally, it also possible to obtain the data recorded by the sensors in the fields.

2.2.1. Parcel list

This service, by providing the owner id as argument, returns the list of the parcels which contain the information of the individual Parcel. The parcels in the list are those associated with the user in of the owner_id argument.

Parcel information contains the partner's name, ID and geometry.

Service name	Parcel list		
Resource Path	</parcel/owner/{data_model:owner_id}>		
Type	HTTP GET		
Data input			
Model name	**Format**	**Attributes**	**Example**
Owner	JSON	owner_id	{owner_id:10}
Data output			
Model name	**Format**	**Attributes**	**example**

Parcel list	JSON	• Parcel_list: Array (Parcel)	{parcel_list:[<parcel>..]}
Parcel	JSON	• Parcel name:string • Parcel id: long • Parcel_Geometry	{"id":1, "name":"tomatoes1", Sensors:<Sensors>, parcel_geometry:<Geometry>}
Parcel_Geometry	GeoJSON	• Type: string • Features : feature _type • Coordinates: Array (Coordinates)	{"type": "FeatureCollection", "features": [{ "type": "Feature", "properties": {}, "geometry": { "type": "LineString", "coordinates": [21 28],[....

Table 18 - Parcel list service

2.2.2. Parcel detail

The Parcel API provides the details of a single Parcel.

The API takes as argument the parcel id and returns the data of a single parcel. The information related to the parcel are: name of the parcel, the id and the list of sensors, which includes the name of the sensor, the id and the location, the API also returns the parcel geometry in GeoJSON.

Service name	Parcel detail		
Resource Path	</parcel/{data_model:parcel_id}>-		
Type	HTTP GET		
	Data input		
Model name	**Format**	**Attributes**	**Example**
Parcel	JSON	parcel_id	{parcel_id:10}
	Data output		
Model name	**Format**	**Attributes**	**example**
Parcel	JSON	• Parcel name:string • Parcel id: long • Sensors: Array (Sensors) • Parcel	{"id":1, "name":"tomatoes1", Sensors:<Sensors>, parcel_geometry:<Geometry>}
Parcel_Geometry	GeoJSON	• Type: string • Features : feature _type • Coordinates: Array (Coordinates)	{"type": "FeatureCollection", "features": [{ "type": "Feature", "properties": {}, "geometry": { "type": "LineString", "coordinates": [21 28],[....
Sensor	JSON	• Id • Name • Sensor_geometry	{"id":1435, "name":"temp1", sensor_geometry:<Geometry>}
Sensor_Geometry	GeoJSON	• Type: string	{"type": "FeatureCollection",

		• Features : feature _type
		• Coordinates: Array (Coordinates)
			"features": [
			{
			"type": "Feature",
			"properties": {},
			"geometry": {
			"type": "LineString",
			"coordinates": [21
			28],[....

Table 19 - Parcel detail service

2.2.3. Parcel data

Parcel data API provides information on the sensory data related to a specific Parcel.

Information returned contains the list of sensors and each sensor in turn contains the list of data recorded by the sensor.

Sensor information are: name, description, geometry of the sensor (position) , type of data and the list of values each value consists of a timestamp and the value itself

Service name	Parcel data		
Resource Path	/parceldata/<parcel_id>		
Type	HTTP GET		
	Data input		
Model name	**Format**	**Attributes**	**Example**
Parcel	String	parcel_id	parcel_id:10
Data_filter	JSON	Data_start Time_start Data_end Time_end	{Data_start:'22122021' Time_start:00:00 Data_end:'22022022' Time_end:00:00 }
Sensors	JSON	Sensor_ids	{Sensors:[1,3,45]}
	Data output		
Model name	**Format**	**Attributes**	**example**
Sensor_list	JSON	• Sensors_List	{sensors_list:[sensors...]
Sensor	JSON	• Id • Name • Sensor_geometry • Sensor_data	{"id":1435, "name":"temp1", sensor_geometry:<Geometry>, Sensor_data[]}

Sensor_Geometry	GeoJSON	• Type: string
		• Features : feature _type
		• Coordinates: Array (Coordinates)
			{"type": "FeatureCollection",
			"features": [
			{
			"type": "Feature",
			"properties": {},
			"geometry": {
			"type": "LineString",
			"coordinates": [21
			28],[....

Sensor_data	JSON	• Timestamp • value	{timestamp:453534, Value:30}

Table 20 - Parcel data service

2.3. BDA services

AgriBIT provides several solutions to improve farm management. In addition to the precision agriculture services, which are ready to be provided to end users, there is the possibility of further processing without having to implement new algorithms or components. In fact, to add new data processing, the third-party users can both directly use the data processed by the AgriBIT components, but they can also create new workflows and new analytics using the Big Data Analytics system, ALIDA[11]. A workflow in Alida is called APP and the processors the run algorithms and that made up the workflow are called services. ALIDA allows users to process fields data by exploiting a set of algorithms to be configured and connected to each other, creating a workflow. Furthermore, it is also possible to insert new algorithms creating new services. The data coming from the field are acquired and stored in special data storage for big data, these are then made available within the platform in the form of a dataset.

The BDA services are the processing that takes place within the ALIDA platform and makes up the workflows. Basically, they are components that allow users to acquire the data from one of the available data sources, process them using different types of algorithms and transmit them into data source or external systems. The services within the GUI are presented with blocks that can be dragged and dropped on a canvas and the workflow can be achieved by connecting and configuring these blocks.

The workflows can be created and processed via GUI on the ALIDA platform and once the workflow has been created, it can be controlled remotely via APIs. Moreover, it is possible to query ALIDA to know the available workflows, datasets, and data sources. The ALIDA APIs available for third-party users are listed and described below.

BDA application is described in the relevant deliverable and in the community platform BDA description page.

2.3.1. BDA Applications list

Returns the list of BDA Apps

Service name	BDA App list		
Resource Path	/api/proxy/bundle/bdas		
Type	HTTP GET		
Data input (Parameters/Body)			
Model	**Format**	**Attributes**	**Example**
-	-	-	-
Data output			
Model	**Format**	**Attributes**	**Example**
-	-	-	-

Table 21 - BDA App list service

2.3.2. BDA Application details

Returns the detail of a BDA App (incorporating all its assets)

Service name	BDA App details
Resource Path	/api/proxy/bundle/bdas/<id>
Type	HTTP GET

[11] https://www.eng.it/en/case-studies/alida-per-migliorare-la-flessibilita-reattivita-del-business

Data input (Paramenter/Body)			
Model	**Format**	**Attributes**	**Example**
id	String	Id:String	102
Data output			
Model	**Format**	**Attributes**	**Example**
-	-	-	-

Table 22 - BDA App details service

2.3.3. BDA services list

Returns the list of BDA Service

Service name	BDA services list
Resource Path	/api/platform/services
Type	HTTP GET

Data input (Paramenter/Body)			
Model	**Format**	**Attributes**	**Example**
-	-	-	-
Data output			
Model	**Format**	**Attributes**	**Example**
-	-	-	-

Table 23 - BDA services list

2.3.4. BDA service detail

Returns the detail of a BDA Services

Service name	BDA service detail
Resource Path	/ api/platform/services/<id>
Type	HTTP GET

Data input (Paramenter/Body)			
Model	**Format**	**Attributes**	**Example**
id	String	Id:String	102
Data output			
Model	**Format**	**Attributes**	**Example**
-	-	-	-

Table 24 - BDA service detail

2.3.5. Machine Learning model list

Returns the list of own, own team and public ML / DP Models

Service name	Machine Learning model list
Resource Path	/api/proxy/bundle/models
Type	HTTP GET

Data input (Paramenter/Body)			
Model	**Format**	**Attributes**	**Example**
-	-	-	-
Data output			
model	format	attribute	example
-	-	-	-

Table 25 - Machine Learning model list service

2.3.6. Machine Learning model details

Returns the detail of the ML / DP Model

Service name	Machine Learning model details		
Resource Path	/api/proxy/bundle/models/<id>		
Type	HTTP GET		
Data input (Paramenter/Body)			
Model	**Format**	**Attributes**	**Example**
id	String	Id:String	102
Data output			
Model	**Format**	**Attributes**	**Example**
-	-	-	-

Table 26 - Machine Learning model details service

2.3.7. Dataset list

Returns the list of own, own team and public datasets

Service name	Dataset list		
Resource Path	/api/proxy/bundle/datasets		
Type	HTTP GET		
Data input (Paramenter/Body)			
Model	**Format**	**Attributes**	**Example**
-	-	-	-
Data output			
Model	**Format**	**Attributes**	**Example**
-	-	-	-

Table 27 - Dataset list service

2.3.8. Dataset detail

Returns the detail of a Dataset

Service name	Dataset detail		
Resource Path	/api/proxy/bundle/datasets/<id>		
Type	HTTP GET		
Data input (Paramenter/Body)			
Model	**Format**	**Attributes**	**Example**
id	String	Id:String	102
Data output			
Model	**Format**	**Attributes**	**Example**
-	-	-	-

Table 28 - Dataset detail service

2.3.9. Data storage list

Returns the list of own, own team and public Data Storage

Service name	Data storage list		
Resource Path	/api/proxy/bundle/datasetSources		
Type	HTTP GET		
Data input (Paramenter/Body)			
Model	**Format**	**Attributes**	**Example**
-	-	-	-
Data output			
Model	**Format**	**Attributes**	**Example**
-	-	-	-

Table 29 - Data storage list service

2.3.10. Data storage details

Returns the detail of the Data Storage

Service name	Data storage details		
Resource Path	/api/proxy/bundle/datastorage/<id>		
Type	HTTP GET		
Data input (Paramenter/Body)			
Model	**Format**	**Attributes**	**Example**
id	String	Id:String	102
Data output			
Model	**Format**	**Attributes**	**Example**
-	-	-	-

Table 30 - Data storage details service

2.3.11. BDA scheduled runs list

Returns the list of running schedules for BDA App

Service name	BDA scheduled runs list		
Resource Path	/api/proxy/scdf/tasks/schedules/instances/<wfName>		
Type	HTTP GET		
Data input (Paramenter/Body)			
Model	**Format**	**Attributes**	**Example**
-	-	-	-
Data output			
Model	**Format**	**Attributes**	**Example**
-	-	-	-

Table 31 - BDA scheduled runs list service

2.3.12. BDA run application command

Schedule the run of a BDA App

Service name	BDA run application		
Resource Path	/ api/proxy/scdf/tasks/schedules? scheduleName=<name>& taskDefinitionName=<wfname>& arguments=& properties=scheduler.cron.expressio		
Type	HTTP POST		
Data input (Paramenters)			
Model	**Format**	**Attributes**	**Example**
Schedule Name	Plain text	scheduleName	-
Task Definition Name	Plain text	taskDefinitionName	
Arguments	Plain text	arguments	
Properties	Plain text	properties	
Data output			
Model	**Format**	**Attributes**	**Example**
-	-	-	-

Table 32 - BDA run application service

2.3.13. BDA application stop

Delete the scheduling of a previously created run

Service name	2.3.2. BDA application stop		
Resource Path	/api/proxy/scdf/tasks/schedules? scheduleName=<name> &taskDefinitionName=<wfname> &arguments= &properties=scheduler.cron.expression		
Type	HTTP POST		
Data input (Paramenters)			
Model	**Format**	**Attributes**	**Example**
Schedule Name	Plain text	scheduleName	-
Task Definition Name	Plain text	taskDefinitionName	
Arguments	Plain text	arguments	
Properties	Plain text	properties	
Model	**Format**	**Attributes**	**Example**
-	-	-	-

Table 33 - BDA application stop service

2.3.14. BDA application delete

Delete the scheduling of a previously created run

Service name	BDA application delete		

Resource Path	/ api/proxy/scdf/tasks/schedules?
	scheduleName=<name>&
	taskDefinitionName=<wfname>&
	arguments=&
	properties=scheduler.cron.expressio		

Type	HTTP DELETE		
Data input (Paramenters)			
Model	**Format**	**Attributes**	**Example**
Schedule Name	Plain text	scheduleName	-
Task Definition Name	Plain text	taskDefinitionName	
Arguments	Plain text	arguments	
Properties	Plain text	properties	
Data output			
Model	**Format**	**Attributes**	**Example**
-	-	-	-

Table 34 - BDA application delete service

3. Conclusions

This document describes the API services for third-party users. Firstly, a technological context has been given, describing the technologies involved in the supply of the APIs. After having described the context, the individual services have been listed, providing the information which represents the current state of their design. The APIs are the reflection of the components and services that make up the entire AgriBIT platform and on the other hand their design is the tool that guides the development of the components; thus, this document gives both the guidelines for coordinating the activities development and also a general idea of the components and the software that will be delivered by the project. Along the design and integration phase, the APIs will be updated and revised if necessary.

List of Abbreviations

Abbreviation	Explanation/Definition
OAS	Open API Specification
WMS	Web Map Service
WFS	Web Feature Service
REST	REpresentational State Transfer
JSON	JavaScript Object Notation
BDA	Big Data Analytics
HTTP	HyperText Transfer Protocol
API	Application Program Interface
YAML	Yet Another Markup Language
OGC	Open Geospatial Consortium
URL	Uniform Resource Locator
GUI	Graphical User Interface

Bibliography

[1] S. Mishra, „Emerging Technologies—Principles and Applications in Precision Agriculture." Data Science in Agriculture and Natural Resource Management,“ Singapore, Springer, 2022, pp. 31-53.

[2] F. R. Thomas, „Architectural styles and the design of network-based software architectures,“ University of California, Irvine, 2000.

Internal Deliverable Review Form

Project Acronym	AgriBIT
Project Title	Artificial intelliGence applied to pRecision farmIng By the use of GNSS and Integrated Technologies
Grant Agreement number	101004259
Call	SU-SPACE-EGNSS-3
Funding Scheme	Innovation Action (IA)
Project duration	36 Months

Document Information			
Deliverable:	D2.4 - AgriBIT Open APIs and interfaces		
Work Package:	WP2	**Task:**	T2.3
Date of Review:	30/06/2022		
Internal Reviewer:	Ifigeneia-Maria Tsioutsia (AgroApps)		
Classification:	Consortium Only		

Topic	Answer	IF "No", classify as "Major" or "Minor" issues	Comments

1. Is the content and structure of the deliverable in accordance with the DoA?	[✓ Yes] No		
2. Is the content of the deliverable scientifically relevant?	[✓ Yes] No N/A		
3. Is the content of the deliverable useful for the subsequent work on the project?	[✓ Yes] No N/A		
4. Is the deliverable suitable to be submitted to the EC?	[✓ Yes] No		
If not:			
4.1. Does it need formatting adjustments?	Yes [✓ No]		
4.2. Does it need content adjustments?	[✓ Yes] No	**Major**	Please check the content of the documentation, because some sentences or paragraphs needs to be revised.
4.3. Does it need to be significantly refined (e.g. content improvement, structure changes, etc.)?	Yes [✓ No]		
Additional comments			
➔ Abbreviations, i.e. WFS, WMS, API, PA, HTTP, etc. : When an abbreviation is firstly mentioned in the text, it is preferable to add the full text (explanation). ➔ It would be great if you could connect/ correlate the deliverable with other deliverables (make a reference).			