HYDROUSA

H2020-CIRC-2-2017

Water in the context of circular economy

Full project title:

Demonstration of water loops with innovative regenerative business models for the Mediterranean region

Deliverable: D32

Relative Number in WP D5.6

Data Repository

Due date of deliverable: 30/06/2020

Actual submission date: 28/02/2022

This project has received funding from the European Union's Horizon 2020 Research and Innovation Programme under Grant Agreement No 776643

This project has received funding from the European Union's Horizon 2020 Research and Innovation Programme under Grant Agreement No 776643

DOCUMENT INFORMATION

Deliverable	**Number**	D5.6	**Title:**	Data repository
Work Package	**Number**	WP5	**Title:**	Monitoring of the demonstration sites and ICT integration

Due date of deliverable	**Contractual**	M24	**Actual**	M44
Version number	1.2			
Format	MS Office Word document			
Creation date	20/12/2021			
Version date	28/02/2022			
Type	☒ R	☐ DEM	☐ DEC	☐ OTHER ☐ ETHICS
Dissemination Level	☒ PU Public		☐ CO Confidential	
Rights	Copyright "HYDROUSA Consortium". During the drafting process, access is generally limited to the HYDROUSA Partners.			

Responsible authors	**Name:**	Zisis Tsiropoulos Evangelos Skoubris	**E-mail:**	tsiropoulos@agenso.gr eskoubris@agenso.gr
	Partner:	AGENSO	**Phone:**	+30 2109234473

Brief Description	This document describes the HYDROUSA's data repository, analysing its architecture and infrastructure, the components design, the authentication mechanisms and the Application Programming Interface (API).
Keywords	Data repository, Application Programming Interface

Version log			
Rev. No.	**Issue Date**	**Modified by**	**Comments**
1.0	20/12/2021	Zisis Tsiropoulos Evangelos Skoubris	First draft of the deliverable
1.1	25/12/2021	Simos Malamis, Eleni Nyktari	Comments from the Coordination Team
1.2	28/02/2022	Zisis Tsiropoulos Evangelos Skoubris	Final version of the deliverable

This project has received funding from
the European Union's Horizon 2020
Research and Innovation Programme
under Grant Agreement No 776643

TABLE OF CONTENTS

LIST OF FIGURES

EXECUTIVE SUMMARY

This report presents the architecture of data repository designed and developed for the purposes of hosting all the readings collected from sensor grids of the HYDROUSA project. Additionally, the open source software components used for the data repository's development is explained, while the API developed for the purpose of easing the data transfer between the repository, the open source platform and the models and tools that will be used for the evaluation of the HYDROUSA systems in WP 6 are described in detail.

More specifically, in the first chapter there is a brief introduction regarding the deliverable.

In the second chapter of the deliverable, the specifications of the server where the data repository is hosted are presented in detail.

The third chapter describes the open-source technologies used in the design and development of the data repository, along with the provision for big data implementation (Apache Hadoop).

The structure of the data repository is presented at the fourth chapter, along with the graphical user interface (GUI) that was developed for making the data repository's functionalities more user friendly.

The fifth chapter of this deliverable is dedicated to examples of interaction with the external API, presenting all possible sets of outputs from the API after a user is successfully authenticated, as well as examples of outputs.

In the sixth chapter of this deliverable, several statistical data regarding the data repository and its operations are presented, as well as what measures are taken to ensure data consistency and integrity.

HYDROUSA has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 776643.

This project has received funding from
the European Union's Horizon 2020
Research and Innovation Programme
under Grant Agreement No 776643

ABBREVIATIONS

API	**Application Programming Interface**
CGI	**Common Gateway Interface**
CPU	**Central Processing Unit**
DDR	**Double Data Rate**
DR	**Data Repository**
ECC	**Error-correcting Code**
ET	**Evapotranspiration**
GA	**Grant Agreement**
GB	**Gigabyte**
GUI	**Graphical User Interface**
HDD	**Hard Disk Drive**
HTML	**HyperText Markup Language**
IMEI	**International Mobile Equipment Identity**
JSON	**JavaScript Object Notation**
KPI	**Key Performance Indicator**
OSS	**Open Source Software**
RAM	**Read Access Memory**
RDBMS	**Relational Database Management System**
SQL	**Structured Query Language**
UI	**User Interface**
URL	**Uniform Resource Locator**

1. INTRODUCTION

For the purposes of storing all the readings from the sensors installed in the HYDROUSA project, a data repository (DR) has been developed based on OSS technologies. This also will be serving as the intermediate between the sensors, the scientific models developed and the monitoring platform.

These readings come from either the sensors installed (soil moisture, water quality, etc.), the weather station or the nodes themselves, and include:

- Date & timestamp of the readings
- Node coordinates
- Node IMEI
- Soil moisture
- Water tank level
- Water temperature
- Water quality
    - TDS
    - pH
    - Turbidity
- Weather station data
    - Temperature
    - Humidity
    - Rainfall
    - Wind Speed
    - Wind Direction
    - Gust Speed
    - Light Intensity
    - UV Power
    - Pressure
    - Battery Level

The purpose of these data is to be used as input in the models that have been developed for the automation of irrigation or as a source of information for end users.

In the following chapters, this report will present the server's specifications, the technologies used for developing the DR, its architecture, examples from the external API output and statistical data for the DR.

2. SERVER SPECIFICATIONS

The DR was installed on a dedicated server used for the purposes of the HYDROUSA project alone.

The server has the specifications described below:

- CPU: AMD Opteron™ (8-core)
- RAM: 32GB DDR3 ECC
- HDD: 2x 2TB SATA II
- Monthly bandwidth: Unmetered data

For complying with the open-source spirit the server's OS is CentOS (64-bit version), a Linux distribution that provides a free and open-source community-supported computing platform.

It is obvious from the specifications above, that the server is more than capable of handling the volume of the current infrastructure (see 3.5.1 for more details), while being also capable of handling whatever future needs might arise.

3. DATA REPOSITORY ARCHITECTURE

In this chapter, the architecture behind the DR that was built by AGENSO will be explained.

Using OSS for the implementation of the HYDROUSA DR was a conscious decision. All the OSS components used for the development of the DR are described in the following sections.

3.1. Apache HTTP Server[1]

The Apache HTTP Server, colloquially called Apache, is a free and open-source cross-platform web server software, released under the terms of Apache License 2.0. Apache is developed and maintained by an open community of developers under the auspices of the Apache Software Foundation.

The vast majority of Apache HTTP Server instances run on a Linux distribution, but current versions also run on Microsoft Windows, OpenVMS, and a wide variety of Unix-like systems. Past versions also ran on NetWare, OS/2 and other operating systems, including ports to mainframes.

Originally based on the NCSA HTTPd server, development of Apache began in early 1995 after work on the NCSA code stalled. Apache played a key role in the initial growth of the World Wide Web, quickly overtaking NCSA HTTPd as the dominant HTTP server. In 2009, it became the first web server software to serve more than 100 million websites. As of January 2021, Netcraft estimated that Apache served 24.63% of the million busiest websites, while Nginx served 23.21% and Microsoft is in third place at 6.85% (for some of Netcraft's other stats Nginx is ahead of Apache), while according to W3Techs, Apache is ranked first at 35.0% and Nginx second at 33.0% and Cloudflare Server third at 17.3%.

Apache supports a variety of features, many implemented as compiled modules which extend the core functionality. These can range from authentication schemes to supporting server-side programming languages such as Perl, Python, Tcl and PHP. Popular authentication modules include mod_access, mod_auth, mod_digest, and mod_auth_digest, the successor to mod_digest. A sample of other features include Secure Sockets Layer and Transport Layer Security support (mod_ssl), a proxy module (mod_proxy), a URL rewriting module (mod_rewrite), custom log files (mod_log_config), and filtering support (mod_include and mod_ext_filter).

Popular compression methods on Apache include the external extension module, mod_gzip, implemented to help with reduction of the size (weight) of web pages served over HTTP. ModSecurity is an open-source intrusion detection and prevention engine for Web applications. Apache logs can be analyzed through a Web browser using free scripts, such as AWStats/W3Perl or Visitors.

Virtual hosting allows one Apache installation to serve many different websites. For example, one computer with one Apache installation could simultaneously serve example.com, example.org, test47.test-server.example.edu, etc.

[1] https://en.wikipedia.org/wiki/Apache_HTTP_Server

Apache features configurable error messages, DBMS-based authentication databases, and content negotiation and supports several GUIs.

It supports password authentication and digital certificate authentication. Because the source code is freely available, anyone can adapt the server for specific needs, and there is a large public library of Apache add-ons.

A more detailed list of features is provided below:

- Loadable Dynamic Modules
- Multiple Request Processing modes (MPMs) including Event-based/Async, Threaded and Prefork.
- Highly scalable (easily handles more than 10,000 simultaneous connections)
- Handling of static files, index files, auto-indexing and content negotiation
- .htaccess per-directory configuration support
- Reverse proxy with caching
- Load balancing with in-band health checks
- Multiple load balancing mechanisms
- Fault tolerance and Failover with automatic recovery
- WebSocket, FastCGI, SCGI, AJP and uWSGI support with caching
- Dynamic configuration
- TLS/SSL with SNI and OCSP stapling support, via OpenSSL or wolfSSL.
- Name- and IP address-based virtual servers
- IPv6-compatible
- HTTP/2 support
- Fine-grained authentication and authorization access control
- gzip compression and decompression
- URL rewriting
- Headers and content rewriting
- Custom logging with rotation
- Concurrent connection limiting
- Request processing rate limiting
- Bandwidth throttling
- Server Side Includes
- IP address-based geolocation
- User and Session tracking
- WebDAV
- Embedded Perl, PHP and Lua scripting
- CGI support
- public_html per-user web-pages
- Generic expression parser
- Real-time status views
- FTP support (by a separate module)

3.2. MySQL[2]

MySQL is an open-source Relational Database management System (RDBMS). A relational database organizes data into one or more data tables in which data types may be related to each other; these relations help structure the data. SQL is a language programmers use to create, modify and extract data from the relational database, as well as control user access to the database. In addition to relational databases and SQL, an RDBMS like MySQL works with an operating system to implement a relational database in a computer's storage system, manages users, allows for network access and facilitates testing database integrity and creation of backups.

MySQL is free and open-source software under the terms of the GNU General Public License, and is also available under a variety of proprietary licenses. MySQL was owned and sponsored by the Swedish company MySQL AB, which was bought by Sun Microsystems (now Oracle Corporation). In 2010, when Oracle acquired Sun, Widenius forked the open-source MySQL project to create MariaDB.

MySQL has stand-alone clients that allow users to interact directly with a MySQL database using SQL, but more often, MySQL is used with other programs to implement applications that need relational database capability. MySQL is a component of the LAMP web application software stack (and others), which is an acronym for Linux, Apache, MySQL, Perl/PHP/Python. MySQL is used by many database-driven web applications, including Drupal, Joomla, phpBB, and WordPress. MySQL is also used by many popular websites, including Facebook, Flickr, MediaWiki, Twitter, and YouTube.

MySQL is written in C and C++. Its SQL parser is written in yacc, but it uses a home-brewed lexical analyzer. MySQL works on many system platforms, including AIX, BSDi, FreeBSD, HP-UX, ArcaOS, eComStation, IBM i, IRIX, Linux, macOS, Microsoft Windows, NetBSD, Novell NetWare, OpenBSD, OpenSolaris, OS/2 Warp, QNX, Oracle Solaris, Symbian, SunOS, SCO OpenServer, SCO UnixWare, Sanos and Tru64. A port of MySQL to OpenVMS also exists.

The MySQL server software itself and the client libraries use dual-licensing distribution. They are offered under GPL version 2, or a proprietary license.

Support can be obtained from the official manual. Free support additionally is available in different IRC channels and forums. Oracle offers paid support via its MySQL Enterprise products. They differ in the scope of services and in price. Additionally, a number of third party organisations exist to provide support and services.

MySQL has received positive reviews, and reviewers noticed it “performs extremely well in the average case” and that the “developer interfaces are there, and the documentation (not to mention feedback in the real world via Web sites and the like) is very, very good”. It has also been tested to be a “fast, stable and true multi-user, multi-threaded SQL database server”.

[2] https://en.wikipedia.org/wiki/MySQL

3.3. PHP[3]

PHP is a general-purpose scripting language geared towards web development. It was originally created by Danish-Canadian programmer Rasmus Lerdorf in 1994. The PHP reference implementation is now produced by The PHP Group. PHP originally stood for Personal Home Page, but it now stands for the recursive initialism PHP: Hypertext Preprocessor.

PHP code is usually processed on a web server by a PHP interpreter implemented as a module, a daemon or as a CGI executable. On a web server, the result of the interpreted and executed PHP code – which may be any type of data, such as generated HTML or binary image data – would form the whole or part of an HTTP response. Various web template systems, web content management systems, and web frameworks exist which can be employed to orchestrate or facilitate the generation of that response. Additionally, PHP can be used for many programming tasks outside the web context, such as standalone graphical applications and robotic drone control. PHP code can also be directly executed from the command line.

The standard PHP interpreter, powered by the Zend Engine, is free software released under the PHP License. PHP has been widely ported and can be deployed on most web servers on a variety of operating systems and platforms.

The PHP language evolved without a written formal specification or standard until 2014, with the original implementation acting as the de facto standard which other implementations aimed to follow. Since 2014, work has gone on to create a formal PHP specification.

W3Techs reports that, as of April 2021, “PHP is used by 79.2% of all the websites whose server-side programming language we know.”

3.4. Swagger[4]

Swagger is an Interface Description Language for describing RESTful APIs expressed using JSON. Swagger is used together with a set of OSS tools to design, build, document, and use RESTful web services. Swagger includes automated documentation, code generation (into many programming languages), and test-case generation.

Swagger's open-source tooling usage can be broken up into different use cases: development, interaction with APIs, and documentation.

3.4.1. Developing APIs

When creating APIs, Swagger tooling may be used to automatically generate an Open API document based on the code itself. This embeds the API description in the source code of a project and is informally called code-first or bottom-up API development.

Alternatively, using Swagger Codegen, developers can decouple the source code from the Open API document, and generate client and server code directly from the design. This makes it possible to defer the coding aspect.

[3] https://en.wikipedia.org/wiki/PHP
[4] https://en.wikipedia.org/wiki/Swagger_(software)

3.4.2. Interacting with APIs

Using the Swagger Codegen project, end users generate client SDKs directly from the OpenAPI document, reducing the need for human-generated client code. As of August 2017, the Swagger Codegen project supported over 50 different languages and formats for client SDK generation.

3.4.3. Documenting APIs

When described by an OpenAPI document, Swagger open-source tooling may be used to interact directly with the API through the Swagger UI. This project allows connections directly to live APIs through an interactive, HTML-based user interface. Requests can be made directly from the UI and the options explored by the user of the interface.

3.5. Apache Hadoop[5]

Apache Hadoop is a collection of open-source software utilities that facilitates using a network of many computers to solve problems involving massive amounts of data and computation. It provides a software framework for distributed storage and processing of big data using the MapReduce programming model. Hadoop was originally designed for computer clusters built from commodity hardware, which is still the common use. It has since also found use on clusters of higher-end hardware. All the modules in Hadoop are designed with a fundamental assumption that hardware failures are common occurrences and should be automatically handled by the framework.

The core of Apache Hadoop consists of a storage part, known as Hadoop Distributed File System (HDFS), and a processing part which is a MapReduce programming model. Hadoop splits files into large blocks and distributes them across nodes in a cluster. It then transfers packaged code into nodes to process the data in parallel. This approach takes advantage of data locality, where nodes manipulate the data they have access to. This allows the dataset to be processed faster and more efficiently than it would be in a more conventional supercomputer architecture that relies on a parallel file system where computation and data are distributed via high-speed networking.

The base Apache Hadoop framework is composed of the following modules:

- Hadoop Common: contains libraries and utilities needed by other Hadoop modules
- Hadoop Distributed File System (HDFS): a distributed file-system that stores data on commodity machines, providing very high aggregate bandwidth across the cluster;
- Hadoop YARN: introduced in 2012, it is a platform responsible for managing computing resources in clusters and using them for scheduling users' applications;
- Hadoop MapReduce: an implementation of the MapReduce programming model for large-scale data processing.
- Hadoop Ozone: introduced in 2020, it is an object store for Hadoop

[5] https://en.wikipedia.org/wiki/Apache_Hadoop

The term Hadoop is often used for both base modules and sub-modules and also the ecosystem, or collection of additional software packages that can be installed on top of or alongside Hadoop, such as Apache Pig, Apache Hive, Apache HBase, Apache Phoenix, Apache Spark, Apache ZooKeeper, Cloudera Impala, Apache Flume, Apache Sqoop, Apache Oozie, and Apache Storm.

Apache Hadoop's MapReduce and HDFS components were inspired by Google papers on MapReduce and Google File System.

The Hadoop framework itself is mostly written in the Java programming language, with some native code in C and command line utilities written as shell scripts. Though MapReduce Java code is common, any programming language can be used with Hadoop Streaming to implement the map and reduce parts of the user's program. Other projects in the Hadoop ecosystem expose richer user interfaces.

3.5.1. About the Apache Hadoop Implementation

At the time of writing this report, the Apache Hadoop was not entirely implemented.

As explained in the above section it is a solution for big data, but currently the datasets from the readings collected from sensor grids (see Task 5.4 for more details) are not large in size – quite the opposite in fact.

Each reading, sends exactly 306 bytes of data. For receiving each reading and before storing it, some additional data are added in each reading, such as a timestamp (for example, October 25th 2021 at 22:45 UTC is formatted as 2021-10-25 T 22:45 UTC) and the IMEI of the sensor sending the reading (for example, 123509243872123).

All in all, assuming that every sensor is updated in the default interval (i.e, 30 minutes), the daily sum of the datasets from all 200 sensors amounts to 200MB, something that makes the implementation of Apache Hadoop unnecessary for the time being.

The infrastructure for its implementation is complete, and the transition will take place when the size of the datasets' size exceeds the capabilities of the current implementation.

The usage of the combination of the above technologies, has helped with keeping with one of the major objectives of the project, and that is low cost and effective technologies.

4. DATA REPOSITORY STRUCTURE

The DR's main purpose is to provide an API for the HYDROUSA products, serving results by supported scientific models and receiving/providing measurements from the connected sensors.

The main components of the DR, include the following:

- the basic user authentication
- the registration of apps
- the API endpoints to be used for fetching results and for accepting the pings from the hardware sensors
- the scientific models

The following figure shows the relations/connections between all the aforementioned components.

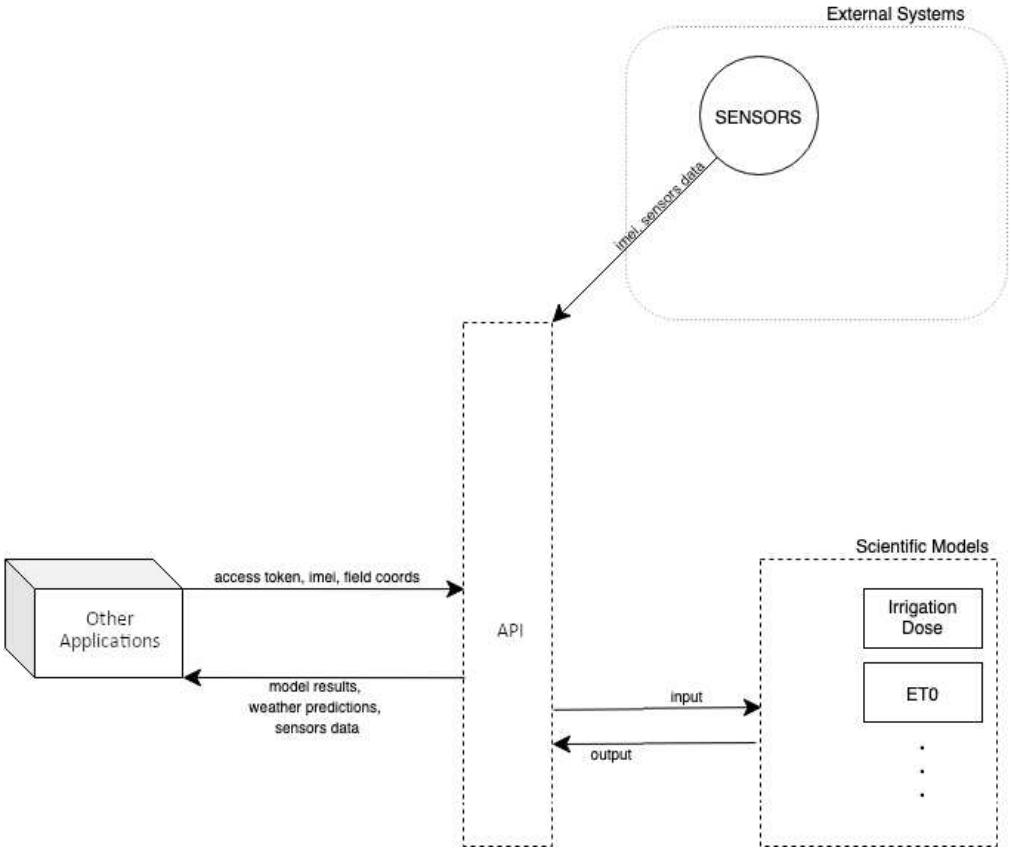

Figure 4.1 Relations/connections between all the DR components

The DR uses a MySQL database (as mentioned in the previous chapter), hosted in the same server. The database is accessible remotely using a MySQL client such as Jetbrains Datagrip or by using phpMyAdmin. The main use of this relational database is to store the users of the application along with the access tokens.

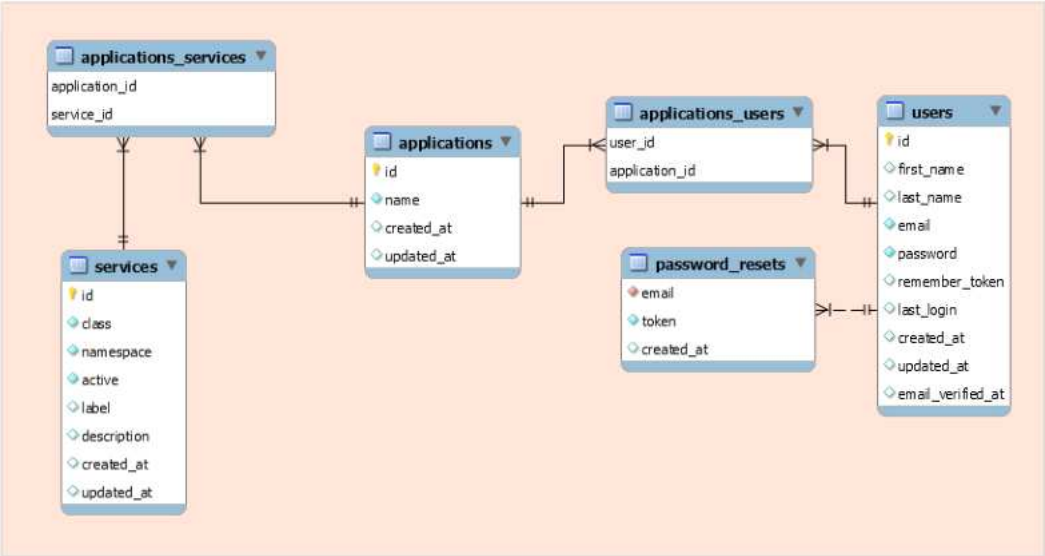

Figure 4.2 Table relationships in the MySQL database

In the DR the following services have been developed and implemented:

- Sensors' Data: this is the service that is responsible for receiving and storing the data originating from the installed sensors that are attached to the different nodes
- Evapotranspiration: this is the service that is responsible for calculating the evapotranspiration. It is calculated using measurements from the attached weather station.
- Irrigation Dose: this is the service that is responsible for calculating the irrigation dose for each field or plot. The calculation is done combining the ET and the measurements from the attached soil moisture sensor.
- Weather Forecast: this is the service that is responsible for receiving the data for the following days' weather forecast. The coordinates used to receive the weather forecast, are those of the node's position.
- IMEI Updater Instructor: this is the service that is responsible for the bidirectional communication between the node and the DR. IMEI is a unique identifier (number) of each node, as the nodes are connected to the cloud through cellular network providers.

For the purposes of making the DR more user-friendly, a dedicated GUI was designed and developed, as can be seen in the figure below (Figure 4.3).

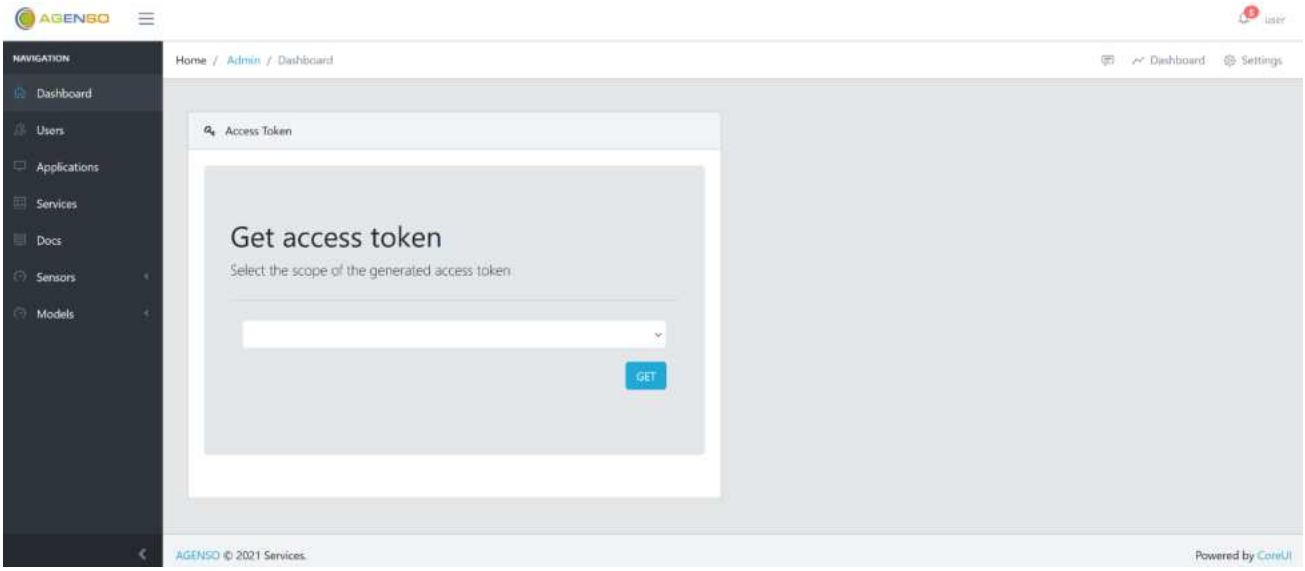

Figure 4.3 Dashboard of the Data Repository GUI

In the central page, users can create access tokens for the DR's API. With this token, every application designed to retrieve data from the API can do so. This token is a security mechanism, used to authenticate each application that has access to the DR's data, aiming to avoid unauthorized data retrieval. For additional security, each token is time-limited and when it does expire, a new token must be issued.

Additionally, there's a menu on the left of the screen with the following choices:

- **Users**
  A user with administrative privileges, can manage the users having access to the DR GUI, as well as their privileges.

- **Applications**
  The user can see what applications have already been created (see Figure 4.4) and manage them (view/edit/delete). In essence, these applications are the ones that have already been authorized with a token and have access to the DR's API. In addition, users can include additional applications or manage the tokens (edit/delete them or renew those expiring).

Figure 4.4 Overview of Applications page

Additionally, the user can create an application. After pressing the designated button, a name must be given to the application, and subsequently, the services (one or more) that will be run on the application must be chosen from a list (see Figure 4.5).

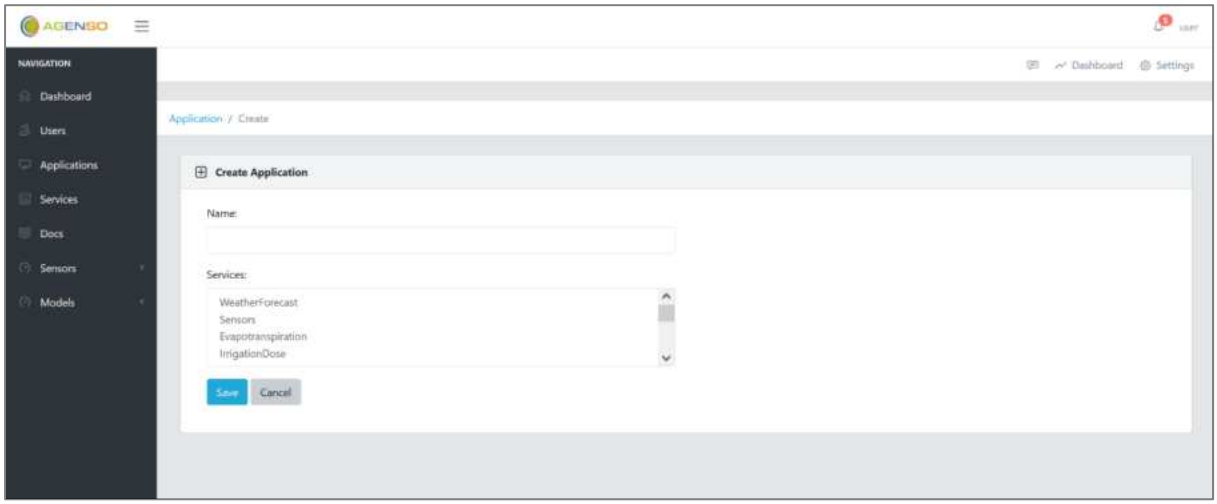

Figure 4.5 Create application feature

- **Services**

  The user can manage the existing applications (view/edit/delete), as well as create a new service (see Figure 4.6). For the purposes of the HYDROUSA project, the following services are up and running (a description of each service has been given before):

  - WeatherForecast
  - Sensors
  - EvapotranspirationInstructor
  - IrrigationDoseInstructor
  - ImeiUpdaterInstructor

  Each one of the aforementioned services, can be modified and/or deleted, if necessary.

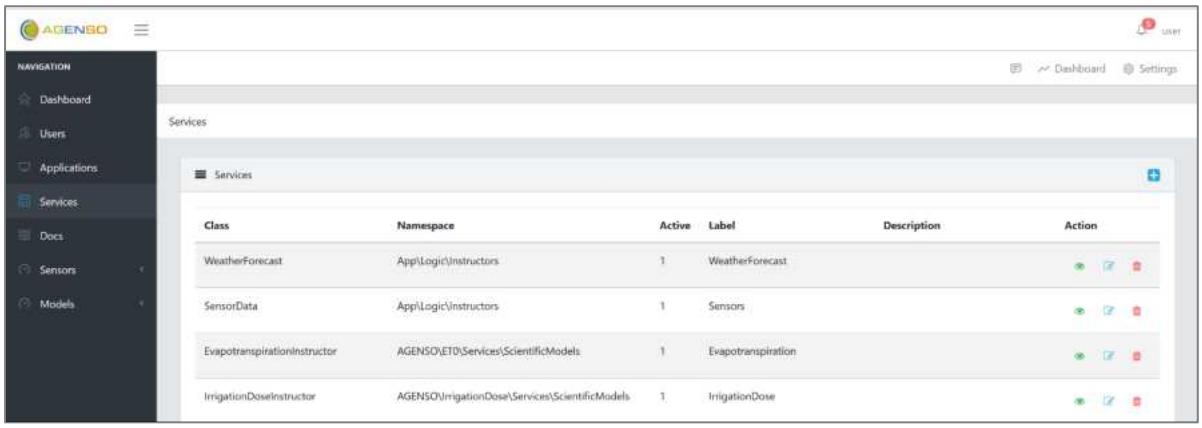

Figure 4.6 Overview of Services page

- **Docs**

  This section (see Figure 4.7) has the form of a wiki page, in which major and minor components of the DR are explained in detail.

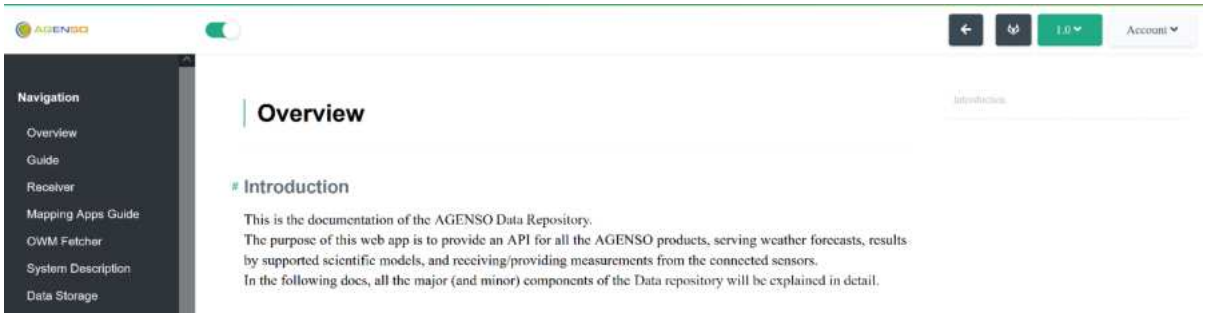

Figure 4.7 Overview of Docs page

- **Sensors**

  In this section, the user can get information from a sensor by using a query of its paired IMEI (see Figure 4.8). Depending on the types of sensors that are attached to the queried node, the relevant data are shown. For example, if a weather station is attached to the node, the following data are being presented:

  - Temperature
  - Humidity
  - Rainfall
  - Wind Speed
  - Wind Direction
  - Gust Speed
  - Light Intensity
  - UV Power
  - Pressure
  - Battery Level

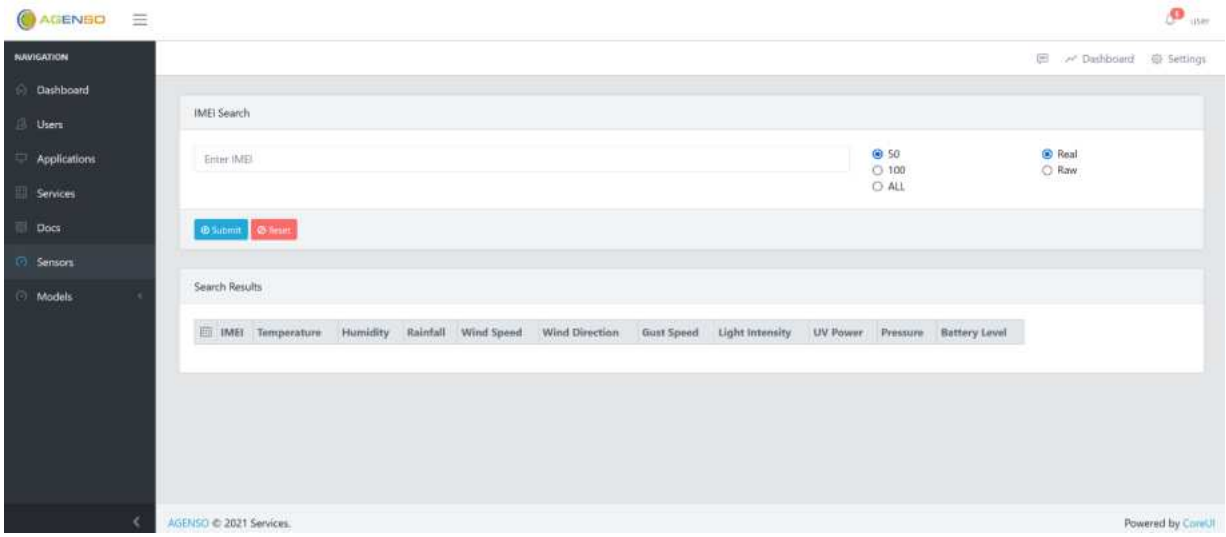

Figure 4.8 Overview of sensors page

- **Models**

  Users can view/edit all the available models running on the DR, if necessary.

This project has received funding from
the European Union's Horizon 2020
Research and Innovation Programme
under Grant Agreement No 776643

5. EXAMPLES FROM THE EXTERNAL API

The external API has been developed so that all applications that have an authorized token from the DR can receive data.

As described in a previous chapter (see Data Repository Architecture), Swagger is used in order to retrieve data from the external API. The API was made using OpenAPI 3.0 Specifications and can export the data in various formats as JSON, XML, CSV and YAML.

When somebody accesses the external API UI, they are greeted with the screen shown below:

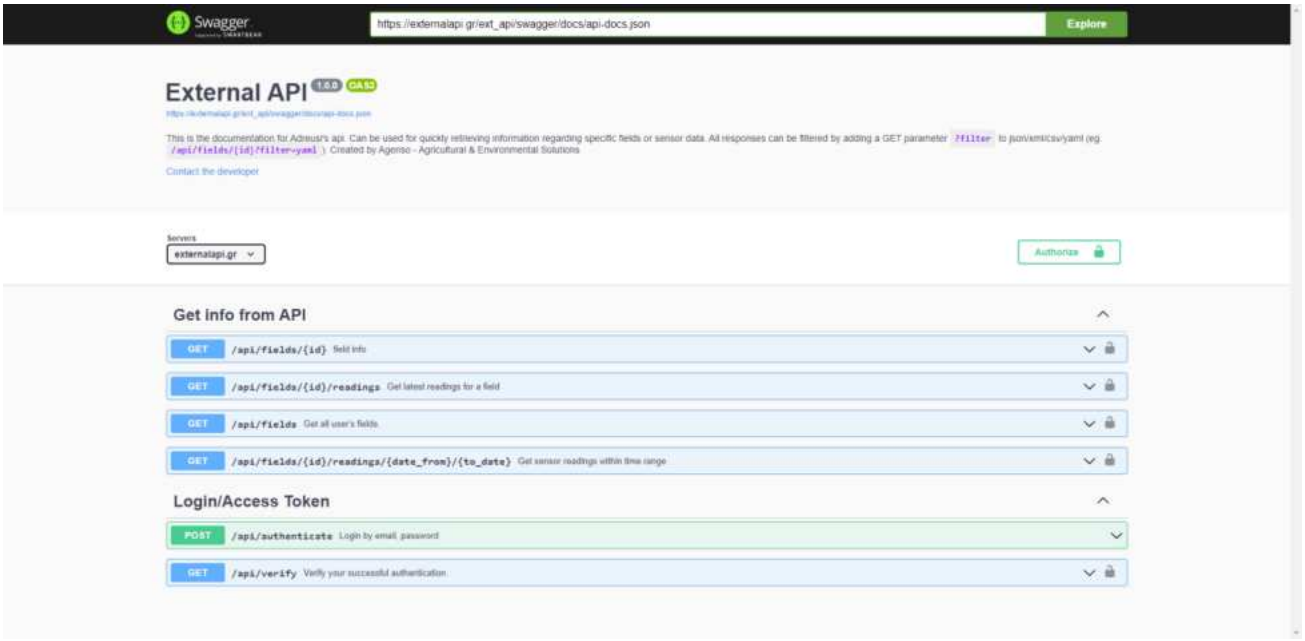

Figure 5.1 Overview of the External API UI

The choices given to the user, include a section where they can retrieve information from the fields through the API ("Get info from the API" section), as well as a section dealing with authorization and tokens ("Login/Access Token" section).

5.1. Login/Access Token

The user must first enter their login data, and after successfully logging in, they receive a unique access token, which they must use in order to authorize and continue with API queries.

What follows below is the user's input and the respective Curl input:

```
{
  "email": "test@email.gr",
  "password": "pswrd"
}
```

Curl

```
curl -X 'POST' \
  'https://externalapi.gr/ext_api/authenticate \
  -H 'accept: application/json' \
  -H 'Content-Type: application/json' \
  -H 'X-CSRF-TOKEN: ' \
  -d '{
  "email": "test@email.gr",
  "password": "pswrd"
}'
```

Request URL

```
https://externalapi.gr/ext_api/authenticate
```

In case of wrong credentials, the user gets the following message and headers:

Response body

```
{
  "message": "Invalid Credentials"
}
```

Response headers

```
cache-control: private,must-revalidate
content-encoding: br
content-type: application/json
date: Fri,22 Oct 2021 10:53:57 GMT
expires: -1
pragma: no-cache
server: nginx
vary: Accept-Encoding
x-ratelimit-limit: 60
x-ratelimit-remaining: 59
```

In case the user enters the correct credentials, they get a message informing them about the success of the procedure, and returns the following:

Response body

```
{
  "message": "success",
  "access_token": "GiI1wi8VqPO7vuEFOeFbDng8wFwajQylSd9Fkco3pHCvWPpos93LFl1TZwsGFYjHHdURrVx6FJz80kf
4tV2xjbak82NzwI787MScV98mk2HT6F969CTfKUtcNVzyXJQVAr7kieSespVO86alGaqK02kxwOMPDq5EoiRRU1Nrp74nNnyMZ
HwVsLashPz8sge9EvEQCJJpx3TpM31oFjRuA1l1mDA3ZhtcMRClPmP8GqmlNZKMfPET9L823PLDVrUFpaSCdfTkwco2PBcvIsq
y1VQjlTehw8SqVAzXm4nzT9KI1MeDgJsd1hoIURIRAXoNLMTytSIt5ASnmgOOJsv0PH7fUsAMBDFSnXTMPWxPDDggTgFetEBkJ
3vLTobuqTmTSKJJfIuYPDHPyCLQtiSSMtfIqBAj3noWDlXq0CLUpwYk61j1eBVMjHokfiUQcC0jp5nEvTRy40IHJxSwbRm75Eh
OA8s8LPA8DmwV9ZGEAsnb6vz12BNcVR60UatHU0FG175rc7PzjwcdcnJdcHcP3z3VFnpN6zBbUHnBLfmUcM7D5ySO4FCsE9son
lRSW43BXAVSSZZlCRQngGg97zRbcE3dbIcKWUO71zNYquYW8JtGq5C2AIaeonf5KGDNhLsxm49V6iKTEA4kXlISOZipkKuVttr
lk2RqaHx1d8oBljqQJP62PiOGrwTZvjgGenJcWlwMs48M0I0utv7dI1L2aYrzmk65lhGsIAnbmkFSeekGwnN25uViirrFVQOLw
AlBoCMyCDHAodX1BMTEaFGe6xLba3oeOeCdo7g4IZBVsOVeC5cJ63AHS88smUP4fXeGcodxtc8V0pJysFliYpEUMp4tNbcLquI
6qbqhHJcB6iCSUQlQJvfXYSLIC8knKtk8taMMbX9ibGw9RQ4K0DZFkzN51UIc3j0gOMxUt6Jy1N6ef0RD2icOI9hQdCqpbVH1l
Wvqm2xilAxyx4Gmk7urb6CxVvVQS"
}
```

Response headers

```
cache-control: private,must-revalidate
content-encoding: br
content-type: application/json
date: Fri,22 Oct 2021 10:57:52 GMT
expires: -1
pragma: no-cache
server: nginx
```

```
vary: Accept-Encoding
x-ratelimit-limit: 60
x-ratelimit-remaining: 59
```

5.2. Get Info from API

5.2.1. Verification of Successful Authentication

The user can check whether they have successfully authenticated or not.

By entering their token in the designated field and executing the script, the user produces the following Curl input:

Curl

```
curl -X 'GET' \
  'https://externalapi.gr/ext_api/verify' \
  -H 'accept: application/json' \
  -H 'Authorization: Bearer
eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJhdWQiOiIxIiwianRpIjoiMGY0MzA0NWU0MmFmMzdkODM0M2U3NzNhOTM3ZTA3NWVlZjBmNTViNDlkNTYxMTRiMTQ2YWEwN2E4YWVjZTYxNzM5OGJjYzgyYjI1ODliZjkiLCJpYXQiOiIxNjM0OTAwMjcyLjE0NjI4MSIsIm5iZiI6IjE2MzQ5MDAyNzIuMTQ2MjkwIiwiZXhwIjoiMTYzNzU3ODY3MS45NDQ1NzQiLCJzdWIiOiIxNSIsInNjb3BlcyI6W119.NtMhlo0oZQRPRab3ZmPk2r3D4-
ej_wSAjLy5c48h6jfRBDAG31_cP4VjmzfskIzwGnrXzffPzFA0G8LvUUp83L8nKtNiLo15LI7dDHHZYg5p9SLeprGbBOeCCifpG-
PtenALhgMqj2cRM0E1zJ_cshAJgYLr2_t_vcdfJ_ErgGffuLBm28H9sx8IZ27NpaPZqGzMG8frxk3ttuhc5UTqDTnaFznjtXuzizrbLJ_Ssm-dsBSuuqebsDSEhWro6VWg1v-mdOF9WO8to1bpKApKbPJskrbC_sV-Ze27rvdLlVJ90fsiChzcixfMN-1_f-ozX7T6djbof8oe_yMhms0Wp3MgtvriDFdb6n-
cE0pN1_Mp1Lp1SxJ4eqtyNKclYVbmlngK72HxpDYDMumH1bkIKzl1_9W3ujExqy_gMPUk5Iwxy8PEBhQqqFc9lWVmmNtojaKHiBrzc1I8AUwKUn7Pnk11q93VEw7ZnLm6fmle3KpiaBTUCA8clV_7W3lJ0xp8Y5SWRcQkxQx2WhH4SFz-werbTtiAUiKyyfHWnT144xVJV--Nhf0Up1sjNvQElhm-6cw4u4HZAK_cqaQwkya415ZfryZ1f-bbnlZobIcTXPA05ds7MGMxJxpTvnhOqD1xYdZ84481QNlX-vtpHyAsVnmZ5lYmt7k' \
  -H 'X-CSRF-TOKEN: '
```

Request URL

```
https://externalapi.gr/ext_api/verify
```

In case of successful authentication, the user gets the following message and headers:

Response body


```
{
  "message": "You made it!",
  "user": {
    "id": 15,
    "first_name": "test",
    "last_name": "email",
    "email": " test@email.gr",
    "password": "$2y$10$dYzkLMSxxHJ.Rz5fTECiHt36nSelGXM6",
    "city": "Athens",
    "postal_code": null,
    "address": null,
    "phone_number": null,
    "email_verified_at": null,
    "role": 1,
    "remember_token": "OKHO7Gzej0dB7NSOLla9dFsZM5DGREHYNOKC82a",
    "created_at": "2020-04-22T21:11:08.000000Z",
    "updated_at": "2021-08-31T17:34:51.000000Z",
    "deleted_at": null,
    "notes": null,
```

```
    "api_access": 1,
    "userToken": "QBpXtghiZs213"
  }
}
```

Response headers

```
cache-control: private,must-revalidate
content-encoding: br
content-type: application/json
date: Fri,22 Oct 2021 11:02:05 GMT
expires: -1
pragma: no-cache
server: nginx
vary: Accept-Encoding,Authorization
x-ratelimit-limit: 60  x-ratelimit-remaining: 59
```

5.2.2. Retrieve Information for a Specific Field

The user can retrieve the basic information for a specific field[6].

After having successfully authenticated and by entering the field ID and executing the script, the user produces the following Curl input:

Curl

```
curl -X 'GET' \
  'https://externalapi.gr/ext_api/fields/50?filter=xml' \
  -H 'accept: application/json' \
  -H 'Accept-Encoding: application/json' \
  -H 'Authorization: Bearer
eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJhdWQiOiIxIiwianRpIjoiMGY0MzA0NWU0MmFmMzdkODM0M2U3NzNhOTM3ZTA3NWVlZjBmNTViNDlkNTYxMTRiMTQ2YWEwN2E4YWVjZTYxNzM5OGJjYzgyYjI1ODliZjkiLCJpYXQiOiIxNjM0OTAwMjcyLjE0NjI4MSIsIm5iZiI6IjE2MzQ5MDAMS45NDQ1NzQiLCJzdWIiOiIxNSIsInNjb3BlcyI6W119.NtMhlo0oZQRPRab3ZmPk2r3D4-
ej_wSAjLy5c48h6jfRBDAG31_cP4VjmzfskIzwGnrXzffPzFA0G8LvUUp83L8nKtNiLo15LI7dDHHZYg5p9SLeprGbBOeCCifpG-
PtenALhgMqj2cRM0E1zJ_cshAJgYLr2_t_vcdfJ_ErgGffuLBm28H9sx8IZ27NpaPZqGzMG8frxk3ttuhc5UTqDTnaFznjtXuzizrbLJ_Ssm-dsBSuuqebsDSEhWro6VWg1v-mdOF9WO8to1bpKApKbPJsk_sV-Ze27rvdLlVJ90fsiChzcixfMN-1_f-ozX7T6djbof8oe_yMhms0Wp3MgtvriDFdb6n-
cE0pN1_Mp1Lp1SxJ4eqtyNKclYVbmlngK72HxpDYDMumiDzzSYXpbHPBH1bkIKzl1_9W3ujExqy_gMPUk5Iwxy8PEBhQqqFc9lWVmmNtojaKHiBrzc1I8AUwKUn7Pnk11q93VEw7ZnLm6fmle3KpiaBTUCA8clV_7W3lJ0xp8Y5SWRcQkxQx2WhH4SFz-werbTtiAUiKyyfHWnT144xVJV--Nhf0Up1sjNvQElhm-6cw4u4HZAK_cqaQwkya415ZfryZ1f-bbnlZobIcTXPA05ds7MGMxJxpTvnhOqD1xYdZ84481QNlX-vtpHyAsVnmZ5lYmt7k' \
  -H 'X-CSRF-TOKEN: '
```

Request URL

```
https://externalapi.gr/ext_api/fields/50?filter=xml
```

In case the user enters a field ID that exists in the database, they get the following message and headers:

Response body

```
<?xml version="1.0" encoding="UTF-8"?>
  <results>
    <id>50</id>
    <label>Outside</label>
```

[6] Each field mentioned hereafter, corresponds to a specific node

```
    <area>Tinos</area>
    <coordinates>
      <lat/>
      <lng/>
    </coordinates>
    <timezone>Europe/Athens</timezone>
    <created_at>2020-11-27T14:52:07.000000Z</created_at>
  </results>
```

Response headers

```
cache-control: private,must-revalidate
content-encoding: br
content-type: application/xml
date: Fri,22 Oct 2021 11:25:47 GMT
expires: -1
pragma: no-cache
server: nginx
vary: Accept-Encoding,Authorization
x-ratelimit-limit: 60
x-ratelimit-remaining: 59
```

5.2.3. Get the Latest Readings from a Specific Field

The user can retrieve the latest readings from the sensor for a specific field.

After having successfully authenticated and by entering the field ID and executing the script, the user produces the following Curl input:

Curl

```
curl -X 'GET' \
  'https://externalapi.gr/ext_api/fields/50/readings?filter=xml' \
  -H 'accept: application/json' \
  -H 'Accept-Encoding: application/json' \
  -H 'Authorization: Bearer
eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJhdWQiOiIxIiwianRpIjoiMGY0MzA0NWU0MmFmMzdkODM0M2U3NzNhOTM3ZTA3NWVlZjBmNTViNDlkNTYxMTRiMTQ2YWEwN2E4YWVjZTYxNzM5OGJjYzgyYjI1ODliZjkiLCJpYXQiOiIxNjM0OTAwMjcyLjE0NjI4MSIsIm5iZiI6IjE2MzQ5MDAyNzIuMTQ2MjkwIidWIiOiIxNSIsInNjb3BlcyI6W119.NtMhlo0oZQRPRab3ZmPk2r3D4-ej_wSAjLy5c48h6jfRBDAG31_cP4VjmzfskIzwGnrXzffPzFA0G8LvUUp83L8nKtNiLo15LI7dDHHZYg5p9SLeprGbBOeCCifpG-
PtenALhgMqj2cRM0E1zJ_cshAJgYLr2_t_vcdfJ_ErgGffuLBm28H9sx8IZ27NpaPZqGzMG8frxk3ttuhc5UTqDTnaFznjtXuzizrbLJ_Ssm-dsBSuuqebsDSEhWro6VWg1v-mdOF9WO8to1bpKApKbPJskrbC_sV-Ze27rvdLlVJ90fsiChzcixfMN-1_f-ozX7T6djbof8oe_yMhms0Wp3MgtvriDFdb6n-
cE0pN1_Mp1Lp1SxJ4eqtyNKclYVbmlngK72HxpDYDMumiDzzSYXpbHPBH1bkIKzl1_9W3ujExqy_gMPUk5Iwxy8PEBhQqqFc9lWVmmNtojaKHiBrzc1I8AUwKUn7Pnk11q93VEw7ZnLm6fmle3KpiaBTUCA8clV_7W3lJ0xp8Y5SWRcQkxQx2WhH4SFz-werbTtiAUiKyyfHWnT144xVJV--Nhf0Up1sjNvQElhm-6cw4u4HZAK_cqaQwkya415ZfryZ1f-bbnlZobIcTXPA05ds7MGMxJxpTvnhOqD1xYdZ84481QNlX-vtpHyAsVnmZ5lYmt7k' \
  -H 'X-CSRF-TOKEN: '
```

Request URL

```
https://externalapi.gr/ext_api/fields/50/readings?filter=xml
```

In case the user enters a field ID that exists in the database, they get the following message and headers, containing the latest readings of the installed sensors at the node. For this example, temperature, humidity, daily and hourly rainfall, wind speed and direction, gust speed, light intensity and UV power, along with the timestamp are retrieved:

Response body


```
<?xml version="1.0" encoding="UTF-8"?>
  <results>
    <0>
      <node_label>Outside</node_label>
      <node_id>70</node_id>
      <sensor_id>616</sensor_id>
      <label>TEMPERATURE</label>
      <reading>23.40</reading>
      <unit>°C</unit>
      <timestamp>22/10/2021 14:15:05</timestamp>
    </0>
    <1>
      <node_label>Outside</node_label>
      <node_id>70</node_id>
      <sensor_id>617</sensor_id>
      <label>HUMIDITY</label>
      <reading>55.00</reading>
      <unit>%</unit>
      <timestamp>22/10/2021 14:15:05</timestamp>
    </1>
    <2>
      <node_label>Outside</node_label>
      <node_id>70</node_id>
      <label>Daily Rainfall</label>
      <reading>0.00</reading>
      <unit>mm</unit>
      <timestamp>21/10/2021 21:38:01</timestamp>
    </2>
    <3>
      <node_label>Outside</node_label>
      <node_id>70</node_id>
      <label>Hourly Rainfall</label>
      <reading>0.00</reading>
      <unit>mm</unit>
      <timestamp>22/10/2021 11:15:05</timestamp>
    </3>
    <4>
      <node_label>Outside</node_label>
      <node_id>70</node_id>
      <sensor_id>619</sensor_id>
      <label>WIND SPEED</label>
      <reading>5.04</reading>
      <unit>km/h</unit>
      <timestamp>22/10/2021 14:15:05</timestamp>
    </4>
    <5>
      <node_label>Outside</node_label>
      <node_id>70</node_id>
      <sensor_id>620</sensor_id>
      <label>WIND DIRECTION</label>
      <reading>NW</reading>
      <unit>degrees</unit>
      <timestamp>22/10/2021 14:15:05</timestamp>
    </5>
    <6>
      <node_label>Outside</node_label>
      <node_id>70</node_id>
      <sensor_id>621</sensor_id>
      <label>GUST SPEED</label>
      <reading>12.24</reading>
      <unit>km/h</unit>
```

```
      <timestamp>22/10/2021 14:15:05</timestamp>
    </6>
    <7>
      <node_label>Outside</node_label>
      <node_id>70</node_id>
      <sensor_id>622</sensor_id>
      <label>LIGHT INTENSITY</label>
      <reading>237.64</reading>
      <unit>watt/m²</unit>
      <timestamp>22/10/2021 14:15:05</timestamp>
    </7>
    <8>
      <node_label>Outside</node_label>
      <node_id>70</node_id>
      <sensor_id>623</sensor_id>
      <label>UV POWER</label>
      <reading>3</reading>
      <unit/>
      <tim
```

Response headers

```
cache-control: private,must-revalidate
content-encoding: br
content-type: application/xml
date: Fri,22 Oct 2021 11:28:33 GMT
expires: -1
pragma: no-cache
server: nginx
vary: Accept-Encoding,Authorization
x-ratelimit-limit: 60
x-ratelimit-remaining: 59
```

5.2.4. Get the information of all fields registered into an user

The users can retrieve the basic information of all their fields (nodes).

After having successfully authenticated and by and executing the designated script, the user produces the following Curl input:

Curl

```
curl -X 'GET' \
  'https://externalapi.gr/ext_api/fields?filter=xml' \
  -H 'accept: application/json' \
  -H 'Accept-Encoding: application/json' \
  -H 'Authorization: Bearer
eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJhdWQiOiIxIiwianRpIjoiMGY0MzA0NWU0MmFmMzdkODM0M2U3NzNhOTM3ZTA3NWVlZjBmNTViNDlkNTYxMTRiMTQ2YWEwN2E4YWVjZTYxNzM5OGJjYzgyYjI1ODliZjkiLCJpYXQiOiIxNjM0OTAwMjcyLjE0NjI4MSIsIm5iZiI6IjE2MzQ5MDAyNzIuMTQ2MjIiOiIxNSIsInNjb3BlcyI6W1QRPRab3ZmPk2r3D4-
ej_wSAjLy5c48h6jfRBDAG31_cP4VjmzfskIzwGnrXzffPzFA0G8LvUUp83L8nKtNiLo15LI7dDHHZYg5p9SLeprGbBOeCCifpG-
PtenALhgMqj2cRM0E1zJ_cshAJgYLr2_t_vcdfJ_ErgGffuLBm28H9sx8IZ27NpaPZqGzMG8frxk3ttuhc5UTqDTnaFznjtXuzizrbLJ_Ssm-dsBSuuqebsDSEhWro6VWg1v-mdOF9WO8to1bpKApKbPJskrbC_sV-Ze27rvdLlVJ90fsiChzcixfMN-1_f-ozX7T6djbof8oe_yMhms0Wp3MgtvriDFdb6n-
cE0pN1_Mp1Lp1SxJ4eqtyNKclYVbmlngK72HxpDYDMumiDzzSYXpbHPBH1bkIKzl1_9W3ujExqy_gMPUk5Iwxy8PEBhQqqFc9lWVmmNtojaKHiBrzc1I8AUwKUn7Pnk11q93VEw7ZnLm6fmle3KpiaBTUCA8clV_7W3lJ0xp8Y5SWRcQkxQx2WhH4SFz-werbTtiAUiKyyfHWnT144xVJV--Nhf0Up1sjNvQElhm-6cw4u4HZAK_cqaQwkya415ZfryZ1f-bbnlZobIcTXPA05ds7MGMxJxpTvnhOqD1xYdZ84481QNlX-vtpHyAsVnmZ5lYmt7k' \
  -H 'X-CSRF-TOKEN: '
```

Request URL

```
https://externalapi.gr/ext_api/fields?filter=xml
```

In case the user has successfully authenticated, they get the following message and headers, containing basic information about all the fields that have been registered:

Response body

```
<?xml version="1.0" encoding="UTF-8"?>
  <results>
    <11>
      <id>42</id>
      <label>Mykonos 2</label>
      <area>Mykonos</area>
      <coordinates>
        <lat/>
        <lng/>
      </coordinates>
      <timezone>Europe/Athens</timezone>
      <created_at>2020-05-30T11:50:52.000000Z</created_at>
    </11>
    <18>
      <id>49</id>
      <label>Hydro 2</label>
      <area>lesvos</area>
      <coordinates>
        <lat/>
        <lng/>
      </coordinates>
      <timezone>Europe/Athens</timezone>
      <created_at>2020-11-20T17:11:11.000000Z</created_at>
    </18>
    <19>
      <id>48</id>
      <label>Hydro 1</label>
      <area>lesvos</area>
      <coordinates>
        <lat/>
        <lng/>
      </coordinates>
      <timezone>Europe/Athens</timezone>
      <created_at>2020-11-20T17:09:01.000000Z</created_at>
    </19>
    <20>
      <id>50</id>
      <label>Outside</label>
      <area>Tinos</area>
      <coordinates>
        <lat/>
        <lng/>
      </coordinates>
      <timezone>Europe/Athens</timezone>
      <created_at>2020-11-27T14:52:07.000000Z</created_at>
    </20>
    <34>
      <id>64</id>
      <label>Greenhouse</label>
      <area>Tinos</area>
```

```
    <coordinates>
      <lat/>
      <lng/>
    </coordinates>
    <timezone>Europe/Athens</timezone>
    <created_at>2021-05-21T06:11:16.000000Z</created_at>
  </34>
  <50>
    <id>80</id>
    <label>Hydro 2b</label>
    <area>Λέσβος</area>
    <coordinates>
      <lat/>
      <lng/>
    </coordinates>
    <timezone>Europe/Athens</timezone>
    <created_at>2021-07-02T15:43:28.000000Z</created_at>
  </50>
  <51>
    <id>81</id>
    <label>Hydro 3 1st tank</label>
    <area>Μύκονος</area>
    <coordinates>
      <lat/>
      <lng/>
    </coordinates>
    <timezone>Europe/Athens</timezone>
    <created_at>2021-07-28T11:37:43.000000Z</created_at>
  </51>
  <52>
    <id>82</id>
    <label>Hydro 3 2nd tank</label>
    <area>Μύκονος</area>
    <coordinates>
      <lat/>
      <lng/>
    </coordinates>
    <timezone>Europe/Athens</timezone>
    <created_at>2021-07-28T12:11:46.000000Z</created_at>
  </52>
  <53>
    <id>83</id>
    <label>Hydro 4 Taratsa</label>
    <area>Μύκονος</area>
    <coordinates>
      <lat/>
      <lng/>
    </coordinates>
    <timezone>Europe/Athens</timezone>
    <created_at>2021-07-28T12:15:15.000000Z</created_at>
  </53>
  <54>
    <id>85</id>
    <label>Hydro 4 Tank 2</label>
    <area>Μύκονος</area>
    <coordinates>
      <lat/>
      <lng/>
    </coordinates>
    <timezone>Europe/Athens</timezone>
    <created_at>2021-07-28T12:22:07.000000Z</created_at>
  </54>
  <55>
```

```
      <id>86</id>
      <label>Hydro 4 - Πηγάδι</label>
      <area>Μύκονος</area>
      <coordinates>
        <lat/>
        <lng/>
      </coordinates>
      <timezone>Europe/Athens</timezone>
      <created_at>2021-07-28T13:35:37.000000Z</created_at>
    </55>
  </results>
```

6. DATA REPOSITORY STATISTICAL DATA

In this section, the most important statistical data for the DR are presented:

- up until writing this report, the DR has an uptime of 99.99%
- over 200 nodes are communicating with the DR
- over 10 million measurements have been retrieved and stored, originating from more than 1,000 sensors
- the approximate amount of data sent by the nodes each month is 280 MB (around 70 MB per week)
- the approximate amount of data sent from the DR each month to external applications (e.g. Grafana) is 4GB, something that translates to 1 million API calls per month.
- despite the large amount of API calls, the average CPU usage does not exceed 1%, while the RAM needed does not exceed 4GB (this number also includes the RAM needed by the OS, etc.)
- the HDD usage is approximately 19% of its maximum capacity (this includes the OS files, backups, etc.)

Additionally, in order to ensure data consistency and integrity and to avoid losing valuable measurements, the data are being stored on the server twice: as records in the MySQL database and as text log files. Moreover, each measurement is sent through the API to a second server for archiving purposes.

During the DR's testing phase, several benchmarks were carried out in order to determine the system's capabilities, as well as check the conformity with the GA's KPIs.

According to the benchmarks, the system was able to support more than 10,000,000 calls per day, thus achieving the relevant KPI. In real-life situations, the system is able to support more than 42,000,000 calls per day – fourfold of the KPI value.

As for the number of concurrent users, it can be as high as 10,000 – tenfold of the KPI value of 10,000.

These numbers can be verified by the log files stored on the server, as well as the server's administrative control panel.

7. CONCLUSIONS

The HYDROUSA DR was designed and developed for storing the readings from the sensors installed for the purposes of the project.

A dedicated server is used for the installation of the DR, with its specifications being more than capable of handling the volume of the current infrastructure, while also having the capability of using Apache Hadoop so as to handle large amounts of incoming data. The server has an uptime of 99.99% and is using various backup techniques in order to ensure data consistency and integrity. Despite the large amount of API calls, the server's CPU usage does not exceed 1%.

The DR was built exclusively using a number of OSS, so as to comply with EU's guidelines.

A number of services have been developed and implemented in the DR, such as sensor's data, irrigation dose, weather forecast, etc.

In order to make the DR more friendly to the end user, a GUI has been designed and developed and it is accompanied by a wiki explaining in details all the DR's components.

Additional to the DR, an external API has been developed so that the applications from the DR that have an authorization token can receive data in various formats, such as JSON, XML, etc. These data include general information from a specific field, the latest readings from the sensors of a specific field, as well as a summation of the general information from all the fields linked to a user account.