Project Title:	ECOPOTENTIAL: IMPROVING FUTURE ECOSYSTEM BENEFITS THROUGH EARTH OBSERVATIONS
Project number:	641762
Project Acronym:	ECOPOTENTIAL
Proposal full title:	IMPROVING FUTURE ECOSYSTEM BENEFITS THROUGH EARTH OBSERVATIONS
Type:	Research and innovation actions
Work program topics addressed:	SC5-16-2014: "Making Earth Observation and Monitoring Data usable for ecosystem modelling and services"

Deliverable No:

D5.7 Database of preexisting and new data

Date of deliverable:	31 May 2018
Actual submission date:	31 May 2018
Version:	v1
Main Authors:	Jordi Prados (STARLAB), Christoph Wohner (EAA), Johannes Peterseil (EAA)

This project has received funding from the *European Union's Horizon 2020 research and innovation programme* under grant agreement No 641762

Project ref. number	641762
Project title	ECOPOTENTIAL: IMPROVING FUTURE ECOSYSTEM BENEFITS THROUGH EARTH OBSERVATIONS

Deliverable title	Database of preexisting and new data
Deliverable number	D5.7
Deliverable version	1
Contractual date of delivery	31-05-2018
Actual date of delivery	31-05-2018
Document status	Draft
Document version	1
Online access	http://www.ecopotential-project.eu/products/deliverables.html
Diffusion	Public
Nature of deliverable	Report
Work package	WP 5
Partner responsible	33 – STARLAB
Author(s)	Jordi Prados (STARLAB), Johannes Peterseil (EAA), Christoph Wohner (EAA)
Editor	
Approved by	
EC Project Officer	Gaëlle Le Bouler

Abstract	Metadata is as important as the data it describes. For this reason, it is essential to have the necessary technology to be able to exploit this information and obtain desired results. The aim of this report is to explain the process of identification, design and development of the technical solution to store and provide access to different in-situ metadata resources.
Keywords	In-situ data, Metadata, DEIMS, INSPIRE, DarwinCore, Protected Areas, ECOPOTENTIAL, ISO 19157, ISO 19139

This project has received funding from the *European Union's Horizon 2020 research and innovation programme* under grant agreement No 641762

Table of Contents

Table of Figures

Table of Tables

Terms and abbreviations

INSPIRE	Infrastructure for Spatial Information in the European Community
ISO	International Organization for Standardization
OGC	Open Geospatial Consortium
WP	Work Package
WP5	WP5: In-situ monitoring data
CSW	Catalogue Service for the Web
DEIMS	Dynamic Ecological Information Management System
XML	Extensible Markup Language
JSON	JavaScript Object Notation
CQL	Common Query Language
WP10	ECOPOTENTIAL Virtual Laboratory Platform
SoS	System of Systems
SOS	Sensor Observation Service
WMS	Web Map Service
WFS	Web Feature Service
API	Application Program Interface
SFSQL	OGC Simple Feature Access SQL
VLab	ECOPOTENTIAL Virtual Laboratory Platform

Executive Summary

Geospatial data is beginning to play a very important role in today's society, where everything is interconnected. Large amounts of this type of data are generated per day in different sources, such as earth observation systems, mathematical models that produce new data, IoT systems in cities and other data generated in a manual way that have an important role in the administration of different areas. It is for this reason that computer systems and software components are vital to maintain, manage and give access to all this amount of data or services that are being generated.

The description of the data, services or any other related information object is as important as these, since without prior knowledge, they have little reuse value. For this reason, metadata is essential to create a good ecosystem of information between different entities. In the case of the ECOPOTENTIAL project, this ecosystem is centered on geospatial data from different protected areas, and it aims to create different tools capable of sharing this information following the same standards and methods. Within this project there are different work packages, each divided into different tasks. One of these work packages is focused on in-situ monitoring data (WP5), whose main objective is to homogenise, prepare and provide access to this type of data through different standards and software tools. In particular, task 5.7 is focused on the search and implementation of the necessary software and IT components to create an ecosystem of in-situ metadata from the different protected areas within the project and other sources.

The architecture of the in-situ metadata ecosystem is based on a series of methods and standards commonly used by different scientific communities, with particular reference to the standards defined by OGC, ISO and INSPIRE. As a result, this metadata ecosystem is composed of different interconnected components called metadata catalogues. Some of them are existing catalogues of different entities, and others were developed within the scope of the project. These catalogues follow the rules and the framework defined by the OGC Catalogue Service for the Web and have the ability to publish and search metadata in different formats (Dublin Core, ISO 19115, ISO 19139, ISO 19119, NASA DIF) for geospatial data and services. These were linked to the ECOPOTENTIAL Virtual Laboratory (WP10) through a global in-situ catalogue that acts as a data broker. Moreover, to meet the requirements of the European Commission H2020 program, the different software components developed within the framework of the project were published in public repositories for free and open access.

1. Introduction

This document describes the ecosystem of in-situ metadata: a set of catalogues that offer an open web service linked to a central catalogue to simplify its use. It is important to mention that some of these catalogues belong to external entities, so their design and implementation will not be described.

The main objective of the in-situ metadata ecosystem is to store, link and give access to in-situ data. The main problem is due to the large variety of formats and rules applied to each data set, which makes building a system capable of understanding each of them a difficult task. Through the use of descriptive information (metadata), the different datasets can be described and, consequently, searched without the need of reading each of them, making the necessary systems and tools simpler at a computational level and easier to use. To achieve this, metadata must follow the same rules, and, in the case of geospatial data, there are different standards that specify how these types of data should be described. Some of the most used have been treated in the work done by WP5 and described in detail in D5.2 “Metadata for pre-existing datasets” [1] and D5.3 “Framework for user-oriented quality evaluation routines” [2].

The present report is based on the different activities carried out during the development of the task 5.7: (a) concepts of data sharing, (b) explanation of the different components that make up the In-situ metadata ecosystem, (c) metadata models and capabilities, (d) data flows, (e) integration with existing catalogues and (f) development and implementation in the final version that will be subsequently linked to the global platform created within the project by the different participants: the ECOPOTENTIAL Virtual Laboratory. In addition, this report is intended to serve as a basis and guide to help different protected areas and other entities to better understand the technologies and tools available to share their data and metadata.

2. Data exchange and data integration: concepts and challenges

The exchange of data is the process that transforms data from one instance under a given scheme (source) to another with a different scheme (target). It is very important to differentiate between data integration and data exchange. In the integration, different data sources are combined to provide a unified (virtual) view of them without any transformation, while in the exchange, the data is transformed into new ones with a format, fulfilling specific rules (see Figure 1). The latter can lead to data loss due to different restrictions and incompatibilities between schemes, so different tools are needed to minimize this loss.

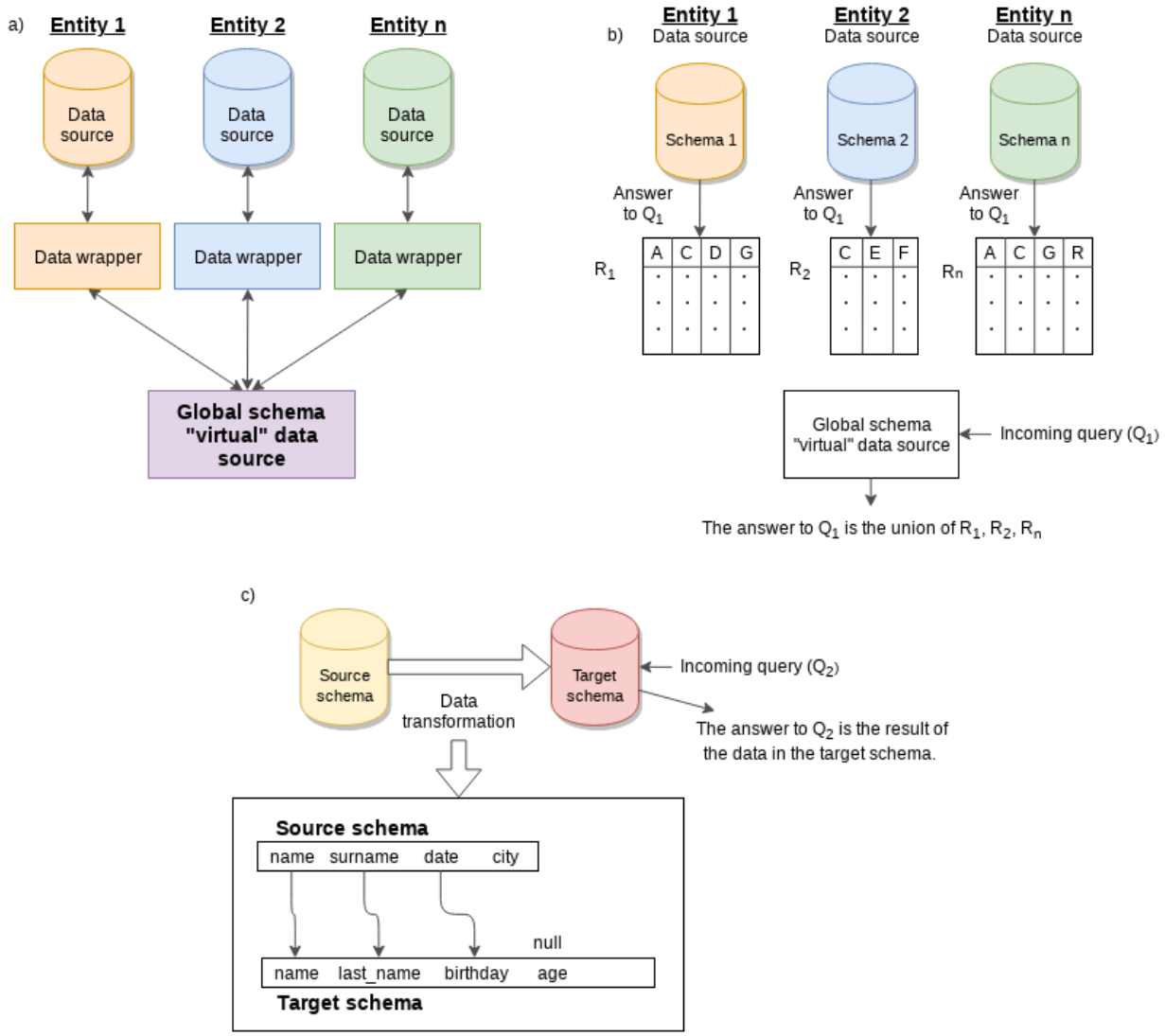

Figure 1: a, b) concept of data integration. In this scenario it is necessary to create components capable of interacting with each different data source. As a result, a general scheme acts as a gateway to be able to query data in a homogeneous manner. c) concept of data exchange. In this scenario, the data in a certain format is converted to another format under other rules. In this case, the queries are simpler since it is only necessary to understand the target schema.

In general, these processes are used between different computer systems that have the objective of sharing their data to offer services to different users. These processes are not incompatible with each other, different systems can be created using the two concepts to build integrated systems of data sharing under the same protocols and schemes. Therefore, if all the systems work under the same framework, the data will be homogenized, the queries will follow the same format and the data will be easily accessible from any external entity that understands the structure of the format.

2.1. System of systems

The objective of the integration and exchange of data is to be able to offer data to different entities or users. In the vast majority of cases, different sources share their data to generate a larger system, uniting

the resources of each one. This is known as a "System of Systems" (SoS) [3], which can be defined as: a *collection of tasks-oriented or dedicated systems that pool their resources and capabilities together to obtain a new, more complex 'meta-system' which offers more functionality and performance than simply the sum of the constituent systems [reference]*. In ECOPOTENTIAL, this concept has an important role, since one of its main challenges is the interconnection of different entities for the creation of larger systems (e.g. ECOPOTENTIAL Virtual Laboratory, In-situ metadata catalogue).

Systems of systems can be centralized, decentralized or distributed depending on their design and purpose (Figure 2). In centralized systems there is a central node that is responsible for controlling the operation of the rest of sub-nodes. In the decentralized, unlike the centralized, each node has a specific function and there is no central node that controls them all. Finally, in the distributed ones, the nodes do not have specific predefined functions. Groups of nodes are created, taking advantage of their hardware and software, to perform different operations using the power of all.

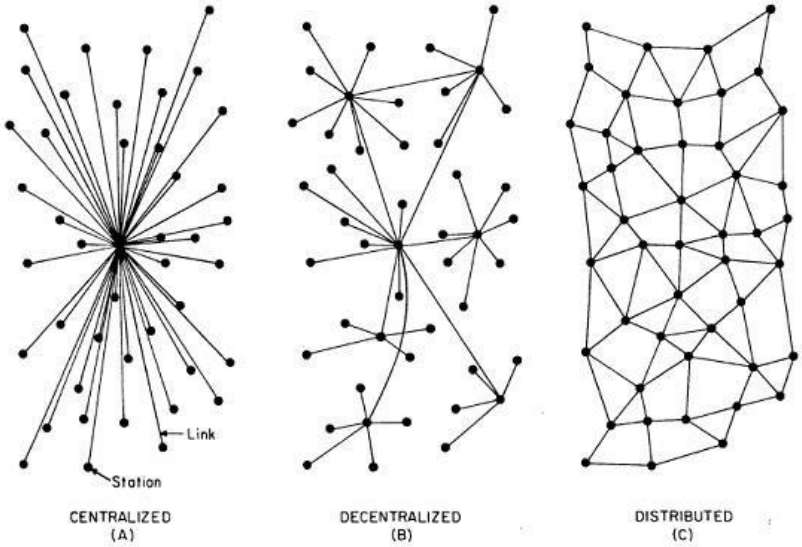

Figure 2: Centralized, decentralized and distributed systems.

In ECOPOTENTIAL, in the in-situ metadata ecosystem, the nodes that form it are decentralized nodes where each one provides different data from different protected areas. In order to create a single link and simplify access to this information, there is a central entity that acts as a gateway to the information of each node. For example, the in-situ metadata ecosystem is a decentralized system linked by a central node, creating a hybrid virtual structure.

From a technical point of view, there are different strategies to link the components of a system of systems: system of brokered systems or a system with different federated entities following the same model:

- In a brokered system (data integration), each entity offers its data or metadata following a certain model that may be different from the rest of the entities. This leads to the creation of specific components (brokers), whose main function is to provide a harmonized access to the data, metadata and functionalities of each entity.

- In a federated system (data integration and exchange), all the entities that compose it use the same model to describe data, metadata and functionalities, as well as a homogenous access to them. It is not strictly necessary that all entities use the same tools but they must use the same standards.

In any of the two previously mentioned strategies, there is a fundamental problem to consider when designing a system: the response time to the user. Being separate systems, their communication is made through the network and the speed can be affected by the volume of data to send and other external factors. Depending on the chosen access method, the response time will be longer or shorter. In the working framework of WP5, two access methods have been taken into account for the design of the in-situ metadata ecosystem architecture, and they are the following:

- "Real time" distributed search: when a request from a user arrives at the system, it is redirected to one or more remote nodes of the network. The results of each node are added to the final result (with an indicator to identify the results of each node) and sent to the user.
  The disadvantage of this method is that the response time will be determined by the slowest node in the system or the complexity of the user's request. However, this concept has an important advantage over other methods, this allows to obtain data up to date (see Figure 3).

- Harvesting: with the objective of improving the response time, data harvesting allows having a copy of the information offered by each node of the system. For this, the central node sends periodic requests to each node to collect its information and store it in a central database. In this way, the response time is considerably reduced since it stops depending on the behavior and communication of each node. However, the data collected and saved may not be up to date, so different techniques should be used to establish the appropriate update interval for each node (see Figure 3).

The two access methods are not incompatible with each other, they can coexist in a system and offer the possibility of using one or the other. Therefore, different users can evaluate their needs and decide if they prefer the data with the latest updates or having a quick response.

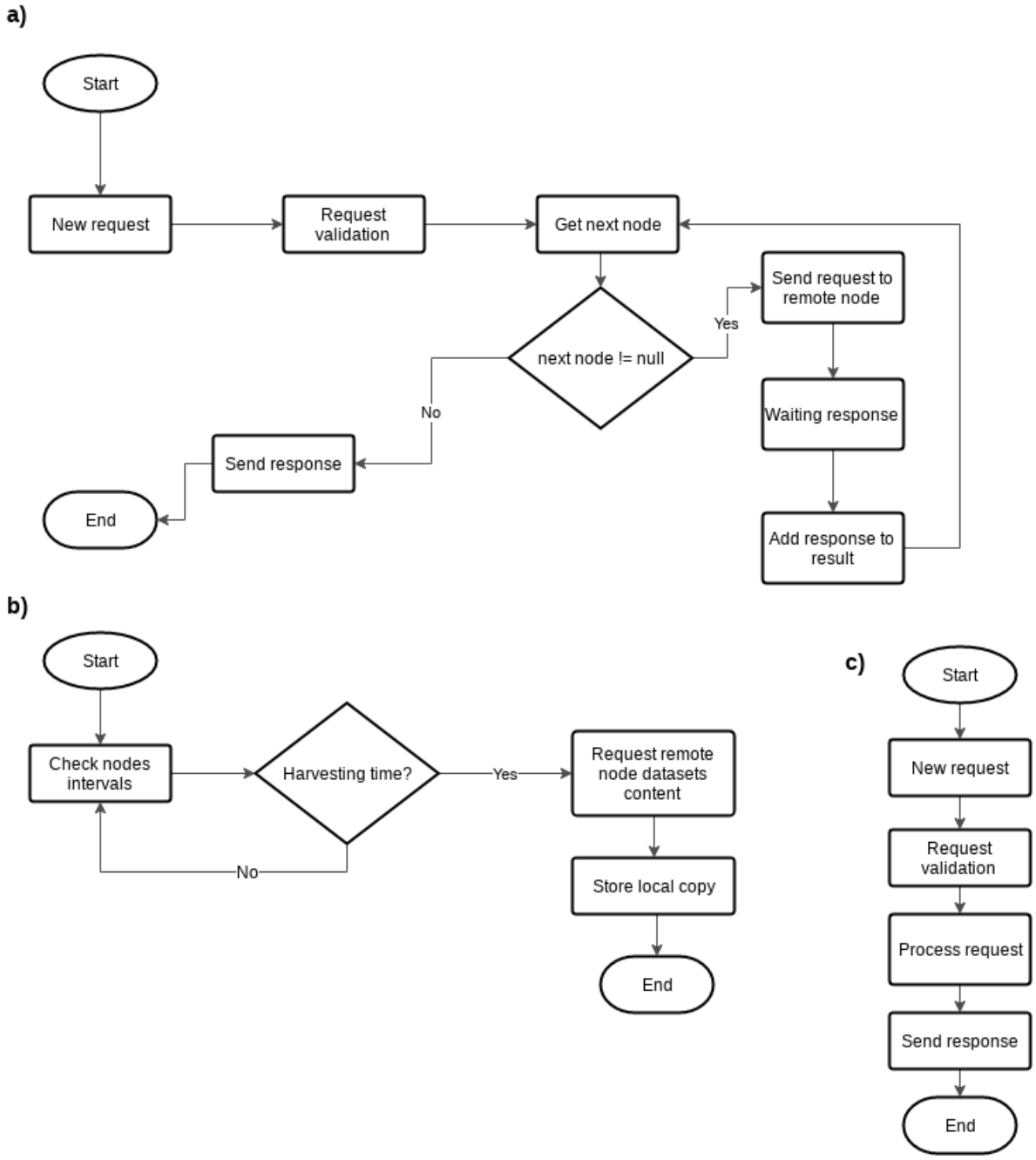

Figure 3: a) Flow diagram of the access method in real time. b) Flow diagram of the process followed to harvest data from remote nodes. c) Flow diagram of a request to previously harvested data.

3. In-situ metadata ecosystem

3.1. General overview

The in-situ metadata ecosystem is composed of different metadata catalogues and services that follow the same standards. These represent nodes in the ecosystem network and are classified into two groups:

(a) existing metadata catalogues or services and (b) the catalogue and tools developed within the framework of task 5.7. The formers usually belong to different entities and protected areas that offer their data in a public or restricted way. The second has as a main objective to act as a central node to obtain information from the rest of the nodes of the ecosystem (federation and harvesting operations). At the same time, it also gives access to data that is not accessible in a standard way through a web service (e.g. data in csv files generated by a protected area). This concept is shown in a conceptual manner in Figure 4.

The metadata sources of the in-situ metadata ecosystem are web applications that offer their data through Application Programming Interfaces (APIs) over the Internet. They are generally based on the OGC CSW [4], SOS [5], WMS [6], WFS [7] standards and use different metadata models to describe datasets. Some of these sources have a user interface to explore data and obtain results. However, the central metadata catalogue (developed in task 5.7) does not have a user interface, simplifying its use and making the system lighter at a computational level.

In this context, the ecosystem of in-situ metadata can be described from different points of view depending on the use that different users want to give it. Within the framework of this ecosystem, different types of users have been identified, as shown in Table 1 and illustrated in Figure 4.

Table 1: Description of the in-situ metadata ecosystem users.

Name	Description
Metadata creator	Entity that produces metadata in a standard or non-standard way using different tools. The offline resources produced by these users are inserted in the central catalogue once they are available (through FTPs or any other tool to share files).
Metadata provider	In the context of Task 5.7, a data provider is a service that provides metadata through a web service. To link with these services, federation and data harvesting techniques are used.
Developer of software applications/Researcher	Users who create applications or small programs to analyse the metadata information. These make use of the API to obtain the data in different formats.
User of the ECOPOTENTIAL Virtual Laboratory	Users who will use in-situ data through the ECOPOTENTIAL Virtual Laboratory (Vlab). The in-situ metadata catalogue created in task 5.7 will be invisible for them, since VLab will act responsible for performing the necessary operations to obtain the data.

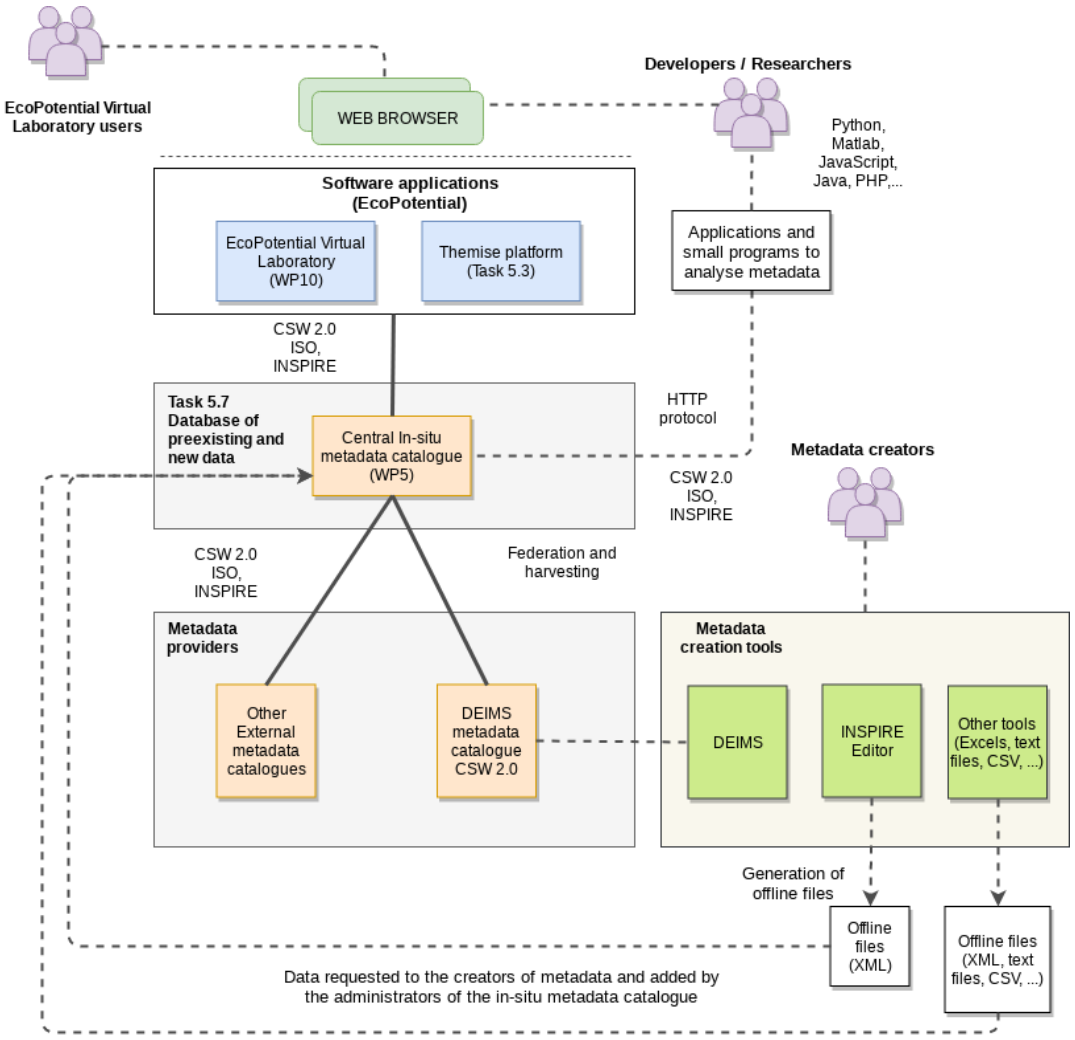

Figure 4: Overview of the ecosystem of in-situ metadata.

3.1. Technical overview

This section describes the different components at the technical level that form the central metadata catalogue in-situ, with special emphasis on the technology used and the different connections with existing metadata catalogues. Only the components developed in the framework of task 5.7 will be described, the rest of the external components mentioned are beyond the scope of this task.

3.1.1. System architecture

The central catalogue of the in-situ metadata ecosystem is based on a multi-tier architecture, divided into 3 parts (see Figure 5Figure 4):

- Delivery tier: its main function is to return the results of the different requests sent to the central catalogue. It has different data caching systems in memory to give an answer in the shortest possible time.

- Aggregation tier: contains the different components that make up the so-called API layer. This layer is responsible for the integration of internal (harvested data) and external (federation) data. The Aggregation tier is the center of all the logic of this central metadata catalogue, providing tools to respond (following the OGC CSW standard) to the different requests and return the results to the users.

- Data tier: Contains the elements to manage the database of the central catalogue. Performs operations of inserting and updating harvested data and searches for data with different filters.

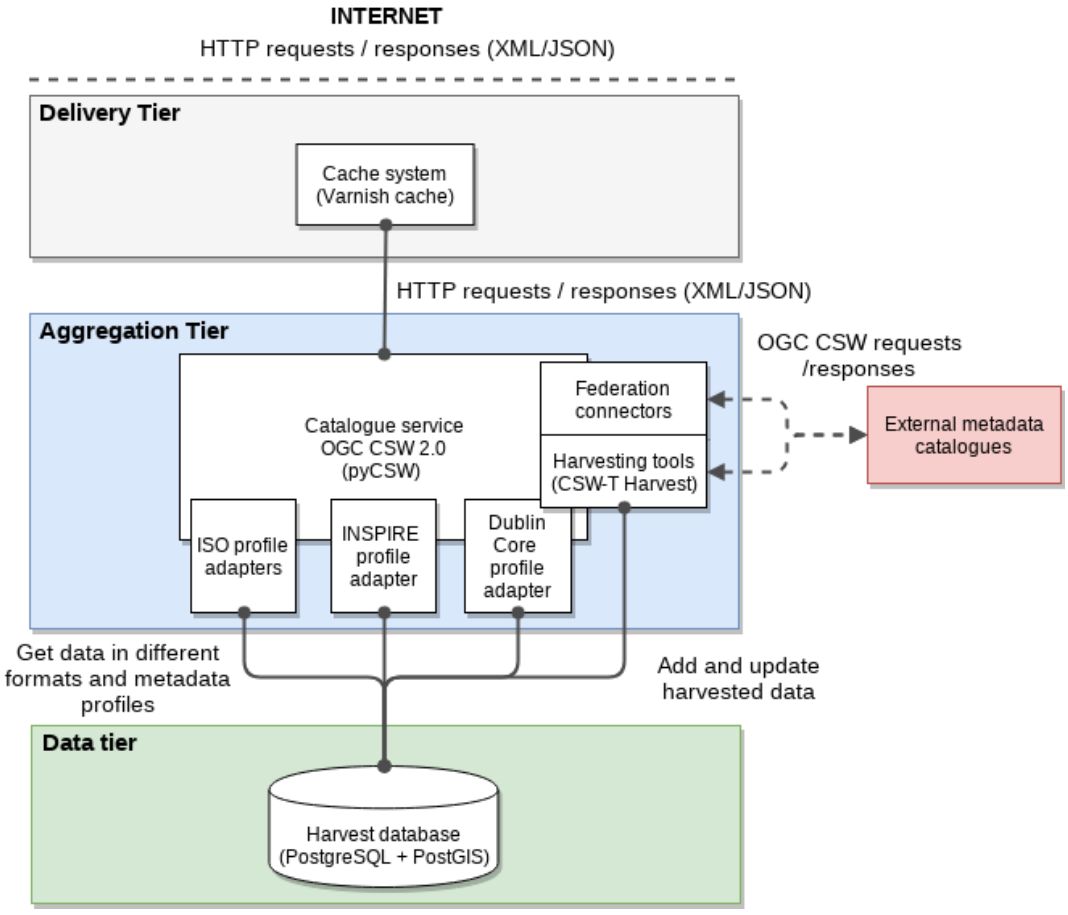

Figure 5: System architecture with the connections between the different components.

3.1.2. Caching system

The caching system acts as the gateway to the in-situ metadata catalogue and is totally invisible to the user. Its main objective is to speed up the response to users, and, to achieve this, it stores the most frequent requests and responses in RAM. In this way, the caching system is able to respond to recurring requests for the same resource without the need to send the request to the metadata catalogue to obtain

the expected results. With this process, the workload of the metadata catalogue is greatly reduced, therefore, less processing capacity is needed to process multiple requests. The graphic illustrated in Figure 6 shows the different steps that the caching system follows to respond to different requests.

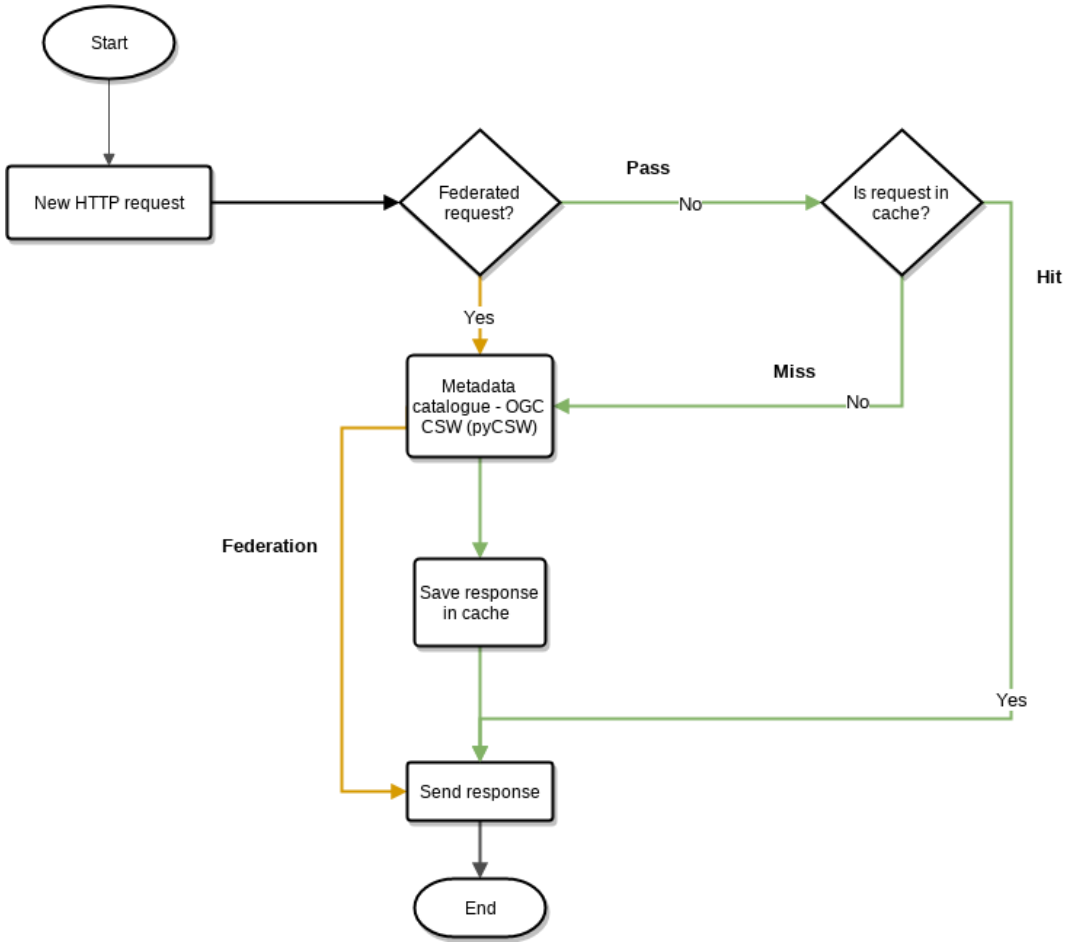

Figure 6: Process followed by the caching system to respond to user requests. In yellow the path followed by requests to federated catalogues. In green the process followed for requests to the harvested data.

The caching system distinguishes between two groups of requests: requests to obtain results with the latest updates and requests for the harvested data. The requests of the first group are sent directly to the catalogue of metadata, which, in turn, are distributed among the different federated catalogues. In this case, no type of data caching operation is carried out since the purpose of these requests is to obtain the data with the latest updates. However, the response time will depend on the different federated catalogues and the complexity of the operations. The second group of requests is aimed at avoiding dependence on federated catalogues, directly consulting the most recent harvested data. In the latter, the caching system performs a series of operations (see Figure 5) to save the content in memory for future requests by the same resource. The data stored in cache does not remain in memory forever, it has an expiration date and it is deleted once that date has been reached. To this end, different HTTP headers are added with different parameters that indicate when the cached data should be deleted. Due to the characteristics of the in-situ metadata ecosystem created in ECOPOTENTIAL, the cached data will be erased once the central catalogue has performed all the metadata harvesting operations of the different

external sources (once a day). The operation to eliminate the data of the system of cache is known as "PURGE", and this can be executed remotely by authorized components of the system.

The technology used to create the cache system is Varnish Cache. This is described as "a web application accelerator also known as a caching HTTP reverse proxy". It allows performing the caching operations previously described, as well as extra functionalities to monitor the use of the different resources.

3.1.3. Database

The main objective of the in-situ metadata catalogue database is to offer the necessary mechanisms and tools to store and manage the information of the metadata harvested from the different remote sources of data. This database follows the specifications defined in the OGC SFSQL standard, which is defined as "schema that supports storage, retrieval, query and update of feature collections via the SQL Call-Level Interface (SQL / CLI) (ISO / IEC 9075-3: 2003). A feature has both spatial and non-spatial attributes. Spatial attributes are geometry valued, and simple features are based on two-or-less dimensional geometric (point, curve and surface) entities in 2 or 3 spatial dimensions with linear or planar interpolation between vertices" [8]. By definition, the catalogue library creates the tables and functions necessary to support the standard. However, there are database engines that incorporate these tables and functions by default. One of these engines is PostgreSQL with the PostGIS extension, which incorporates a series of mechanisms to manage geospatial data.

The catalogue generates a table called "records", which is composed of 73 columns. These columns can be divided into different groups (see Table 2)

Table 2: Database fields description

Group	Description	Database table fields
Core	Core of the metadata table. These are the mandatory fields required by each of the metadata in the database.	identifier, typename, schema, mdsource, insert_date, xml, anytext, language
Identification	This group contains descriptive information of the dataset	type, title, title_alternate, abstract, keywords, keywordstype, parentidentifier, relation, time_begin, time_end, topicategory, resourcelanguage
Attribution	Information about the creators of the metadata record.	creator, publiser, contributor, organization
Security	Information about the different restrictions that may exist to access the data described by the metadata.	securityconstraints, accessconstraints, otherconstraints
Date	Relevant information about the dates of creation, publication, modification and revision of the metadata.	date, date_revision, date_creation, date_publication, date_modified
Geospatial	Group of fields that are related to the area of	crs, geodescode, denominator,

	action of the dataset.	distancevalue, distanceuom, wkt_geometry
Service	Information about the service(s) that give access to the data.	Servicetype, servicetypeversion, operation, couplingtype, operateson, operatesonidentifier, operatesoname
Additional	Additional information related to the quality of the metadata (DQ_DataQuality) and the format and source (external or local) of these. Although not all the fields related to the quality of the metadata are included, these can be added within the xml field.	Degree, classification, conditionapplyingtoaccessanduse, lineage, responsiblepartyrole, specificationtitle, specificationdate, specificationdatetype format, source
Distribution	Information about the online resource to access the data described by the metadata.	links

3.1.4. In-situ metadata catalogue

The metadata catalogue is the most important component of the in-situ metadata ecosystem of ECOPOTENTIAL. The idea behind this is to integrate as many in-situ metadata catalogues as possible. This is the starting point for searching the entire ecosystem, using tools to provide a transversal and homogeneous access to the metadata data of all the metadata services that make up the ecosystem.

This catalogue is a web application based on pyCSW [9], a Python library that offers the necessary tools to create a server capable of sharing metadata following the standards defined by OGC, INSPIRE and ISO. The kernel is based on the operations described in OGC CSW 2.0.2 and 3.0.0 and offers support for different types of metadata models and formats. In addition, different changes and improvements in the code of this library have been introduced to adapt the functionality to the objectives of this catalogue. It is important to mention that it lacks a user interface, and only users with knowledge in XML and JSON formats can interpret the information it offers. This seeks to offer a simple and lightweight system and thus allow easy integration into different external applications (e.g. ECOPOTENTIAL VLab, Themise). In relation to access to data, it lacks authentication systems through HTTP headers, therefore, the data is open access without any restriction of use.

The process followed for the requests of the users is illustrated in Figure 6. In this, different components interact to validate the requests, extract their information and obtain the results to subsequently respond to the user.

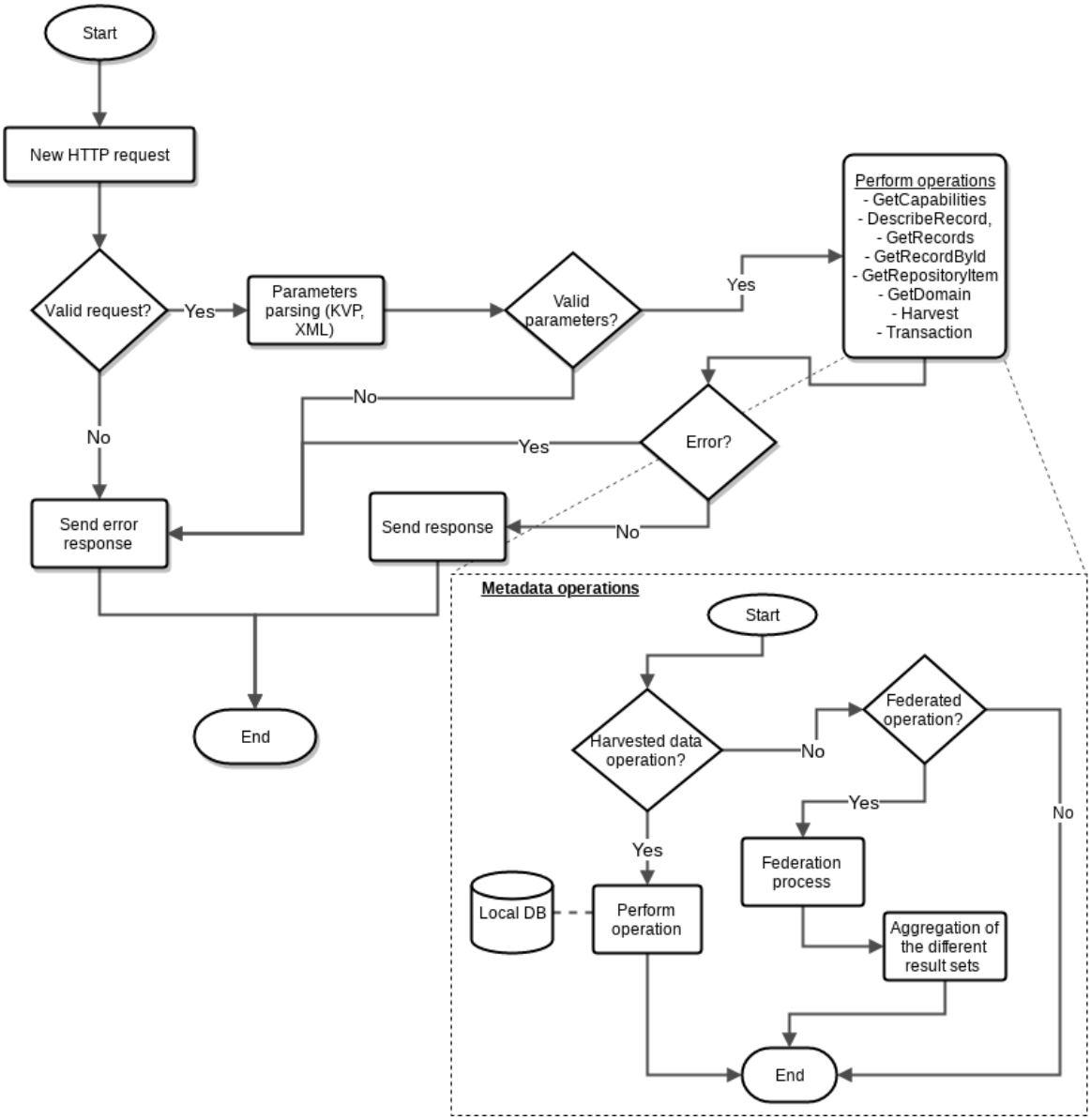

Figure 7: Diagram with the steps followed by the in-situ metadata catalogue to give an answer to the different requests of the users.

3.1.4.1. Standard support

The ecosystem's central catalogue is based on different metadata standards, some of them related to mechanisms to share metadata and others related to their structure. The most important and considered are those described in Table 3. With the implementation of these, a homogeneous access is achieved using the same language among the different external applications that use the metadata of this ecosystem. In addition, the integration of remote metadata catalogues is simplified thanks to the different tools to perform federated searches and harvest metadata from different models and metadata formats. PyCSW implements all these standards, offering a system of easy configuration and adaptable to the different previously mentioned standards.

Table 3: Standards support.

Standard name	Description
OGC CSW	Set of specifications that define methods of accessing metadata. The metadata catalogue described in this document is based on this standard, offering the main functions of this: GetCapabilities, DescribeRecord, GetRecords, GetRecordById, GetRepositoryItem, GetDomain, Harvest (WMS, WFS, WCS, WPS, WAF, CSW, SOS), Transaction
OGC Filter	Specifications to create a homogenous data query language between different standards. This describes the structure in KVP (HTTP GET) or XML (POST or SOAP) of the requests to search and filter data.
OGC GML	Structure in XML defined by OGC to describe geographical features. The metadata catalogue uses this scheme to insert new metadata and return the user results following this structure.
Dublin Core	Set of terms to describe digital resources. It is composed of fifteen generic properties that allow describing a wide range of digital resources. In the same way as OGC GML, it is used as input and output metadata schema.
ISO 19115, 19139, 19119	Standards of metadata models (XML based) that describe geographic information and services.
OAI-PMH	Set of rules that form a framework for the harvest of metadata.

3.1.4.2. Supported operations

As previously mentioned, OGC CSW defines a series of functions to discover and manage the data of a metadata catalogue. In many of these operations, different types of filtering operations can be applied to obtain more accurate results. This in-situ metadata catalogue supports different filtered types: logical and geospatial data:

Table 4: Supported filtering operations.

Logical filters	Spatial filters
• Between • EqualTo • LessThanEqualTo • GreaterThan • Like • LessThan • GreaterThanEqualTo • NotEqualTo • NullCheck	• BBOX • Beyond • Contains • Crosses • Disjoint • DWithin • Equals • Intersects • Overlaps • Touches

	• Within These operations need geometries defined using the structures defined in the GML schema [GML reference]: • gml:Point • gml:LineString • gml:Polygon • gml:Envelope

This section aims to provide a brief description of the different operations with technical examples of their use in different formats.

GetCapabilities

Function that returns metadata with the service information in XML. This XML contains information about the different versions, formats, schemas and filters of the supported standards. The parameters accepted by this operation are described in the following table.

Table 5: GetCapabilities parameters description

Parameter	Required	Description
AcceptVersions=2.0.2	yes	CSW version
SERVICE=CSW	yes	Fixed parameter
REQUEST=GetCapabilites	yes	Operation value (In this case GetCapabilities)
outputFormat=application/xml	no	Output format of the response. GetCapabilities only accepts xml formats
Sections	no	Sections(ServiceIdentification, ServiceProvider, OperationsMetadata, or Filter_Capabilities sections) included in the response. If the sections parameter is not specified, all sections are returned.

Table 6: GetCapabilities request examples.

Request (GET)
HTTP GET http://eco.starlab.es/csw?request=GetCapabilities&service=CSW&acceptVersions=2.0.2&acceptForm

ats=application/xml
Request (POST)
HTTP POST http://eco.starlab.es/csw Headers: Content-Type: application/xml Body: <?xml version="1.0" encoding="UTF-8"?> <csw:GetCapabilities xmlns:csw="http://www.opengis.net/cat/csw/2.0.2" service="CSW"> <ows:AcceptVersions xmlns:ows="http://www.opengis.net/ows"> <ows:Version>2.0.2</ows:Version> </ows:AcceptVersions> <ows:AcceptFormats xmlns:ows="http://www.opengis.net/ows"> <ows:OutputFormat>application/xml</ows:OutputFormat> </ows:AcceptFormats> </csw:GetCapabilities>

GetRecords

Function that returns a list of all available metadata in the catalogue. This operation allows metadata to be obtained from federated catalogues (updated data) or from data previously collected (local database). In addition, the different types of filtering operations can be applied as well as obtaining the results in different formats and metadata schemas (see Table 7 and Table 8).

Table 7: GetRecords parameters description.

Parameter	Required	Description
AcceptVersions=2.0.2	yes	CSW version
SERVICE=CSW	yes	Fixed parameter
REQUEST=GetCapabilites	yes	Operation value (In this case GetRecords)
TypeName=gmd:MD_Metadata	yes	Type of schema to search. It can be a list of comma-separated values. The most common values: gmd:MD_Metadata and csw:Record
outputFormat=application/xml	no	Output format (JSON/XML) of the response.
resultType	no	Detail of the response. This parameter determines whether the result is a summary of the data set (hits) or returns one or more records (results) or validates the request and then is answered asynchronously (validates).
outputSchema	no	Specifies the schema of the returned records. The default value is http://www.opengis.net/cat/csw/2.0.2.
startPosition	no	Specifies the catalogue record number by which the

		response should be generated.
maxRecords	no	Maximum number of records returned in the result.
ElementSetName	no	Level of detail of the response (brief, full, summary)
ElementName	no	Specifies the elements of the schema to be displayed of the different metadata.
ConstraintLanguage	no	Type of constraint: FILTER or CQL_TEXT (case sensitive)
Constraint	no	Specifies the filter for the query.
SortBy=Title:A	no	Properties separated by commas to sort the result (A: ascending and D: descending)
DistributedSearch=true	no	Parameter to indicate if the request is for federated catalogues or local data (harvested)
hopCount=2	no	Indicates the number of "jumps" to the distributed server before the search ends. By default, it is 2 and, to avoid circular requests, the in-site metadata catalogue will not perform operations with a value other than 2.
ResponseHandler=URI	no	Response method, by default it is URI (synchronous) but other asynchronous methods can be specified (email notification if configured)

Table 8: GetRecords request examples.

Request (GET)
HTTP GET http://eco.starlab.es/csw?SERVICE=CSW&version=2.0.2&REQUEST=GetRecords&resultType=results&elementSetName=brief&outputSchema=http://www.isotc211.org/2005/gmd&typeNames=gmd:MD_Metadata
Request (POST Geospatial filter)
HTTP POST http://eco.starlab.es/csw Headers: Content-Type: application/xml Body: <?xml version="1.0" encoding="ISO-8859-1" standalone="no"?> <csw:GetRecords xmlns:csw="http://www.opengis.net/cat/csw/2.0.2" xmlns:gml="http://www.opengis.net/gml" xmlns:ogc="http://www.opengis.net/ogc" service="CSW" version="2.0.2" resultType="results" startPosition="1" maxRecords="5" outputFormat="application/xml" outputSchema="http://www.opengis.net/cat/csw/2.0.2" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://www.opengis.net/cat/csw/2.0.2 http://schemas.opengis.net/csw/2.0.2/CSW-discovery.xsd"> <csw:DistributedSearch hopCount="2"/> <!-- Federation option --> <csw:Query typeNames="csw:Record">

```
    <csw:ElementSetName>brief</csw:ElementSetName>
    <csw:Constraint version="1.1.0"> <!-- Filtering section -->
    <ogc:Filter>
        <ogc:BBOX>
                <ogc:PropertyName>ows:BoundingBox</ogc:PropertyName>
                <gml:Envelope>
                <gml:lowerCorner>40.41 -0.24 </gml:lowerCorner>
                <gml:upperCorner>42.59 4.3</gml:upperCorner>
                </gml:Envelope>
        </ogc:BBOX>
    </ogc:Filter>
   </csw:Constraint>
   </csw:Query>
  </csw:GetRecords>
```

GetRecordById

Obtains the information of one or several registers using its identifiers. To perform this operation, it is necessary to have prior knowledge of the identifiers to be searched (GetRecords). The allowed parameters are described in Table 8 and Table 9.

Table 9: GetRecordById parameters description.

Parameter	Required	Description
AcceptVersions=2.0.2	yes	CSW version
SERVICE=CSW	yes	Fixed parameter
REQUEST=GetRecordById	yes	Operation value (In this case GetRecordById)
outputFormat=application/xml	no	Output format (JSON/XML) of the response.
outputSchema	no	Specifies the schema of the returned records. The default value is http://www.opengis.net/cat/csw/2.0.2.
ElementSetName	no	Level of detail of the response (brief, full, summary)
id=...	no	Identifier of one or more records to return.

Table 10: GetRecordById request examples.

Request (GET)
http://eco.starlab.es/csw?SERVICE=CSW&version=2.0.2&Request=GetRecordById&id=urn:uuid:455a6ce0-2b33-428f-979a-bc42dc925bd9,cdecfe47-1780-4dda-9fdf-e0a099e4ba9b
Request (POST)
HTTP POST http://eco.starlab.es/csw

Headers:
Content-Type: application/xml

Body:

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="no"?>
<csw:GetRecordById
   service="CSW"
   version="2.0.2"
   outputFormat="application/xml"
   outputSchema="http://www.opengis.net/cat/csw/2.0.2"
   xmlns="http://www.opengis.net/cat/csw/2.0.2"
   xmlns:csw="http://www.opengis.net/cat/csw/2.0.2"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
   <csw:Id>urn:uuid:455a6ce0-2b33-428f-979a-bc42dc925bd9</csw:Id>
   <csw:Id>cdecfe47-1780-4dda-9fdf-e0a099e4ba9b</csw:Id>
   <ElementSetName typeNames="csw:Record">full</ElementSetName>
</csw:GetRecordById>
```

Table 8: GetRecordById request examples

DescribeRecord

Function that returns descriptive information about all the elements of the metadata models supported by the catalogue. The following tables show the description of the accepted parameters and their use.

Table 11: DescribeRecord parameters description.

Parameter	Required	Description
AcceptVersions=2.0.2	yes	CSW version
SERVICE=CSW	yes	Fixed parameter
REQUEST=DescribeRecord	yes	Operation value (In this case DescribeRecord)
outputFormat=application/xml	no	Output format (JSON/XML) of the response.
Typename=csw:Record	no	Type of schema to describe. By default, all the scheme will be described.
schemaLanguage=XMLSCHEMA	no	Squema language that should be used to describe the schemas. The default value is XMLSCHEMA.

Table 12: DescribeRecord request examples.

Request (GET)
http://eco.starlab.es/csw?Request=DescribeRecord&service=CSW&version=2.0.2&NAMESPACE=xmlns(csw=http://www.opengis.net/cat/csw/2.0.2),xmlns(gmd=http://www.isotc211.org/2005/gmd)&schemaLanguage=http://www.w3.org/XML/Schema&outputFormat=application/xml&TypeName=csw:Record,gmd:MD_Metadata

Request (POST)
HTTP POST http://eco.starlab.es/csw Headers: Content-Type: application/xml Body: <?xml version="1.0" encoding="ISO-8859-1" standalone="no"?> <csw:DescribeRecord service="CSW" version="2.0.2" outputFormat="application/xml" schemaLanguage="http://www.w3.org/XML/Schema" xmlns:csw="http://www.opengis.net/cat/csw/2.0.2" xmlns:gmd="http://www.isotc211.org/2005/gmd"> <csw:TypeName>csw:Record</csw:TypeName> <csw:TypeName>csw:MD_Metadata</csw:TypeName> </csw:DescribeRecord>

Table 10: DescribeRecord request examples

Harvest

This operation offers a series of mechanisms to harvest metadata from remote sources and thus create, update or delete metadata from the local database indirectly. This operation can be executed through GET and POST requests (see Table 11 and Table 12) synchronously or asynchronously, making it possible to execute it periodically. However, the use of this operation is restricted by IP and only authorized computer machines can perform this operation.

Table 13: Harvest parameters description.

Parameter	Required	Description
AcceptVersions=2.0.2	yes	CSW version
SERVICE=CSW	yes	Fixed parameter
REQUEST=Harvest	yes	Operation value (In this case Harvest)
source=target	yes	URL of the remote metadata source.
resourceType	yes	Type of metadata schema (http://www.opengis.net/cat/csw/2.0.2, http://www.isotc211.org/schemas/2005/gmd/)
requestHandler	no	This parameter allows to notify via email or FTP the results of the metadata harvest operation asynchronously. The in-situ metadata catalogue incorporates a new notification method by sending the results to a web API using the HTTP POST method.

Table 11: Harvest parameters description

Table 14: Harvest request examples.

Request (GET - Synchronous)
http://<repository_url>/csw?request=Harvest&service=CSW&version=2.0.2&Source=http://eco.starlab.es&ResourceType=http://www.opengis.net/cat/csw/2.0.2
Request (POST - Synchronous)
HTTP POST <REPOSITORY_URL> Headers: Content-Type: application/xml Body: <?xml version="1.0" encoding="UTF-8"?> <csw:Harvest xmlns:csw="http://www.opengis.net/cat/csw/2.0.2" xmlns:gmd="http://www.isotc211.org/2005/gmd" service="CSW" version="2.0.2"> <csw:Source>http://eco.starlab.es/csw</csw:Source> <csw:ResourceType>http://www.opengis.net/cat/csw/2.0.2</csw:ResourceType> </csw:Harvest>
Request (GET - Asynchronous)
http://<repository_url>/csw?request=Harvest&service=CSW&version=2.0.2&Source=http://eco.starlab.es&ResourceType=http://www.opengis.net/cat/csw/2.0.2&ResponseHandler=<url or email>
Request (POST - Asynchronous)
HTTP POST <REPOSITORY_URL> Headers: Content-Type: application/xml Body: <?xml version="1.0" encoding="UTF-8"?> <csw:Harvest xmlns:csw="http://www.opengis.net/cat/csw/2.0.2" xmlns:gmd="http://www.isotc211.org/2005/gmd" service="CSW" version="2.0.2"> <csw:Source>http://eco.starlab.es/csw</csw:Source> <csw:ResourceType>http://www.opengis.net/cat/csw/2.0.2</csw:ResourceType> <CSW:ResponseHandler>[URL or email]</csw:ResponseHandler> </csw:Harvest>

Transaction

The Transaction operation (CSW-T) defines a series of methods for creating, updating and deleting records from the metadata catalogue. Only HTTP POST operations are allowed and, in the same way as metadata harvest operations, the use of these is restricted by IP. In the update and delete operations, different filters can be applied using the structure defined by OGC Filter [reference]. Some examples are described in the following table.

Table 15: Transaction request examples.

Request (POST - Insert)

HTTP POST http://eco.starlab.es/csw

Headers:
Content-Type: application/xml

Body:

```
<?xml version="1.0" encoding="UTF-8"?>
<csw:Transaction xmlns:csw="http://www.opengis.net/cat/csw/2.0.2" version="2.0.2"
service="CSW">
  <csw:Insert>
      <gmd:MD_Metadata xmlns:gmd="http://www.isotc211.org/2005/gmd"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:gml="http://www.opengis.net/gml" xmlns:gco="http://www.isotc211.org/schemas/2005/gco"
xmlns:srv="http://www.isotc211.org/2005/srv">
    …
    …
   </gmd:MD_Metadata>
  </csw:Insert>
</csw:Transaction>
```

Request (POST - Update)

HTTP POST http://eco.starlab.es/csw

Headers:
Content-Type: application/xml

Body:

```
<?xml version="1.0" encoding="UTF-8"?>
<csw:Transaction xmlns:csw="http://www.opengis.net/cat/csw/2.0.2"
xmlns:dc="http://purl.org/dc/elements/1.1/" xmlns:ogc="http://www.opengis.net/ogc"
version="2.0.2" service="CSW">
          <csw:Update>
            <csw:RecordProperty>
              <csw:Name>apiso:Title</csw:Name>
              <csw:Value>test</csw:Value>
            </csw:RecordProperty>
            <csw:Constraint version="1.1.0">
              <ogc:Filter>
                <ogc:PropertyIsEqualTo>
                   <ogc:PropertyName>apiso:Identifier</ogc:PropertyName>
                  <ogc:Literal>urn:uuid:455a6ce0-2b33-428f-979a-bc42dc925bd9</ogc:Literal>
                </ogc:PropertyIsEqualTo>
              </ogc:Filter>
            </csw:Constraint>
   </csw:Update>
</csw:Transaction>
```

Request (POST - Detele)

HTTP POST http://eco.starlab.es/csw
Headers:

```
Content-Type: application/xml

Body:
<?xml version="1.0" encoding="UTF-8"?>
<csw:Transaction xmlns:csw="http://www.opengis.net/cat/csw/2.0.2"
xmlns:dc="http://purl.org/dc/elements/1.1/" xmlns:ogc="http://www.opengis.net/ogc"
version="2.0.2" service="CSW">
        <csw:Delete>
          <csw:Constraint version="1.1.0">
            <ogc:Filter>
              <ogc:PropertyIsEqualTo>
                <ogc:PropertyName>apiso:Identifier</ogc:PropertyName>
                <ogc:Literal>urn:uuid:455a6ce0-2b33-428f-979a-bc42dc925bd9</ogc:Literal>
              </ogc:PropertyIsEqualTo>
            </ogc:Filter>
          </csw:Constraint>
    </csw:Delete>
</csw:Transaction>
```

3.1.5. Data exchange and integration

The metadata of the in-situ metadata ecosystem is exposed using the central catalogue explained in the previous section. In addition to the access functions of the catalogue, it also incorporates a series of mechanisms to add new metadata sources and thus be able to perform federated searches or metadata harvesting. This section aims to explain the technical implementation of the concepts of federation and metadata harvesting, as well as different examples of use cases and metadata integration from external sources.

3.1.5.1 Federation

The federation allows communication between different nodes that follow the same standards and metadata models. Technically, the central catalogue validates all incoming requests and then sends them to the different federated catalogues to which it has access. PyCSW provides a series of mechanisms to perform these operations as well as the possibility of adding new links to the list of federated catalogues available to the central node.

Federated catalogues must comply with the OGC CSW standard in order to be linked to the central catalog, and since pyCSW does not have a graphical interface, the link between catalogues must be done manually by the administrator. PyCSW has a configuration file, called "default.cfg", in which catalogue configuration parameters are specified. Within this file there is a variable called "federatedcatalogues" (see Figure 8) to specify the URLs of the different remote metadata catalogues. It must be borne in mind that, by default, this operation does not consider any type of authentication system, so access to this data must be open. In the future, and, in particular cases, some modifications must be added to offer the possibility of applying restrictions to access the catalogues.

Figure 8: Federated catalogues list structure.

3.1.5.2 Harvest

The process of harvesting metadata is divided into two parts: first harvest and updating of previously harvested metadata (see Figure 9).

- The first part is a manual process in which a metadata harvest request must be sent, as explained in the previous section. However, this operation can result in duplicate metadata conflicts from different sources. To avoid this, these harvesting operations will be carried out in a table of the local database that will act as a temporary table. Subsequently, the data of this local table will be copied to the main table and, in the case of duplicate metadata, it will be decided which of them remains in the main table. Once this operation is done, the temporary table is deleted to avoid redundancy of data in the system. The problem with this process is that it is usually time consuming and, in some cases, different types of errors may appear (e.g. HTTP 504 Gateway Timeout). For this reason, it is important to perform this operation asynchronously, specifying the "ResponseHandler" parameter in the metadata harvest request.
- The second part refers to the update of the metadata. This operation is performed once a day automatically in a parallel process. The different external sources of the metadata are consulted to verify possible changes in the metadata and, in this way, the local database will have a copy of all the metadata of the different remote catalogues, making the incoming requests resolved in a short time frame. PyCSW incorporates a tool to execute a series of commands related to the management of metadata. Within the list of these commands, there is one called "refresh_harvested_records" whose objective is to consult the sources of external metadata to verify possible changes in these. This operation can be executed within a time-based job scheduler or cron job.

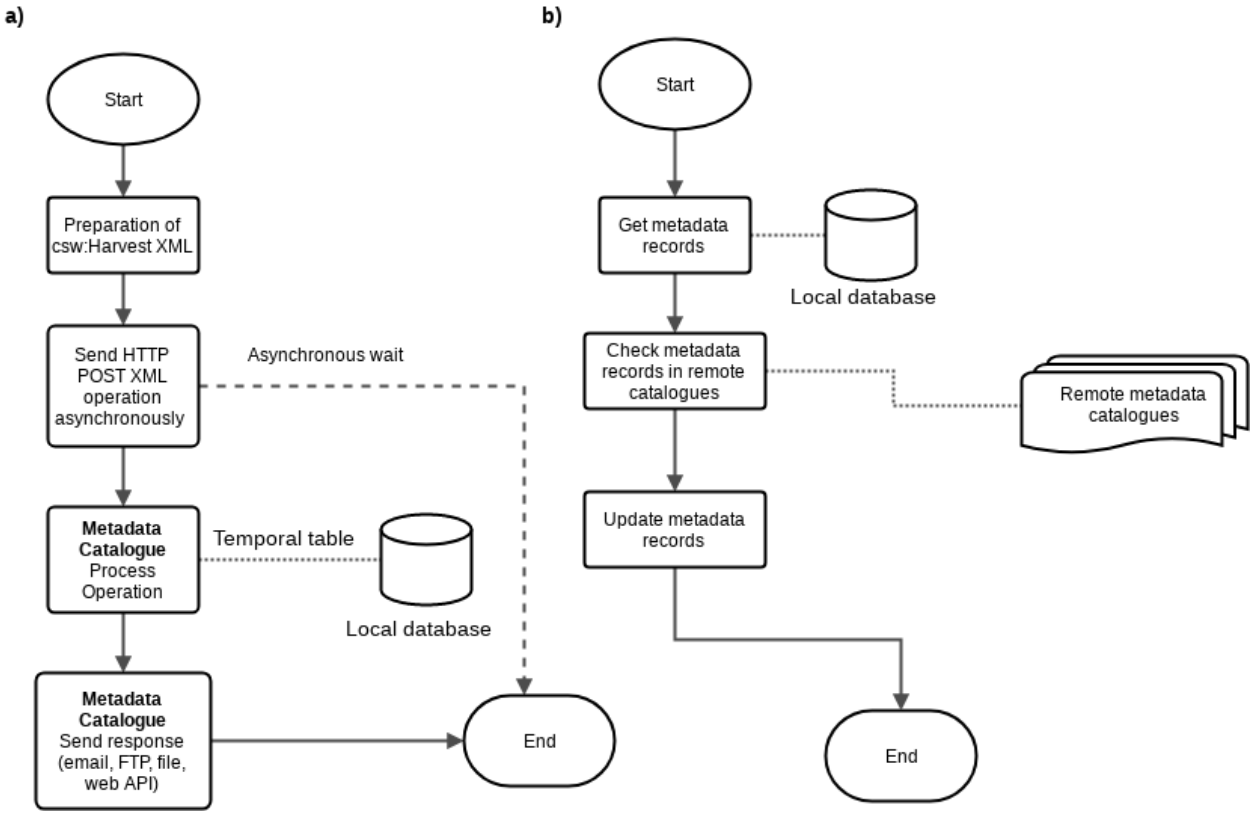

Figure 9: Harvesting process. a) First metadata harvest, b) Process to update previously harvested metadata.

3.1.5.3 Integration examples

DEIMS metadata catalogue

Metadata on DEIMS-SDR is exposed using pyCSW, an OGC CSW server implementation written in Python, that allows to run multiple CSW endpoints with a single install while being “Headless”, meaning that there is no GUI, only scriptable command line configuration and deployment, making it an ideal choice for an online production service such as DEIMS-SDR.

The actual files containing the metadata information exposed through pyCSW are batch-generated in the ISO19139 format once a week. Those generated records feature all data entities on DEIMS-SDR, i.e. site, dataset and data product information. The central metadata catalogue uses the CSW API offered by DEIMS-SDR to perform federated searches and harvest of metadata. Since they use similar technologies and standards, communication between both platforms is fluid and robust.

ThemisE

ThemisE is defined as "user-oriented spatial data quality evaluation routines and procedures (data quality management) for the assessment of the internal and external quality of pre-existing data, based on quality measures extracted from (spatial) metadata, accessible as a web -based application based on open-sources technologies. Within ECOPOTENTIAL's scope, these tools are important to support the quality-

driven identification / selection of relevant data or the identification of data quality gaps enabling the planning of targeted data collection "[Reference D5.3].

In this case ThemisE does not act as a data provider, but as an external application that requires access to metadata catalogues following the OGC CSW standard. It sends requests (e.g. GetRecords, GetRecordById) to different metadata catalogues, including ECOPOTENTIAL's central in-situ metadata catalogue.

ECOPOTENTIAL Virtual Laboratory

ECOPOTENTIAL Virtual Laboratory is defined as "innovative solutions that will provide open and unrestricted access to interoperable ecosystem Earth Observation data and information" [10] . It is a platform with web access and is composed of different sections to process, consult and produce new data and metadata. It has a section to explore metadata from different external sources with different models and different standards. In the same way as ThemisE, this platform uses the operations defined by the OGC CSW standard to perform searches in the in-situ metadata catalogue described in this document. It is important to mention that only the harvested data are shown in this platform, since the operations to federated catalogues may not be stable in certain cases.

4. Conclusions

The use of different standards and tools for the sharing of data and metadata have been widely adopted by different entities that operate on the Internet but less used in contexts such as in the administration of Protected Areas. During the course of the different activities carried out in Task 5.7, it has been possible to observe how the vast majority of Protected Areas have data and metadata, but in many cases the access and structure of these does not follow any standard, making the use of these difficult. The creation of computer architectures for the sharing of data can be expensive, and, in many cases, an unjustified expense. For this reason, the catalog of in-situ metadata developed in task 5.7 together with the different tools offered within ECOPOTENTIAL represent a great opportunity to show the potential and the possible benefits that these technologies can offer to the different institutions that participate in the project.

The architecture and the different technical components described in this document respond to different requirements on the part of the users. These requirements were collected and analyzed iteratively during task 5.7 and are mainly related to the need to have a fast and robust access to data from different sources in different formats. For this reason, the decision to adopt georeferenced metadata sharing standards such as OGC CSW was vital to meet the expectations and objectives set.

The final implementation of the in-situ metadata catalogue was implemented in a modular manner, resulting in a web application without a user interface but with tools to link its content with other applications in a standardized way. In addition, the concept of federation and harvest of metadata offers an extra bonus to the platform since it allows users to control the response time of their requests.

In order to offer a better experience to different users, future developments or improvements have to be carried out, some of them can be:

- Creation of a user interface to be able to manage the catalogue more easily without deep technical knowledge on the platform.
- Support for more filtering options.
- Allow the results to be exported in other formats (e.g. xls; ...)
- Descriptive online documentation with real examples of use for users not technically advanced.

Finally, all the concepts explained in this document have been taken into account for the final development of the platform, whose code can be found in the following link for free access: https://github.com/starlab-bcn/ecopotential-insitu-metadata-catalogue

5. References

[1] P. Dimitris, «ECOPOTENTIAL D5.2 Metadata for pre-existing datasets,» . Available: http://www.ecopotential-project.eu/images/ecopotential/documents/D5.2.pdf.

[2] P. C. Joaquim Alonso, «ECOPOTENTIAL D5.3 Framework for user-oriented quality evaluation routines,» . Available: http://www.ecopotential-project.eu/images/ecopotential/documents/D5.3.pdf.

[3] A. T. Pierre Dersin, «IEEE-Reliability Society. Technical Committee on ‘ Systems of Systems’,» 15 10 2014. . Available: https://rs.ieee.org/component/content/article/9/77-system-of-systems.html.

[4] Open Geospatial Consortium (OGC), «OGC Catalogue Service for the Web (CSW),» 10 06 2016. . Available: http://docs.opengeospatial.org/is/12-168r6/12-168r6.html.

[5] Open Geospatial Consortium (OGC), «OGC Sensor Observation Service (SOS),» 16 04 2012. . Available: http://www.opengeospatial.org/standards/sos.

[6] Open Geospatial Consortium (OGC), «OGC Web Map Service (WMS),» 15 03 2006. . Available: http://www.opengeospatial.org/standards/wms.

[7] Open Geospatial Consortium (OGC), «OGC Web Feature Service (WFS),» 03 05 2005. . Available: http://www.opengeospatial.org/standards/wfs.

[8] Open Geospatial Consortium (OGC), «OGC Simple Feature Access SQL (SFSQL),» 4 8 2010. . Available: http://portal.opengeospatial.org/files/?artifact_id=25354.

[9] PyCSW, «PyCSW Documentation guide,» 20 03 2018. . Available: http://docs.pycsw.org/en/2.2.0/.

[10] P. M. M. S. Stefano Nativi, «ECOPOTENTIAL D10.1 Design of the ECOPOTENTIAL Virtual Laboratory,» . Available: http://www.ecopotential-project.eu/images/ecopotential/documents/D10.1v2.pdf.

[11] Federation of Earth Science Information Partners, «Data Understanding - Quality (ISO-19157),» 06 10 2016. . Available: http://wiki.esipfed.org/index.php/Data_Understanding_-_Quality_(ISO-19157).

[12] DATYPIC, «DataQuality profile,» . Available: http://www.datypic.com/sc/niem20/s-dataQuality.xsd.html.

[13] P. P. W.-C. T. Mauricio A. Hernandez, «Data Exchange with Data-Metadata Translations,» . Available: http://dit.unitn.it/~p2p/RelatedWork/Matching/datametadata.pdf.