D7.3: ENVRI-FAIR Knowledge Base for RI Service Interoperation and Competence

Work Package	WP7
Lead partner	UvA
Status	Final
Deliverable type	Report
Dissemination level	Public
Due date	31-08-2020
Submission date	07-12-2020

Deliverable abstract

The overarching goal of ENVRI-FAIR is for all participating RIs to improve their FAIRness and prepare the connection of their data repositories and services to the European Open Science Cloud (EOSC). With the development of FAIR implementations from the participating RIs and integrated services among the environmental subdomains, these data and services will be brought together at a higher level (for the entire cluster), providing more efficient services for researchers and policy makers.

This deliverable introduces the building of the Knowledge Base in the ENVRI-FAIR context, describes the approach chosen for the knowledge construction, its support for sharing technical practices, identifying common problems and solutions, searching existing solutions for interoperability challenges among environmental RIs, and knowledge-based decisions.

The objective of this task is to build a cluster-level Knowledge Base in order to share technical practices, identify common data and service requirements and design patterns, and facilitate search and analysis of existing RI solutions for interoperability challenges that are shared among environmental RIs.

ENVRI-FAIR (www.envri-fair.eu) has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 824068

DELIVERY SLIP

	Name	Partner Organization	Date
Main Author	Zhiming Zhao	UvA	08-11-2020
Contributing Authors	Xiaofeng Liao, Doron Goldfarb Markus Stocker Siamak Farshidi Barbara Magagna	UvA EAA TIB UvA EAA	
Reviewer(s)	Keith Jeffery Peter Thijsse Christian Pichot	NERC/EPOS MARIS/SeaDataNet INRA/ANAEE	
Approver	Andreas Petzold	FZJ	30-11-2020

DELIVERY LOG

Issue	Date	Comment	Author
V 1.0	13-07-2020	First Draft	Xiaofeng Liao
V 2.0	09-11-2020	Second Draft	Zhiming Zhao
	13-11-2020	Comments from Reviewer 1	Keith Jeffery
	23-11-2020	Comments from Reviewer 2	Peter Thijsse
	26-11-2020	Comments from Reviewer 3	Christian Pichot
V 3.0	26-11-2020	Final version	Zhiming Zhao

DOCUMENT AMENDMENT PROCEDURE

Amendments, comments and suggestions should be sent to the Project Manager at manager@envri-fair.eu.

GLOSSARY

A relevant project glossary is included in Appendix A. The latest version of the master list of the glossary is available at http://doi.org/10.5281/zenodo.3465753.

PROJECT SUMMARY

ENVRI-FAIR is the connection of the ESFRI Cluster of Environmental Research Infrastructures (ENVRI) to the European Open Science Cloud (EOSC). Participating research infrastructures (RI) of the environmental domain cover the subdomains Atmosphere, Marine, Solid Earth and Biodiversity / Ecosystems and thus the Earth system in its full complexity.

The overarching goal is that at the end of the proposed project, all participating RIs have built a set of FAIR data services which enhances the efficiency and productivity of researchers, supports innovation, enables data- and knowledge-based decisions and connects the ENVRI Cluster to the EOSC.

This goal is reached by: (1) well defined community policies and standards on all steps of the data life cycle, aligned with the wider European policies, as well as with international developments; (2) each participating RI will have sustainable, transparent and auditable data services, for each step of data life cycle, compliant to the FAIR principles. (3) the focus of the proposed work is put on the implementation of prototypes for testing pre-production services at each RI; the catalogue of prepared services is defined for each RI independently, depending on the maturity of the involved RIs; (4) the complete set of thematic data services and tools provided by the ENVRI cluster is exposed under the EOSC catalogue of services.

TABLE OF CONTENTS

List Of Images

List Of Tables

D7.3 - ENVRI-FAIR Knowledge Base for RI Service Interoperation and Competence

1 Introduction

This document describes the results and the progress made by WP7 regarding the Knowledge Base (KB) construction and introduces the main components of the Knowledge Base for RI Service Interoperation and Competence.

A cluster-level Knowledge Base, which is obviously not a new idea and had been discussed since at least the prior project ENVRIplus, is now being implemented to enable different users in ENVRI (e.g. RI developers, data managers and users) to effectively share their technical practices, identify common data and service requirements and design patterns, and facilitate search and analysis of existing RI solutions for interoperability challenges that are shared among environmental RIs. The Knowledge Base provides Knowledge-as-a-Service for the RI development communities to document the development and operation of RI services and to address engineering problems.

More specifically, the Knowledge Base will

1. Ingest technical results from ENVRIplus, FAIR assessment (reported in T5.1), key output from task forces (organised by WP5), sub domains and other tasks using a formal language for knowledge representation and proven semantic technologies;
2. Provide services and tools to enable RI developers and data managers to browse, search, retrieve and compare RI technical statuses and technical solutions to development problems via available content;
3. Provide content management tools for specialists in the ENVRI community to ingest new knowledge and control the quality of content;
4. Also provide interfaces to other existing semantic resources, e.g. the service catalogue of a future ENVRI-HUB, to enhance knowledge discovery and cross-RI search, between knowledge services and the online presence of ENVRI resources.

The process started with utilising resources from previous projects, discussing with the community, agreeing on objectives, defining user stories, and implementing prototypes to demonstrate selected functionality.

The deliverable summarises the requirements for the Knowledge Base (section 2), the review of the state of the art (section 3), the architecture design (section 4) and the current implementation (section 5). The deliverable also demonstrates the current system in different scenarios (section 6) and discusses the development agenda for the next phase and the sustainability plan.

2 Requirements

The idea of an ENVRI community Knowledge Base was initially proposed in the ENVRIplus project for documenting the engineering status of each research infrastructures. The initial user stories for the ENVRI knowledge mainly focus on the data manager, RI service or Virtual Research Environment (VRE) developers, e.g., for enabling a developer to check the existence or details of data management solutions from different RIs. A detailed requirement analysis has been made at that time, the output has been summarised in the recent ENVRI book [1] and presented in a conference [2].

After the ENVRI-FAIR project started, we extended the scope of the initial requirements, for supporting the sharing of FAIRness assessment, for identifying the gaps of current FAIRness, for searching knowledge from the broader potential data sources (e.g., metadata catalogue of emerging ENVRI-HUB), and for enabling future integration with Virtual research environments (e.g., from EOSC or other communities).

We derived the following prioritised technical requirements from the early phase:

1. Compatible with Semantic Web technologies. As the most common type for knowledge storage, representation, reasoning, the support of RDF is the core requirement in design and developing of our Knowledge Base. This requirement can include the following specific options, like: RDF import/export, RDF storage, owl import, SPARQL and GeoSPARQL support.
   It is acknowledged that while providing many advantages especially in the context of integrating and operating on heterogeneous knowledge sources and of linking to existing external resources, RDF, but also the overall concept of operating on a non-monolithic set of data

collections, comes with specific limitations as well, such as lack of support for referential integrity. It is nevertheless assumed that the nature of the KB content is of rather non-volatile nature, shifting this aspect more into the background.

2. Semantic search & Query functionality. An interface for search and discovery of Knowledge Base content should be provided, this could be the conventional keyword-based search or a faceted search. Rather than strict adherence to a single controlled vocabulary or keyword set, a semantic search function is further expected to permit search based on 'similar' or 'related' terms, across multiple ontologies/controlled vocabularies.
3. Open and flexible knowledge ingestion. Due to the variance of source types in the ENVRI community, various methods should be supported for knowledge acquisition, like form-based manual RDF ingestion, Questionnaire-based RDF triple generation, existing RDF integration, structured and unstructured information transformation, etc. Certain measures should be considered to facilitate non-technical users adding knowledge in a straight-forward way.
4. Provenance and version control of the knowledge. Considering the typical case where multiple users contribute to the Knowledge Base, provenance is of fundamental importance for monitoring and tracking issues, for example enabling a third party to reproduce the scientific workflow, for an authority to audit the whole process. This especially refers to the tracking of individual additions, deletions, and updates and their administration, i.e. approval, rejection, reversion.
5. User friendly and customisable user interface. A clear and straight-forward user interface is needed for users to fulfil their objectives, like query, (semantic) search. Advanced services like comparison, recommendation are also needed for interested users. Considering the difference between general public users and professional users, two different user interfaces should be provided for them respectively.
6. Scaling and increasing performance. To tackle the growing size of the Knowledge Base, a choice between centralised or distributed storage should be considered. Also should be considered includes the dynamic resource scheduling facing concurrent search/query requests. Other features like collaborative editing are required to enable comment on contributions by other users.
7. API interface. An application programming interface (API) abstraction layer can help make knowledge accessible through applications to facilitate the transaction of knowledge via APIs.

Among those technical requirements, the ENVRI Knowledge Base should play a key role in the ENVRI communities for helping the development of the FAIR data services, and for sharing their best practices.

3 The state-of-art

As there exist off-shelf systems/tools and technologies for each aspect, in this section, we briefly review existing technologies and discuss the selection of the technical choices for our development.

3.1 Knowledge Base overview

In its most simple form, a Knowledge Base can represent a collection of documents dedicated to a specific topic, which can be common solutions to frequently arising problems, such as usually provided in dedicated "Questions and Answers" sections, or a collection of information and advice about a specific topic, such as provided for different research methods in the "Research Methods Knowledge Base" [3]. In the context of environmental research infrastructures, the ENVRI Wiki[1] (see Figure 1) can similarly be considered to be a Knowledge Base, providing information about project context and outputs.

[1] http://mediawiki.envri.eu/index.php?title=Category:Data_for_Science

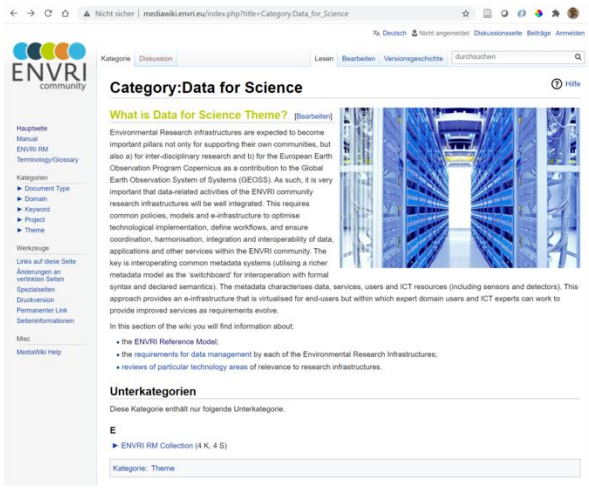

Figure 1. Online wiki of the ENVRI community.

While such approaches are targeted at a human audience and commonly do not even provide any search functionality beyond free text search, other applications seek to provide descriptions of entities relevant for the research infrastructure and processes in a machine- readable/understandable, structured form and to make them available following established representation standards such as RDF and standardised interfaces for querying, such as SPARQL. This allows the creation of sophisticated search and recommendation functionality and especially in the case of Linked Data (RDF) representations, the extensive interlinking of the descriptions with related context information from different sources. Using contextual information which is structured in the form of taxonomies or ontologies lays the foundation to move from mere machine-readability towards machine understanding.

Esonet Yellow Page is another example developed throughout the EU FP6 ESONET Network of Excellence project, the ambition was to collect information about available products for Deep-Sea Observatories and to provide a platform for searching and exploring them. Collected entities included "sensors", "hardware components", "deep sea services" and "manufacturers", especially the former two including not only technical specifications but also information about compatibility and standardisation procedures.

Figure 2. Esonet Yellow pages.

ESONET has a dedicated component called Knowledge Base described in [4]. This approach was based on the concept of a Web platform to aggregate and visualise existing structured information about sensors and observatories obtained from a sensor registry, which allowed the registration of individual sensor installations using sensor metadata derived from ESONET Yellow Pages described above, combined

with related measurement data derived from an archive, the ESONET data catalogue, and from real-time measurements provided via OGC SOS[2] . This concept of the Knowledge Base thus rather followed the vision of a data portal, which was also reflected in its public name[3].
Structured knowledge can be combined with inference mechanisms that can derive explicit facts from implicitly hidden relations found in the available information. Such approaches have in common that the Knowledge Base usually represents an extensive collection of very basic, "low-level" facts which are interrelated via rules of varying complexity and described using dedicated domain ontologies relating the classes of the involved entities on a conceptual level.
While the Knowledge Bases serving the facts for such dedicated knowledge-based systems have traditionally been built and maintained by experts within clearly confined, domain-specific boundaries of knowledge, developments such as the establishment of OBO[4] foundry ontologies, described in [5], have led to a more open approach to collect related facts. A very recent approach, described in [6], goes one step further and proposes Wikidata (as shown in Figure 3), the open Knowledge Base, as a source for life sciences related tasks, including integrative queries, crowdsourced curation, phenotype-based disease diagnosis, and drug repurposing, the latter two based on data-mining approaches. Wikidata, described in [7] was initially conceived as a common fact base for multilingual Wikipedia pages to serve language-agnostic information across multiple language versions of articles about the same thing. In the meantime, it has evolved into an extensive repository of cross-domain knowledge, fed by initiatives from very different domains such as Cultural Heritage[5] or Molecular Biology[6].

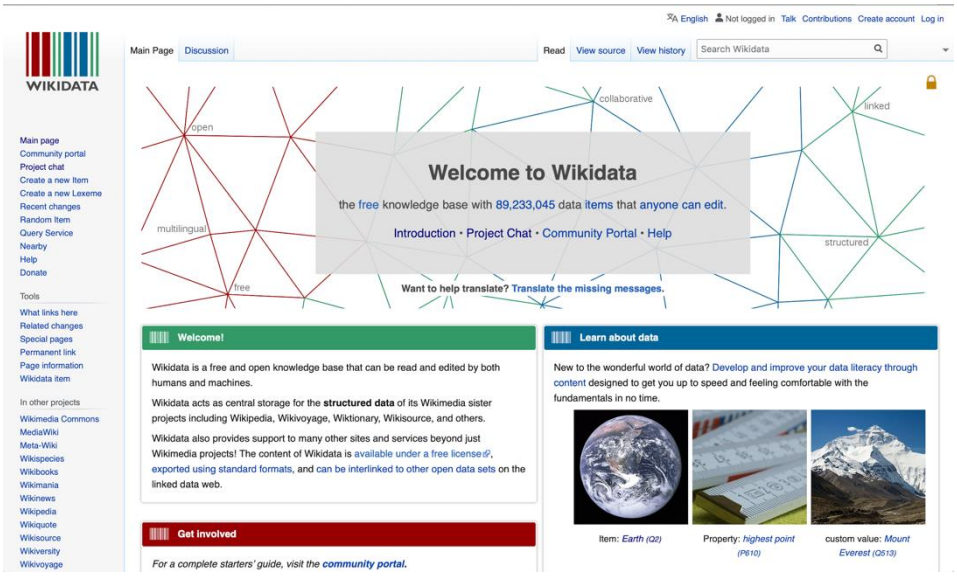

Figure 3. An open Knowledge Base example.

3.2 Technologies involved in knowledge development

We will review the relevant technologies involved in the development of the Knowledge Base according to the key aspects identified by the requirements in the previous section.

[2] The Open Geospatial Consortium (OGC) Sensor Observation Service (SOS) is a specification for Web services allowing to query sensor data in real-time and as time series.
[3] https://dataportals.pangaea.de/esonet/
[4] The Open Biological and Biomedical Ontologies (OBO) Foundry is a group of researchers dedicated to building and maintaining ontologies for the life sciences
[5] https://www.wikidata.org/wiki/Wikidata:WikiProject_Cultural_heritage
[6] https://www.wikidata.org/wiki/Wikidata:WikiProject_Molecular_biology

3.2.1 Knowledge storage

RDF or similar triple-graph-like data structures are widely used in knowledge representation; A range of existing approaches can be used for knowledge storage, from traditional RDBMS to dedicated Graph databases.

Relational database management systems (RDBMS), such as MYSQL[7], are for example used by the popular Mediawiki software to drive the Wikipedia ecosystem. Wikibase, the engine behind Wikidata, introduced in the previous section, is another example in this regard. Additional services such as the SPARQL-based Wikidata query service operate on RDF data which is created via MYSQL-RDF exports taking place in regular intervals.

Triplestores are specialised graph databases dedicated to storing RDF data. Apache Jena[8] is a Java-based framework for handling RDF data. Jena can be used as a hybrid store operating on top of MYSQL (Jena SDB) or as a native Triplestore using its own infrastructure (Jena TDB). TDB can be operated in-memory or using a disk index. Jena Fuseki is a SPARQL server, which can run as an operating system service, as a Java web application (WAR file), and as a standalone server, using TDB to as a robust, transactional persistent storage layer.

Virtuoso[9] is a high-performance and scalable Multi-Model RDBMS, Data Integration Middleware, Linked Data Deployment, and HTTP Application Server Platform. It combines the functionality of a traditional Relational database management system (RDBMS), Object-relational database (ORDBMS), virtual database, RDF, XML, free-text, web application server, and file server functionality in a single system. Virtuoso's RDF engine is a hybrid design operating on top of Virtuoso's RDBMS.

General purpose graph databases can handle any type of node-link based information, while Triplestores are designed to operate solely on RDF-based data. One fundamental distinction between RDF graphs and general-purpose graphs is that RDF does not allow for the annotation of individual triples (i.e. adding attributes to individual links between two instances) without relatively complicated mechanisms such as reification. So-called labelled property graphs in turn assign individual IDs for each link, allowing to attach attributes such as values, categories, etc. Neo4J[10] is an example for such a labelled property graph DB, offering a dedicated query language (Cypher) to access the stored information. In contrast to Triplestores with their standard SPARQL query engines, however, there is currently no comparable general standard available, limiting the usability of individual solutions to the respective platforms.

3.2.2 Knowledge management

Assuming RDF as target data format for the ENVRI-FAIR KB, this section therefore explores solutions to allow data managers to access the content of the Knowledge Base using GUI tools with the aim to explore, search and edit the KBs content in a user friendly manner. In contrast to end user interfaces, however, the considered approaches are rather targeted at administrative tasks.

Five existing solutions to presenting/managing RDF-based content were compared for basic features such as the support for exploring and/or visualising triples, but also for technology related issues such as native RDF/SPARQL support or open-source related aspects such as availability via Github and the time of the last edits performed there. Besides supporting content exploration on triple level, three of the five compared solutions also allowed for content management (Upload, individual editing) and are presented in more detail below, followed by the overall results of the comparison summarised in **Table 1**.

Ontowiki, first described in [8], is a collaborative knowledge engineering platform for RDF based data. With its latest version introduced in [9], it provides a browser-based interface for editing and browsing collections ("Knowledge Bases") of RDF statements and puts heavy emphasis on collaborative editing features such as social comment features and statement-level provenance with history. Implemented in PHP, it can use different data backends, including relational DBMS such as MySQL or Triplestores such as Openlink Virtuoso. Another flexible feature is its plugin-based architecture, enabling the addition of different features such as data visualisation.

Semantic Mediawiki (SMW) is an extension to the popular Mediawiki software driving the well-known Wikipedia universe. It is mainly intended to augment classical Mediawiki markup pages with semantic annotation, serving e.g. to dynamically update certain facts such as dates or quantities. Using Mediawiki as a foundation, it offers all the collaboration features of the software, i.e. discussion and version history.

[7] https://www.mysql.com
[8] https://jena.apache.org
[9] https://virtuoso.openlinksw.com
[10] https://neo4j.com

By default, SMW uses the standard Mediawiki data infrastructure for storage but can be extended to use a Triplestore in parallel. A significant difference to OntoWiki is that in SMW, structured information is used only for annotating entities in Wikitext (In an RDFa like fashion) and to display lists, it is not meant to "drive" the Wiki content itself.

Wikidata uses a proprietary data format that provides a number of features that require workarounds to be represented as RDF, such as n-ary relations or statement-level provenance. Wikidata uses a set of Mediawiki extensions called Wikibase[11] for its data architecture, which therefore includes similar collaboration features as SMW. As described in [10] the data can be converted into an RDF representation for export and a parallel triplestore representation of Wikidata content offers SPARQL-based querying. As of today, however, there is no means to directly import RDF into Wikidata.

Table 1. Comparison of existing platforms/products.

	Semantic Mediawiki	**Wikibase (Wikidata)**	**OntoWiki**
Editor	X (In fulltext or via Page Forms[12]	X	X
Triple Viewer	X	X	X
Visualisation	Optional (e.g. graphextension)[13]	Several built-in features[14]	Optional (Cubeviz)[15]
Built-in UI Features	https://www.semantic-mediawiki.org/wiki/Help:Browsing_interfaces		
Example raw page	https://sandbox.semantic-mediawiki.org/wiki/Main_Page	https://www.wikidata.org/wiki/Q5593	http://aksw.org/model/info/?m=http%3A%2F%2Faksw.org%2F
Example application site	https://practicalplants.org/wiki/Practical_Plants	https://wikidp.org/	https://demo.amsl.technology/OntoWiki/list
Original Purpose	Semantic Annotation of Wiki pages	Knowledge Base + Fact Editor for populating Wikipedia Info boxes	Knowledge Base Editor
Full source on Github	Yes	Yes	Yes
Github URL	https://github.com/SemanticMediaWiki/SemanticMediaWiki	https://github.com/wikimedia/mediawiki-extensions-Wikibase	https://github.com/AKSW/OntoWiki
Last Edit on Github	17.06.2019 / Active	14.06.2019 / Active	11.07.2017/Currently no further dev https://github.com/AKSW/OntoWiki/issues/440
Frontend Technology	Browser	Browser	Browser
Server Technology	PHP	PHP	PHP

[11] https://en.wikipedia.org/wiki/Wikibase
[12] http://edutechwiki.unige.ch/en/Page_Forms)
[13] https://www.mediawiki.org/wiki/Extension:Semantic_MediaWiki_Graph)
[14] https://www.wikidata.org/wiki/Wikidata:SPARQL_query_service/Wikidata_Query_Help/Result_Views
[15] http://aksw.org/Projects/CubeViz.html)

Storage Technology	Relational (can optionally be complemented with TripleStore running in parallel[16]	Relational with mirror Triplestore for Queries	Triplestore or Relational
Native RDF	When using complementary Triplestore	No	Yes
RDF Export (Only for editor)	Yes	Yes	Yes
RDF Import	Via tools, e.g. RDFIO Tool for RDF import, convertion to internal format[17]	Needs longer conversion workflow,[18] would need creation of or mapping to existing Wikibase classes/properties	Yes
Builtin OWL import	No (Could be done via script, converting owl features having counterparts in internal class/property structure)	No (Could be done via script, converting owl features having counterparts in internal class/property structure)	Yes (With limitation)
SPARQL Support	When using complementary Triplestore	SPARQL queries against parallel Triplestore	Yes
Class/Category System available for instantiation	yes; proprietary; subclass/subproperty relationships possible	yes; proprietary; subclass/subproperty relationships possible	Yes; based on OWL
Constraints (Only for Editor)	uniqueness (Property can only be used once per instance) Permitted Values (Controlled list of values to be entered for property) datatype check (One of [19])	Custom[20],[21]	No
Provenance	Statement Level Provenance (references)	Statement Level Provenance (references)	Statement Level Provenance (references)
Revision Tracking	Yes	Yes	Yes

3.2.3 Search Interface

Though some of the knowledge management tools investigated in 3.2.2 provide the GUI tools for querying and searing the Knowledge Bases, their main target users are Knowledge Base administrators in terms of the expertise required to use these tools. Thus, a more general user interface is still expected for easy search experience and context-aware exploration of the Knowledge Base.

[16] https://www.semantic-mediawiki.org/wiki/Help:Using_SPARQL_and_RDF_stores)
[17] https://www.semantic-mediawiki.org/wiki/SMWCon_Fall_2016/Batch_import_of_large_RDF_datasets_using_RDFIO_or_the_new_rdf2smw_tool
[18] https://www.wikidata.org/wiki/Wikidata:Data_Import_Guide
[19] https://www.semantic-mediawiki.org/wiki/Help:List_of_datatypes
[20] https://www.mediawiki.org/wiki/Extension:Wikibase_Quality_Extensions
[21] https://www.wikidata.org/wiki/Help:Property_constraints_portal

We investigate two systems, open semantic search and Elastic search, which provide comprehensive search solutions.
Open Semantic Search is an Integrated tool for easier searching, monitoring, analytics, discovery & text mining of heterogeneous & large document sets & news with free software on the user's own server.
Elastic Search is a search engine based on the Lucene library. It provides a distributed, multitenant-capable full-text search engine with an HTTP web interface and schema-free JSON documents.
A comparison of these two platforms in terms of our requirements is presented in **Table 2.**

Table 2. Comparison between Open Semantic Search and ElasticSearch.

	Open Semantic Search[22]	**ElasticSearch**[23]
Visualisation	1. Visualising like trend charts, word clouds, interactive maps 2. graph/network analysis view[24] 3. Alternatively, enable the Open Source ETL plugin for integration with the Neo4J database and present visualisation with Neo4j browser by Cypher graph query language.	via Kibana(Pie, Bar, Map, etc)
Builtin UI Features	Solr-PHP-UI[25]	Search UI
Full source on Github	Yes	Yes
Frontend Technology	Browser	HTTP web interface
Server Technology	Solr or Elasticsearch open-source enterprise-search	Lucene
Storage Technology	Inverted Index	Inverted Index
Native RDF	native graph storage for graphs including RDF triple stores	index RDF data in JSON format
RDF Export (Only for editor)	Yes[26]	No
RDF Import	Yes[27]	index RDF data in JSON format
Builtin OWL import	Yes[28]	No
SPARQL Support	1. Neo4j has Cypher that covers the need for a structured graph query language 2. But there is workaround like [29]	Elasticsearch provides a full Query DSL (Domain Specific Language) based on JSON to define queries

[22] https://github.com/opensemanticsearch
[23] https://github.com/elastic/elasticsearch
[24] https://www.opensemanticsearch.org/doc/analytics/graph
[25] https://www.opensemanticsearch.org/solr-php-ui
[26] https://www.opensemanticsearch.org/etl/export/rdf
[27] https://www.opensemanticsearch.org/connector/rdf
[28] https://neo4j.com/docs/labs/nsmntx/current/importing-ontologies/
[29] https://community.neo4j.com/t/sparql-for-neo4j/19583/3

3.2.4 Knowledge Graph Visualisation

To present the content in the Knowledge Base with a knowledge graph visualisation, we investigate several tools with network graph features. They are:

D3.js is a JavaScript library for manipulating documents based on data. Different kinds of data can be bound to a DOM and then different kinds of functions may be executed on it. One of those functions includes generating an SVG, canvas, or HTML visualisation from the data in the DOM. The complicated part of D3 (or any embeddable library that doesn't have direct Neo4j connection) is converting the graph data into the expected map format for export. D3 expects two different collections of graph data - one for nodes and one for links (relationships). Each of these maps includes arrays of properties for each node and relationship, that d3 then converts into circles and lines. Version 4 and 5 of D3.js also support force-directed graphs, where the visualisation adjusts to the user's view pane.

Popoto.js is one kind of the tools that are embeddable tools with built-in Neo4j connections. This kind of embeddable tool can be included as a dependency within an application and can easily be configured and styled for an application and Neo4j. This tool can easily connect to an instance of the Neo4j graph database using configuration properties and allows to style the visualisation based on nodes, relationships, or specific properties. Popoto.js is a JavaScript library that is built upon D3.js.

Neo4j Bloom is a standalone product tool that helps data exploration and visualises data in the graph and allows users to navigate and query the data without any query language or programming.

Grao.fo a tool for visually designing knowledge graphs with online, collaborative, real-time editing features. Grafo has reused the existing WebVOWL standard rather than reinventing the wheel.

WebVOwl a web application for the interactive visualisation of ontologies. An example of using WebVOWl can be found at[30].

Several practical issues need to be considered when choosing from these tools. A comparison of these tools in terms of the several practical issues is presented **in Table 3.**

Table 3. Comparison between Knowledge Graph Visualisation Options.

	Popoto.js[31]	**D3.js[32]**	**Neo4j Bloom[33]**	**Grafo[34]**	**WebVOwl[35]**
Open source	Open source licensed under GNU General Public License v3.0	open source licensed under BSD license.	open source licensed under GPLv3.	No. offers a Free/Student tier with basic functionality.	Open source released under the MIT license
RDF support	NA	NA`	Yes, via the neosemantics (n10s) plugin.	supports importing OWL (RDF/XML) and Turtle file formats.	visualisations are automati-cally generated from JSON files into which the ontologies need to be converted
SPARQL Support	JavaScript naturally fits for querying a SPARQL endpoint which provides a REST service returning the result in the JSON format	JavaScript naturally fits for querying a SPARQL endpoint which provides a REST service returning the result in the JSON format	No, Neo4j has Cypher that covers the need for a structured graph query language.	No	Yes

[30] http://www.visualdataweb.de/webvowl
[31] popoto.js
[32] D3.js
[33] https://neo4j.com/bloom/
[34] http://gra.fo/
[35] https://github.com/VisualDataWeb/WebVOWL

3.3 Gap Analysis

Here we revisit the requirements and analyse the gap for the tools or platforms we investigated in terms of the requirements identified in section 2.

1. **Compatible with Semantic Web technologies**. As the most common type for knowledge storage, representation, reasoning, the support of RDF is the core requirement in design and developing our Knowledge Base. This requirement can include following specific options, RDF import/export, RDF storage, owl import, SPARQL support, etc.

   The two storage solutions (Apache Jena and Virtuoso) are triplestores that are dedicated to storing RDF data, thus fully meeting the requirements of semantic web technology compatibility.

   *Regarding the knowledge management solutions, as the comparison in **Table 1** indicates, both Semantic Mediawiki and Ontowiki are RDF compatible.*

2. **Semantic search & Query functionality.** An interface for search and discovery of Knowledge Base content should be provided, this could be the conventional keyword-based search or faceted search. Rather than strict adherence to a single controlled vocabulary or keyword set, a semantic search function is further expected to permit search based on 'similar' or 'related' terms.

   Though the knowledge management tools investigated (like Ontowiki, Semantic Mediawiki) allow users to explore, search and edit the content of the Knowledge Base via GUI tools, they are still lacking easy user experience in terms of the technology required. The original purpose of both Semantic Mediawiki and Ontowiki are semantic annotation of wiki pages, and as Knowledge Base editor respectively.

3. **Open and flexible knowledge ingestion.** Due to the variance of source types in the ENVRI community, various methods should be supported for knowledge acquisition, like form-based manual RDF ingestion, Questionnaire-based RDF triple generation, existing RDF integration, structured and unstructured information transformation, etc. Certain measures should be considered to facilitate non-technical users adding knowledge in a straight-forward way.

   *As shown in **Table 1**, several knowledge management tools, like Semantic Mediawiki and Ontowiki, support RDF import, which facilitates the ingestion of knowledge. However, to prepare RDF triples, or transform the information needed into knowledge, some customised tools needed to be designed and implemented considering the diversity of information sources in our project.*

4. **Provenance and version control of the knowledge.** Considering the typical case where multiple users contribute to the Knowledge Base, provenance is of fundamental importance. This especially refers to the tracking of individual additions, deletions, and updates and their administration, i.e. approval, rejection, reversion.

   As far as the considered knowledge management platforms are concerned, Ontowiki meets the requirements by providing detailed user management and statement-level provenance for RDF data, allowing to track and potentially edit individual user contributions to the Knowledge Base.

5. **User friendly and customisable user interface**. A clear and straight-forward user interface is needed for users to fulfil their objectives, like query, (semantic) search. Advanced services like comparison, recommendation are also needed for interested users. Considering the difference between general public users and professional users, two different user interfaces should be provided for them respectively.

   As already analysed, although the knowledge management tools provide a GUI for search and query, their targeted users are Knowledge Base administrators considering the technology

barriers. For general users without much technological knowledge of the SPARQL or triplestores, an easy and straight-forward user interface for searching and exploration is expected to increase the user experience.

6. **Scaling and increasing performance**. To tackle the growing size of the Knowledge Base, a choice between centralised or distributed storage should be considered. Also should be considered includes the dynamic resource scheduling facing concurrent search/query requests. Other features like collaborative editing are required to enable comment on contributions by other users.

   Apache Jena Fuseki doesn't currently support horizontal scale up, but there are workaround solutions like by coordinating the updates from a staging server and being a publishing (read-only) to the external clients.

Based on the comparison, it is clear that no one single solution satisfies all the requirements. The optimal solution should be a combination of existing options and other softwares such as Blazegraph could be a candidate.

4 System Design

In this section, we first present the architecture design of the Knowledge Base, then describe how it works in several scenarios with corresponding sequence diagrams.

4.1 Architecture of KB

We will describe the Knowledge Base architecture using the multi view approach based on the ODP approach [11,12,13,14]. This approach has also been used to develop the ENVRI RM.

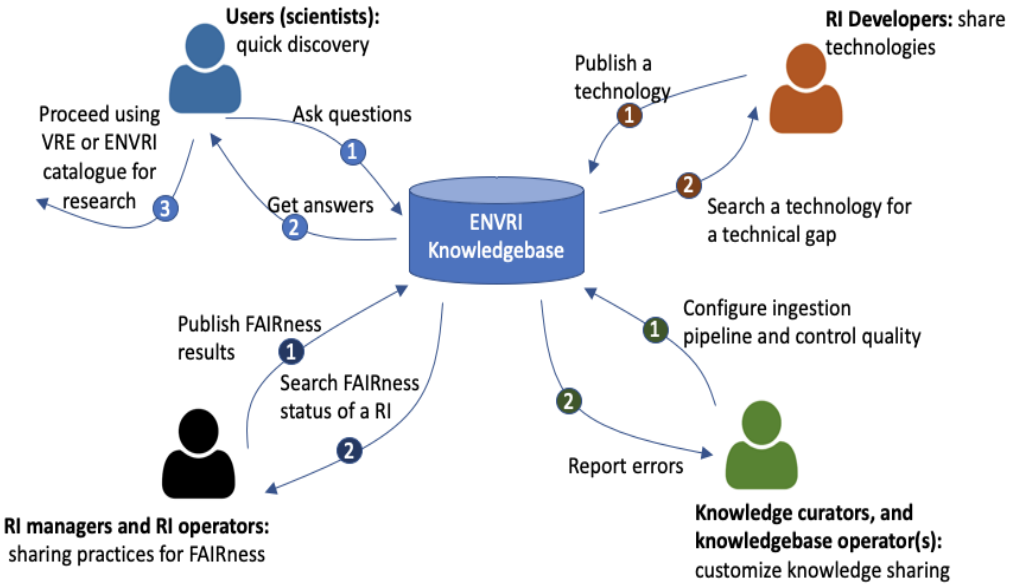

Figure 4. An enterprise view of the envri Knowledge Base. The enterprise view highlights the key stakeholders (namely communities in the ODP term) and their interaction scenarios with the Knowledge Base. The numbered circles indicate the possible orders of the interactions.

In **Figure 4**, four key user types are highlighted:

1. **End users** may use the KB to find answers to their general questions about available sources of data, services and tools, and to use the discovered information to perform further research

activities using the other tools like Virtual Research Environments, or services like the RI catalogues of data or services.

2. **RI managers or operators** may use the KB to check the status of the FAIRness of specific repositories, or update the state of their own RIs. The update process often needs the KB tools for ingestions of FAIRness output from the other tools, e.g. the assessment wizard tool.
3. **RI developers** may use the KB to check the existing technologies, e.g. those development results in the ENVRI portfolio, or the demonstrators prepared for some known FAIRness gaps. They can also publish or update the technical descriptions using the KB tools, like the tool description online form.
4. **Knowledge curator and the Knowledge Base operators** may use the KB to ingest content from new sources, and respond to the possible errors occurred during the ingestion, or during the operation.

Based on those scenarios, we designed the functionality components of the Knowledge Base from the computational views. **Figure 5** shows the key components via three layers:

1. **The interface layer** atop contains components dealing with user related activities. The Knowledge Base will be an open system for community users; the user management component is not for acquiring and processing users' personal information, but more for providing customised user support based on their interaction or contexts. A user can log into the system using an open identity provider. The User Interface (UI) components provide the mechanism for users to interact with the application. They format data and render it into different presentations to meet different users' needs, and acquire and validate data entered by users.

2. **The service layer** abstracts the functionality that the Knowledge Base offers; it can be roughly split into three sub-layers, namely:
    - The Application sub-layer provides customised application logic (e.g., FAIRness gap gnalysis, engineering support, or discovery knowledge from ENVRI community) based on the data passed from the underlying discovery sub-layer, and passes those results up to the User Interface Component.
    - The Discovery sub-layer provides the functionality for searching the Knowledge Base, ranking the results, and recommending relevant content.
    - The Content sub-layer provides functionality for managing the content in the Knowledge Base, typically in a pipeline covering: ingesting information, transformation from information to knowledge, quality control of the knowledge generation, CRUD (Create, Read, Update, Delete) of the Knowledge Base content, and the provenance of these activities.

3. **The storage layer** at the bottom is responsible for data storing and access. The data storage options needed in this project includes: RDF Triple Store and Inverted Index.

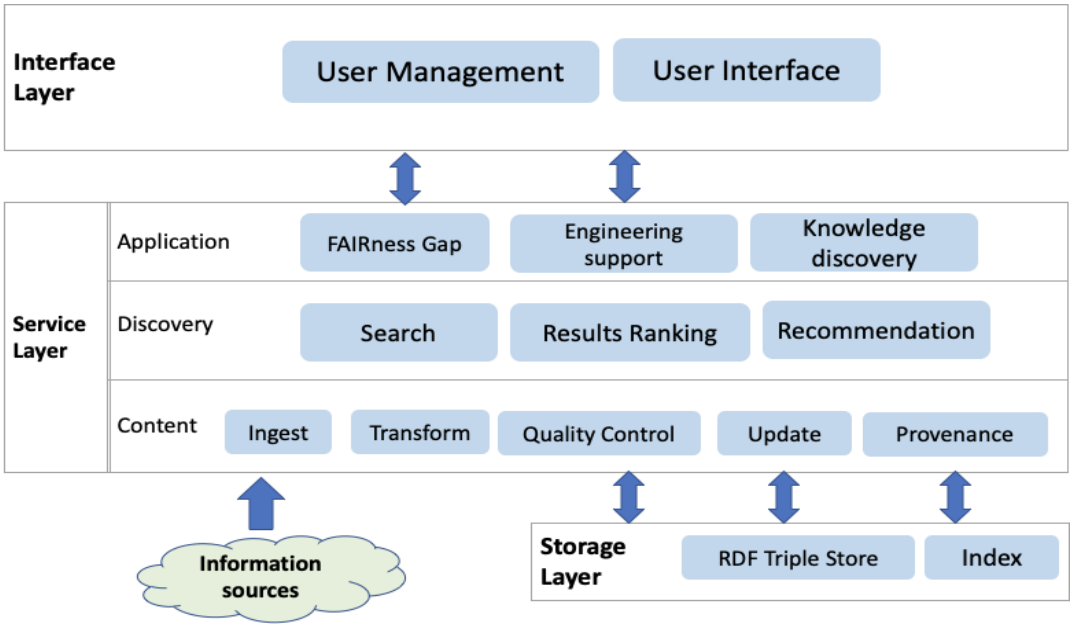

Figure 5. Architecture layers.

Currently, information collected in the knowledge consists of two main parts, as illustrated in the **Figure 6** below.

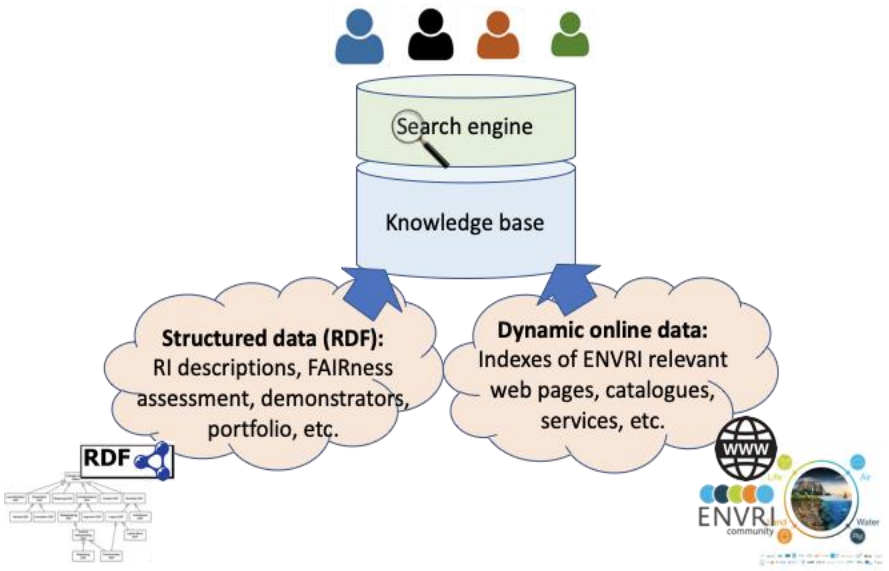

Figure 6. Knowledge Base content components.A Knowledge Base search engine is provided for different end users to search different contents.

The structured data in the Knowledge Base is based on RDF, and mainly includes:

1. OIL-e (ontology of the ENVRI Reference Model) based ENVRI RI description,
2. Description of the service portfolio from the previous project, and the possible new ones in ENVRI-FAIR,
3. FAIRness principles, and the results of the assessment of the ENVRI research infrastructures (D5.1), and
4. Demonstrators for tackling the known gaps, e.g. those being identified during the FAIRness assessment

The versions of the structure data currently can be managed via version control systems. Currently GitHub is used.
The dynamic data in the Knowledge Base will be ingested from different online sources of the ENVRI communities. Figure 7 depicts the basic information flow of the knowledge ingestion.

1. A significant amount of KB relevant information is represented in human readable form, residing in Wikis, other content management systems or even static Web-pages, in "offline" text found in various documents such as books, project deliverables or scientific publications. In the ENVRI-FAIR context, the research infrastructure websites are a good resource of related information, including news/events, background knowledge, etc. The community websites, like ENVRI[36], ENVRI-FAIR[37], also contain lots of related information, like news/events, community introduction, community landscape, projects information, progress, etc. These information sources have different formats, like webpage, word document, pdf file, etc.
2. Another approach to populate the KB would therefore be to process such free-text information with the aim to extract structured, machine readable information. Named entity recognition would represent a first step in this regard, while the application of more complicated Natural Language Processing operations could be a valuable field of research in its own regard.
3. Information from the available catalogues of data and services. It should be clear that the indexes generated from those sources will not aim to replicate the entire catalogues, but for providing quick searching capability for community users. For some RI such information will be already managed in RDF format and accessible from triplestores

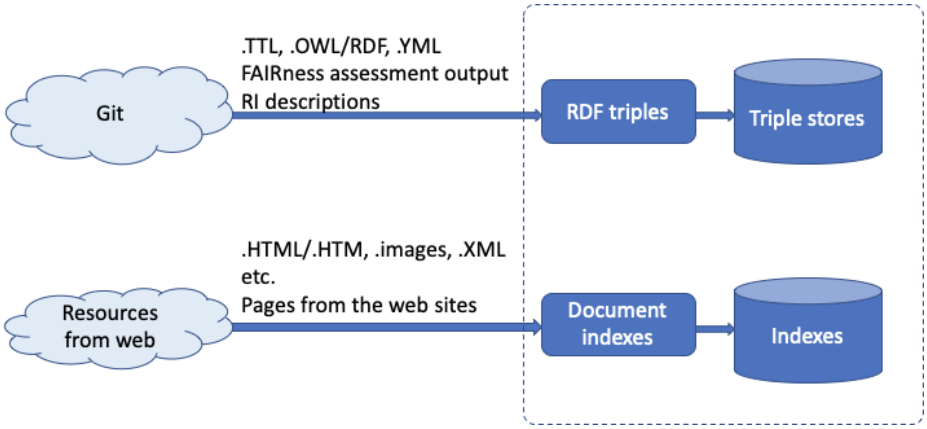

Figure 7. Basic information flow of the knowledge ingestion.

4.2 How does it work

We use three sequence diagrams to explain how the Knowledge Base architecture works. **Figure 8**. depicts how a user interacts with the functional components of the Knowledge Base to perform search activities. In the scenario, a user can send questions to the Knowledge Base via the user interface components, and those questions will be transformed as queries to the backend knowledge storage component (including both RDF and indexes). The output will be sent back to the user after being ranked.

[36] https://envri.eu
[37] https://envri.eu/home-envri-fair/

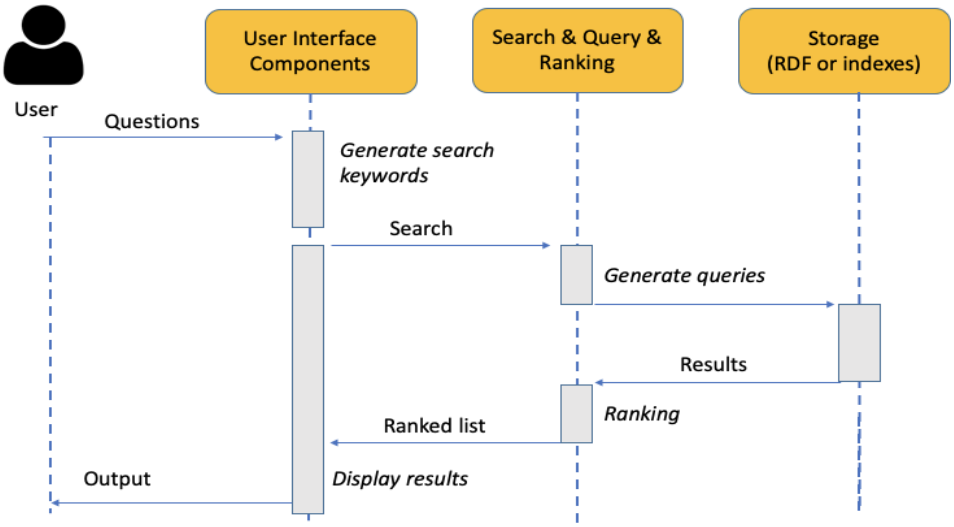

Figure 8. Search the Knowledge Base.

Figure 9. depicts how a technology provider (e.g. a RI data management service developer) shares the technology via Knowledge Base using the description form (portfolio input tool) provided by the Ingestion component. The ingest component and quality component will check the input and interact with the user to store the validated input to the storage.

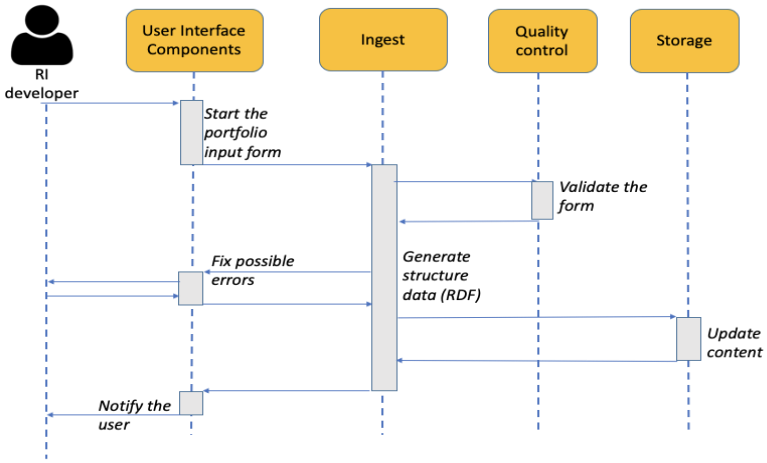

Figure 9. Describing a technology using the portfolio input tool provided by KB.

Figure 10 shows the basic activity sequence of a knowledge operator to ingest knowledge from different sources. The ingestion pipeline may use API or interface of those information sources, for instance if source is from GitHub, the git interface will be used. In this scenario, a Knowledge Base operator can configure the pipeline using the user interface component, and responses to the possible errors generated by the quality control component during the ingestion.

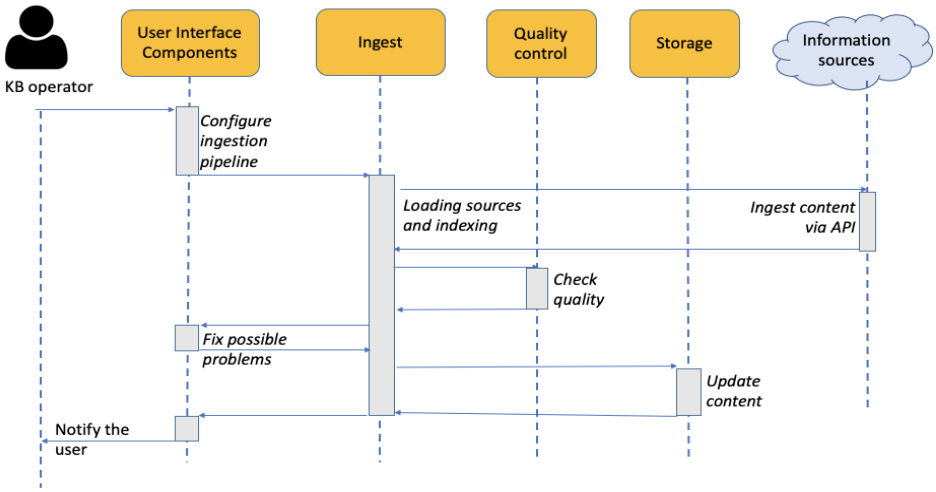

Figure 10. Knowledge ingestion from different information sources.

5 Implementation

5.1 Current prototype

The Knowledge Base development follows an interactive approach, in which prioritised user stories have been analysed, and technical choices were selected based on state-of-the-art review done in the section 3. In the current prototype, we use Ontowiki to manage the RDF triples and open semantic search to develop the search tool for the Knowledge Base. A number of tools were developed for ingesting specific knowledge, e.g. a technology description forms for describing service portfolio, interactive graph visualiser for the search results, and dynamic online data ingestion pipeline. These tools will be described in the following sections.

5.2 Knowledge storage

The comparison of existing RDF content management platforms summarised in Table 1 in section 3.2.2. It was suggested to consider OntoWiki for managing RDF content. The main reasons for this decision were as follows:

- Direct operation on RDF triples: Ontowiki can directly operate on a Triplestore as the underlying storage layer and provides an API to populate it with RDF.
- Integrated User management and statement-level provenance: Ontowiki supports user management with varying permissions and also offers a detailed create/update/delete-history on RDF statement level.
- Named-Graph based separation of RDF content and administrative data: RDF data ingested via Ontowiki is directly written as-is into the underlying Triplestore, while all the administrative statements such as provenance etc. are stored separately.
- Plugin-based extensions: Ontowiki offers a framework for developing plugin extensions

The choice of Ontowiki had a direct effect on the choice of the underlying Triplestore, since Ontowiki provides a pre-configured connector to the Openlink Virtuoso data management system, which members of the KB team already had experience with from previous projects. The open-source edition[38] of Openlink Virtuoso (Version 7.2.5.1) was therefore deployed for that purpose and configured for Ontowiki (and vice-versa).

[38] http://vos.openlinksw.com/owiki/wiki/VOS

5.3 Tools for ingesting knowledge

The population of the Knowledge Bases can take different routes. On the one hand, existing collections of information can sometimes be transformed so that they can be "bulk" imported into the Knowledge Base, which includes rearrangements and mappings of existing collections of structured information but potentially also the extraction of structured content from unstructured sources such as free text, which by no means an easy task considering the complexity in the natural language processing/understanding. On the other hand, it is usually also possible to add Knowledge Base content manually, "fact by fact", though manual input can be slow, tedious and error-prone if not supported by dedicated tools. In the context of the ENVRI Knowledge Base, it should be possible to provide content in both ways.

5.3.1 Data acquisition for service portfolio based on Web-input-forms

Emanating from the initial requirement to find a suitable structured representation for describing the ENVRI Service Portfolio, the data model shown in Figure 11 was designed for representing information about services (left), software (upper right), use cases (centre) and documents (bottom right). Considered to be instantiated as RDF, it aimed at reusing existing related schemas as much as possible, with future integration with other data sources in mind. While the data schema for services was mainly based on the FITSM39 approach as implemented for the EOSC catalogue in eInfraCentral, the data schema for software was derived from the Software Ontology (SWO) from the OBO Foundry universe. The simple data schema for use cases was created from scratch and documents were represented using the standard Dublin Core dcterms element set. A focus was put on reusing additional existing vocabularies for properties, mainly schema.org, as much as possible. In order to foster interoperability on value level, the eInfraCentral terminology40 for classifying services was reused wherever feasible.
Due to potential GDPR issues, the current version of the data model explicitly refrains from representing dedicated person records, which would allow representing responsibilities in an efficient and reusable way. It offers simple literals for entering the appropriate information instead, allowing users to enter less sensitive information such as department names/email addresses, explicitly accepting potential ambiguities in this regard. Moreover, for the sake of simplicity in the RDF representation, links between the described entity records are currently expressed as simple triples and not as complex relationship assertions, allowing faster querying and more intuitive understanding of the structure of the information at the cost of reduced expressiveness. Depending on user feedback and experiences with using the data model in practice could lead to changes in future data model revisions.

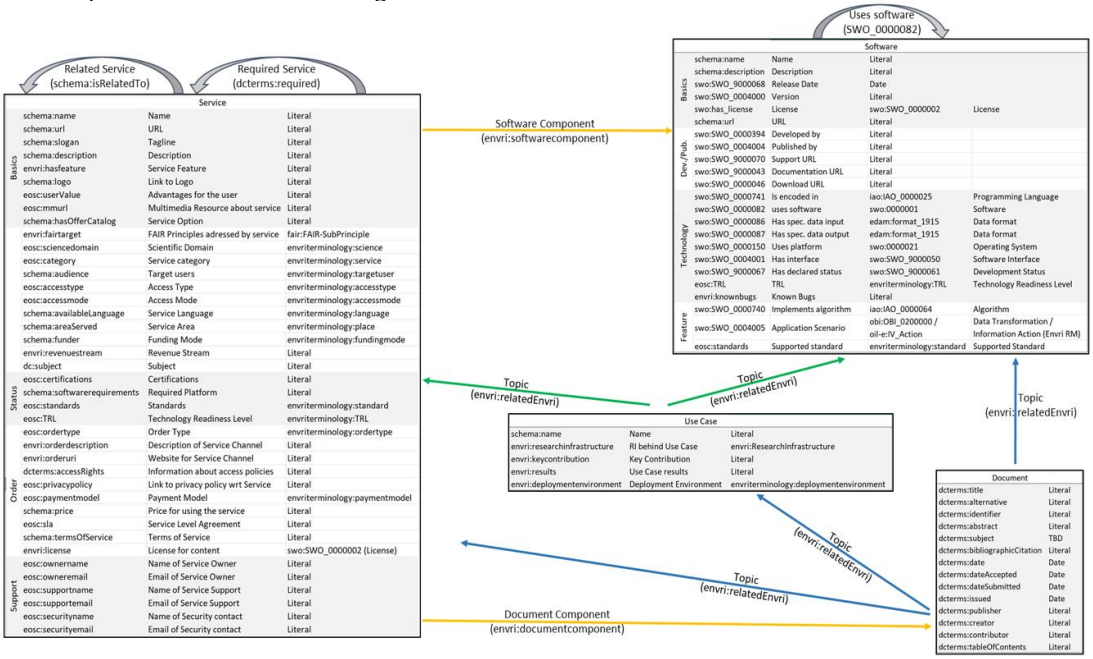

Figure 11. Data model for describing services, software, related use-cases and documents.

[39] FitSM is a free and lightweight standards family aimed at enabling effective IT service management.
[40] https://github.com/eInfraCentral/docs/blob/master/eInfraCentral_ServiceClassification_v2.0.xlsx

The rdforms library was chosen as the tool to automatically create Web-input-forms for entering the information outlined in **Figure 11**. This was mainly motivated by the open-source nature of the package, its state of active development and the ability to represent custom constraints which could be handled by extending predefined stubs in plugin-like handlers. Such custom constraints were crucial features which enabled the creation of re-usable dynamic option menu components loading their available choices directly from the Triplestore using standard SPARQL 1.1 requests. **Table 4** provides a rdforms specification snippet describing such a dynamic option menu loading class labels from an external resource where the selected classes must all be a subclass of "programming language" (IAO_0000025).

Table 4. Example for rdforms template for dynamic option menu.

```
{
  "type": "choice",
  "nodetype": "RESOURCE",
  "property": "http://www.ebi.ac.uk/efo/swo/SWO_0000741",
  "cardinality": { "min": 1, "pref": "1"},
  "constraints": {"http://www.w3.org/2000/01/rdf-schema#subClassOf":"http://purl.obolibrary.org/obo/IAO_0000025"},
  "OntologyUrl": "http://90.147.102.53/OntoWiki/index.php/EnvriServicePortfoliowithexternalTerminology/",
  "label": { en": "Is encoded in"},
  "description": { "en": "Programming language(s) the software is encoded in" },
  "styles": ["multiline", "pathExpr"]
}
```

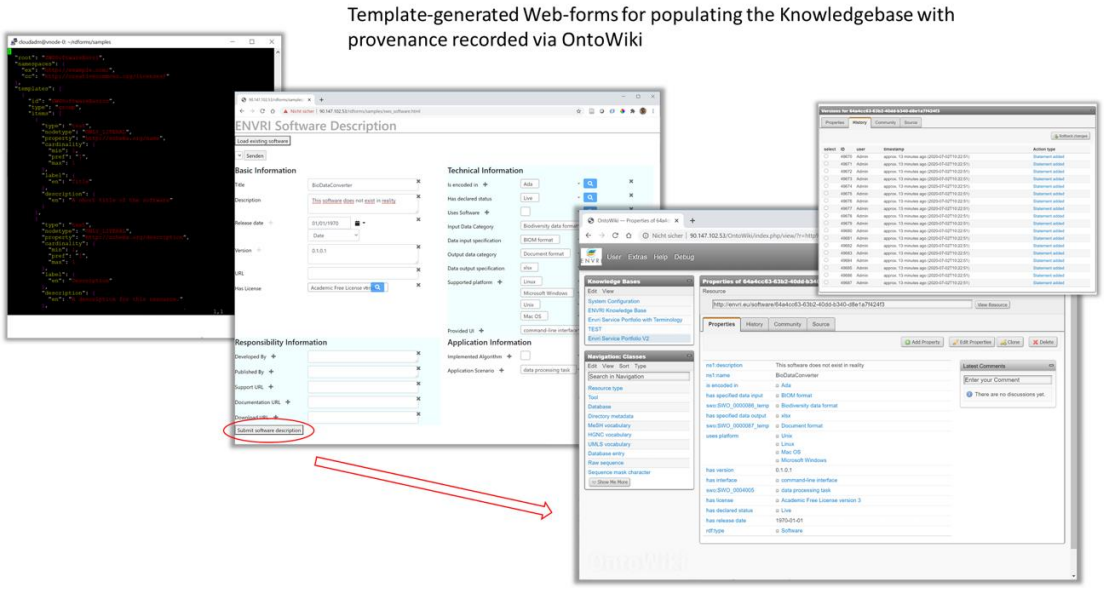

Figure 12. From rdforms specifications to triples.

Figure 12. provides a sketch of the "flow" from a rdform specification to the respective triples. Sets of specifications as shown in Table 4 are read by the rdforms library and converted into sets of automatically created html elements which operate on an RDF graph storing all user form inputs accordingly. The resulting input graphs are submitted to Ontowiki, where the individual statements can be explored, including a detailed history when they were added by whom. Ontowiki in turn writes the provided RDF into the connected Virtuoso Triplestore which provides direct SPARQL access to the collected data.

5.3.2 From Questionnaire to FAIRness

The full process of how WP5 has collected FAIRness assessment results together with WP8-11 has been described in D5.1. A short summary of the process follows here.
The questionnaire of the FAIR Convergence Matrix with 53 questions were used to collect information from participating RIs. The GO FAIR team provided a spreadsheet of these questions with explanations and example answers. In addition, references to the FAIR principles where appropriate were linked to the questions. In addition to this approach, it was decided to use the FAIR Maturity Indicator (hereafter FMI) 'generation 1' questionnaire with 25 questions.

The surveys were distributed through the leads of the subdomains (WP8-11 leads) to representatives of the participating RIs. The survey was conducted in the period between March and May 2019 using Google Forms. All responses from the RIs were collected in a Google spreadsheet.
The answers (which were collected in spreadsheets-XLS as mentioned above) were converted, and the extracted key information was transformed into a structured form in YAML (Yet Another Markup Language) format, following a template also written in YAML. This format was chosen for its conciseness and readability as well as for the fact that it requires minimal extra information to encode answers. The sequence of the YAML attributes is aligned with the questions in RDM+. While making this conversion the answers were translated as much as possible from free text to reference lists (same label for same concept/responses). Concretely, a Knowledge Base was implemented in the form of a triple store using RDF as the data model. Hence, as an additional step, the information in YAML was converted into RDF. The YAML documents were converted into an RDF document (data.trig file) using a fully automated script implemented in Python as a Jupyter notebook that can be executed on EGI Notebooks service.
In the next phase of the project the exercise of assessment will be repeated, however in that case the DS-Wizard will be used which will automatically have RDF triples as output.

5.3.3 Online content ingestion pipeline

The Search Engine of the Knowledge Base is implemented using the open semantic search technology; it also enables end-users to ingest data from pre-existing datasets and Knowledge Bases. The data can be the result of a SPARQL query and should be converted to a CSV format. **Figure 13** illustrates the interface for the ingestion of this type of data.

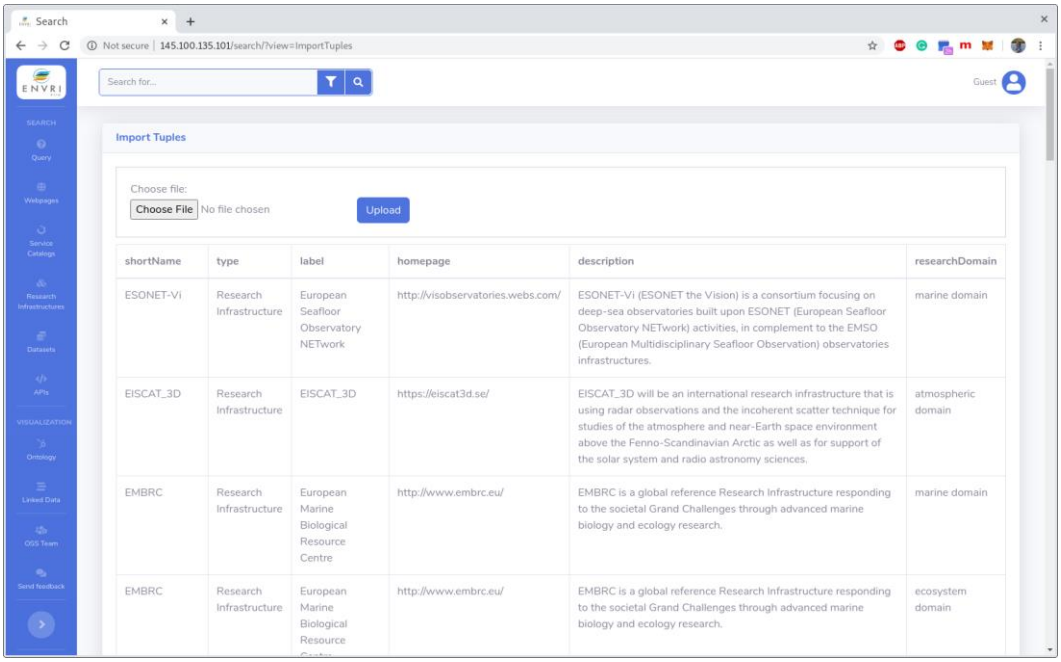

Figure 13. Ingestion of pre-existing datasets and Knowledge Bases.

6 Demonstration

We will demonstrate the current Knowledge Base via four typical user stories:

1. **As a data manager** in a RI, I have to improve the FAIRness of my data, and I want to check if other RIs face same problem or have had working solutions;
2. **As a data management service developer**, I developed a useful service, and I want to share it with the community. Can I make the information available in the Knowledge Base, so that the other colleagues can find it?
3. **As a semantic web specialist**, I want to develop tools for some new innovations, e.g., for semantic search or recommendation. I want to check the content in the Knowledge Base, in particular in the form of RDF, triples, and the end points etc.
4. **As a user** in the ENVRI community, I am curious what resources or services the infrastructures provide, and I want to use the Knowledge Base to search relevant information;

6.1 FAIRness status sharing and gap analysis

A prototype is developed to support the discovery of gaps in FAIR principle implementation at the granularity of RI repositories and the discovery of possible technology solutions to address such gaps. The prototype can be accessed at the following address https://envri-fair.github.io/knowledge-base-ui/ and **Figure 14** is a screenshot of the application.

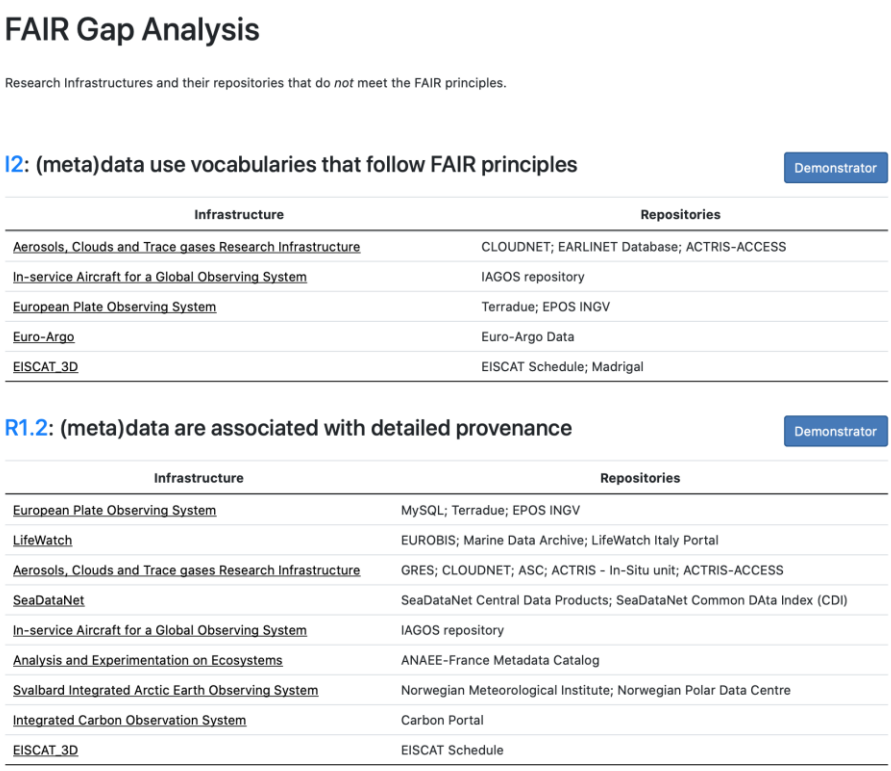

Figure 14. Screenshot of the running the prototype for discovery of FAIR-ness gaps at the granularity of RI repositories and corresponding Technology Demonstrators.

The list in the screenshot is dynamically created by querying the ENVRI KB using the OntoWiki SPARQL endpoint. Indeed, by modifying the FAIR-ness Assessment of a repository of a particular RI—which is functionality natively supported by OntoWiki—for instance the information on whether or not the repository has machine readable provenance information, the interface automatically adapts to either include or exclude the corresponding repository under the relevant FAIR principle (R1.2 in case of provenance). By selecting an RI, the user interface presents a summary view for the RI. A further

example showing the summary of repositories belonging to Aerosols, Clouds and Trace gases Research Infrastructure is presented in **Figure 15**.

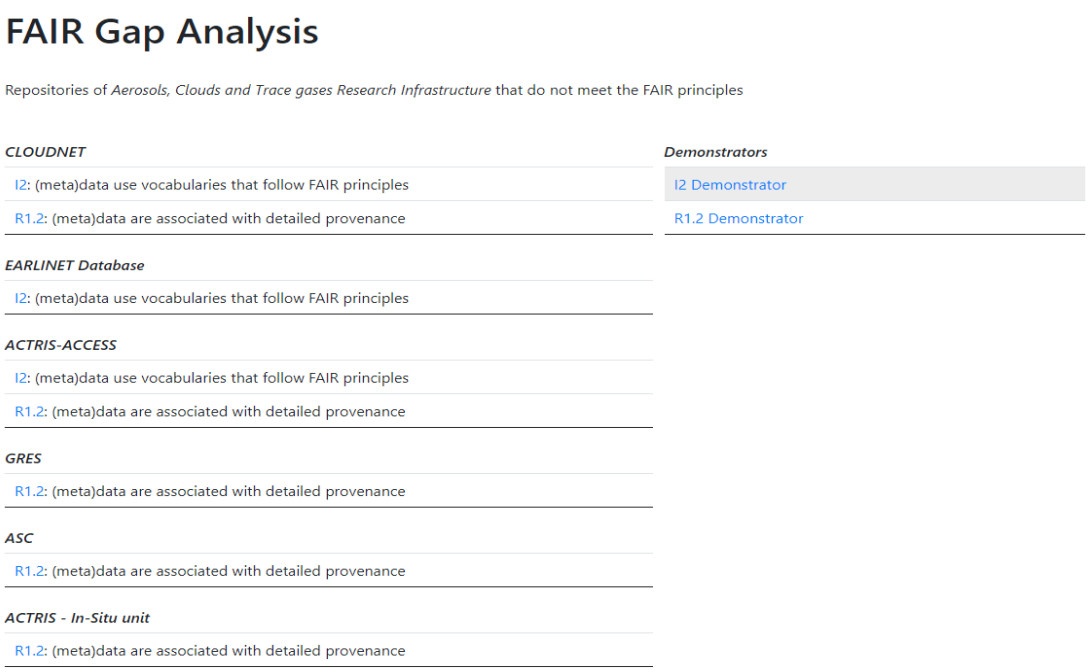

Figure 15. Summary of related repositories of Aerosols, Clouds and Trace gases Research Infrastructure.

6.2 Ontowiki as Knowledge Management Platform

As described in Section 3.2.2, Ontowiki was found to be a suitable RDF data management platform. A test instance was configured to run at http://ontowiki.envri.eu/ and slightly customised to use the ENVRI logo and to display the ENVRI RSS news feed at the front page, as illustrated in **Figure 16** below. It currently serves as a data gateway for, primarily, the facts added via forms, described in Section 6.2 and the gap analysis data described in Section 6.1, based on the FAIRness analysis.
User accounts for Ontowiki can be provided upon request. No account is needed for read-only access.

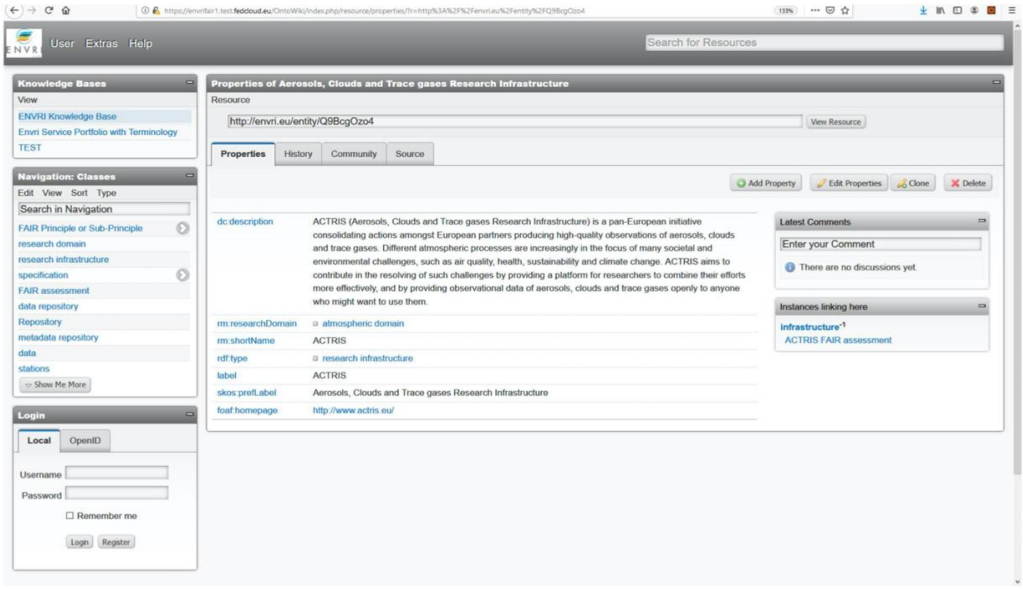

Figure 16. Ontowiki User Interface.

Ontowiki was found to perform well as RDF "middleware" used to ingest data provided from the RDF forms. Some issues were discovered regarding the cross-referencing of statements between Knowledge Bases (=named graphs). A workaround published in a newsgroup provided a potential fix for static data but would have to be extended for a continuously growing data collection. A possible solution would be to store information which is expected to change/grow, e.g. the entity descriptions and the user terminology collected from the RDF forms, in a common named graph and to configure Ontowiki filters for its efficient navigation, while storing more static content, such as external ontologies, in separate graphs. While Ontowiki supports flexible navigation and editing of data at the RDF statement level, the interface is arguably not appropriate for the vast majority of RI managers or developers. Indeed, we conducted some experiments with the atmospheric domain but RIs didn't engage with the user interface. This is to be expected since Ontowiki relies on a good understanding of the RDF data model. Moreover, and more importantly, presenting information at the granularity of the RDF statement is typically inadequate for high level information needs, e.g. discovery of FAIR gaps in the data centers of an RI. We thus suggest that Ontowiki can act as an RDF-based middleware that powers high level user applications and services. One important aspect of using Ontowiki to manage the generated RDF data will be the question of versioning. While built-in features such as the statement-level provenance in principle allow detailed tracing of changes/revisions of the provided data, a backup strategy using external means should be considered as well. One straight-forward step would be to export complete RDF dumps of the provided content in regular intervals and to track their versions in source code repositories such as Github.

6.3 Describing new tools in Knowledge Base

The objective of this demonstrator was to enable users to add facts about RI-relevant services, software, documents and related use cases to the Knowledge Base, without having to deal with complex data formatting issues. It was implemented as an RDF-generating Web-form based on Rdforms described above, using OntoWiki as the user and data management layer operating on top of a Virtuoso Triplestore. Currently, there are four forms following the data model outlined in Figure 11, shown in **Figures 17, 18, 19 and 20**.

Software[41] **(Figure 17).** Description of software loosely based on the Software Ontology (SWO). A described software can use another software etc. enabling the description of complex dependencies. It is possible to add one or more specific application scenario(s) to each description, which also includes IV actions from the ENVRI RM.

[41] http://90.147.102.53/rdforms/samples/software_description.html

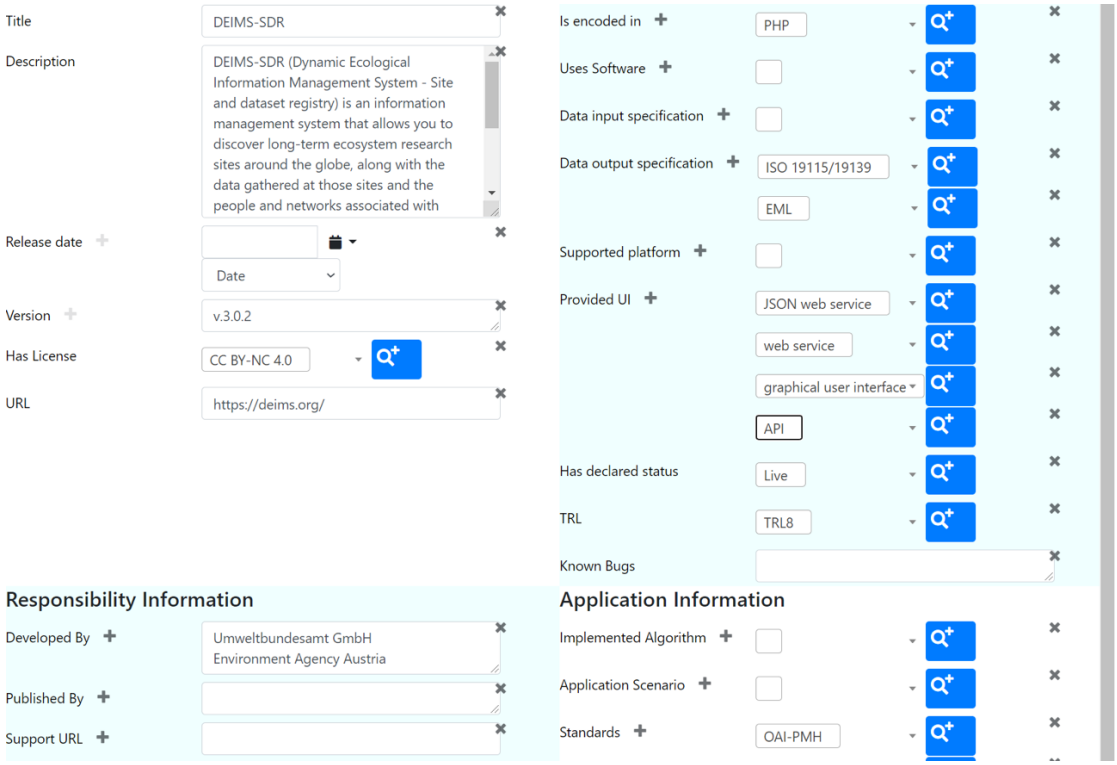

Figure 17. Describing software components.

Documents[42] **(Figure 18).** Documents play multiple roles in the data model. On the one hand, a document can be a "standalone" entity, such as a field manual, a book, etc. On the other hand, a document can be part of a service, e.g. teaching material. Last but not least, a document can also be a publication about an ENVRI service, software or use case. This has to be considered when adding information about a document. Dedicated fields "Related ENVRI software" and "Related ENVRI service" should be used when describing documents which were published about the respective service or software (i.e. a scientific paper, a manual, etc), resulting in triples having the document as subject and the software/service as object, while documents serving as components of a service should be linked from the service, resulting in triples pointing from service to document.

[42] http://90.147.102.53/rdforms/samples/document_description.html

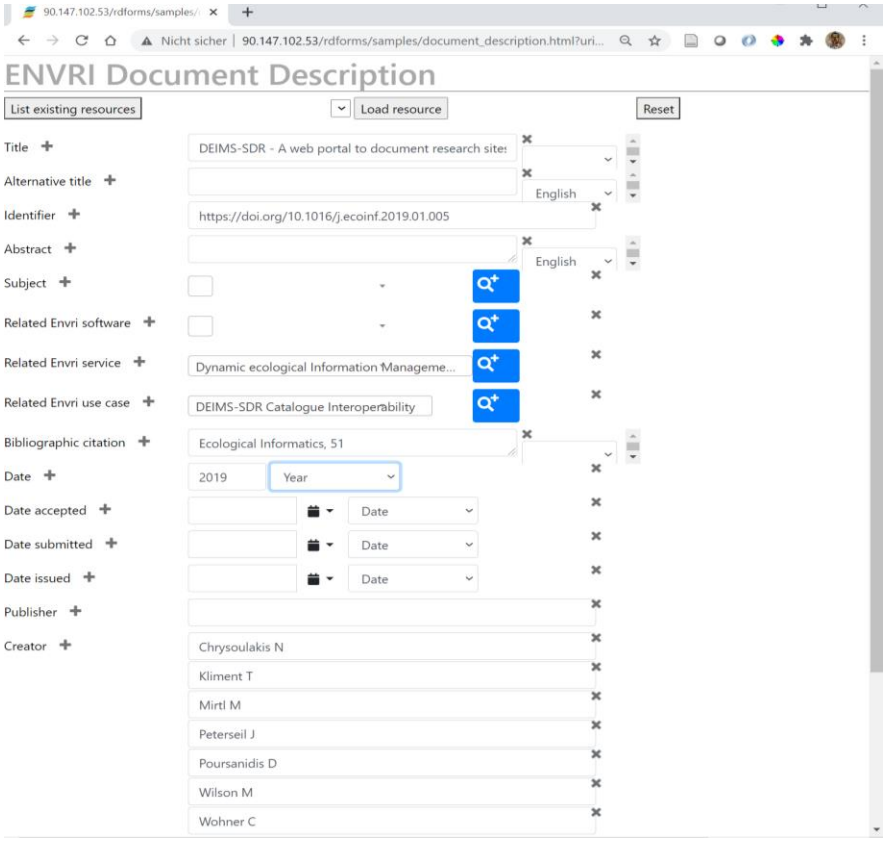

Figure 18. Describing documents.

Services[43] **(Figure 19)** Service descriptions can target many different types of services, computational but also "human 2 human", such as teaching. Following the FITSM approach, their description is thus separated into a common part which serves many different service types, and a "component" part which links to the different building blocks, currently software or document. Therefore, software or documents serving as such components must be described and stored before describing the service, in order to be able to reference them from the service description "Software component" and "Document component" fields.

[43] http://90.147.102.53/rdforms/samples/service_description.html

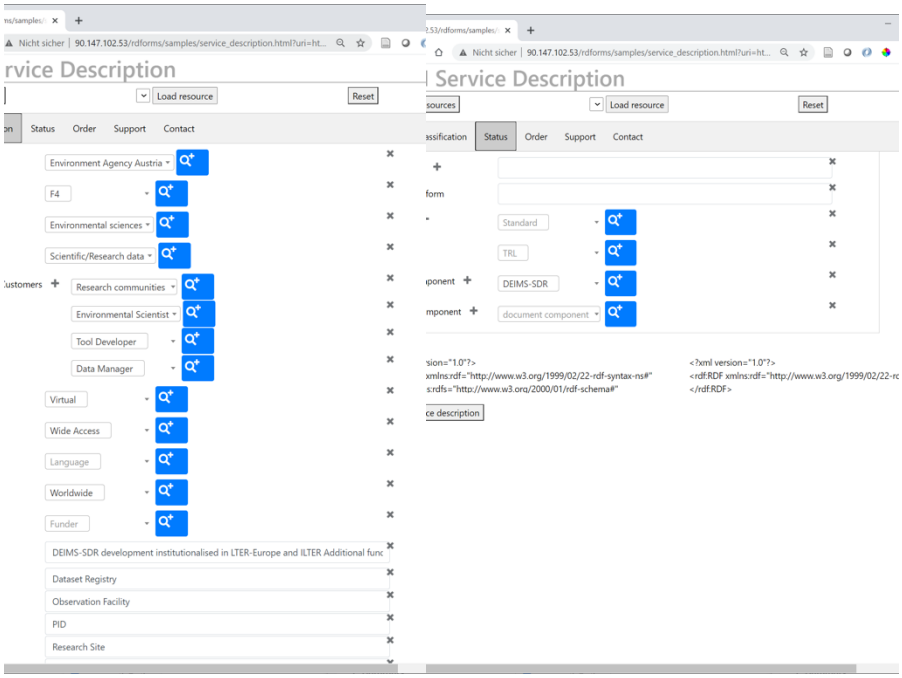

Figure 19. Describing services.

Use Case[44] **(Figure 20)** Use cases are brief descriptions of deployment of services and/or software, highlighting key contributions and/or results of the deployment. In order to describe a use case, the related service or software should have been submitted to the TripleStore beforehand.

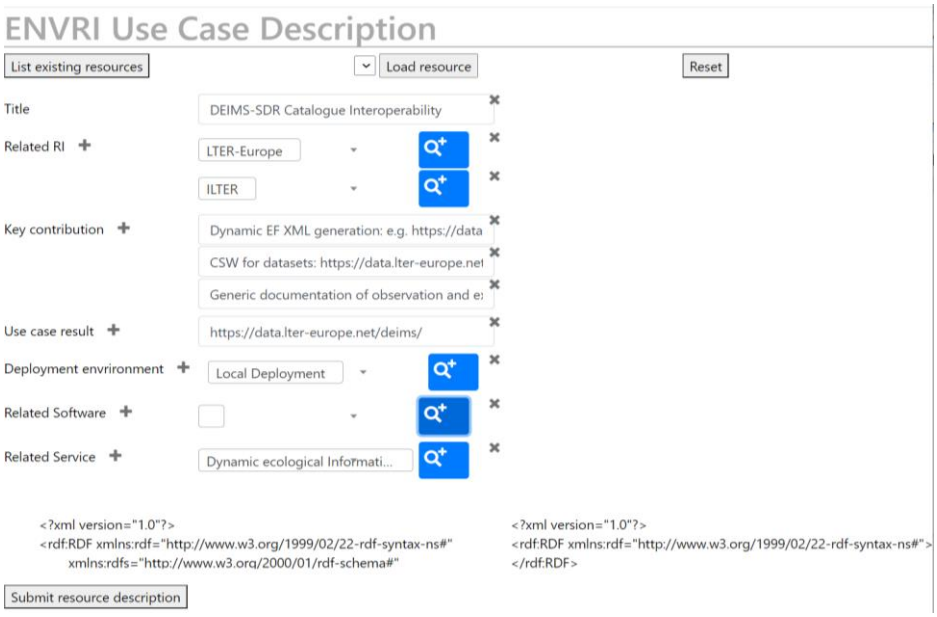

Figure 20. Describing use_cases.

All form interfaces provide (simple) elements to load existing entities and to submit the results. The latter requires a user account for the Ontowiki instance. In addition, the current state of the input graph is shown on the bottom left, the state of the custom user terminology graph on the bottom right, both formatted as rdf/xml which can be copied for manual use in third party settings.

[44] http://90.147.102.53/rdforms/samples/usecase_description.html

In order to demonstrate the functional state of the rdforms-based RDF generation, eight service outputs from the ENVRIPlus project - the D4Science Service Portfolio - were described using the available forms. The resulting collection of RDF facts can be accessed in Ontowiki (http://ontowiki.envri.eu/index.php/EnvriServicePortfoliowithexternalTerminology/), as shown in **Figure 21**.

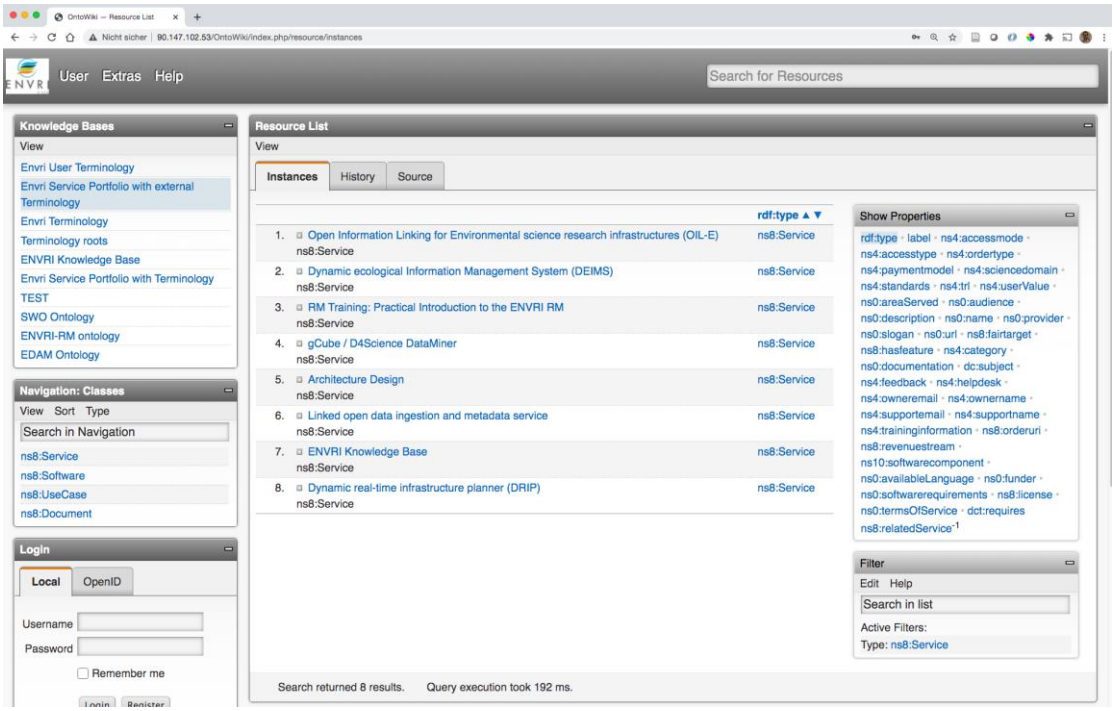

Figure 21. Eight services described using rdforms shown in Ontowiki.

As demonstrated, the current state of the rdforms-protoype allows users to describe services, software, related use cases and documents in RDF without the need to take format-specific considerations into account. Interlinking with existing Knowledge Base concepts is supported via a custom lookup widget which can be activated by clicking on the "loupe+" symbols next to the relevant fields. When having a valid Ontowiki user with correct permissions, users can submit their descriptions directly from the form to Ontowiki, where their changes/additions get tracked accordingly. Existing descriptions can also be loaded into each respective form. Next steps will include the refinement of the data model and the improvement of the forms interface, e.g. by providing better search for existing entities.

6.4 Knowledge Base Search Engine

Though Ontowiki provides navigation functionality over the Knowledge Base, it mainly operates at the RDF triple level, which poses strong technical requirements on users' expertise. To facilitate the general users to easily explore the Knowledge Base, we build the Knowledge Base Search Engine based on the fundamental concepts and components of the Open Semantic Search. This demonstrator exemplifies the search knowledge sequence diagram in section 4.2.
The search instance is available at http://search.envri.eu **Figure 22** below illustrates the search interface.

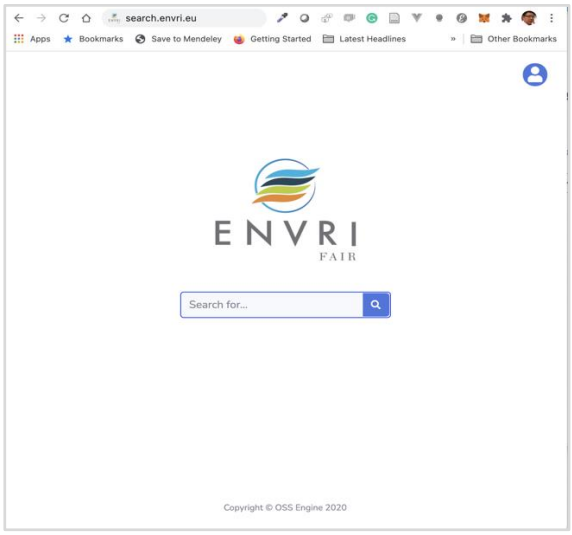

Figure 22. The starting page of the Knowledge Base Search interface.

6.4.1 Search Results

By simply typing in a keyword and hit search button, a result list is returned and displayed as in the following **Figure 23**.

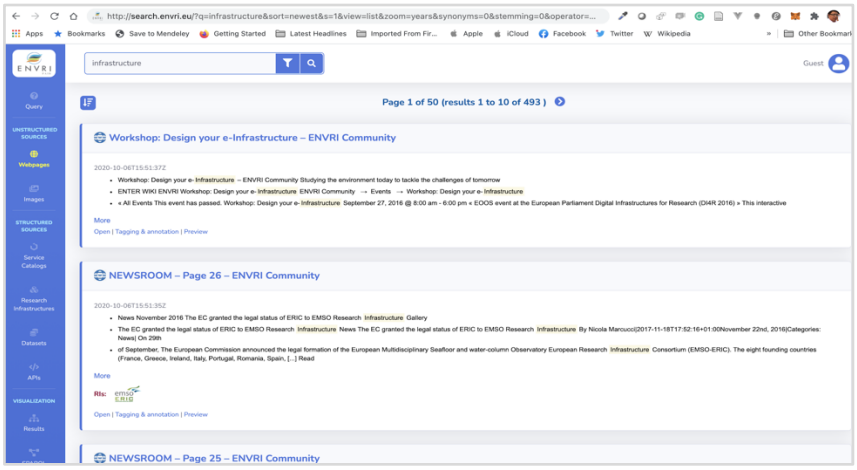

Figure 23. An example of search results.

The results can currently be ranked based on their (1) relevance, and (2) publish date, as shown in **Figure 24**. The results can be sorted and suggested based on the search history of the end-users. In other words, by storing and retrieving search queries of each user, in the meantime compatible with GDPR (General Data Protection Regulation) requirements, the search engine can predict users' interests and recommend a set of search results. Note, the latter option is still under development, and it will be ready soon after conducting research on the decision model for recommending search results to end-users.

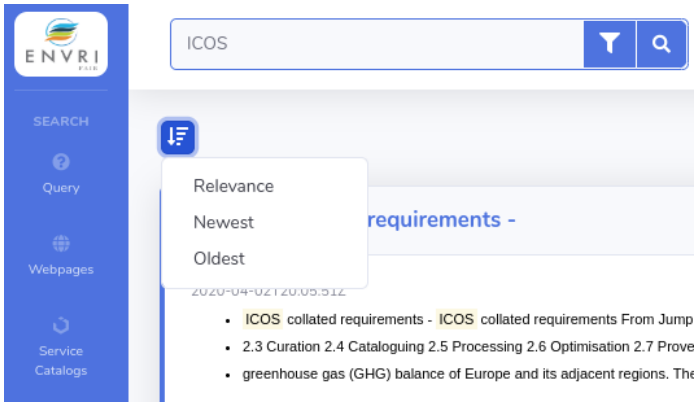

Figure 24. Ranking criteria enabled.

The end users can manipulate the search results by clicking on the "Advanced search" button to consider the following search operators, as shown in **Figure 25**: (1) At least one word (OR), (2) All words (AND), (3) Exact expression (Phrase). Moreover, the search results can be based on semantic search and fuzzy search to include (1) other word forms (grammar & stemming) and (2) Synonyms and aliases.

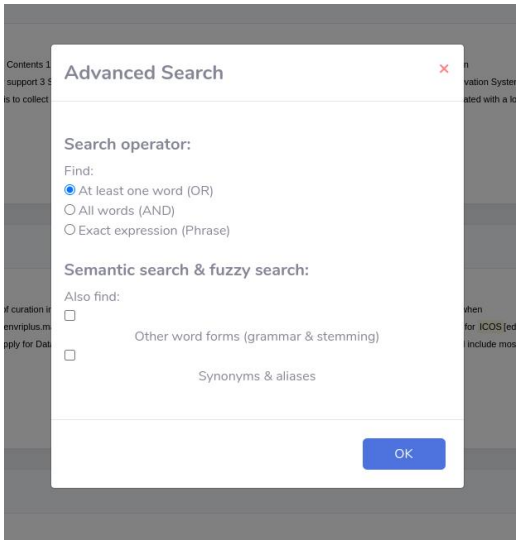

Figure 25. Advanced Search options.

6.4.2 Search Categories

The following five search categories have been considered on the Knowledge Base search engine (KBSE), as shown in **Figure 26**: (1) webpages, (2) Service Catalogs, (3) Research Infrastructures, (4) Datasets, and (5) APIs. The first category's search results are based on the web pages, whitepapers, scientific articles, fact sheets, technical reports, wikis, forums, videos, images, and webinars to map the search queries. The KBSE crawls and indexes all webpages (unstructured knowledge) regarding Service Catalogs, Research Infrastructures, Datasets, and APIs. Such unstructured knowledge is stored in the Knowledge Base of the search engine as separated documents and retrieved when an end-user is looking for a particular search query. The rest of the search categories can be employed to search for specific results within the context of any of the named structured knowledge, consisting of Service Catalogs, Research Infrastructures, Datasets, and APIs.

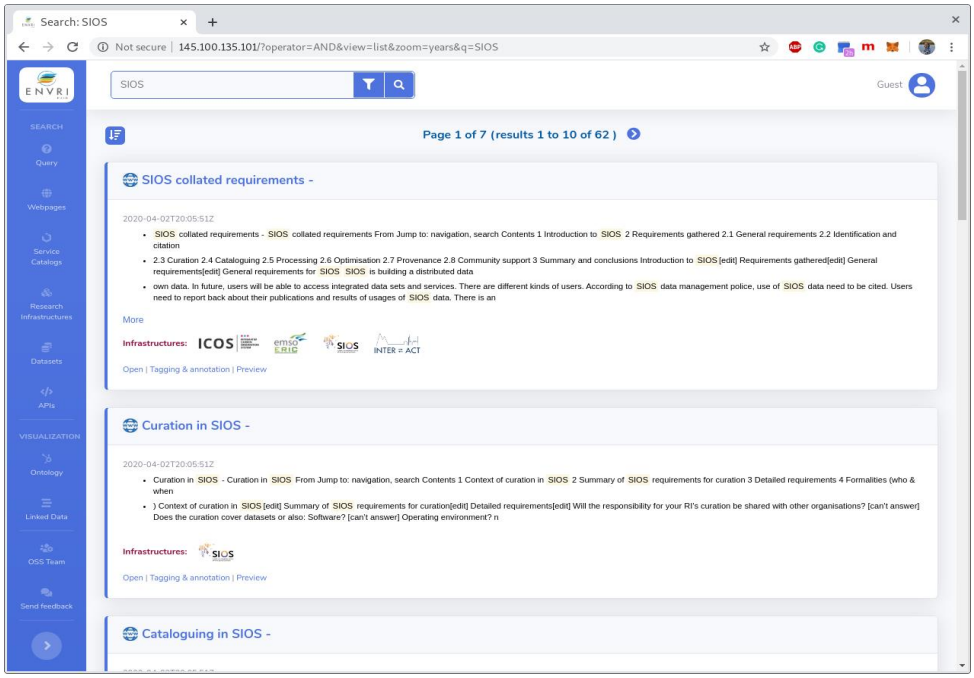

Figure 26. An example of search categories enabled on the Knowledge Base search engine.

6.4.3 Visualisation

The Knowledge Base Search Engine enables end-users to visualise the unstructured and structured knowledge in terms of ontologies and linked data. In other words, the search results are combined with a graph that is rendered in a force-directed layout and represents the ontology to support end-users with a better understanding and analysis of the underlying data. An example illustration is provided in **Figure 27**.

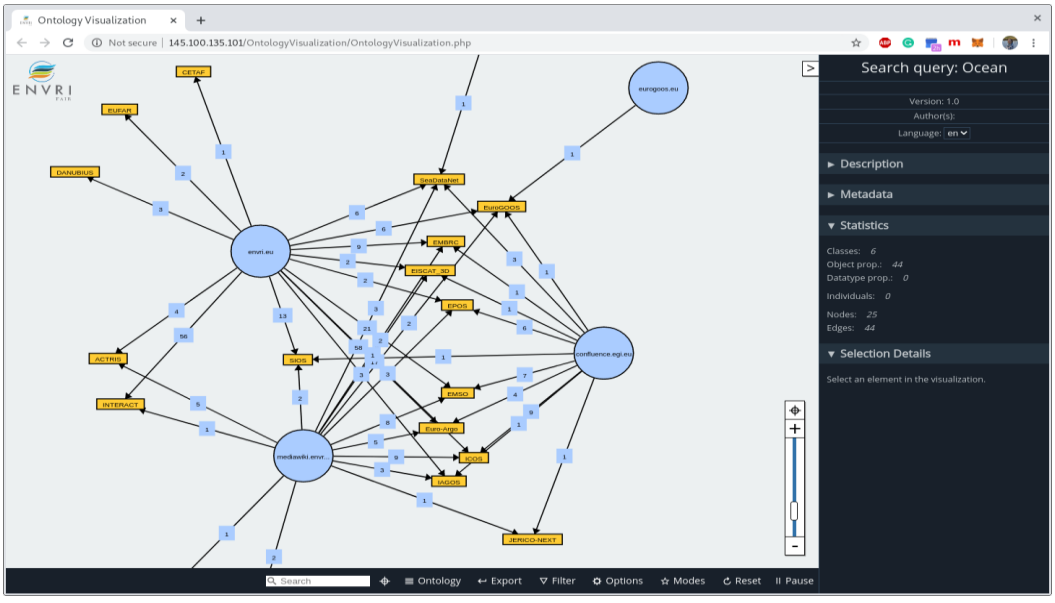

Figure 27. A graph representation of the search output.

6.4.4 Thesaurus editor for editing, structuring & linking vocabulary or dictionary of topics, concepts & names

Open Semantic Thesaurus Editor and Thesaurus Manager are the integrated Django (Python) based open source web app as user interfaces for editing, linking, managing, and structuring a controlled vocabulary or domain knowledge. So end-users can manage important terms, words or concepts, names, topics, persons, organisations, or places in a custom thesaurus for editing names, entities, or concepts, their alternate labels like aliases, synonyms, or typos and misspellings and its structure or relations like hyponyms.
Thesaurus entries are used for automatic tagging for additional structure for analysis and named entity extraction or named entity linking for exploratory search or as tags for news pipes or alerts. Based on the thesaurus entries, the named entity tagger or named entity extraction can find the name or label, alternate labels like synonyms and misspellings, and add the name (preferred label) to the configured facets for an aggregated overview, interactive filters, and analytics. For example, if an end-user add the entity "Open Semantic Search" with/to the facet "Software", she will be able to use this entity or name as an interactive filter and will get an aggregated overview of the count of documents matching this entity while searching for other queries. Additionally, using the alternate labels, aliases, or synonyms, the semantic search can precisely find the terms the user searches for, but the search engine will find documents with alternate terms like synonyms. **Figure 28** illustrates the configuration of Open Semantic Thesaurus Editor and Thesaurus Managerfor in our KBSE.

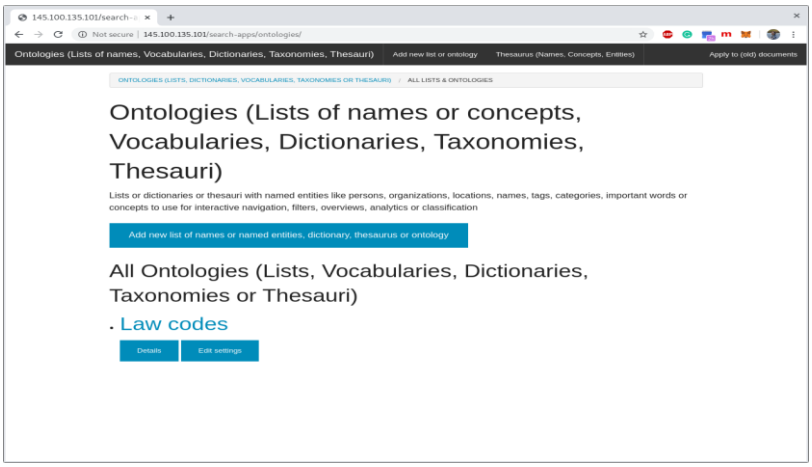

Figure 28. Open Semantic Thesaurus Editor and Thesaurus Manager configured in KBSE.

6.4.5 Optical Character Recognition results

The text stored in image formats like JPG, PNG, TIFF, or GIF (i.e., scans, photos, or screenshots) can not be found by standard full-text search. The KBSE enriches metadata of images like filename, format, and size, resulting from automatic text recognition or optical character recognition (OCR) by free, open-source OCR software like Tesseract[45]. Since much information is not searchable by full-text search because it is in graphical formats embedded in PDF documents or Powerpoint presentations (i.e., screenshots instead of text format), the KBSE extracts images from PDF files for automatic text recognition (OCR), too. **Figure 29** illustrates our settings on KBSE in terms of the OCR functionality.

[45] https://github.com/tesseract-ocr

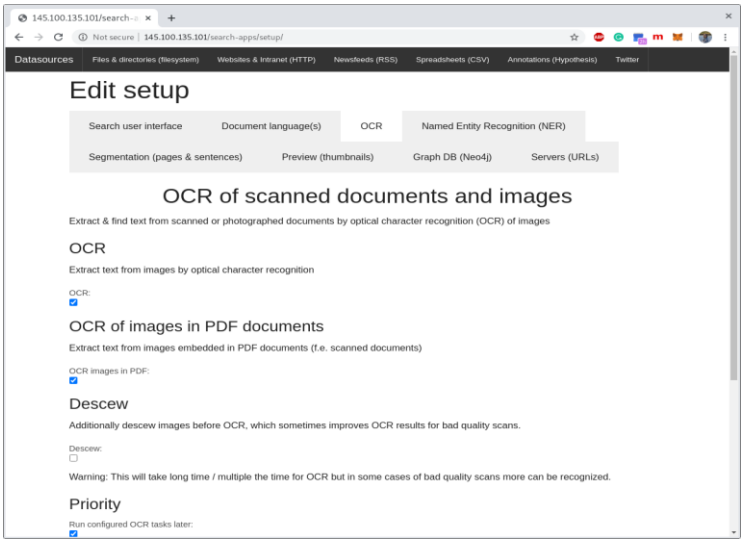

Figure 29. Settings for OCR in KBSE.

6.4.6 Index Datasources

The KBSE (re)indexes web pages regularly to keep its Knowledge Base always up to date. A setting regarding indexing on KBSE is illustrated in **Figure 30**.

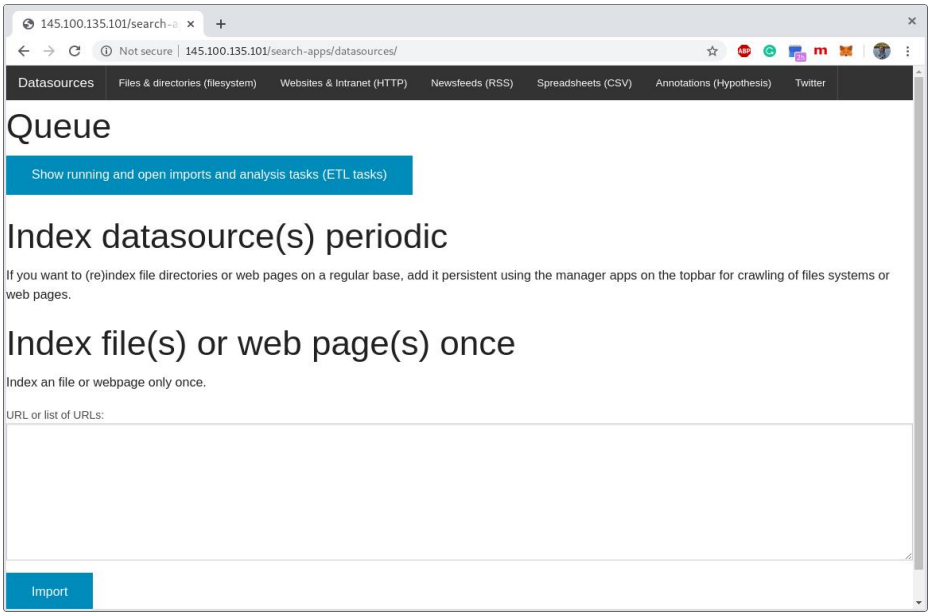

Figure 30. Index settings on KBSE.

7 Summary

The development and operation of the ENVRI Knowledge Base will be continuous. It will grow during the project while the development results and knowledge accumulate. In this summary section, we will briefly discuss how the Knowledge Base effort is embedded in the project with the other development efforts, and what will be the development agenda for the coming two years.

7.1 Embedded in community effort

The Knowledge Base development and operation has a clear dependency on the development effort from the ENVRI subdomains and research infrastructures. Not only the Knowledge Base should play a role for supporting developers from RIs to share best practices and to find existing solutions, but also the ENVRI community provides valuable input to the Knowledge Base and keep the Knowledge Base alive.

Currently, the Knowledge Base team closely interacts with the other subdomain developers (via workshops, meetings and work groups organised by subdomains), and the joint task forces at the cluster level. The Knowledge Base core members or members of WP7 are directly involved in almost all the tasks forces. Through members, there is valuable input of the catalogue of services (TF1), Authentication and authentication (TF2), persistent identifier (TF3), triple store (TF4), license and usage track (TF5), and ENVRI-HUB (TF6).
Moreover, the Knowledge Base team has also a close connection with the FAIRness assessment done from WP5 in interaction with WP8-11.
The Knowledge Base team also closely interacts with semantic search work groups in sub domains, e.g., a semantic search use case in ACTRIS[46], reported in the Semantic Search Working Group Final Report[47].

7.2 Future development

The Knowledge Base will continue in the rest of the ENVRI-FAIR project. In the next phase, the development effort will mainly focus on following aspects:

1. Continuous content ingestion and curation. The Knowledge Base team will improve the knowledge ingestion tool, and continuously ingest the description (metadata) of high quality results from the ENVRI community (e.g. task forces, sub-domain or RI developers), including development results (e.g., best practices, software technologies, recommendations, updated FAIRness assessment possibly generated by new tools) in the Knowledge Base, and make those descriptions FAIR for the community.
2. Continuous improvement of the Knowledge Base based on the feedback received from the community. Extra features e.g. for Knowledge Base discovery and recommendation, will be further explored.
3. The development and operation of the Knowledge Base will also follow the software engineering DevOps practices. The continuous testing, integration and deployment pipeline will be established.
4. We will also extend the content maintenance to community specialists. In this way, we hope the community will play a key role in the Knowledge Base.

[46] https://github.com/xiaofengleo/actris
[47] https://docs.google.com/document/d/1zlPNB1Z1lSGPHAn88B4_DdjybvnGp5pYaIhBqV-oc0A/edit?ts=5ec25164#heading=h.i58j0up5xhwk

8 References

1. Zhao, Z. (2020). Towards Interoperable Research Infrastructures for Environmental and Earth Sciences: A Reference Model Guided Approach for Common Challenges (Vol. 12003). Springer Nature. https://doi.org/10.1007/978-3-030-52829-4
2. Zhao, Z. et al., "Knowledge-as-a-Service: A Community Knowledge Base for Research Infrastructures in Environmental and Earth Sciences," 2019 IEEE World Congress on Services (SERVICES), Milan, Italy, 2019, pp. 127-132, doi: 10.1109/SERVICES.2019.00041.
3. Trochim, W. M., Donnelly, J. P., & Conjoint.ly. (2020 [First published in 2001]). *Research methods Knowledge Base*. https://conjointly.com/kb/
4. Huber, R., Behnken, A., Carval, T., Delory, E. and Robin, C. (2010). *D43-D44 - Data Infrastructure Productive Version ESONET Knowledge Base*. ESONET European Seas Observatory Network of Excellence Project Deliverable. http://www.esonet-noe.org/content/download/42248/file/D43-D44_final.pdf
5. Smith, B., Ashburner, M., Rosse, C., Bard, J., Bug, W., Ceusters, W., Goldberg, L. J., Eilbeck, K., Ireland, A., & Mungall, C. J. (2007). The OBO Foundry: Coordinated evolution of ontologies to support biomedical data integration. *Nature biotechnology*, *25*(11), 1251–1255.
6. Waagmeester, A., Stupp, G., Burgstaller-Muehlbacher, S., Good, B. M., Griffith, M., Griffith, O., Hanspers, K., Hermjakob, H., Hudson, T. S., Hybiske, K., Keating, S. M., Manske, M., Mayers, M., Mietchen, D., Mitraka, E., Pico, A. R., Putman, T., Riutta, A., Queralt-Rosinach, N., … Su, A. I. (2020). Wikidata as a FAIR knowledge graph for the life sciences. *BioRxiv*, 799684. https://doi.org/10.1101/799684
7. Vrandečić, D., & Krötzsch, M. (2014). Wikidata: A free collaborative knowledgebase. *Communications of the ACM*, *57*(10), 78–85.
8. Auer, S., Dietzold, S., & Riechert, T. (2006). OntoWiki – A Tool for Social, Semantic Collaboration. In I. Cruz, S. Decker, D. Allemang, C. Preist, D. Schwabe, P. Mika, M. Uschold, & L. M. Aroyo (Hrsg.), *The Semantic Web—ISWC 2006* (S. 736–749). Springer. https://doi.org/10.1007/11926078_53
9. Frischmuth, P., Arndt, N., Martin, M. (2016). OntoWiki 1.0. In SEMANTiCS 2016, Poster & Demo Session, http://ceur-ws.org/Vol-1695/paper11.pdf
10. Erxleben, F., Günther, M., Krötzsch, M., Mendez, J., & Vrandečić, D. (2014). Introducing Wikidata to the Linked Data Web. In P. Mika, T. Tudorache, A. Bernstein, C. Welty, C. Knoblock, D. Vrandečić, P. Groth, N. Noy, K. Janowicz, & C. Goble (Hrsg.), *The Semantic Web – ISWC 2014* (S. 50–65). Springer International Publishing. https://doi.org/10.1007/978-3-319-11964-9_4
11. ISO/IEC, "ISO/IEC 10746-1: Information technology--Open Distributed Processing--Reference model: Overview," *ISO/IEC Standard*, 1998.
12. ISO/IEC, "ISO/IEC 10746-2: Information technology--Open Distributed Processing--Reference model: Foundations," *ISO/IEC Standard*, 2009.
13. ISO/IEC, "ISO/IEC 10746-3: Information technology--Open Distributed Processing--Reference model: Architecture," *ISO/IEC Standard*, 2009.
14. ISO/IEC, "ISO/IEC 10746-4: Information technology--Open Distributed Processing--Reference model: Architecture Semantics," *ISO/IEC Standard*, 1998

9 Appendix 1: Glossary

Table 5. Glossary.

ACTRIS	Aerosols, Clouds, and Trace gases Research InfraStructure network
Catalogue (Metadata)	A collection of metadata, usually established to make the metadata available to a community. A metadata catalogue has an access service.
CERIF	Common European Research Information Format
CODATA	Committee on data for Science and Technology
DOM	Document Object Model (DOM) is the data representation of the objects that comprise the structure and content of a document on the web.
ENVRI	(1) The ENVRI Community of Environmental Research Infrastructures. (2) FP7 project on Implementation of common solutions for a cluster of ESFRI infrastructures in the field of Environmental Sciences.
ENVRIplus	ENVRIplus is a Horizon 2020 project bringing together Environmental and Earth System Research Infrastructures, projects and networks together with technical specialist partners to create a more coherent, interdisciplinary and interoperable cluster of Environmental Research Infrastructures across Europe.
ENVRI-FAIR	An EU-funded project which stands for ENVironmental Research Infrastructures building Fair services Accessible for society, Innovation and Research.
FAIR	Findability, Accessibility, Interoperability, and Reusability of digital assets
Elastic Search	Elasticsearch is a search engine based on the Lucene library.
EOSC	European Open Science Cloud
FITSM	The name for a family of standards for lightweight IT service management (ITSM).
GDPR	General Data Protection Regulation, a regulation in EU law on data protection and privacy in the European Union and the European Economic Area.
GO FAIR	A bottom-up international approach for the practical implementation of the European Open Science Cloud (EOSC).
GUI	A GUI (graphical user interface) is a system of interactive visual components for computer software.
H2020	Horizon 2020, European level research funding scheme
Knowledge Base (KB)	(1) A store of information or data that is available to draw on. (2) The underlying set of facts, assumptions, and rules which a computer system has available to solve a problem.
LifeWatch	European e-Science infrastructure for biodiversity and ecosystem research
Metadata	Data that describes other data. Metadata summarises basic information about data, which can make finding and working with particular instances of data easier.
NetCDF	A file format
RM-ODP	Reference Model of Open Distributed Processing (RM-ODP) is a reference model in

	computer science, which provides a co-ordinating framework for the standardisation of open distributed processing (ODP)
OIL-e	Ontology of the ENVRI Reference Model
Ontology	(In computer science and information science) an ontology is a formal naming and definition of the types, properties, and interrelationships of the entities that really or fundamentally exist for a particular domain of discourse.
Ontowiki	A free and open-source semantic wiki application, meant to serve as an ontology editor and a knowledge acquisition system.
Open Semantic Search	A free software for building own Search Engine, an explorer for discovery of large document collections, media monitoring, text analytics, document analysis & text mining platform based on Apache Solr or Elasticsearch.
OWL	Web Ontology language
Provenance	The pathway of data generation from raw data to the actual state of data
RDA	Research Data Alliance
RDBMS	A software system used to maintain relational databases
RDF	Resource Description Framework
RI	Research Infrastructure
SPARQL	SPARQL is an RDF query language—that is, a semantic query language for databases—able to retrieve and manipulate data stored in Resource Description Framework format.
Semantics	The encoding of meaning using a formal language.
Semantic Mediawiki	Semantic MediaWiki is an extension to MediaWiki that allows for annotating semantic data within wiki pages, thus turning a wiki that incorporates the extension into a semantic wiki.
Triple	A triple is a data entity composed of subject-predicate-object
Triplestores	A triplestore or RDF store is a purpose-built database for the storage and retrieval of triples through semantic queries.
VRE	virtual research environment
Wikidata	A collaboratively edited multilingual knowledge graph hosted by the Wikimedia Foundation.
W3C	World Wide Web Consortium
WP	Work Package
YAML	A human-readable data-serialisation language.