Shelter

Sustainable Historic Environments hoListic reconstruction through Technological Enhancement & community-based Resilience

D.1.3 Data Lake

Version number	2.0
Dissemination level	PU
Lead partner	LINKS Foundation
Due date	30.11.2020
Type of deliverable	report
Status	Delivered

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 821282

Published in the framework of:

SHELTER - Sustainable Historic Environments hoListic reconstruction through Technological Enhancement & community-based Resilience

Authors:

Federico Oldani (LINKS Foundation)

Reviewers:

Marco Folegani (SISTEMA), Maria Luisa Quarta (SISTEMA), Aitziber Egusquiza Ortega (TECNALIA), Claudio Rossi (LINKS Foundation)

Revision and history chart:

VERSION	DATE	AUTHOR/EDITOR	COMMENT
0.1	27/10/2020	Federico Oldani (LINKS Foundation)	First draft
0.2	03/11/2020	Federico Oldani (LINKS Foundation)	Add Messaging system details
0.3	13/11/2020	Federico Oldani (LINKS Foundation)	Adaptation based on the reviewer comments
0.9	19/11/2020	Marco Folegani (SISTEMA), Maria Luisa Quarta (SISTEMA)	Revision completed
1.0	24/11/2020	Aitziber Egusquiza Ortega (TECNALIA)	Final revision
2.0	05/05/2021	Federico Oldani (LINKS Foundation), Claudio Rossi (LINKS Foundation)	A deeper discussion about Data Lake technologies, geospatial data and geospatial database has been added, including scientific references, whenever appropriate.

Disclaimer:

Table of Content

List of tables

List of Figures

Glossary

Acronym	Full name
API	Application Programming Interface
CH	Cultural Heritage
CIAM	Customer Identity Access Management
CMS	Content Management System
DMF	Data Mapping Form
GIS	Geographic Information System
GUI	Graphical User Interface
IaaS	Infrastructure as a Service
IDI	I-REACT Data Interface
JWT	JSON Web Token
NGO	Non-Governmental Organization
REST	REpresentational State Transfer
RPC	Remote Procedure Call
SDI	Spatial Data Infrastructure
URL	Universal Resource Locator
WP	Work Package

1 Executive summary

The objective of the work in this deliverable is to describe the adopted solutions for Data Lake management in terms of platform and software used.

A great amount of geospatial and non-geospatial data and information will be collected and produced in SHELTER (as satellite imagery, sensor data, geo-environmental and social big data, building stock databases and crowdsourcing). The main aim of the report is to present the implemented solution for data management. In order to explain the provided solution, a survey of the most advanced available technologies for data storage and data management are described. The report introduces all the components needed, specifying the concept of Data Lake, reporting the advantages of the in-cloud solution instead of the on-premises data storage. Following, the concept of the data catalog is introduced, exploring the existing solutions and analyzing their functionalities. Whether SHELTER Data Lake stores structured or unstructured data, data needs to have predefined metadata that summarize the information contained therein, to ensure a simple and agile search. The metadata structure follows the technical guidelines of the INSPIRE Metadata directive, which aims to create a European Union spatial data infrastructure for the purposes of EU environmental policies and policies or activities which may have an impact on the environment. This document finally provides an explanation of the integration of the components with a special focus on the data catalog component, with the description of the customization to adapt it to the SHELTER needs.

The report thus contributes to describe the data management infrastructure of the SHELTER project. It delivers a set of components to ensure a high-reliability service of data storage and efficiency in collecting the information. Adopting the most efficient solutions for storing the relevant information collected and produced by SHELTER, the report delivers technical documentation of the implemented Data Lake and data management.

2 Introduction

2.1 Aims and objectives

One of the objectives of the SHELTER project is to use a data-driven framework to improve the resilience of Historic Areas. Data from diverse sources (satellite data, sensor data, data from social and crowdsourcing) is important to create a deep knowledge of the local risks and vulnerabilities of Historical Areas and for elaborate a decision-masking criterion in case of disaster scenarios. By a deep understanding of the hazard, the exposure and the vulnerability of the historic area, the local dynamics and the provision of innovative governance and community-based models, it is possible to provide useful methodologies, tools and strategies to enhance resilience and secure sustainable reconstruction. Furthermore, by considering information about historical events, best practices and results from linked research initiatives, it is possible to build a collaborative best/next practices observatory, that can be used for consultancy works since it will advise on the suitability of different solutions depending on the element characteristics, location and damage based on the knowledge gathered in the repository. In order to reach the aim, it is crucial to collect the knowledge and data generated during the project activities and to provide a tool to make them efficiently accessible for analysis.

Due to the information complexity and the diverse data sources, the SHELTER framework will be implemented in a multiscale and multisource data-driven platform, able to provide the necessary information for planning and adaptive governance. The objective of the work in this deliverable is to describe the identified solutions for data management in terms of services and software used.

The report thus contributes to describe the data management infrastructure of the SHELTER platform. It provides a technical analysis of the components used to create a Data Lake which takes care of all the SHELTER data lifecycle. The SHELTER Data Lake includes two main components: the Big Data Storage and the Data Catalog. For the Big Data Storage component, in-cloud and on-premises solutions are considered and compared. The Big Data Storage stores different data types coming from diverse sources, facilitating the retrieval of such information which can be difficult due to its heterogeneous nature. Therefore, the concept of Data Catalog is introduced, which through Metadata ensures data consistency and ease of search through heterogeneous data. Since the Data Lake has to manage heterogeneous types of file including geospatial dataset, the Metadata used are compliant with the INSPIRE Implementing Rules on interoperability of spatial data sets and services and Technical Guidelines (Data Specifications) which specify common data models, code lists, map layers and additional metadata on the interoperability to be used when exchanging spatial datasets. By using INSPIRE compliant metadata, the SHELTER platform ensures easy interoperability with European Union Data Catalogs, which provide satellite datasets of critical events. An analysis of the advantages of Open-Source Software rather than proprietary software is

then described and a critical comparison is provided to select the software for Data Catalog that better suits the SHELTER use case. Finally, this document provides a technical guide on the functionalities implemented in Data Catalog software and how to use them.

Specifically, the Data Lake is addressing the following main aspects:

- The collection of heterogeneous datasets in scalable and efficient data storage and the guarantee of fast access to the information
- The metadata content and format: metadata are commonly used to describe data, maps, and services, to facilitate the accessibility and the harvesting in global catalogues. They describe which metadata will be used within SHELTER while being compliant with INSPIRE recommendations.
- The data interoperability: Provide the structure of data for the integration with the web mapping services and other SHELTER platform components.
- Data Catalog functionalities: provide a technical description on how to exploit the functionalities of the Data Catalog

2.2 Relations to other activities in the project

The SHELTER project has been structured in 9 Work Packages (WP) to ensure fertilization among the different steps and partners. In particular, the objective of WP1 (Knowledge base: operationalizing existing data and knowledge) is to identify the existing data, the information and knowledge structures where all the data are collected to be exploited during the project. Within WP1, the aim of Task 1.3 (Data Lake) is to identify the most suitable solution to collect the data from diverse data sources, how to store them and finally how to make them accessible for the users.

The interactions with the other WPs are described in Figure 1:

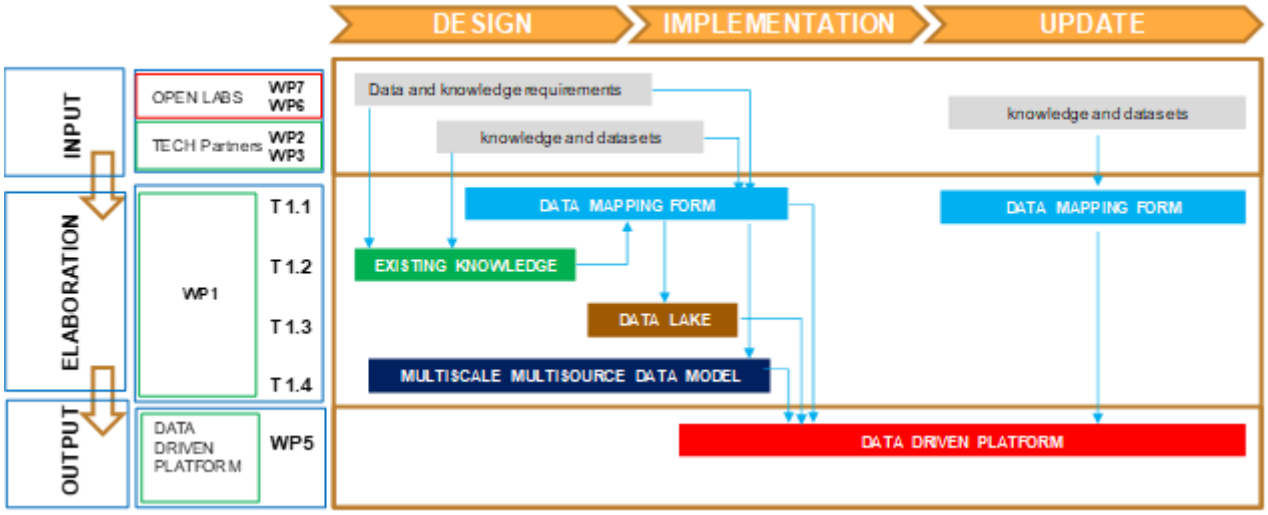

Figure 1. Interactions between WP1 and the other WPs in SHELTER

The data and the existing knowledge that will be collected in the Data Lake are identified and described in D1.1 (Data Sources and Knowledge). To collect the information in a structured way, the T1.1 provides a structure called Data Mapping Form (DMF) inspired by the dataset description template preliminarily outlined in D9.3 – DataManagementPlan_V1 (see Annex A of D9.3 and the related Sharepoint link for accessing the DMF). The DMF serves as a tool for the identification of the data and the existing knowledge generated by the other partners during the SHELTER project. The definition of the data that will be fed into the Data Lake has been performed in the DMF and any change concerning this initial definition will be managed in WP5. Such datasets represent the input for the T1.3 which is in charge of storing the information and providing the tools through which access to it. All the data that will be managed by the T1.3 are compliant to the main principles and standard outlined in the Data Management Plan (D9.3). Furthermore, the metadata described in this document includes the keywords that are compliant to the ontology identified in D1.2. All the data collected and generated during the SHELTER project will contribute to build the "Best/next Practices Observatory" (D1.2) that includes the solution repository which could be used for consultancy works, since it will advise on the suitability of different solutions depending on the element characteristics, location and damage based on the knowledge gathered in the repository. Therefore, Data Lake will be enhanced also by other project results: characterization of the solutions (T3.4-portfolio of solutions), governance schemes maps (T6.4), Co-creation processes blueprints (T6.3) and resilience financing and business models (T6.6).

The design described in this document will be used as input for the WP5 Data-Driven Platform activities, which is in charge of developing the platform that supports the diagnosis, decision making, implementation and monitoring of the natural and cultural

heritage sites. Data Lake and its integration with the other components described will be implemented in ST5.1.1 (Data Lake), ST5.1.2 (AA Server) and ST5.1.3 (Messaging System). They will take into account the design choices of this deliverable in terms of components identified and metadata structure.

The geospatial information of the Open Labs stored in the Data Lake, will be used by T1.4 (Multiscale multisource data model) which is aimed to build a model that will allow the visualization of the information at different scales, from city to region, of relevant cultural heritage and climate change information. The Messaging System component will serve the ST5.1.4 (Data importer and mapper) that will implement an importing and mapping service.

Datasets, existing knowledge (T1.2), Data Lake (T1.3) and Multiscale Multisource Data Model (T1.4) are all fundamental pillars to upon which the Data-Driven Platform must be designed, developed, and implemented.

2.3 Report structure

The document is structured as follows:

- The first section introduces the principal components of the Data Lake and the geospatial data that will be stored in the Data Lake
- Following a critical analysis of the existing technologies and solutions for the Data Lake components is provided.
- The last section describes the implemented solution for the SHELTER use case.

2.4 Contribution of partners

The following table (Table 1) details the contribution of each partner:

Table 1. Contribution of partners

Partner	Contribution
LINKS	Responsible for the coordination of the task and deliverable.
SISTEMA	Review of the whole document.
TECNALIA	Final review

2.5 Applicable and reference documents

The applicable documents are listed in the table below:

ID	Document	WP
1	D1.1 - Data Sources and Knowledge	WP1
2	D1.2 - Building of best/next practices observatory	WP1
3	D1.4 - Multiscale data model	WP1
4	D9.3 - DataManagementPlan_V1	WP9

For other external references please refer to Section 7.

3 Introduction to Data Lake and geospatial data

Data are growing at a rapid pace, and storing these data allows to perform analysis and provide availability in case of further needs. The rate at which data are created increases exponentially and in 2025 the amount of worldwide data generated will reach 79 Zettabytes [1]. Organizations can successfully generate business value from their data performing advanced analytics, but it is crucial to have a reliable place (on-premises or in-cloud) to store all the information and provide it when necessary. The data collected can be of such size and structure that exceed the capabilities of traditional programming tools (databases, software, etc.) for data collection, storage and processing in a reasonable time and a-fortiori exceed the capacity of their perception by a human. Data can be structured, semi-structured and unstructured that makes it impossible to manage and process them effectively in a traditional way [2]. In the era of big data, a new term called "Data Lake" came into view of the digital universe. The simplest intention of data lake is to munge every data produced by an organization to give more valuable insight into finer granularity. Big Data technologies are sometimes considered as destructive technologies as they revolutionized the traditional ways of doing things in this data intensive era. The idea of Data Lake was first initiated by Pentaho CEO Jame Dixon [3]. If a data warehouse or data mart is seen as a bottle of water cleaned and ready for consumption, then "Data Lake" is a whole lake of data that is cleaned for ready use. All data generated by an organization regardless of types, structures, or formats will be stored in Hadoop clusters or other similar frameworks in their original forms. When parts of the organizations need to use the data, that stored data will be loaded and transformed as required by that organization's parts. Due to these, concepts in "Data Lake" seem to be challenging to the traditional ways of storing data i.e. data warehouse and data marts. A Data Lake is usually composed of a Big Data storage, where all the data are stored in its raw format, a Management component (in this document called Data Catalog) that orchestrates and handles the data ingestion and data request, and finally, a Computing Engine to perform data analytics.

3.1 Big Data

The criteria for determining the difference between big data and traditional data are three "V": volume – very large volumes of data; velocity – very high data transfer rate; variety – weakly structured data, which is primarily understood as data structure irregularity and difficulty of extracting homogeneous data from a stream and identifying some correlations [4]. The big data solutions are based on two main components: the big data storage system and the big data processing component.

The former clusters a large number of commodity servers attached to high-capacity disk to support analytic software written to crunch vast quantities of data. The system relies on Massively Parallel Processing databases to analyze data ingested from a variety of

sources. Big data often lack structure and comes from various sources, making it a poor fit for processing with a relational database. The Apache Hadoop Distributed File System (HDFS) is the most prevalent file system for big data, and is typically combined with some flavor of a NoSQL database (e.g., MongoDB [5], DocumentDB [6]). Some Data Lakes include also a Computing Engine able to analyze a great amount of data using batch processing and parallel computation. The most common data analytics programming paradigms are the MapReduce belonging to the Apache Hadoop framework [7] and the Apache Spark framework [8].

3.2 Data catalog

In Big Data Storage, the data is stored in their natural formats such as satellite imagery, sensor data, geo-environmental and social big data, building stock databases, and crowdsourcing. Big Data Storage can take input from multiple sources, making them a functional single repository for an organization to use as a collection point. The data remains in their raw format until a schema is applied to it for processing ("*schema on read*"), allowing analysts and data scientists to pick and choose what and how to work with the Data Catalog. Structured information is needed to make the datasets easily accessible to the users. A Data Catalog creates an informative and searchable inventory of all data assets in an organization. These assets can include (but are not limited to):

- Structured (tabular) data
- Unstructured data, including documents, web pages, email, social media content, mobile data, images, audio, and video
- Reports and query results
- Data visualizations and dashboards
- Machine learning models
- Connections between databases

This inventory enables the users to search through all of the available data assets of the organization and help themselves to retrieve the most appropriate data for their analytical or business purposes.

A Data Catalog typically includes capabilities for collecting and continually enriching the metadata associated with each data asset to make each asset easier to identify, evaluate, and use properly. The Catalog also provides tools that enable users to do the following:

- Search the catalog
- Automate the discovery of potentially relevant data for which they did not specifically search
- Govern the use of the data in compliance with organization regulations

3.2.1 Metadata

Metadata summarizes basic data information, making finding and working with particular instances of data, and Data Catalog is a collection of metadata, combined with data management and search tools. Shelter platform considers diverse data sources with heterogeneous data types; therefore, the metadata ensures data consistency and ease of search. Since the Shelter platform has to support also geospatial information, the metadata that will be used are compliant with the INSPIRE metadata conform to ISO 19115/ISO 19119 [9] metadata standard which specifies common data models, code lists, map layers and additional metadata on the interoperability to be used when exchanging datasets. By using INSPIRE compliant metadata, the SHELTER platform ensures easy interoperability with European Union Data Catalogs which provide satellite dataset of historical-critical events. In the Appendix, at the end of this document, the definition of INSPIRE compliant metadata is fully listed.

3.3 Geospatial data

The Shelter Project will collect, store, transform, and analyze big amount of geospatial data to support historic site management activities and provide information. Hence, a big data architecture, which is described in Section 5, is needed.

The GIS-based applications are used to process geospatial data and visualize the results of the analyses on maps, associated with the most appropriate layers. Data are stored in databases with geospatial features, which are specifically designed for efficiently storing and querying spatial data by means of ad-hoc data structures (e.g., spatial indexes). Points, lines, and polygons can be represented and queried by means of geospatial databases. In Chapter 4.2, we describe a set of databases with geospatial features that can be used to store and manage geospatial and geo-referenced data. However, we first provide a description of standard file formats that are usually used by GIS to store and share geospatial data. Two commonly used standard formats are ESRI Shapefile [10] and GeoJSON [11]. These standards will also be used to represent the majority of the data collected and analyzed by the Shelter Project.

3.3.1 Popular GIS file formats

A set of standard GIS file formats are used to encode geographical information, i.e., GIS data and maps. The GIS file formats are usually classified into two main categories: raster data formats (e.g., GeoTIFF [12]) and vector formats (e.g., GeoJSON and Shapefile). The raster data formats represent geospatial data as a geo-referenced surface divided into a regular grid of cells (i.e., a matrix of cells). Each cell is associated with a value representing the information of interest (e.g., the elevation value if an

elevation surface map is represented). Raster models are useful for storing values that vary continuously with a high resolution (the minimum granularity is given by the size of the areas associated with the cells), as in a satellite image or an elevation surface. Differently, the vector file formats represent the world using geo-referenced points, lines, and polygons (i.e., basic types of geometry). Each geometry type, represented in a vector-based file, is associated with a set of attributes describing it. For instance, a polygon describing a lake may be associated with the lake's depth or the water quality.

Raster data file formats can have spatial inaccuracies due to the limits imposed by the cell dimensions and the generated files are potentially very large, depending on the enforced resolution, because all cells of the represented area are stored in the file. Differently, the vector format, which represents the object as points, lines or polygons, is usually more accurate and generates smaller files because, for instance, it stores only the vertexes of a polygon and not all the points in the area of the polygon. Scaling operations are usually easier and more accurate when vector data formats are considered, whereas raster data formats are usually computationally less expensive to render than vector graphics.

In the following, we describe the main characteristics of two well-known and frequently used standard vector data formats: GeoJSON and shapefile and one commonly used raster format: GeoTIFF. Many of the data collected and processed by Shelter will exploit these file formats.

3.3.1.1 GeoJSON

GeoJSON [11] is an open standard that is widely used for representing geographical objects, characterized by a shape corresponding to a geometry type (e.g., point, linestring, polygon, multipoint). In particular, GeoJSON is an extension of the JSON file format [13] commonly used for representing data, which enables the description of a variety of geographic data structures in a concise way, using the JSON syntax. The following geometry types are supported by GeoJSON: Point, LineString, Polygon, MultiPoint, MultiLineString, MultiPolygon, GeometryCollection, Antimeridian Cutting.

Every one of the mentioned geometry types defines the syntax to describe an array of values, which individuates the position of the object by means of the projected coordinate reference system or the geographic coordinate projected system. The geographic data must be inserted with respect to the [longitude, latitude] form, on WGS84 coordinates system. GeoJSON is a standard of the IETF [11] and it is released under the Creative Commons Attribution v3.0 license. This file format is natively supported by many databases with spatial features and extensions for big data frameworks. For instance, both the MongoDB database and the Magellan package for Apache Spark support GeoJSON. However, it is important to highlight that some systems do not have a

complete implementation of the standard and implement only a subset of the data types defined by GeoJSON.

3.3.1.2 Shapefile

Another vector data format that is usually used to represent geospatial data is the ESRI shapefile format [10]. The shapefile format was introduced by the ESRI (Environmental Systems Research Institute) company in the 1990s. It became a de facto standard and it is frequently used to achieve interoperability among the ESRI systems and other GIS software products. Many GIS applications support the shapefile format. Similarly to GeoJSON, also the shapefile format can describe vector features: Points, Lines, Polygons and more. Each item represents a real object, such as, for example rivers, lakes, buildings and each item is usually also characterized by other (non spatial) attributes (metadata) that describe it (e.g., name or temperature).

Since both GeoJSON and shapefile are frequently used, third-party software is available to convert data from one format to the other if they are not natively supported by the used GIS system or the exploited database with geospatial features. For instance, the ogr2ogr [14] is a tool that can be used to convert files from one geospatial data format to another. This conversion tool supports many data formats (e.g., GeoJSON, shapefiles, GML).

3.3.1.3 GeoTIFF

GeoTIFF is a frequently used raster format for geospatial data based on the TIFF data format. Specifically, the GeoTIFF format extends the TIFF format by defining a set of ad-hoc TIFF tags. The tags formalized in the GeoTIFF specification allow describing cartographic information for satellite images, maps, etc.

The images based on the GeoTIFF format can be transformed into other popular data formats by using common conversion tools such as QGIS.

4 Data Lake technologies

Several solutions for big data querying and analytics have been proposed in the last years. As briefly discussed in Section 3.1, the most popular big data frameworks are Apache Hadoop and Apache Spark, usually coupled with databases (relational and non-relational) for storing metadata. The common phases of the big data value chain are the following [15]:

Data acquisition: it is the process that collects, transforms and pre-processes data;

Data storage: it consists of persistently storing and managing large datasets;

Data analysis & visualisation: this phase leverages algorithms and tools to inspect and mine knowledge from data, in order to extract value. Furthermore, it comprises software to visualise the results for example using a map server software.

Traditional frameworks, data catalog and databases for big data are usually focused on non-spatial data. However, geospatial and geo-referenced data are available in many application contexts and the Shelter project context is one of them. The big data systems that manage geospatial data are based on the same macro-architecture and the big value chain of the solutions for non-spatial data [16]. However, they pose new challenges and opportunities because the basic (standard) components of Apache Hadoop and Apache Spark are not designed for spatial data [16]. Hence, geospatial databases must be exploited, to effectively store and query big spatial data.

The Big data processing frameworks (e.g., Apache Hadoop and Apache Spark) are usually used to perform (batch/offline) data analytics operations that extract knowledge by analysing the complete data collection, whereas the geospatial databases (used within data catalog) are usually exploited to perform (operational/real-time) queries that aim at selecting a small subset of the data collection. In a big data system, both components are important since they address complementary issues. In this Chapter we analyze the the most popular Big Data Storage solutions, the databases natively supporting GIS data, and the data catalogueing solutions in a view of selecting the technical solutions that better fit the Shelter project purposes (Section 4.4).

4.1 Big Data Storage systems

Big Data storage system allows to store a great amount of data in a reliable way to avoid any data loss. According to the nature of big data, a single server is not a wise decision for storage, and it is better to configure a cluster of multiple hardware elements as the distributed storage system. As far as the availability of a system is concerned, it suggests quick access to data storage resources [17]. For this purpose, data are replicated among

different servers, which may be placed on the same location or distant locations to make the data highly available to users at their nearby sites, thus increasing big data retrieval efficiency ([18] [19]). In other words, minimum downtime and promptness of a system for ad hoc access requests define its availability [20]. Node failure is very common in distributed storage systems. To make it fault-tolerant, multiple copies of data are created and placed on the same node and/or different nodes of the storage cluster. Replication not only makes the system highly available but it is also useful for fault tolerance [21]. Nowadays, data not only comprise tuples; documents and objects are also part of big data. Therefore, a predefined schema cannot deal with varying data structures [22]. Regardless of the distribution, storage systems for big data ensure the data be complete and correct. Changes made by users are committed under defined rules [20]. Eventual consistency has become a widely adopted mechanism to implement consistency in NoSQL databases. Changes are propagated eventually, and the system becomes consistent after propagating the changes. However, instantaneous propagation leads to the development of a strongly consistent system, yet it results in frequent access locks. Research outcomes of exploring storage technologies for big data advocate different aspects of designing storage mechanisms. These reliable and highly available mechanisms contribute to improving data access performance. Improved data access performance drives better quality of data analysis. These technologies offer scalable storage solutions for growing big data with enhanced data structures and support fault tolerance [23].

The principal key of Big Data storage is the file system used to control how the data is stored and retrieved. Many different file systems have been developed in last years to guarantee service reliability, performance and fault tolerance, but the most popular solution is the open source Apache HDFS. The Hadoop Distributed File System (HDFS) is a distributed, scalable storage system designed as a core of Apache Hadoop to run on inexpensive commodity hardware, initially designed as infrastructure of Apache Nutch. HDFS is a suitable solution for data-intensive applications, typically at gigabyte to terabyte scales, which require high throughput. HDFS provides quick fault detection and automatic recovery, as it comprises a large number of components. However, there is a probability of block failure and nonfunctioning [24]. Block replication is offered to avoid node failure and unavailability or loss of data [25]. Replication ensures not only the availability but also the reliability of the system and it is automatically handled by HDFS NameNode. Rather than just being a storage layer of Hadoop, HDFS is a standalone distributed file system that helps improve the throughput of the system. HDFS has a namespace-separated architecture. Metadata is stored on the master node, which is called NameNode, whereas block-split files are stored on a number of DataNodes. NameNode performs mapping of data on DataNodes and namespace operations such as open, close, and rename file. DataNodes fulfill read-write requests and create block and replicas.

In the last years many tech companies implement their own file system in order to provide a unique service on the market. Among these, we find IBM (IBM Cloud Storage), Amazon (AWS S3), Google (Google Cloud Storage) and Microsoft (Azure Blob Storage). The majority of the services can be deployed in the cloud only, only IBM offers a hybrid solution to have data on-premises. The data is stored in local hardware and this allows to have full control of it and the accesses. However, on-premises solutions require dedicated hardware and if remote access has to be allowed, a set of security measures must be implemented to avoid data leaks. Furthermore, the scalability of this solution depends on the availability of the hardware and it would require effort in time and resources. On the contrary, cloud storage in form of IaaS (Infrastructure as a Service) offers greater flexibility, accessibility, and fast and cost-effective scalability. The main drawback of cloud solutions is data outsourcing and the inability to have full control of the data. This last observation raises the necessity to rely only on cloud storage providers able to guarantee the cutting-edge technologies to protect the data. Table 2 summarizes the benefits and the drawbacks of cloud and local storage.

Since the SHELTER partners are spread across the European Union and they can have the possibility to access to the data, the most convenient solution is cloud storage in form of IaaS (Infrastructure as a Service), because it allows the accessibility from any internet device, potential infinite scalability, and strong reliability and security of the service.

Table 2. Comparison between in-cloud and on-premises solutions

	Cloud Storage	Local Storage
Cost	Cheap - as you don't need the hardware	Expensive - as you need to buy the hardware
Accessibility	Any internet enabled device	Restricted to local access
Flexibility	Highly flexible and will grow with your data	New hardware may need to be brought as data grows
Disaster Recovery	Very safe from on site disasters	Susceptible to on site disasters
Speed	Reliant on the speed of the internet	Limited only by the hardware used - can be very fast
Security	Can be good but totally dependant on provider	The data is as safe as your network is protected
Control	Inability to retain complete control	You have 100% total control over the data

In the following, we analyze the main players in cloud storage services that could suit Shelter's needs.

4.1.1 IBM Cloud Object Storage

IBM Cloud Object Storage [26] is a service offered by IBM for storing and accessing unstructured data. The object storage service can be deployed on-premise, as part of IBM Cloud Platform offerings, or in hybrid form [27]. The offering can store any type of object which allows for uses like data archiving and backup, web and mobile applications, and as scalable, persistent storage for analytics [28]. The interaction with Cloud Object Storage is based on Rest APIs. IBM Cloud Object Storage stores objects that are organized into buckets identified within each bucket by a unique, user-assigned key. All requests are authorized using an access control list associated with each bucket and object. Bucket names and keys are chosen so that objects are addressable using HTTP URLs. IBM Cloud Object Storage offers different storage classes, identical in data protection, security, durability and resiliency. The classes differ in data pattern and availability needs [29].

4.1.2 Microsoft Azure Blob Storage

Azure Blob Storage [30] helps to create data lakes for analytics needs and provides storage to build powerful cloud-native and mobile apps. It is optimized for long-term data and provides flexibly scale up for high-performance computing and machine learning workloads. It consists of a file system implemented as an extension built on top of the HDFS APIs and is in many ways HDFS. Azure Blob Storage is built into HDInsight (Microsoft's Hadoop on Azure service) and is the default file system. Azure storage stores files as a flat key/value store without formal support for folders. The hadoop-azure file system layer simulates folders on top of Azure storage. Figure 2 shows how the files are stored.

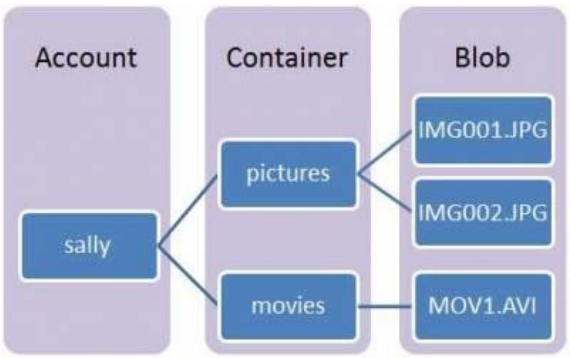

Figure 2. Windows Azure Blob Storage architecture

4.1.3 AWS S3

AWS S3 [31] provides easy-to-use management features to organize the data and configure finely-tuned access controls to meet specific business, organizational, and compliance requirements. Amazon S3 is designed for durability, and stores data for millions of applications for companies all around the world.

4.1.4 Google Cloud Storage

Google Cloud Storage [32] is a RESTful online file storage web service for storing and accessing data on Google Cloud Platform infrastructure. The service combines the performance and scalability of Google's cloud with advanced security and sharing capabilities. It is an Infrastructure as a Service (IaaS), comparable to Amazon S3 online storage service.

4.1.5 Big Data storage comparision

In Table 3, the comparison of the Big Data Storage services is presented.

Table 3. Cloud Storage comparison analysis [33]

	IBM Cloud Object Storage	Azure Blob Storage	Google Cloud Storage	AWS S3
Object limits	Unlimited	Unlimited	Unlimited	Unlimited
Object size limit	**10 TB**	4.75 TB	5 TB	5 TB
Durability	11 9’s	11 9’s	11 9’s	11 9’s
First Byte Latency	N/A	milliseconds	milliseconds	milliseconds
Encryption	SSE 256-bit AES	SSE 256-bit AES	SSE 256-bit AES	SSE 256-bit AES
Cost USD/GB/month	$0.0238	**$0.0184**	$0.020	$0.023
Internal staff training	required	**not required**	required	required

4.2 Databases with geospatial features

Databases can be classified into two macro-categories: relational databases and NoSQL databases. Relational databases are usually the most appropriate and efficient choice when data are structured (i.e., when data are characterized by a fixed set of attributes known at design time). Differently, NoSQL databases, such as document-oriented

databases, are more appropriate when the input data collection contains unstructured or semi-structured data, or when the structure (attributes) of the data evolves overtime and it is (partially)-unknown at design time.

The majority of the traditional databases are mainly focused on non-spatial data. However, since geospatial data play an important role in many application domains, several spatial databases, relational and not, have been proposed to manage and query geospatial data. Spatial databases, or in general databases with (geo)spatial features, are usually an extension of traditional databases in which ad-hoc (geo)spatial data types, query operators and indexes have been integrated. The most common use of geospatial databases consists in executing queries that select all the objects contained in a geographical area or the k-Nearest Neighbours of an input object.

Some commonly used state-of-the-art relational databases with geospatial features are PostGIS [34], SQL Server and its cloud version, which name is Azure SQL Database [35], while two established NoSQL databases with geospatial features are MongoDB [5] and DocumentDB [6]. In the following, we describe the main characteristics of the four selected state-of-the-art databases with spatial features.

4.2.1 PostGIS

PostGIS [34] is a spatial database extender of the relational PostgreSQL database that enables the definition of geospatial objects and operations. It is currently available at version 3.1, released under the GNU GPL v2 license. PostGIS follows the Simple Features for SQL specification from the Open Geospatial Consortium (OGC) [36] [37].

The following standard geometry data types are defined in PostGIS: Point, LineString, Polygon, MultiPoint, MultiLineString, MultiPolygon, GeometryCollection. There are also many standard geospatial operators defined in PostGIS, such as: Distance, Within, Intersects, Closest, Contains Disjoint, Length, Overlaps, Touches and so on. It also supports three types of spatial indexes: B-trees (binary trees), R-trees (sub-rectangles trees) and GiST (Generalized Search Trees) to speed up the execution of spatial queries.

Finally, it is possible to natively import/export data from/to PostreSQL using the WKT, Well-Known Binary (WKB) and ESRI shapefile file formats. GeoJSON files can be imported/exported by using specific conversion libraries. PostGIS is also compatible with state of the arts geographic information systems (GIS) such as ArcGIS [38], QGIS as well as with map server software such as GeoServer [39].

4.2.2 Azure SQL Database

Azure SQL Database [35] is a relational Database as a Service e (DaaS) in the cloud. It has the same functionalities of the centralized Microsoft SQL Server 2019 [40] but it is provided as a service in the cloud. Hence, it scales easily to large data collections since it can be distributed on multiple servers. It can also be easily integrated with HDInsight, which is the cloud implementation on Microsoft Azure of the Apache Hadoop and Apache Spark technology stacks.

Both Azure SQL Database and Microsoft SQL Server 2019 allow defining and querying spatial objects. They support two main categories of data types:

1. Geometry: represents data on a Euclidian coordinate system, using flat XY coordinate pair.
2. Geography: represents data on an earth-like spherical coordinate system, in longitude-latitude shape.

Azure SQL Database can manage the following standard spatial objects: Point, MultiPoint, LineString, CircularString, MultiLineString, CompoundCurve, Polygon, CurvePolygon, MultiPolygon, GeometryCollection. The geography type has also the FullGlobe object. The following are some of the standard spatial operations available for geography objects: Equals, Disjoint, Intersection, Distance, Difference. However, many other standard spatial operations are available because Azure SQL Database implements the Simple Features for SQL specification from the Open Geospatial Consortium (OGC) [36] [37].

Azure SQL Database and Microsoft SQL Server 2016 support also spatial indexes, which are implemented using the B-Tree (i.e., Binary Tree) data structure. It is also possible to import and export spatial data using one of the follow file format: WKT, WKB and GML (Geography Markup Language, an XML-like file format). GeoJSON and shapefile files can be imported/exported by using specific conversion libraries.

Azure SQL Database and Microsoft SQL Server 2016 are compatible with state-of-the-art geographic information systems (GIS) such as ArcGIS [38] as well as with map server software such as GeoServer [39].

4.2.3 MongoDB

MongoDB [5] is an open source NoSQL document-oriented database, based on JSON-like documents. The currently available release is the 4.4.4 version, released under the GNU GPL and Apache licenses. To perform queries and for storage purposes on geospatial data, MongoDB needs an initial definition of the surface type used for running operations on its data.

The supported surfaces are:

- Spherical. It involves calculation based on an Earth-like sphere, with data stored as GeoJSON objects. It is also possible to define the data as legacy coordinate pairs (a couple of longitude, latitude values), which are then indexed using 2dsphere using a translator to the GeoJSON's point type.
- Flat. This surface considers a Euclidian plane with 2d coordinates, stored as legacy coordinate pairs. For indexing purposes, it uses the 2d index.

The current version of MongoDB supports the following GeoJSON data types: Point, LineString, Polygon, MultiPoint, MultiLineString, MultiPolygon, GeometryCollection.

MongoDB implements the following basic spatial operations/query types: inclusion, intersection, and proximity. Hence, the types of possible queries are limited with respect to those provided by the relational state of the art spatial databases, such as PostGIS and Microsoft Azure SQL Database. Finally, MongoDB supports only basics 2-dimensional indexes (2dsphere, 2d).

The mongo import tool enables the possibility to import into MongoDB data from CSV (comma separated values), TSV (tab separated values) or JSON (and GeoJSON) format files.

Another useful feature of MongoDB is the native implementation of a MapReduce paradigm to perform aggregations on data in his system. Hence, MongoDB can be used to both query data and perform (simple) data analytics operations by means of MapReduce programs.

MongoDB has also a lot of drivers for different languages and frameworks: C/C++, C#, Java, web (PHP, Node.js), Python, Ruby, Perl, Motor and Scala.

MongoDB supports horizontal scalability (i.e., the scalability achieved by adding new commodity servers in a cluster environment when the size of the data increases) and distributed execution of queries by exploiting the sharding technique. The basic idea exploited by the sharding technique consists in partitioning the input data collection is chunks and store each chunk on a different server. When a query is executed, each server executes the query on its chunk of data, parallelizing the execution of the query.

The partitioning of the data is based on the value of the selected sharding attribute. The choice of the sharding attribute is crucial in order to achieve a balance distribution of the data in the servers. It is important to highlight that MongoDB supports also the use of geospatial attributes as sharding attribute.

Another interesting feature of MongoDB is the availability of libraries that can be used to import/export data from/to HDFS. Spark and Hadoop applications can read and store

data in MongoDB by means of the MongoDB Spark Connector [41] and MongoDB Connector for Hadoop [42], respectively.

4.2.4 DocumentDB

DocumentDB [6] is a NoSQL document-oriented database designed by Microsoft. Similarly to Azure SQL Database, also DocumentDB is available as a service. Hence, it can manage large data collections by distributing data on a set of servers. It can also be easily integrated with HDInsight, a cloud system of Microsoft including a service version of Apache Hadoop and Spark.

DocumentDB implements the same spatial operations and data types supported by MongoDB and it is compatible with the protocol used by MongoDB. Hence, the applications written for MongoDB

can use DocumentDB as data store. The applications, by using the existing drivers for MongoDB, can transparently communicate with DocumentDB, by simply changing the connection string.

4.2.5 Comparison of databases with geospatial features

In this Section, we report a summary of the functionalities of the considered databases with geospatial features described in the previous chapters and draw conclusions about their pros and cons.

The summary of the comparison of the functionalities of the databases with geospatial features is reported in Table 4.

The comparison is based on the following key features:

- Types of supported geometry objects: types of objects that can be represented (e.g., points, lines, polygons)
- Implemented geometry functions: types of analysis and queries that can be executed by means of built-in functions.
- Spatial index support and types of supported indexes: the availability of indexes allows enhancing the performance of the queries.
- Support for GeoJSON and shapefile file formats: the native support of these data formats is important because many data sources use one of these two file formats

We leave the comment and the selection of the technical solutions to Section 4.4.

Name of the database	Supported Geometry objects	Main supported geometry functions	Supported Spatial indexes	Compatibility with GeoServer	Support for GeoJSON and Shapefile	DaaS
Postgresql with PostGIS extension	Point LineString Polygon MultiPoint MultiLineString MultiPolygon GeometryCollection	Inclusion Intersection Distance/Proximity Union Difference Overlap PostGIS supports the Open Geospatial Consortium (OGC) methods on geometry instances	B-Tree index R-Tree index GiST index	Yes	GeoJSON (import/export based on the GDAL OGR conversion library) Shapefile (import/export specific based on a tool/function of PostGIS)	No
Azure SQL Database	Point LineString Polygon MultiPoint MultiLineString MultiPolygon GeometryCollection	Inclusion Intersection Distance/Proximity Union Difference Overlap SQL Server supports the Open Geospatial Consortium (OGC) methods on geometry instances	2d plane index B-trees	Yes	GeoJSON (import/export based on the GDAL OGR conversion library) Shapefile (import/export based on the GDAL OGR conversion library)	Yes (Microsoft Azure cloud computing platform)

MongoDB	Point LineString Polygon MultiPoint MultiLineString MultiPolygon GeometryCollection	Inclusion Intersection Distance/Proximity	2dsphere index 2d index	Yes (based on the external MongoDB plug-in included in GeoTools) Note. The plug-in is in the unsupported branch of the current version of GeoTools	GeoJSON (native support) Shapefile (the files must be converted to the GeoJSON format)	Yes (MongoDB Atlas cloud service)
DocumentDB	Point LineString Polygon MultiPoint MultiLineString MultiPolygon GeometryCollection	Inclusion Distance/Proximity	2d plane index quadtree	Yes (by exploiting the external plug-in available for MongoDB) Note. The plug-in is in the unsupported branch of the current version of GeoTools	GeoJSON (native support) Shapefile (the files must be converted to the GeoJSON format)	Yes (Microsoft Azure cloud computing platform)

Table 4. Geospatial DB comparison

4.3 Data Catalog solutions overview

Data Catalog is a key component to search and obtain the information stored in the Big Data Storage and to keep track of the data stored. In this paragraph, the principal Data Catalog solutions are presented. Similarly to the Big Data Storage, this component can be on-premises or it can be exploited as a cloud service. Since metadata in compliance with INSPIRE Metadata are adopted, Data Catalog needs to have the possibility to support this type of metadata. INSPIRE metadata have been created for geospatial datasets and thus an important feature to consider is the possibility of querying the data stored using geospatial information. The in-cloud Big Data Storage providers cited in Section 184.1 offer also a Data Catalog software as a service, but they do not support INSPIRE metadata and they are generally more expensive. The on-premises software instead are various and can be proprietary software or open-source solutions. In the open-source software, the source code is released under a license in which the copyright holder grants users the rights to use, study, change, and distribute the software to anyone and for any purpose. On the contrary, proprietary software legally remains the property of the organization, group, or individual who created it, and the code is not publicly available. The following table (Table 5) compares open-source software with proprietary software according to the main aspects of security, stability, cost, warranty, community, and support.

Table 5. Comparison between Open Source and Proprietary software

	Open-Source Software	Proprietary Software
Security	Because anyone can view and modify open-source software, any developer might spot and correct errors or omissions that a program's original authors might have missed.	Security is totally in charge of the creators.
Stability	Stable versions of the software are released but not at regular time intervals. In case creators stop working to the project, it continues to get implemented by the community	Creator gives software which it was probed. If the software house stops distributing the software it is no more available.
Warranty	Limited or no warranty	Full warranty
Community	Large community of developers around open-source software	None or small community
Cost	Code is available for free	High cost per license
Support	Support is given by the community of developers, but it is not assured	Support is given by the software house
Additional functionalities	It is possible to edit the code publicly available to add new functionalities	Beside the functionalities implemented by the authors, it is not possible to add new custom features.

The cost and the possibility to tailor the software to the specific SHELTER needs represent the two main reasons for selecting the open-source software as the most appropriate for this project. There exists plenty of open-source Data Catalog software. The most suitable are listed in the following paragraphs. They are evaluated in terms of stability, community support, Big Data Storage integration, Geo spatial DB integration and ease to add new functionalities.

4.3.1 GeoNetwork

GeoNetwork [43] is an open-source catalog application to manage spatially referenced resources. It provides powerful metadata editing and search functions as well as an interactive web map viewer. It is currently adopted by several Spatial Data Infrastructure initiatives across the world. It provides an easy-to-use web interface to search geospatial data across multiple catalogs. The search function provides full-text search as well as faceted search on keywords, resource types, organizations, scale. The metadata editor supports INSPIRE standards used for spatial resources. Based on user profiles (e.g., reviewer, editor), a dashboard provides easy access to their information and tasks. The GeoNetwork community implemented the integration with Amazon S3 storage, but it can be adapted to other cloud storage services. GeoNetwork is open-source software written in Java. A mid-size community around the project actively propose improvements which need to be embedded modifying the original code.

4.3.2 GeoNode.js

GeoNode [44] is a Content Management System (CMS) for geospatial datasets, which allows the user to publish geospatial datasets with powerful visualization options, advanced search functionality, create interactive maps and collaborate with other users. GeoNode is also a powerful platform for developers for developing Geographic Information Systems (GIS) and for deploying Spatial Data Infrastructures (SDI). GeoNode CMS is built with Django, INSPIRE metadata integration is under evaluation and the integration with cloud storage services has not been implemented yet.

4.3.3 CKAN

CKAN [45] is an open-source data portal designed to allow publishing, sharing, and managing data sets. It provides many functionalities to managers and end-users, like full-text search, multi-lingual support, reporting tools and a powerful API to access the

data. CKAN comes with many options for data scientists including reporting, Geospatial options to publish share location-based data. The most important feature of CKAN is the possibility of extending its functionalities through plugins. CKAN is the most popular choice for governments and Non-Governmental Organizations (NGOs) to publish/share their datasets. Speed in querying the catalog is ensured by the adoption of Apache SOLR, a highly reliable, scalable and fault tolerant, distributed indexing method. The community has developed several numbers of plugins including cloud services integration, customizable metadata and spatial information support. Through plugins it is possible enabling geo spatial functions such as geo queries and map previews. This modification extends the PostgreSQL DB with PostGIS to store data spatial information.

CKAN is internally divided into different components, the behavior of which can be modified by a plugin implemented ad hoc. The lower layer is composed of the databases: the file storage that can be customize through plugin in order to use a cloud big data storage, a PostgreSQL to store metadata information and an instance of Apache SOLR, an Apache opensource project which allows to perform heavy and load-balanced queries to ensure fast and efficient data searches. The logic component encloses all the business logic of the data catalog including the authorization system and it is in charge of handle and perform user requests. CKAN capabilities can be exploited through a visual interface (Views component) or API requests (API component). On top of the last two components, the Routes component defines the connection between CKAN URLs and views that process requests and provide responses.

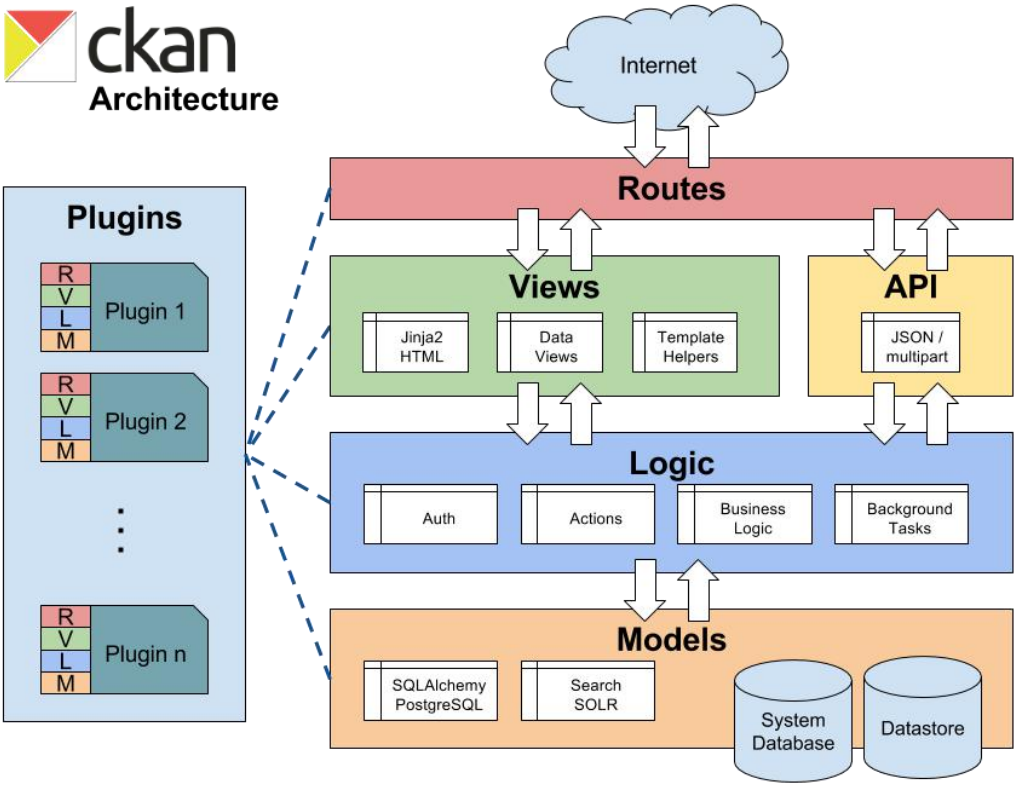

Figure 3. Ckan architecture

4.3.4 I-REACT Data Interface

The I-REACT Data Interface has been developed by LINKS on occasion of the I-REACT project [46], the first European-wide platform to integrate emergency management data coming from multiple sources, including that provided by citizens through social media and crowdsourcing. The I-REACT Data Interface (IDI) has been defined in terms of two separate components: a software module and an associated data format. The former is composed by a series of multiple adapters, each of which is used to extract data and metadata from the data formats at hand, accordingly. On the other hand, the defined data format specifies the format that must be used in the communication protocol between external modules and services for data exchange (Figure 4. IDI Architecture).

In this scenario, the IDI seemingly integrates with the Data Layer, acting either as a broker for communication to and from the data layer with external services and as a

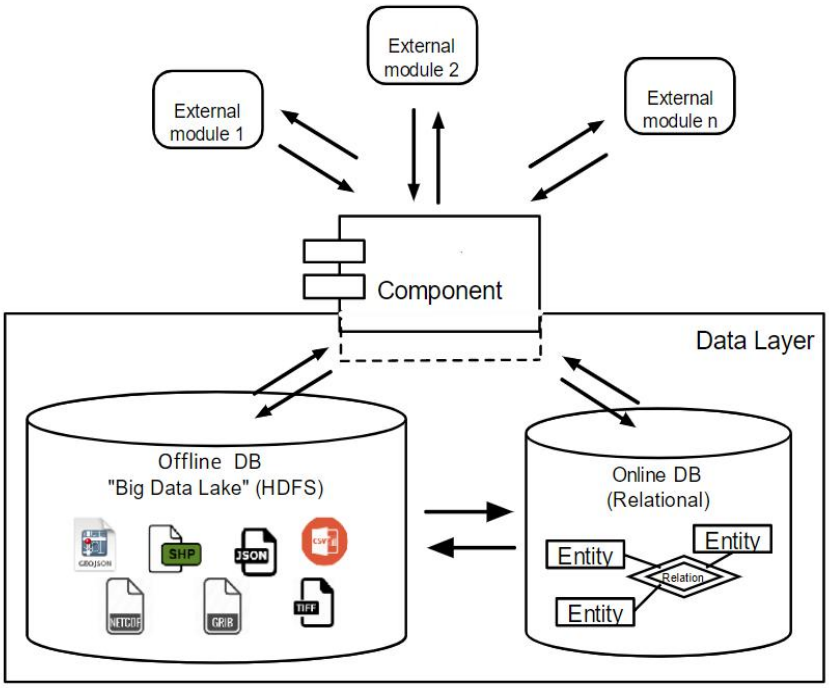

Figure 4. IDI Architecture

mediator for both the Offline and Online Storage. Once data and metadata are provided to the IDI, proper adapters are triggered. These adapters embed the specific rules and algorithms required to manipulate data and metadata from a specific data format. The IDI module instructs the different adapters so that rules and conditions to get the data are specified. Then, each adapter selects and fetches the data to be stored into the data storage. On the other hand, in case an external service asks the Data Layer for specific data, the IDI is responsible to fetch the data from the (online) storage, and to transform the obtained results into the IDI defined format to enable the data exchange process.

I-REACT Data Interface is fully compatible with INSPIRE Metadata and it can be integrated with Azure Blob Storage. The missing of a graphic interface imposes the users to access the data through HTTPS Requests, making the consultation of the catalog not very user friendly. Furthermore, in case of a consistent amount of data, querying the Online database can be slow and it needs a constant wipe of old and dismissed data.

4.4 Selection of the technical solutions

The designed Data Lake Architecture of Shelter is general and it can be implemented/instantiated by using the big data framework and geospatial database that best complies with the characteristics of the data (structured or not) and the types of analyses and queries that we will implement (based on the requirements collected by the other concurrent tasks). This also means that we can potentially have multiple instances of the proposed Data Lake architecture. For example, an instance based on open source and free tools and another instance, more easily manageable and more scalable, based on proprietary systems. Based on the qualitative comparison of the available Big Data Storage (see Table 3) and geo spatial DB (see Table 4) we think that the implementation of the Data Lake Architecture that better fits the requirements of Shelter is based on Azure Blob Storage for big data storage and PostgreSQL+PostGIS for metadata DB. The Data Catalog CKAN can be built joining these two requirements and enhance the solution with Apache SOLR to improve the database queries.

The service offered by the providers of Big Data storage is comparable, thus the decision is left to the cost analysis which yields to take into consideration the Microsoft Azure Blob Storage. Among the available databases, the selection of PostgreSQL with PostGIS is motivated by the following considerations:

- PostGIS provides advanced features for geospatial queries, which are not currently supported by MongoDB and DocumentDB
- PostgreSQL is more tightly integrated and supported by the GeoServer software than MongoDB and DocumentDB
- PostgreSQL with PostGIS overperforms the Azure SQL Database in spatial queries

The analysis of the Data Catalog solutions yields to choose CKAN Data Catalog because, although it is originally a simple portal for non-geospatial data, its community offers a wide number of plugins which enhance the functionalities and build a complete and performant Data Catalog. The popularity, the modularity and the possibility of easily developing own plugins have been the key reasons that guided the choice towards CKAN.

The technical choices that has been taken in this Section could change in case newer and most performant technologies will emerge during the implementation activity.

5 The SHELTER Data Lake

In this Chapter, a detail description of the solution implemented for the Data Lake of SHELTER project is given.

5.1 Architecture

The Shelter Data Lake aims to implement a repository for storing data and geospatial data produced within the project. Data will be uploaded to the platform by means of exposed APIs or Web based interface that will offer functionalities for uploading, retrieving and filtering both INSPIRE compliant metadata and data. This module will use a Big Data storage to store data in its raw format and a structured geospatial DB to store the associated metadata compliant with INSPIRE metadata directive. The geospatial DB allows executing queries that select all the objects contained in a geographical area or any other queries on the metadata fields. A Data Management System handles the operations of the aforementioned data storages (Big Data and geospatial DB) that can be performed through API or web based interface and allows uploading, retrieving, and viewing the data and metadata to the authorized users. Data Lake will exchange messages with other modules of the system by means of a message broker following the publisher/subscriber paradigm, that will notify subscribed services about new data available. Figure 5 shows the architerctural details. As stated in Section 4.4, for Shelter Data Lake implementation we have chosen to use CKAN data catalog as data management system, PostgreSQL with postGIS extension as geospatial DB and Microsoft Azure Blob Storage as big data storage.

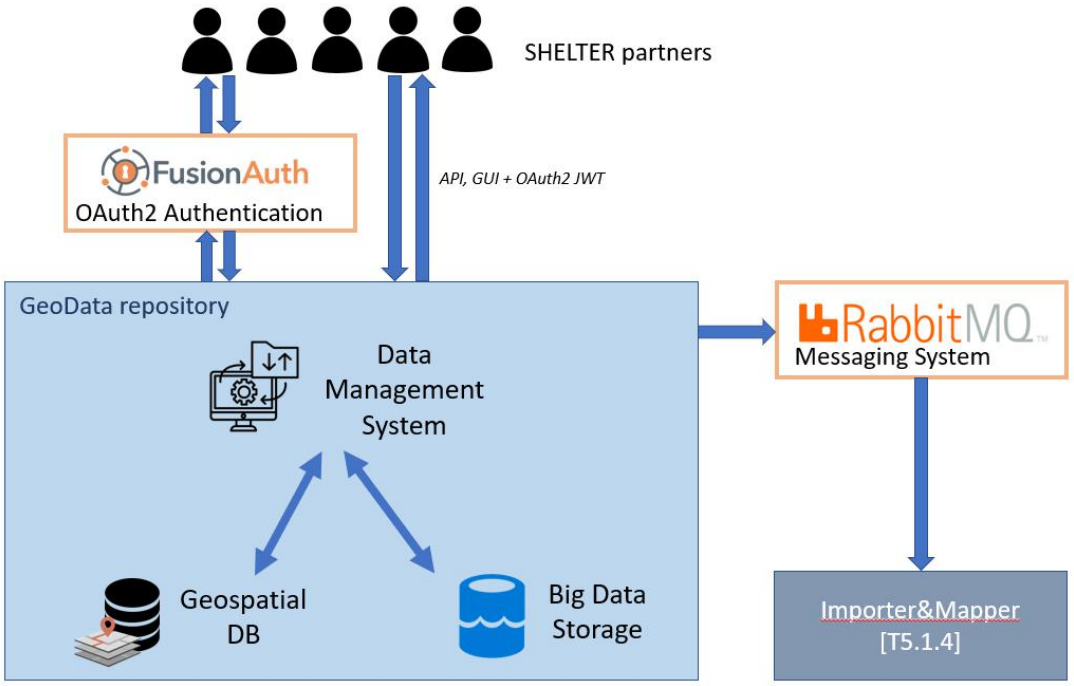

Figure 5. Data Lake Architecture

5.2 Metadata personalization

By adopting INSPIRE metadata (listed in Appendix), Selter platform ensures an accurate description of all the data stored in the Data Lake. To better support internal operations, additional metadata fields are introduced and listed in Table 9: List of additional SHELTER metadata (Appendix).

Among the INSPIRE metadata fields, the field "keyword" will support values originate from a dedicated SHELTER ontology. This domain ontology, identified in D1.2 - Building of best/next practices observatory, is an effective codification of existing knowledge and existing ontologies tailored to the requirements of the project in order to obtain adaptive governance for CH management, local knowledge, historical event and linked projects analysis.

5.3 User Authentication

To upload and explore the datasets in the Data Lake, the user must be logged in with a valid e-mail and password. The authentication process is delegated to FusionAuth [47], a Customer Identity Access Management (CIAM) which implemented OAuth 2.0 standard authentication flows for Single Sign-On. When a user authenticates, FusionAuth returns a JSON Web Token (JWT) to the client application and a refresh token if suitable.

5.4 Data Management System customization

Data Management System aims to handle operations of upload, modify, retrieve and search data stored in the databases. It can be used through API or web based interface, for Shelter implementation we have chosen to use CKAN data catalog. The CKAN base version has been expanded with other functionalities through plugins, some of which were already implemented by the community, and others implemented by our own.

According to the requirements, the first plugin added allows several functionalities about the geospatial data. The "spatial" plugin [48] introduces the possibility of adding a spatial extent to the metadata and search data querying the spatial field, and the geoview plugin allows to obtain a preview map of the data stored in the Big Data Storage.

For the scope of SHELTER platform, it is essential to search data that are valid within certain datetime intervals, thus the plugin "datesearch" [49] is adopted and adapted to query the database and obtain data valid in specifying a datetime interval.

To upload and access to the data, the users must be logged into the platform. The login process takes place through a plugin "OAuth2" [50] developed by the community that interfaces with FusionAuth chosen as project user authentication. FusionAuth provides authentication, authorization and user management using OAuth2 standard protocol.

The integration with the Big Data Storage provided by Microsoft Azure service occurs through the plugin "cloudstorage" [51] which ensures that the data are uploaded directly to the blob storage and map the data Universal Resource Locator (URL) into the database to retrieve it when the user accesses the resource.

The metadata fields are customized through the plugin "scheming" [52] and are set up to be compliant to the INSPIRE metadata.

The customization of the CKAN base version has been carried out adopting the most useful plugins implemented by the community. Furthermore, other plugins have been developed in-house to enhance CKAN functionalities. In particular, a theme plugin has been implemented to customize the user interface and a notify plugin has been adopted to send notifications to a messaging system at every modification of the data storage (upload, modification or deletion).

5.4.1 Users

CKAN's authorization system controls which users are allowed to carry out specifications on the site. All actions that users can carry out on a CKAN site are controlled by the authorization system. For example, the authorization system controls who can register new user accounts, delete user accounts, or create, edit, and delete datasets and organizations.

Organizations are the primary way to control who can see, create and update datasets in CKAN. Each dataset can belong to a single organization, and each organization controls access to its own datasets.

Datasets can be marked as public or private. Public datasets are visible to everyone. Private datasets can only be seen by logged-in users who are members of the dataset's organization. Private datasets are not shown in general dataset searches but are shown in dataset searches within the organization.

When a user joins an organization, an organization admin provides them with one of the three following roles: member, editor, or admin.

An organization admin can:

- View the organization's private datasets
- Add new datasets to the organization
- Edit or delete any of the organization's datasets
- Make datasets public or private.
- Add users to the organization, and choose whether to make the new user a member, editor or admin
- Change the role of any user in the organization, including other admin users
- Remove members, editors or other admins from the organization

- Edit the organization itself (for example: change the organization's title, description or image)
- Delete the organization

An editor can:

- View the organization's private datasets
- Add new datasets to the organization
- Edit or delete any of the organization's datasets

A member can:

- View the organization's private datasets.

Users can access the Data Catalog functionalities through HTTPS RPC-style API and graphic interface published at the following address: https://datalake.shelter-project.cloud. Only authenticated users can view and upload datasets. The login procedure takes place through OAuth 2.0 SHELTER portal with the credentials provided to each partner of the project. If the authentication process succeeds, the user lands to the *dashboard* page where he can find all the last activities and updates on the datasets he is following.

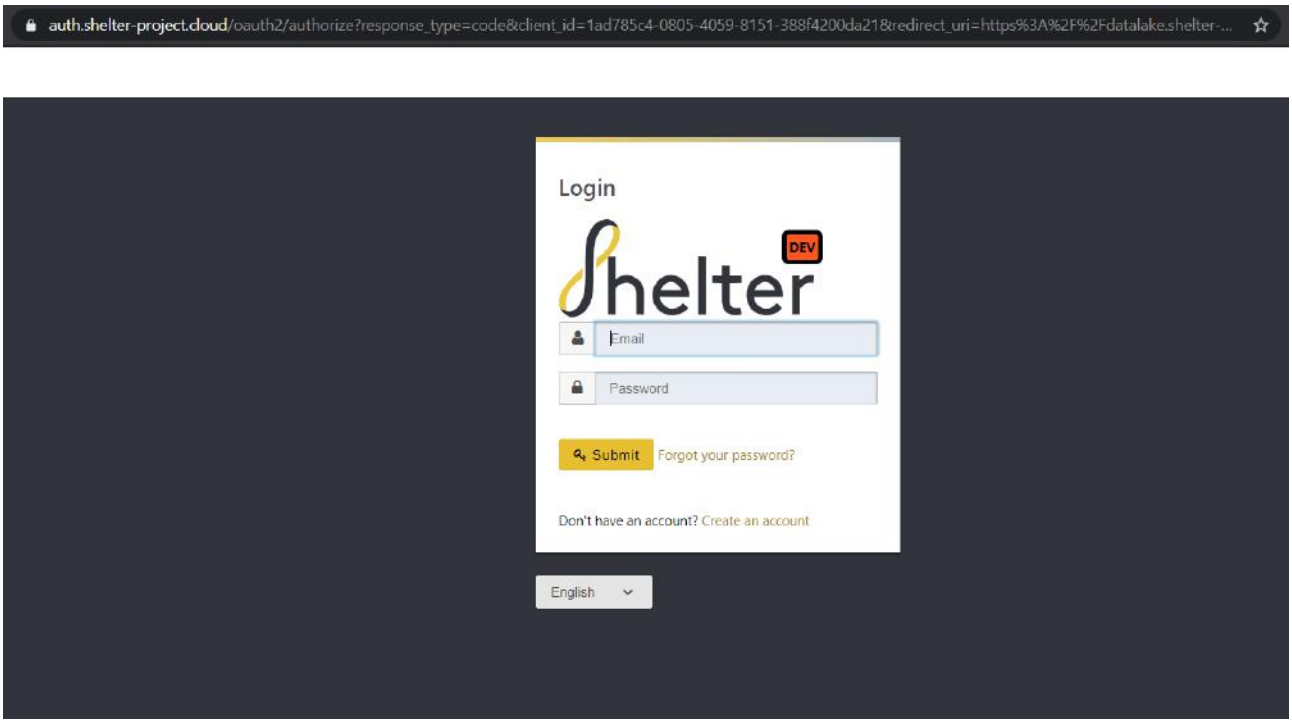

Figure 6. Login page

5.4.2 User Interface

CKAN provides a user interface through which the user can explore the datasets stored in the Data Lake, search through filters, and manage his/her own datasets.

5.4.2.1 User authentication

In order to have access to the resources stored in the Data Lake, the user must be logged in. The login page is accessible through the login button on the top left corner of the CKAN frontend. The button redirects to the FusionAuth component which provides the user authentication through e-mail and password (Figure 6. Login page). When the user is authenticated, the Data Catalog dashboard page is displayed. This page contains a summary of the last actions performed by the user in the tool.

5.4.2.2 Adding a new dataset

Only users registered in an organization with the role of "editor" or "admin" can upload a new dataset.

From the Dataset tab, it is possible to add a new dataset by clicking on the button "Add Dataset". The web page will prompt to fill the metadata fields of the new resource the user wants to add. The fields are compliant with the INSPIRE metadata (ISO 19115/ISO 19119) and they are described in the appendix at the end of this document. If the dataset needs to be not publicly accessible, the editor must select "Private" in the "Visibility" field (Figure 7. Add a new metadata procedure).

Once the metadata form is filled, by clicking on "Next: Add Data" button at the bottom of the page, it is possible to upload the resources. The user selects "Upload" if the file to be uploaded is locally stored, or in the alternative he/she can click on "Link" to specify the URL address of the resource. For the other fields:

- Name: Name of the resource
- File start date: validity start date and timestamp
- File end date: validity end date and timestamp
- Format: user can specify the file extension. If the zip file contains a shapefile, "zip" must be written in this field to see a map preview of the resource when users consult the catalog.

If the user has more than one resource (file or link) to add to the dataset, he/she must select the "Save & add another" button. Once finished adding resources, he/she must select "Finish".

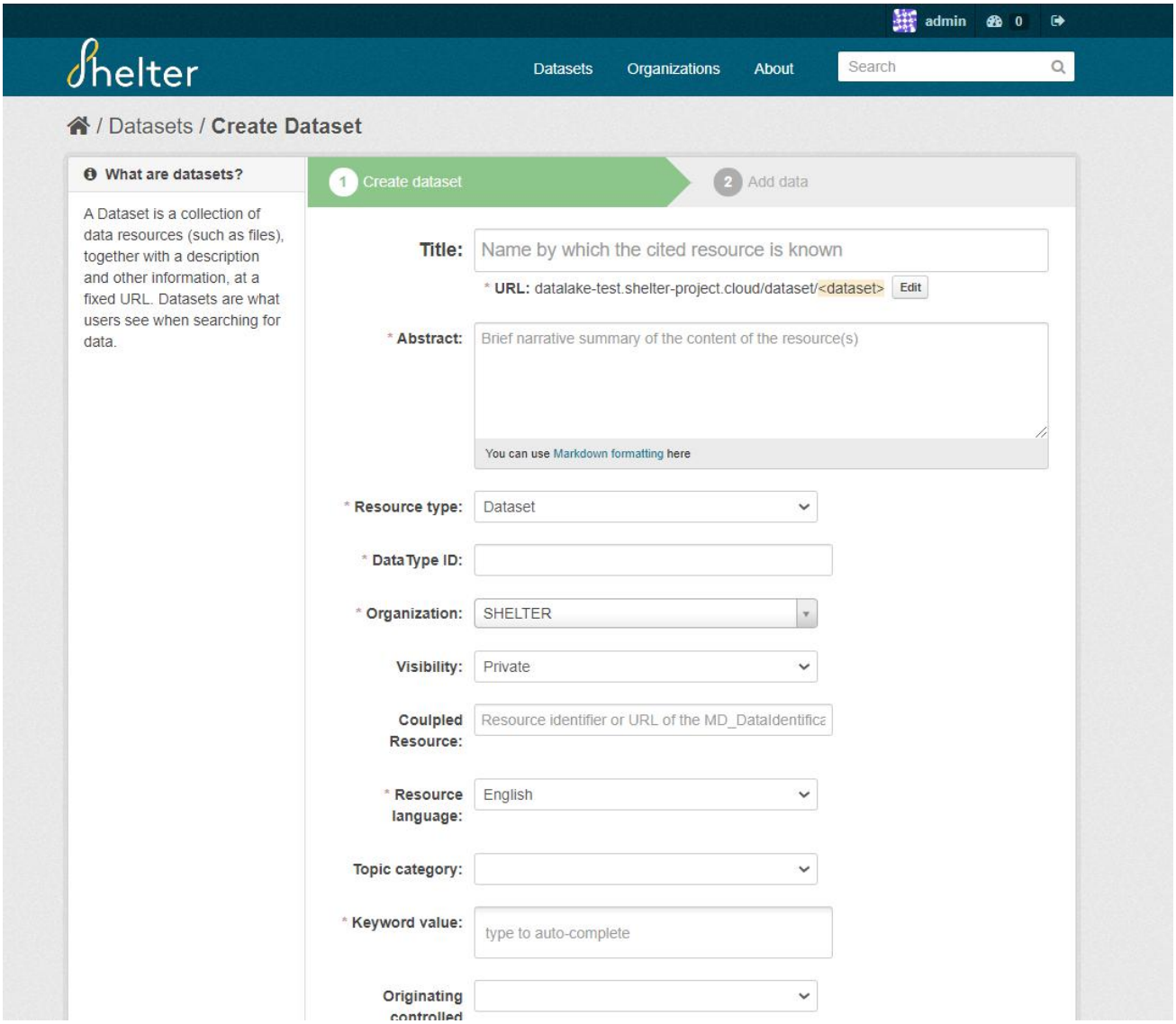

Figure 7. Add a new metadata procedure

5.4.2.3 Datasets search

In the "Dataset" tab (Figure 8) the user can browse the existing data stored in the Data Lake, and perform a search (various filters are available). Through the upper search bar, it is possible to find keywords in the metadata fields and resource descriptions.

The sidebar is useful to search across specific fields of the metadata. The temporal filter "Filter by date" allows selecting metadata valid in the specified datetime interval. Hence, the catalog checks if the metadata "Start Date" and "End Date" have any intersection with the chosen dates.

The spatial filter "Filter by location" allows selecting the datasets that refer to a geographical area. By clicking on the pencil icon on the upper right corner of the map, it is possible to draw the wished bounding box where to search for datasets.

Tags filter is used for selecting the dataset according to the keywords specified in the metadata during the upload process (see metadata fields provided in the Appendix). The keywords in SHELTER platform must conform to the SHELTER ontology identified in D1.2 which adapts existing ontologies to better suit to the CH management, local knowledge and historical event. Furthermore, it is possible to filter for the format of the resources uploaded and their license.

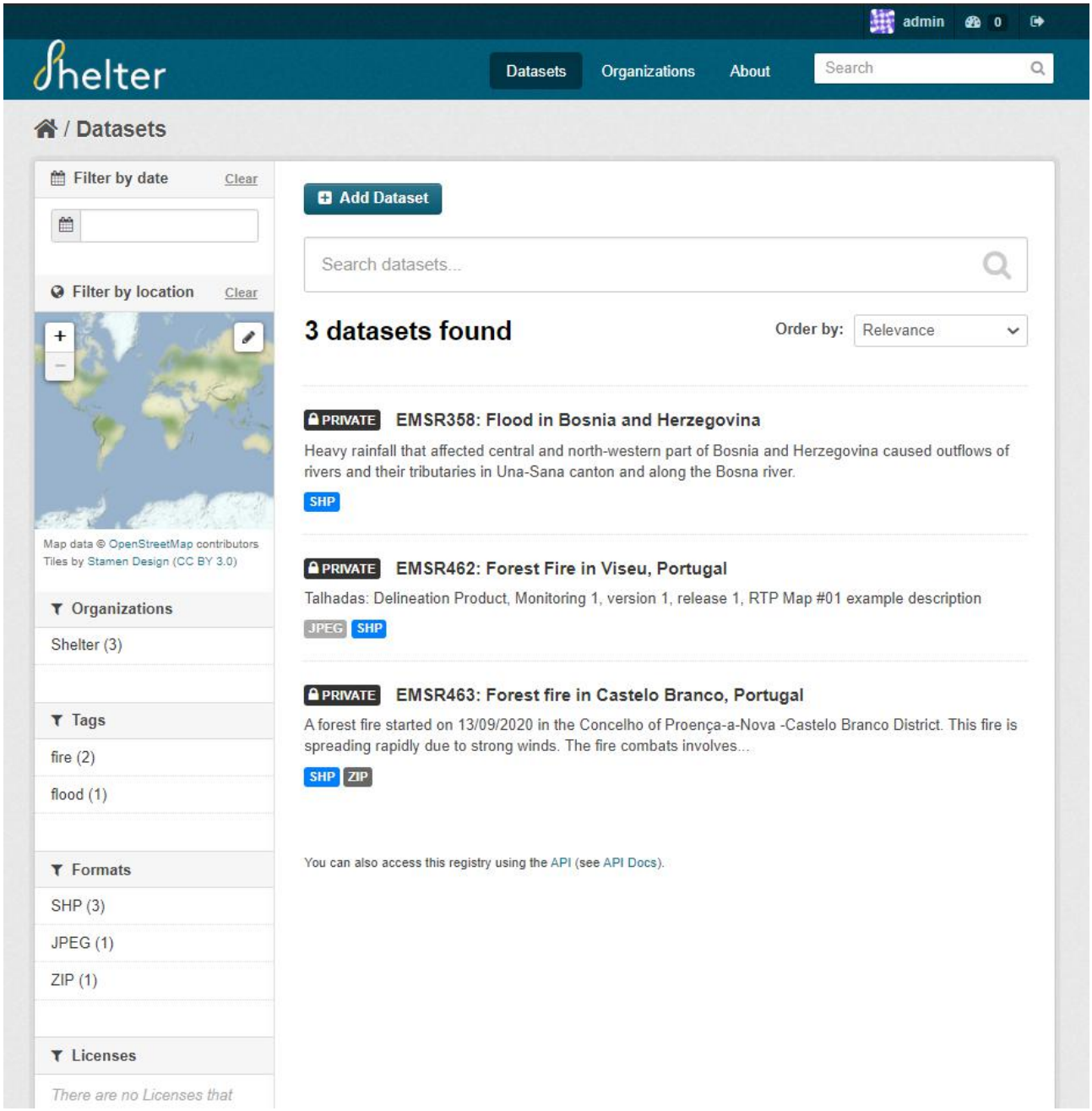

Figure 8. Dataset Tab

5.4.2.4 Datasets view

By clicking on a Dataset, it is possible to explore all the metadata fields and, if the file extension is supported, to have a preview of the resource stored in the Big Data Storage (Figure 9. Data preview from Data Catalog). In particular, in case of shapefiles, the Data Catalog allows to navigate a map and to see to which location the data refers. Naturally, the URL to download the resource from the Big Data Storage is also provided.

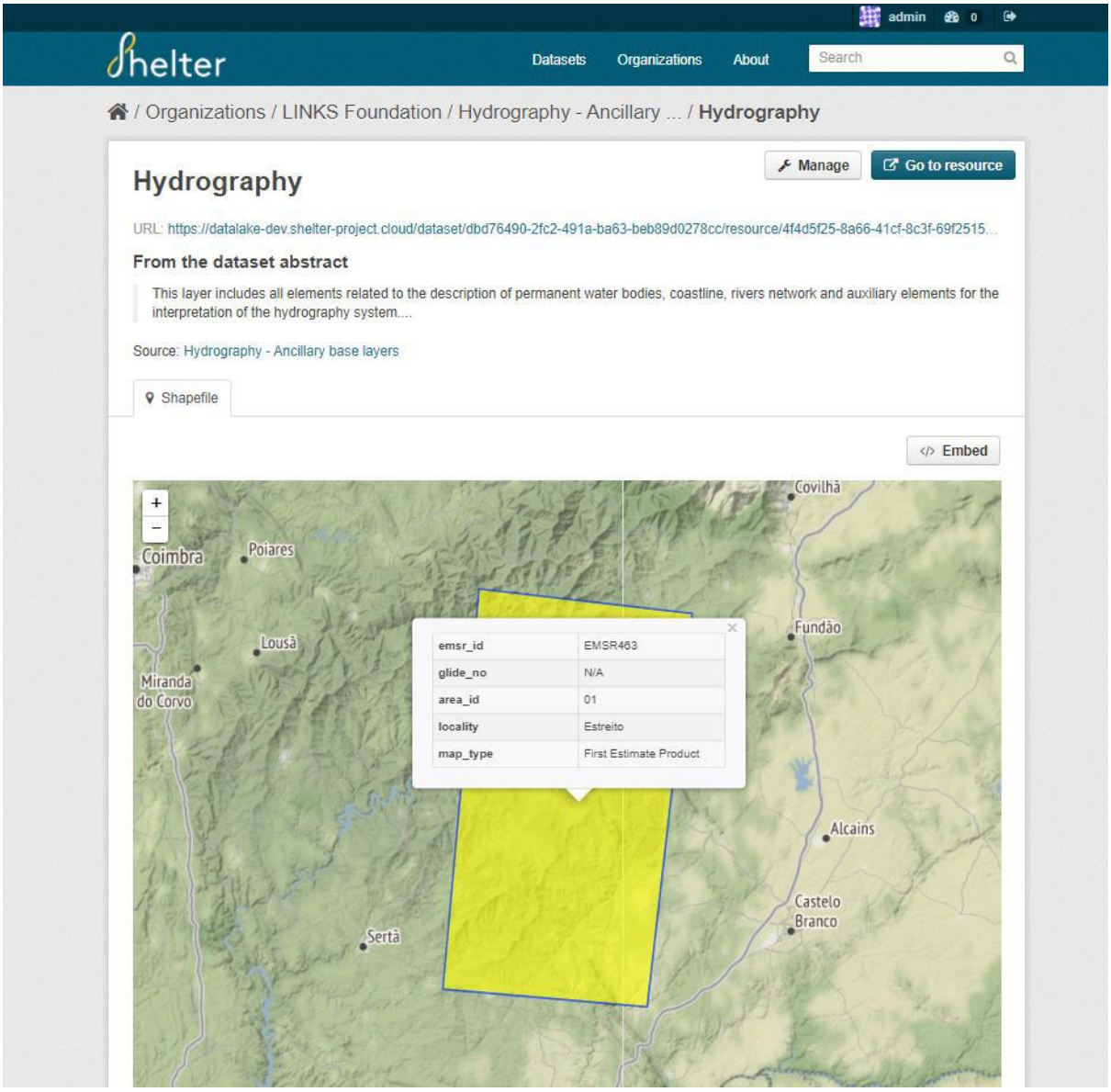

Figure 9. Data preview from Data Catalog

5.4.3 Use of CKAN through API

The very same actions described in the previous paragraph can be performed using CKAN API. CKAN's Action API is a powerful, RPC-style API that exposes all CKAN's core features to API clients. All the CKAN website's core functionalities can be used by external code that calls the CKAN Action API. To call these APIs, the user can send an HTTPS Request to https://datalake.shelter-project.cloud/api/3/action/[API_name]. The complete list of the available API can be found here: https://docs.ckan.org/en/2.6/api/.

Some API functions require authorization. The API uses the same authorization functions and configuration as the web interface, so if a user is authorized to do something in the web interface, he will be authorized to do it via the API as well. The authorization is based on OAuth2.0 protocol and it is provided by FusionAuth (as explained in paragraph 5.3). Hence, the API requests need to also include the Access Token to the Request Headers.

5.4.3.1 User authentication

In order to send valid API requests to the Data Catalog, every API call must include the JWT in the Authorization header using the Bearer schema. To obtain the JWT, the user needs to exploit FusionAuth API [53] https://auth.shelter-project.cloud/api/login. The login request must contain a JSON body message with user e-mail, user password and SHELTER application id. Furthermore, an API Key in the Authorization header is required to obtain a valid response. Below, an example of the login API request.

Request body message:

```
{
      "loginId": user@shelter.cloud,
      "password": 12345,
      "applicationId": applicationID,
      "noJWT": False
}
```

Request header:

```
{
      "Authorization": API_key
}
```

The API_key and the ApplicationID will be provided to the SHELTER Data Catalog users to allow them to get valid JWT.

The response to this request consists of a JSON dictionary containing user information and the JWT access token (‘token’ field). The JWT must be added in the Authorization header of all of the CKAN APIs described in the following paragraphs. An example of Authorization header for CKAN API is the following:

```
headers = {
        "Authorization": 'Bearer [JWT]',
    }
```

where [JWT] is the access token obtained through the Fusion Auth login.

5.4.3.2 Adding a new dataset through APIs

The adding process is performed through two consecutive POST requests. The former creates new metadata calling the API "package_create". In the body of the request, it is possible to specify a JSON dictionary in which the keys are the metadata fields. In Appendix the table with the description of the metadata fields is reported, the column "CKAN field name" indicates the name to be used as a dictionary key. Furthermore, an example of the JSON dictionary required for this API call is given. If the request succeeds, the response contains the id of the metadata created. For adding the resources linked to the metadata a second POST request is necessary using the API "resource_create" that takes a JSON dictionary containing the information specified in the Table 6, and the file to be uploaded.

Table 6. Dictionary keys for resource upload

Dictionary key	Description
package_id	The id returned by the POST request to insert the metadata
name	Name of the file
format	File extension
file_date_start	Start datetime of validity (ISO 8601 format)
file_date_end	Stop datetime of validity (ISO 8601 format)

The following Python script contains an example of the POST request to add the resource to the Data Lake:

```
response = requests.post(url,
                         data=resource_body,
                         headers=headers,
                         files=[('upload', file)])
```

where:

- "url" is the full address containing the API_name (for example https://datalake.shelter-project.cloud/api/action/resource_create)
- "resource_body" is the JSON dictionary containing the information specified in Table 6
- "headers" contains the Authorization key as stated in paragraph 5.4.3.1
- "file" is the python pointer to the resource that the user wants to upload

5.4.3.3 Datasets search using APIs

To search a dataset, the user can use a GET or a POST request. The API "package_search" allows writing a query to filter the search results.

The main query parameters are:

- "include_private": it allows to search and return private datasets that are the datasets not publicly visible.
  Example:
  ```
  { "include_private": true }
  ```

- "ext_bbox": here the user can specify the bounding box within which performing the search.
  Example:
  ```
  { "ext_bbox": "14.238281,44.119142,17.666016,45.050240"}
  ```

- "ext_startdate" and "ext_enddate": the datetime in ISO 8601 format to filter the *Temporal Extent* metadata fields.
  Example:
  ```
  {"ext_startdate":"2019-05-01T00:00:00Z",
  "ext_enddate":"2020-10-30T23:59:59Z"}
  ```

- "fq_list": additional filter queries to apply.
  Example:
  ```
  { "fq_list":
        ["datatype_id:5103",
        "responsable_organization_email:rapidmapping@ems-copernicus.eu"]
  }
  ```

5.4.4 Messaging system

When new metadata are created and the related resource is associated, a notification is sent through the SHELTER message broker that has been developed in T5.1 - Common components of the data-driven platform. The software identified for this purpose is RabbitMQ, an open-source message broker that accepts messages from producers and delivers them to consumers through a queue system (Figure 9).

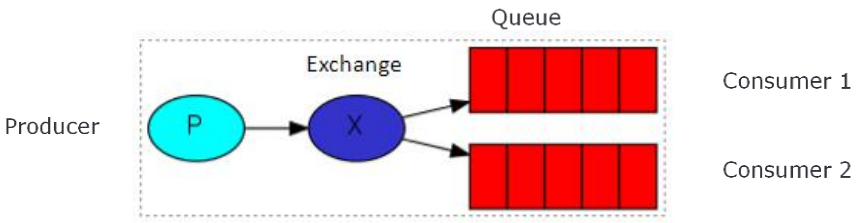

Figure 10 RabbitMQ components

RabbitMQ implements the common messaging system that allows the asynchronous exchange of small data and signal following the publish and subscribe paradigm. The paradigm implemented by this model is composed of one or more producers sending a message to a queue and one or more consumers reading the messages.

In SHELTER messaging system implementation, messages are not directly published to a queue. Instead, the producer sends messages to an exchange. Exchanges are message routing agents, defined by the virtual host within RabbitMQ. An exchange is responsible for routing the messages to different queues with the help of header attributes, bindings, and routing keys. A binding is a "link" that has to be set up to bind a queue to an exchange. The routing key is a message attribute the exchange looks at when deciding how to route the message to queues.

In RabbitMQ, four types of exchanges route the message using different parameters and bindings setups. In the SHELTER implementation, the "topic" exchange named "shelter.b2b" is used. Topic exchanges route messages to queues based on wildcard matches between the routing key and the routing pattern, which is specified by the queue binding. Messages are routed to one or many queues based on a matching between a message routing key and this pattern. The routing keys adopted are composed as follows:

`newexternaldata.WP.Task.DataTypeID`

where "WP" is the number of the Shelter Work Package of reference, "Task" is the task number generating the dataset and "DataTypeID" is the data type id number specified in the metadata.

Following, a queue for each project partner has been created in order to receive and collect all the notifications with routing key matching to newexternaldata.# (the symbol "#" means one or more words). From the queue, each partner project can read the notification and use the information into other Shelter components.

Through the herein described messaging system, it is possible to notifiy the insertion, modification and deletion of new data into the Data Lake, and thus to activate the services of the Map Layer. The message inserted to the queue is a JSON dictionary and

it contains basic information about the metadata (described in Table 7. Information published on RabbitMQ queue). Following an example of the message:


```
{
    "geoJSON": {
        "type":"MultiPolygon",
        "coordinates":[[[[-8.114502355456352,39.956701348548584],
                         [-8.114502355456352,40.501895973348496],
                         [-7.598144933581352,40.501895973348496],
                         [-7.598144933581352,39.956701348548584],
                         [-8.114502355456352,39.956701348548584]]]]
                    },
    "dateOfCreation": "2019-09-15T23:59:59",
    "end": "2019-09-15T23:59:59",
    "start": "2019-09-15T00:00:00",
    "datatype_id": "51001",
    "action": "update/create",
    "id": "dbd76490-2fc2-491a-ba63-beb89d0278cc"
}
```


Table 7. Information published on RabbitMQ queue

Field name	Description
id	Metadata id
datatype_id	Type of metadata inserted. It can be also a list.
action	"update/create" or "delete" based on the action performed by the user
dateOfCreation	Metadata date of creation
start	Start validity date
end	End validity date
geoJSON	Metadata spatial information in geoJSON format

6 Conclusion

The main objective of this deliverable is to provide the design of the identified solution for the Data Lake which is the SHELTER component in charge of managing the heterogeneous data identified within the project. Diverse solutions have been compared, identifying the ones that better suit the SHELTER use case.

The document delivers a set of components to ensure a high-reliability service of data storage and efficiency in collecting the information. Analyzing the efficient solutions for storing the relevant historical information of Open Labs sites, the report delivers a highly adaptive tool for the diverse Historic Environments. The Data Lake ensures a high level of data quality and accessibility for users and allows the application of data analytics techniques. The Metadata used to describe the heterogeneous data uploaded in the Data Lake are compliant with the INSPIRE Implementing Rules on interoperability of spatial data sets and services (IRs) and Technical Guidelines (Data Specifications) which specify common data models, code lists, map layers and additional metadata on the interoperability to be used when exchanging datasets. By using INSPIRE compliant metadata, the SHELTER platform ensures easy interoperability with European Union Data Catalogs which provide satellite dataset of critical events. The data collected in the Data Lake can be a great source of information but to exploit them to their full potential, the metadata must contain precise information about the dataset. Incorrect metadata can cause the non-traceability of the dataset during the user searches.

The Data Lake has been designed considering the need of having a general-purpose Big Data storage that could support data exchange between SHELTER tools and the components of the Data-Driven Platform. Specifically, the use of INSPIRE compliant metadata allows to achieve the FAIR principle in the SHELTER data management, thus making data Findable, Accessible, Interoperable and Reusable. Therefore, the Data Lake can be used to store and retrieve data produced any of the tasks included in SHELTER. For example, it can host historical events & social memory DB, collaborative best/next practices, portfolio of solutions (T6.3), governance schemes maps (T6.4), Co-creation processes blueprints (T6.3) and resilience financing and business models (T6.6), etc..

The next steps in SHELTER should be focused on the implementation of the Data Lake in WP5 and the insertion of the datasets identified within T1.1 and generated by the other WPs, reporting any further needs which require a modification of the behavior of the Data Lake described in this document. The design proposed in this document aims also at bringing the data beyond SHELTER. The implementation of the Data Lake allows the creation of a dataset repository containing historical datasets needed to perform an accurate study of the historic areas considered in the SHELTER project. The definition of the metadata with a dedicated ontology will reinforce the interoperability and reusability of the datasets and the knowledge ensuring their operationalization after the end of the project.

7 References

[1] "IDC report, Worldwide Global DataSphere IoT Device and Data Forecast, 2019-2023".

[2] "Natalia Miloslavskaya, Alexander Tolstoy, Big Data, Fast Data and Data Lake concepts, Procedia Computer Science, Volume 88, 2016, Pages 300-305, ISSN 1877-0509,".

[3] "James Dixon, Pentaho, Hadoop and Data Lakes, Retrieved 10 Aug 2017. jamesdixon.wordpress.com/2010/10/14/pentaho-hadoop-and-data-lakes/".

[4] " Russom, Philip. "Big data analytics." TDWI best practices report, fourth quarter 19.4 (2011): 1-34.".

[5] "MongoDB https://www.mongodb.com/".

[6] "Microsoft Azure DocumentDB https://azure.microsoft.com/en-us/services/documentdb/".

[7] "Apache Hadoop http://hadoop.apache.org/".

[8] "Apache Spark http://spark.apache.org/".

[9] "https://inspire.ec.europa.eu/documents/inspire-metadata-implementing-rules-technical-guidelines-based-en-iso-19115-and-en-iso-1".

[10] "ESRI Shapefile Technical Description http://www.esri.com/library/whitepapers/pdfs/shapefile.pdf 01/07/1998".

[11] "[RD09] The GeoJSON Format https://datatracker.ietf.org/doc/rfc7946/".

[12] "GeoTIFF https://trac.osgeo.org/geotiff/".

[13] "JSON http://www.json.org/".

[14] "GDAL - ogr2ogr," [Online]. Available: https://gdal.org/programs/ogr2ogr.html.

[15] "H. Hu, Y. Wen, T. S. Chua and X. Li, "Toward Scalable Systems for Big Data Analytics: A Technology Tutorial," in IEEE Access, vol. 2, no. , pp. 652-687, 2014.".

[16] "Jae-Gil Lee, Minseo Kang, "Geospatial Big Data: Challenges and Opportunities," Big Data Research, Volume 2, Issue 2, June 2015, Pages 74-81, ISSN 2214-5796, 2015.".

[17] "Bohlouli, M., Schulz, F., Angelis, L., et al., 2013. Towards an integrated platform for big data analysis. In: Fathi, M. (Ed.), Integration of Practice-Oriented Knowledge Technology: Trends and Prospectives. Springer Berlin Heidelberg, p.47–56".

[18] "Azeem, R., Khan, M.I.A., 2012. Techniques about data replication for mobile ad-hoc network databases. Int. J. Multidiscipl. Sci. Eng.".

[19] "Wang, X., Sun, H.L., Deng, T., et al., 2015. On the tradeoff of availability and consistency for quorum systems in data center networks. Comput. Network".

[20] "Oracle, 2015a. Managing Consistency with Berkeley DBHA (White Paper).".

[21] "Hilker, S., 2012. Survey Distributed Databases—Toad for Cloud".

[22] "Cattell, R., 2010. Scalable SQL and NoSQL data stores. SIGMOD Rec.".

[23] "Siddiqa, A., Karim, A. & Gani, A. Big data storage technologies: a survey. Frontiers Inf Technol Electronic Eng 18, 1040–1070 (2017).".

[24] "Borthakur, D., 2008. HDFS Architecture Guide.".

[25] "Shvachko, K.V., 2010. HDFS scalability: the limits to growth.".

[26] "https://www.ibm.com/cloud/storage".

[27] "https://www.computerweekly.com/news/252449283/Object-storage-On-prem-in-the-cloud-and-hybrid".

[28] B. L. e. all., IBM Cloud Object Storage Concepts and Architecture - System Edition, vol. http://www.redbooks.ibm.com/redpapers/pdfs/redp5537.pdf, http://www.redbooks.ibm.com/redpapers/pdfs/redp5537.pdf.

[29] "https://www.ibm.com/cloud/object-storage/storage-class-tiers-archive," [Online].

[30] "https://azure.microsoft.com/en-us/services/storage/blobs".

[31] "https://aws.amazon.com/s3/".

[32] "https://cloud.google.com/storage".

[33] RightScale, "Cloud Storage Comparison: AWS vs Azure vs Google vs IBM," [Online]. Available: https://www.slideshare.net/rightscale/cloud-storage-comparison-aws-vs-azure-vs-google-vs-ibm.

[34] "PostGIs http://postgis.net/".

[35] "Microsoft Azure SQL Database https://azure.microsoft.com/en-us/services/sql-database/".

[36] "OGC Specifications, Simple Feature Access Part 1 - Common Architecture http://www.opengeospatial.org/standards/sfa".

[37] "OGC Specifications, Simple Feature Access Part 2 – SQL Options http://www.opengeospatial.org/standards/sfs".

[38] "ArcGIS http://www.esri.com/software/arcgis".

[39] "GeoServer http://geoserver.org/".

[40] "Microsoft SQL Server 2019 https://www.microsoft.com/en-us/sql-server/sql-server-2019".

[41] "MongoDB Spark Connector https://github.com/mongodb/mongo-spark".

[42] "MongoDB Connector for Hadoop https://docs.mongodb.com/ecosystem/tools/hadoop/".

[43] "https://geonetwork-opensource.org/".

[44] "http://geonode.org/".

[45] "https://ckan.org/".

[46] "http://project.i-react.eu/".

[47] "https://fusionauth.io/," [Online].

[48] "https://github.com/ckan/ckanext-spatial," [Online].

[49] "https://github.com/geosolutions-it/ckanext-datesearch," [Online].

[50] "https://github.com/conwetlab/ckanext-oauth2," [Online].

[51] "https://github.com/open-data/ckanext-cloudstorage," [Online].

[52] "https://github.com/ckan/ckanext-scheming," [Online].

[53] "https://fusionauth.io/docs/v1/tech/apis/login," [Online].

[54] "opensource.com," [Online]. Available: https://opensource.com/resources/what-open-source.

[55] "GeoNetwork," [Online]. Available: https://geonetwork-opensource.org/.

[57] "Jeffrey Dean and Sanjay Ghemawat. 2008. MapReduce: simplified data processing on large clusters. Commun. ACM 51, 1 (January 2008), 107-113. 2008".

[58] "Quix C., Hai R. (2018) Data Lake. In: Sakr S., Zomaya A. (eds) Encyclopedia of Big Data Technologies. Springer, Cham. https://doi.org/10.1007/978-3-319-63962-8_7-1".

[59] "Sanjay Ghemawat, Howard Gobioff, and Shun-Tak Leung. 2003. The Google file system. In Proceedings of the nineteenth ACM symposium on Operating systems principles (SOSP '03). ACM, New York, NY, USA, 29-43. 2003".

[60] "Jeffrey Dean and Sanjay Ghemawat. 2008. MapReduce: simplified data processing on large clusters. Commun. ACM 51, 1 (January 2008), 107-113. 2008".

[61] M. C. T. D. A. D. J. M. M. M. M. J. F. S. S. a. I. S. Matei Zaharia, ""Resilient distributed datasets: a fault-tolerant abstraction for in-memory cluster computing." In Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation (NSDI'12). USENIX Association, Berkeley, CA, USA, 2-2, 2012".

[62] M. C. M. J. F. S. S. a. I. S. Matei Zaharia, ""Spark: cluster computing with working sets." In Proceedings of the 2nd USENIX conference on Hot topics in cloud computing (HotCloud'10). USENIX Association, Berkeley, CA, USA, 10-10, 2010".

[63] "INSPIRE Metadata," [Online]. Available: https://inspire.ec.europa.eu/metadata/6541.

Appendix

All the datasets that will be produced shall be accompanied by metadata. Metadata will be delivered in an INSPIRE Directive (2007/2/EC) conform ISO 19115/ISO 19119 metadata standard.

The overall structure of the ISO 19115/ISO 19119 metadata set supporting the requirements expressed in the INSPIRE Implementing rules for metadata, is included and explained in the "INSPIRE Metadata Implementing Rules: Technical Guidelines based on EN ISO 19115 and EN ISO 19119" document, which is available online (http://inspire.ec.europa.eu/documents/Metadata/MD_IR_and_ISO_20131029.pdf).

The minimum set of metadata elements necessary to comply with INSPIRE Directive have been provided to the partners.

In Table 8: INSPIRE compliant metadata, the INSPIRE[1] compliant metadata are listed. It has been defined selecting the mandatory field of the INSPIRE directive and it is applied to all data that will be inserted into the Geo Importer because through the IDI it implements a data repository capable of sharing data (files) and service map layers through OGC standards.

Table 8: INSPIRE compliant metadata

	Minimum set of metadata elements necessary to comply with Directive 2007/2/EC

[1] https://inspire.ec.europa.eu/

	A) Metadata according to INSPIRE Metadata Regulation (COMMISSION REGULATION (EC) No 1205/2008 of 3 December 2008 implementing Directive 2007/2/EC of the European Parliament and of the Council as regards metadata)			
ref	**group of metadata elements**	**metadata element**	**CKAN API Field name**	**description**
1	IDENTIFICATION			
1.1		Resource Title	title	This a characteristic, and often unique, name by which the resource is known. The value domain of this metadata element is free text.
1.2		Resource Abstract	notes	This is a brief narrative summary of the content of the resource. The value domain of this metadata element is free text.
1.3		Resource Type	identification_ResourceType	This is the type of resource being described by the metadata. The value domain of this metadata element is defined in Part D.1. 1.1. Spatial data set series (series) 1.2.

				Spatial data set (dataset) 1.3. Spatial data services (services)
1.4		Resource Locator		The resource locator defines the link(s) to the resource and/or the link to additional information about the resource. The value domain of this metadata element is a character string, commonly expressed as uniform resource locator (URL).
1.5		Unique resource identifier	name	A value uniquely identifying the resource. The value domain of this metadata element is a mandatory character string code, generally assigned by the data owner, and a character string namespace uniquely identifying the context of the identifier code (for example, the data owner).
1.6		Coupled Resource	identification_CoupledResource	If the resource is a spatial data service, this metadata element identifies, where relevant, the target spatial data set(s) of the service through their unique

				resource identifiers (URI). The value domain of this metadata element is a mandatory character string code, generally assigned by the data owner, and a character string namespace uniquely identifying the context of the identifier code (for example, the data owner).
1.7		Resource language	identification_ResourceLanguage	The language(s) used within the resource. The value domain of this metadata element is limited to the languages defined in ISO 639-2.
2	CLASSIFICATION OF SPATIAL DATA AND SERVICES			
2.1		Topic category	classification_TopicCategory	The topic category is a high-level classification scheme to assist in the grouping and topic-based search of available spatial data resources. The value domain of this metadata element is defined in Part D.2. (Topic categories in

				accordance with EN ISO 19115) - see CELEX_32008R1205_EN_TXT.pdf for details
2.2		Spatial Data Service Type	classification_SpatialDataServiceType	This is a classification to assist in the search of available spatial data services. A specific service shall be categorised in only one category. The value domain of this metadata element is defined in Part D.3. (discovery, view, download, transformation, invoke)
3	KEYWORD			If the resource is a spatial data service, at least one keyword from Part D.4 shall be provided. If a resource is a spatial data set or spatial data set series, at least one keyword shall be provided from the general environmental multilingual thesaurus (GEMET) describing the relevant spatial data theme as defined in Annex I, II or III to Directive 2007/2/EC. For each

				keyword, the following metadata elements shall be provided:
3.1		Keyword value	keyword_KeywordValue	The keyword value is a commonly used word, formalised word or phrase used to describe the subject. While the topic category is too coarse for detailed queries, keywords help narrowing a full text search and they allow for structured keyword search. The value domain of this metadata element is free text.
3.2		Originating controlled vocabulary	keyword_OriginatingControlledVocabulary	If the keyword value originates from a controlled vocabulary (thesaurus, ontology), for example GEMET, the citation of the originating controlled vocabulary shall be provided. This citation shall include at least the title and a reference date (date of publication, date of last revision or of creation) of the originating controlled vocabulary. There can be more than one because the keywords

				can own to more than one vocabulary
4	GEOGRAPHIC LOCATION			
4.1		Geographic Bounding Box	spatial	This is the extent of the resource in the geographic space, given as a bounding box. The bounding box shall be expressed with westbound and eastbound longitudes, and southbound and northbound latitudes in decimal degrees, with a precision of at least two decimals.
5	TEMPORAL REFERENCE			This metadata element addresses the requirement to have information on the temporal dimension of the data as referred to in Article 8(2)(d) of Directive 2007/2/EC. At least one of the metadata elements referred to in points 5.1 to 5.4 shall be provided. The value domain of the metadata elements referred to in points 5.1

				to 5.4 is a set of dates. Each date shall refer to a temporal reference system and shall be expressed in a form compatible with that system. The default reference system shall be the Gregorian calendar, with dates expressed in accordance with ISO 8601.
5.1		Temporal extent	data_temporal_extent_begin_date data_temporal_extent_end_date	The temporal extent defines the time period covered by the content of the resource. This time period may be expressed as any of the following: — an individual date, - an interval of dates expressed through the starting date and end date of the interval, — a mix of individual dates and intervals of dates.
5.2		Date of publication	temporalReference_dateOfPublication	This is the date of publication of the resource when available, or the date of entry into force. There may be more than one date of publication.

5.3		Date of last revision	temporalReference_dateOfLastRevision	This is the date of last revision of the resource, if the resource has been revised. There shall not be more than one date of last revision.
5.4		Date of creation	temporalReference_dateOfCreation	This is the date of creation of the resource. There shall not be more than one date of creation.
6	QUALITY AND VALIDITY			
6.1		Lineage	quality_and_validity_lineage	This is a statement on process history and/or overall quality of the spatial data set. Where appropriate it may include a statement whether the data set has been validated or quality assured, whether it is the official version (if multiple versions exist), and whether it has legal validity. The value domain of this metadata element is free text.

6.2		Spatial Resolution	quality_and_validity_spatial_resolution_latitude quality_and_validity_spatial_resolution_longitude quality_and_validity_spatial_resolution_scale quality_and_validity_spatial_resolution_measureunit	Spatial resolution refers to the level of detail of the data set. It shall be expressed as a set of zero to many resolution distances (typically for gridded data and imagery-derived products) or equivalent scales (typically for maps or map-derived products). An equivalent scale is generally expressed as an integer value expressing the scale denominator. A resolution distance shall be expressed as a numerical value associated with a unit of length.
7	CONFORMITY			The requirements referred to in Article 5(2)(a) and Article 11(2)(d) of Directive 2007/2/EC relating to the conformity, and the degree of conformity, with implementing rules adopted under Article 7(1) of Directive 2007/2/EC shall be addressed by the following metadata elements:

7.1		Specification	conformity_specification_title conformity_specification_dataType conformity_specification_date	This is a citation of the implementing rules adopted under Article 7(1) of Directive 2007/2/EC or other specification to which a particular resource conforms. A resource may conform to more than one implementing rules adopted under Article 7(1) of Directive 2007/2/EC or other specification. This citation shall include at least the title and a reference date (date of publication, date of last revision or of creation) of the implementing rules adopted under Article 7(1) of Directive 2007/2/EC or of the specification.
7.2		Degree	conformity_degree	This is the degree of conformity of the resource to the implementing rules adopted under Article 7(1) of Directive 2007/2/EC or other specification. The value domain of this metadata element is defined in Part D.

8	CONSTRAINT RELATED TO ACCESS AND USE			
8.1		Conditions for Access and Use	constraints_conditions_for_access_and_use	This metadata element defines the conditions for access and use of spatial data sets and services, and where applicable, corresponding fees as required by Article 5(2)(b) and Article 11(2)(f) of Directive 2007/2/EC. The value domain of this metadata element is free text. The element must have values. If no conditions apply to the access and use of the resource, 'no conditions apply' shall be used. If conditions are unknown, 'conditions unknown' shall be used. This element shall also provide information on any fees necessary to access and use the resource, if applicable, or refer to a uniform resource locator (URL) where information on fees is available. Additional Requirements

				on Metadata according to regulation 1312/2014: The technical restrictions applying to the access and use of the spatial data service shall be documented in the metadata element "CONSTRAINT RELATED TO ACCESS AND USE" set out in Regulation (EC) No 1205/2008.
8.2		Limitations on Public Access	constraints_limitation_on_public_access	When Member States limit public access to spatial data sets and spatial data services under Article 13 of Directive 2007/2/EC, this metadata element shall provide information on the limitations and the reasons for them. If there are no limitations on public access, this metadata element shall indicate that fact. The value domain of this metadata element is free text.
9	RESPONSIBLE ORGANISATION			Organisations responsible for the establishment, management, maintenance and distribution of spatial data sets and services

9.1		Responsible party	responsable_organization_name responsable_organization_email	This is the description of the organisation responsible for the establishment, management, maintenance and distribution of the resource. This description shall include: — the name of the organisation as free text, — a contact e-mail address as a character string. Additional Requirements on Metadata according to regulation 1312/2014: The responsible party set out in Regulation (EC) No 1205/2008 shall at least describe the custodian responsible organisation, corresponding to the Custodian responsible party role set out in Regulation (EC) No 1205/2008.
9.2		Responsible party role	responsable_organization_role	This is the role of the responsible organisation. The value domain of this metadata element is defined in Part D.

10	METADATA ON METADATA			
10.1		Metadata Point of Contact	point_of_contact_name point_of_contact_email	This is the description of the organisation responsible for the creation and maintenance of the metadata. This description shall include: - the name of the organisation as free text, - a contact e-mail address as a character string.
10.2		Metadata Date	temporalReference_date	The date which specifies when the metadata record was created or updated. This date shall be expressed in conformity with ISO 8601.
10.3		Metadata Language	metadata_language	This is the language in which the metadata elements are expressed. The value domain of this metadata element is limited to the official languages of the Community

				expressed in conformity with ISO 639-2.
	B) Additional metadata elements according to INSPIRE Implementing Rules for interoperability of spatial data sets and services (Commission Regulation (EU) No 1089/2010; Art. 13 Metadata required for Interoperability) and relevant amendments			
		Coordinate Reference System	coordinatesystemreference_code coordinatesystemreference_codespace	Description of the coordinate reference system(s) used in the data set.
		Character encoding	character_encoding	The character encoding used in the data set.

Additional metadata have been defined in order to manage the internal operations. The most important additional metadata is the DataTypeID, which is adopted to identify the WP and Task that generates the dataset. The list of additional metadata is reported in Table 9.

Table 9: List of additional SHELTER metadata

11	**SHELTER internal metadata**			Metadata required by SHELTER

11.1		Metadata name	name	Name of the metadata
11.2		Owner Organization	owner_org	Organization which owns the data (written in lower case, default: shelter-project)
11.3		DataTypeID	datatype_id	SHELTER task in number of five digits, where the first is the SHELTER WP number, the second is the Task number and the following digits are an incremental number that specifies the data type. It can be a list if more files are uploaded
11.4		Visibility	visibility	Visibility of the dataset: "private" if only authenticated Data Lake users can access to the dataset, "public" to make it publicly available.

Following, an example of the JSON dictionary with the metadata information required to upload a new resource to the Data Lake:

```
{
    "title": "EMSR142: Flood in Croatia",
    "name": "12345567890",
    "private": true,
```

```
    "notes": "Since 15 October 2015 heavy rains have affected several areas of Croatia in the Karlovac area causing floods affecting a number
 of buildungs and roads.",
    "identification_ResourceType": "dataset",
    "datatype_id": "76050",
    "owner_org": "shelter-project",
    "identification_CoupledResource": "",
    "identification_ResourceLanguage": "eng",
    "classification_TopicCategory": "inlandWaters",
    "classification_SpatialDataServiceType": "",
    "keyword_KeywordValue": "Riverine Flood",
    "keyword_OriginatingControlledVocabulary": "shelter_ontology",
    "data_temporal_extent_begin_date": "2015-10-16T13:00:00",
    "data_temporal_extent_end_date": "2015-10-16T13:00:00",
    "temporalReference_dateOfPublication": "2015-10-21T23:31:26",
    "temporalReference_dateOfLastRevision": "2015-10-21T23:31:26",
    "temporalReference_dateOfCreation": "2015-10-21T23:31:26",
    "quality_and_validity_lineage": "Quality approved",
    "quality_and_validity_spatial_resolution_latitude": "0",
    "quality_and_validity_spatial_resolution_longitude": "0",
    "quality_and_validity_spatial_resolution_scale": "25000",
    "quality_and_validity_spatial_resolution_measureunit": "m",
    "conformity_specification_title": "COMMISSION REGULATION (EU) No 1089/2010 of 23 November 2010 implementing Directive 2007/2/EC of the Eu
ropean Parliament and of the Council as regards interoperability of spatial data sets and services",
    "conformity_specification_dateType": "publication",
    "conformity_specification_date": "2010-12-08T00:00:00",
    "conformity_degree": true,
    "constraints_conditions_for_access_and_use": "",
    "constraints_limitation_on_public_access": "",
    "responsable_organization_name": "Copernicus EMS Rapid Mapping Team",
```

```
    "responsable_organization_email": "mapping@copernicus.com",
    "responsable_organization_role": "author",
    "point_of_contact_name": "Copernicus EMS Rapid Mapping Team",
    "point_of_contact_email": "mapping@copernicus.com",
    "temporalReference_date": "2020-11-17T00:00:00",
    "metadata_language": "eng",
    "coordinatesystemreference_code": "4326",
    "coordinatesystemreference_codespace":"EPSG",
    "character_encoding":"UTF-8",
    "spatial": "{ \"type\": \"MultiPolygon\", \"coordinates\": [[[[15.427551269531248, 45.25362179991922], [16.842041015625, 45.2536217999192
2], [16.842041015625, 45.637087095718734], [15.427551269531248, 45.637087095718734], [15.427551269531248, 45.25362179991922]]]] }"
    }
```