This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 870811

SPICE

Social cohesion, Participation, and Inclusion through Cultural Engagement

D4.2 Linked Data server technology: integrating feedback from use case requirements

Deliverable information	
WP	Specify WP
Document dissemination level	PU Public
Deliverable type	R Document, report
Lead beneficiary	OU
Contributors	UNIBO, UNITO, IMMA, CELI, UH, UCM
Date	26/04/2022
Document status	Final
Document version	V1.0

INTENTIONALLY BLANK PAGE

Project information

Project start date: 1st of May 2020

Project Duration: 36 months

Project website: https://spice-h2020.eu

Project contacts

Project Coordinator

Silvio Peroni

ALMA MATER STUDIORUM - UNIVERSITÀ DI BOLOGNA

Department of Classical Philology and Italian Studies – FICLIT

E-mail: silvio.peroni@unibo.it

Scientific Coordinator

Aldo Gangemi

Institute for Cognitive Sciences and Technologies of the Italian National Research Council

E-mail: aldo.gangemi@unibo.it

Project Manager

Adriana Dascultu

ALMA MATER STUDIORUM - UNIVERSITÀ DI BOLOGNA

Executive Support Services

E-mail: adriana.dascultu@unibo.it

SPICE consortium

No.	Short name	Institution name	Country
1	UNIBO	ALMA MATER STUDIORUM - UNIVERSITÀ DI BOLOGNA	Italy
2	AALTO	AALTO KORKEAKOULUSAATIO SR	Finland
3	DMH	DESIGNMUSEON SAATIO - STIFTELSEN FOR DESIGNMUSEET SR	Finland
4	AAU	AALBORG UNIVERSITET	Denmark
5	OU	THE OPEN UNIVERSITY	United Kingdom
6	IMMA	IRISH MUSEUM OF MODERN ART COMPANY	Ireland
7	GVAM	GVAM GUIAS INTERACTIVAS SL	Spain
8	PG	PADAONE GAMES SL	Spain
9	UCM	UNIVERSIDAD COMPLUTENSE DE MADRID	Spain
10	UNITO	UNIVERSITA DEGLI STUDI DI TORINO	Italy
11	FTM	FONDAZIONE TORINO MUSEI	Italy
12	CELI	MAIZE SRL	Italy
13	UH	UNIVERSITY OF HAIFA	Israel
14	CNR	CONSIGLIO NAZIONALE DELLE RICERCHE	Italy

Executive summary

SPICE is an EU H-2020 project dedicated to research on novel methods for citizen curation of cultural heritage through an ecosystem of tools co-designed by an interdisciplinary team of researchers, technologists, and museum curators and engagement experts, and user communities. This technical report D4.2 presents the interim deliverables of Work Package 4 of the SPICE project, focusing on validating the requirements and technical implementations introduced in D4.1 - the Linked Data Hub - against the current status of open data and content management strategies of the museums. Alongside presenting novel functionalities of the Linked Data Hub, we illustrate current technical developments of the SPICE pilots in relation to the provisions of WP4 and collect feedback received from Pilots' implementers about the Linked Data server technologies. Furthermore, we discuss future work in fulfilling the needs of systems implementing the citizen curation methodology, based on the input of the SPICE pilots. This is an interim report that will be followed by D4.3 in the third year of the project.

Document History

Version	Release date	Summary of changes	Author(s) -Institution
V0.1	19/01/2022	First draft released	Enrico Daga (OU)
V0.2	04/04/2022	Revisions and integration. Final contributions from pilot study authors.	All partners
V0.3	06/04/2022	Task status table. Final conclusions added and integrated feedback comments. Image captions and labelling.	Jason Carvalho (OU)
V0.4	11/04/2022	Internal review	Guillermo Jimenez Diaz (UCM) Antonio Lieto (UNITO)
V0.5	26/04/2022	Incorporated comments from internal review	Jason Carvalho (OU), Enrico Daga (OU)
V1.0	26/04/2022	Final version submitted to REA	UNIBO

Table of Contents

1 Introduction

1.1 Description of the work package and objectives

SPICE is an EU H-2020 project dedicated to citizen curation of cultural heritage. To support citizen curation, the project research upon and develops an ecosystem of methods and tools co-designed by an interdisciplinary team of researchers, technologists, domain experts, and user communities. Work Package 4 develops a Distributed Linked Data social media layer, including the technical architecture, the specification of the communication protocols, and the back-end support to the pilot studies developed within the project. Objective of the WP is to research on the application of Linked Data principles to connect cultural objects, collections, and citizen contributions, into an infrastructure for interoperability and knowledge exchange within citizen curation activities. While the WP aims at providing the infrastructure for interoperability within the project, by doing that, its goal is researching on a social media platform that can support museums and technologists with:

(1) privacy-aware content sharing methods, so that museums can expose their catalogue and digital assets in a safe and controlled data environment;
(2) methods for expressing and reasoning over fine-grained policies and constraints associated to digital assets;
(3) linking assets and metadata to support search and discovery capabilities (on top of a secure and controlled data environment); and
(4) content provenance, usage tracing, and monitoring in order to support large scale analyses of user-generated, (anonymised) content.

1.2 Description of the work done so far in relation to the objectives

Based on these objectives, this document describes the second year of activity on such a Linked Data Ecosystem, mainly referring to Tasks 4.1 “Linked Data server technology” (M1--M36) and T4.2 “Distributed privacy and policy layer” (M1--M24), focussed on the development of a Linked Data Hub for ingesting, connecting, and distributing data through a SPICE network of systems. However, the work done also relates to Task 4.3 “Linking and discovering digital assets” (M13--M24) and Task 4.4 “Provenance and process analysis”. As such, this deliverable is mainly reporting on Task 4.1, but also include initial work towards the fulfilment of the other objectives, in particular in relation to data acquisition, linking, and analysis (Tasks 4.2, 4.3 and 4.4).

The main requirements of the Linked Data Infrastructure, its core functionalities, and relation to the state of the art, are discussed in D4.1. In D4.1, we provided insights on the Citizen Curation paradigm and performed an analysis of the state of art technologies for Linked Data content management, and provided insights on the status of data management in SPICE museums. In addition, in D4.1 we devised requirements for a data infrastructure based on the performed analyses, illustrating the design of a Linked Data Layer, a collection of components and protocols for data communication and exchange across the SPICE network. These include the SPICE Linked Data Hub, a data management and publishing infrastructure, and the SPARQL Anything tool, to support knowledge engineers in transforming heterogenous resources into Linked Data.

In this deliverable, we focus on incremental work undertaken in the second year of the SPICE project, and report on how such infrastructure is being applied in SPICE pilots, collecting feedback from developers, and plan for future developments. Specifically, we focus on describing new functionalities of the Linked Data Hub and validate usability in relation to museum’s open data strategies and content management systems. In addition, we illustrate current developments of the SPICE Pilots and discuss them in relation to the requirements identified in D4.1 and the current facilities provided by the Linked Data Hub, collecting feedback and lessons learnt. Crucially, we discuss them in relation to further developments to be conducted in the last year of the SPICE project.

The deliverable is structured as follows. The next Section is dedicated to incremental work performed on the **SPICE Linked Data Hub**, illustrating new functionalities in relation to access control, file management, and activity monitoring. Section 3 reports on **strategies for Integrating Museums' Data**, validating the current Linked Data Infrastructure against possible strategies for open data and content management of museums in the sector (taking the SPICE museums as exemplary cases). Section 4 provides an overview of **SPICE pilots: applications and feedback** in relation to the Linked Data Hub infrastructure. We discuss the current state of developments and we plan for **integrating the feedback from SPICE pilots** in Section 5, before providing a summary of main contributions of this deliverable in the **conclusions** section.

2 The SPICE Linked Data Hub

2.1 Overview

This SPICE Linked Data Hub (LDH) was developed as a data infrastructure to support the acquisition and management of dynamic data from a variety of sources including: museum collection metadata and digital assets, social media events and user activities, systems' activities (e.g., recommendations, reasoning outputs), ontologies and linked data produced by pilot case studies. A layout of the system is provided in Figure 2.1.1.

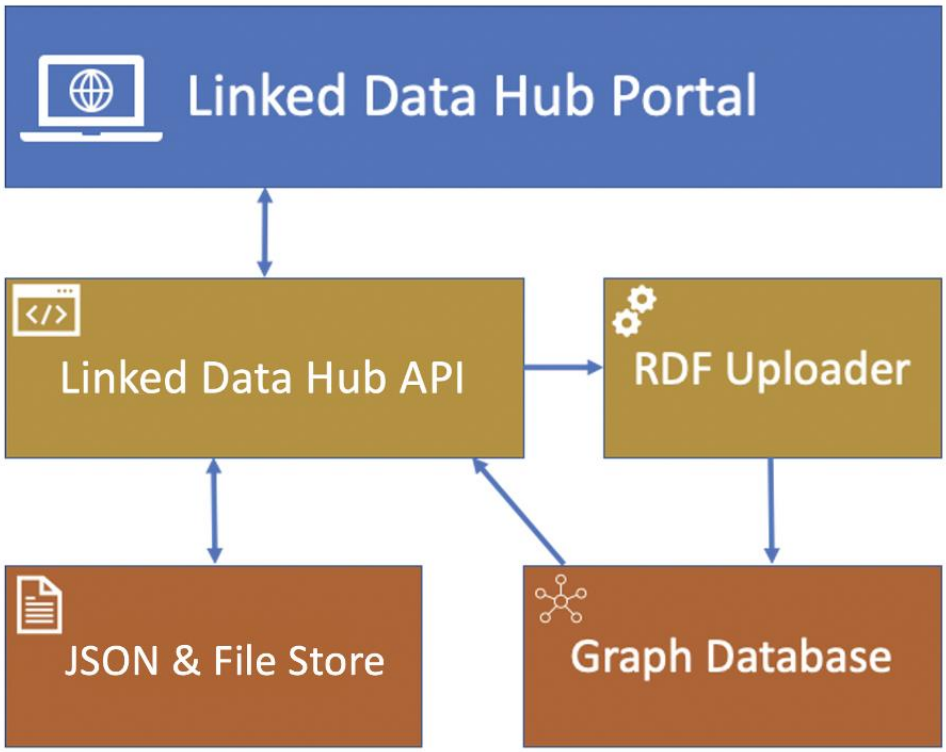

Figure 2.1.1. SPICE Linked Data Hub layout

The SPICE Linked Data Hub is made up of a number of components, the most visible being the front-end web portal. The portal provides a facility for data providers to create and catalogue datasets as well as manage their privacy, licences and provenance. The web portal enables users to browse catalogues and registries of datasets and their corresponding metadata and data models. These datasets are largely made up of social media activities within the SPICE network. Sitting behind the web portal and driving most of its functionality is the LDH API that exposes a range of REST-based API functionality to support the production, management and consumption of data. This API is directly available to all, enabling developers to side-step the web portal and integrate SPICE LDH read/write operations with existing automated systems. The API also offers a range of extended functionality over and above that offered by the web portal such as:

- Full read/write (CRUD) access to datasets for users with appropriate permissions
- Enhanced browsing capabilities
- Advanced querying and data filters (using both JSON-style queries and SPARQL)
- A read-only SPARQL endpoint

Datasets take the form predominantly of JSON documents or static files, so anything that can be encoded as a JSON string can be stored and accessed using the SPICE LDH's full range of features. All JSON documents pushed into the SPICE LDH via the API are also replicated as RDF to a graph database for read-only query access via SPARQL.

2.2 New functionalities from D4.1

The first iteration of the SPICE LDH put in place the foundations of the data infrastructure that would support the initial linked data requirements for the SPICE project. Now, as pilot applications continue to be developed, the requirements of the LDH also continue to grow. Using a feature-driven approach to development, new functionality has been launched to support the ongoing needs of pilots.

New features added, since D4.1, include

- Access control
- File handling
- Activity monitoring
- RDF replication and linked data transformations

2.2.1 Access control pushed to the next level

The initial release of the SPICE LDH was based around the concept of the dataset owner having read/write access to their dataset with their chosen access key. Where access controls permitted, other users were able to 'subscribe' to this dataset with a single read-only key. This initial functionality proved to be somewhat limiting and a number of potential use cases arose that did not fit this model, including

- *Users or applications require write-only keys.* Dataset owners may wish to grant write access to users or applications that need to contribute data to the dataset without having read access to other data potentially submitted by other users.
- *Users may require multiple keys with the same access to a single dataset*. A single dataset may collect data from a number of different applications. Even in the case where these applications are managed by a single LDH user, it would be preferable for these applications to use distinct access keys both for audit trail purposes and also to deal with scenarios where access may need to be revoked for a single application.
- *Revoking access to specific keys*. Where access policy has changed or an application may have become compromised, there are scenarios where a key owner may wish to remove their own access to a dataset or a dataset owner may want to disable specific user or key access.

In addition to these initial shortcomings, a more sophisticated access control layer was required as a foundation for further proposed developments to the Policy Management Layer of the LDH:

- *Single user can register multiple keys on a dataset.*
- *Distinguish read, write and read/write keys*. The LDH has been extended to not only support read, write and read/write keys, but also to offer write access to non-dataset owners (where permissions allow). Previously, only dataset owners were able to create keys with write permission.
- *Dataset owners can manage developers' keys on their dataset.* Key access to datasets can be removed by key owners but can now also be disabled via dataset owners. The LDH makes a subtle distinction between keys access that has been removed and key access that has been disabled. In the case of a key owner removing their access to a dataset, that key is completely disassociated with

the dataset and, if access control still allows, can subsequently be reassociated with the same dataset to restore access if the need arises. In the case of a dataset owner disabling access for a particular key on their dataset, the key-dataset association is maintained and set to a disabled state. Further to this, the key owner can no longer remove this association and reassign the key to the same dataset due to it being locked in a disabled state. Dataset owners can manage whether the user can assign further new dataset keys to their dataset by setting the access controls accordingly. A valid scenario was highlighted where a dataset owner may wish to disable a specific user key but be happy for that user to register further keys with the same dataset, for example where that specific key was built into an application where the security of the key had become compromised.

Alongside these new access control features, both dataset owners and users now have finer grained visibility of who has what kind of access to which datasets. Dataset owners and custodians can view, at a glance a summary of which users have keys assigned to their datasets and users can similarly get an overview of their access to datasets. The work on these components addresses WP4.2 requirements 1, 23 and 36 (*Table 3.1*).

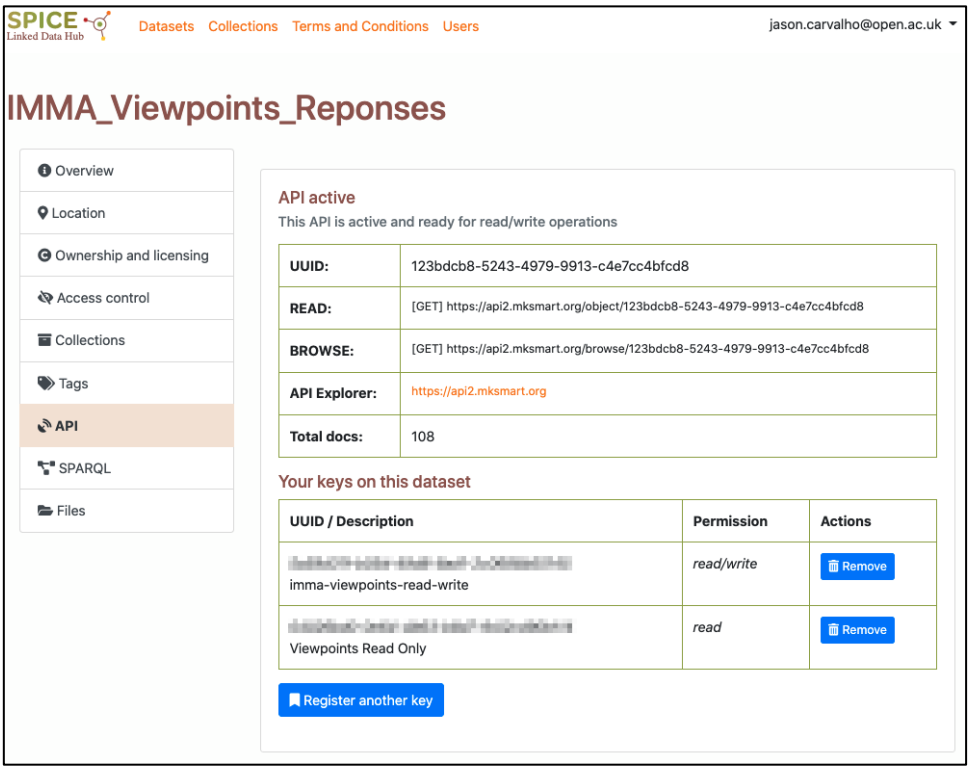

Figure 2.2.1.1. Registering multiple keys on a single dataset

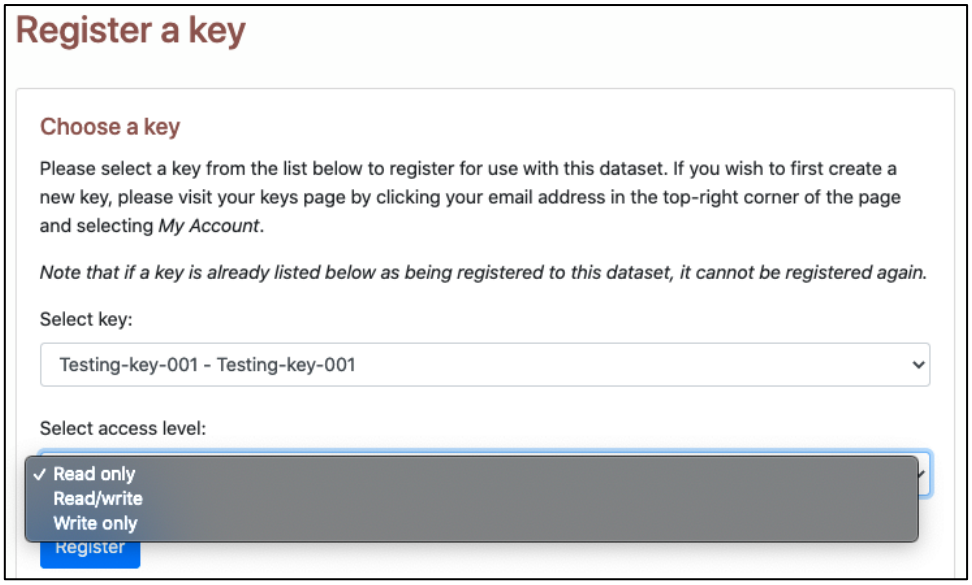

Figure 2.2.1.2. Selectable access levels for registering keys

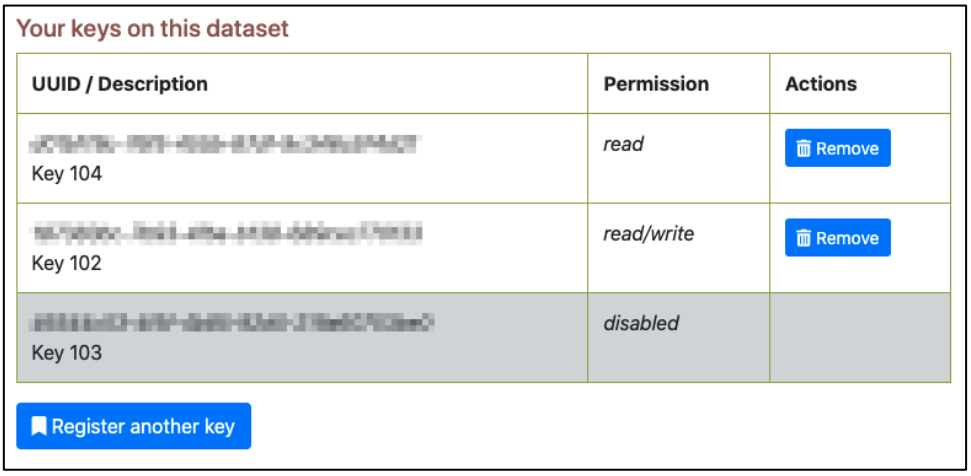

Figure 2.2.1.3. Managing multiple keys on a single dataset

2.2.2 File repository

In D4.1 the LDH web portal provided a separate facility for uploading and retrieving files to dedicated file-specific datasets which were stored and handled separately from the main JSON API-based datasets. The backend LDH API had no knowledge of these file-based datasets and access was limited to dataset owners only via the web interface, making integration into SPICE pilot applications problematic.

In the latest release of the LDH API, file handling functions have been incorporated into the API itself, enabling dataset users to perform typical create/read/update/delete (CRUD) operations on both JSON documents and binaries files interchangeably and within a single dataset, using a single key and set of access controls. SPICE pilot applications such as IMMA Viewpoints now make use of both JSON data and image resources directly through the LDH API. Alongside API access, the LDH portal also provides file listing and manipulation tools via the web interface as a proxy to the new API file functions.

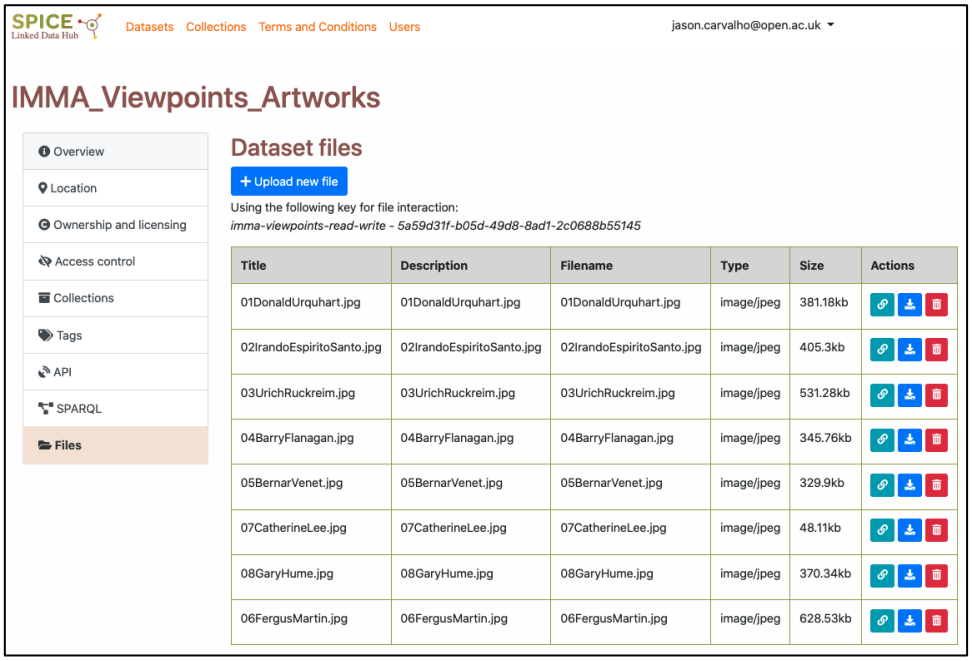

Figure 2.2.2.1. Dataset file listing

Since access to LDH file resources is done through the API, it should be noted that these calls require authentication with an appropriate API key. The facility is designed to enable application programmers to make authenticated API calls for resources from within their applications, mixing JSON and binary file data interchangeably. The same access controls and API key registration policies that apply to all JSON data within a dataset are also now applied in the same way for requests to retrieve or manipulate file-based content. That is to say, a single dataset access policy is applied to all dataset operations, regardless of data type. The work on this functionality addresses point 33 in the WP4.2 list of requirements (*Table 3.1*).

2.2.3 Activity monitoring

As part of the preparations for the Process Analysis Layer and also WP4.2 requirements 1, 5, 29 and 30 (*Table 3.1*), the LDH API now implements a detailed and comprehensive activity log providing a full audit trail of all API operations. Log entries take the form of JSON-LD documents and are managed internally by the API using the same JSON document store that is used for all other SPICE LDH data.

Since the activity log itself resides within a regular LDH dataset, the full suite of searching, filtering, sorting and paging operations made available by the API can be used on log entries. This functionality enables us to further develop process analysis applications and layers that are able to reconstruct timelines of activity pertaining to specific segments of interest within the LDH. This feature also opens up the possibility of plugging in replication agents alongside the LDH that can rebuild either all or parts of LDH datasets on external platforms that may be better suited to application-specific querying, workflows or reasoning layers.

Below is a typical application log entry:


```
{
      "_id": "621dd7bd0763c253c53a1de9",
      "@id": "621dd7bd0763c253c53a1de9",
      "@context": "https://mkdf.github.io/context",
      "@type": [
        "al:Create",
        "al:ActivityLogEntry"
```

```
      ],
      "al:summary": "Create[POST] - http://api2.mksmart.org/object/<datasetID> -
Dataset: <datasetID> - Key:***removed*** - Create new document",
      "al:request": {
        "@type": "al:HTTPRequest",
        "al:agent": {
          "@type": "al:Agent",
          "al:key": "***removed***"
        },
        "al:endpoint": "http://api2.mksmart.org/object/<datasetID>",
        "al:httpRequestMethod": "al:POST",
        "al:parameters": [],
        "al:payload": "{\"_id\":\"1099\",\"category\":\"52-a\",\"value\":44.7}"
      },
      "al:documentId": "1099",
      "al:datasetId": " <datasetID> ",
      "_timestamp": 1646122941,
      "_timestamp_year": 2022,
      "_timestamp_month": 3,
      "_timestamp_day": 1,
      "_timestamp_hour": 8,
      "_timestamp_minute": 22,
      "_timestamp_second": 21
}
```


Activity log entries and operations are available to LDH administrators and LDH plugin components such as the RDF replication agent. The next phase of development with respect to the LDH activity log is to develop an API endpoint that lets users access a subset of details from these log entries. Combined with filtering and sorting functions, as well as appropriate access controls, this will enable third parties to build and run their own replication agents for individual datasets.

2.2.4 Jobs: Automating and Customising Linked Data Transformations

Using the activity logging features mentioned in 2.2.3, we run an *RDF Uploader* component alongside the LDH that is responsible for the RDF replication of LDH data that drives the SPARQL querying API endpoint. This work addresses point 5 in the list of WP4.2 requirements (*Table 3.1*).

Specifically, this component continuously monitors the Activity log and reacts to the logging events creating, deleting, or updating JSON documents and static files. Whenever an event of this kind is detected, the RDF Uploader retrieves from the Activity log's entry the dataset and the document targeted by the logged activity and updates the triple store consequently.

A Blazegraph[1] server is used for storing the RDF triples that are generated by the RDF Uploader. Blazegraph can host multiple triple stores, each one is defined as a *namespace* and corresponds to an LDH dataset. Each Blazegraph namesapce contains a number of RDF graphs, with each graph corresponding to a document or file within the LDH.

The behaviour of the RDF uploader upon the detection of the logged event is described in the following Table.

Activity Log Entry	RDF Uploader Reaction
Create Dataset	The RDF Uploader creates a new Blazegraph namespace which will contain the RDF transformation of the content dataset.

[1] https://blazegraph.com/

Create Document / File	The RDF Uploader transforms into RDF (according to the strategies developed for SPARQL Anything) and uploads the result into a graph. The graph is stored in the namespace created for the dataset containing the document.
Delete Document / File	The RDF Uploader removes the graph storing the RDF replication of the Document/File.
Update Document / File	The RDF Uploader transforms into RDF the new version of the document/file, clears the graph already created for the file and uploads the RDF replication of the file content.

As a result, LDH components are univocally associated with the RDF entities. In particular, each LDH dataset is associated with a Blazegraph namespace which is used to store the RDF replication of the content of the dataset. Documents and Files of a dataset correspond to RDF graphs stored into the namespace created for the dataset. The identifiers assigned to namespaces and graphs allows us to trace back to the datasets, documents and files from which they were generated.

Further development of this component now offers users the ability to construct their own custom RDF graphs, materialising views on their data using SPARQL CONSTRUCT statements. These constructed graphs are stored within the same namespace as their parent dataset and are then available to be queried via the API's SPARQL endpoint.

Requests for the construction of new graphs are submitted via a new SPARQL Jobs component of the LDH web portal. The requests are stored as JSON documents into a special dataset of the LDH, called rdf_jobs. Such JSON documents have the following structure:

```
{
        "_id": "JobId",
        "dataset": "datasetid",
        "job-type": "CONSTRUCT",
        "query": "CONSTRUCT {…} WHERE {…}",
        "target-namespace": "spice_datasetid",
        "target-named-graph": "some-graph-name",
        "status": "PENDING",
        "message": "some message 1",
        "history": [
                {
                        "message": "some message 2",
                        "timestamp": 893478298213
                },
                {
                        "message": "some message 3",
                        "timestamp": 89347498211
                },
                {
                        "message": "some message 4",
                        "timestamp": 893478299287
                }
        ],
        "scheduled": 893478298213,
        "submitted-by": "jason.carvalho@open.ac.uk",
        "modified-by": "jason.carvalho@open.ac.uk",
        "_timestamp": 893478298213
}
```

Once the RDF Uploader has retrieved the job, it evaluates the construct query over the target-namespace and stores the result into the target-named-graph. Finally, it updates the status of the job according to the result of activity and possibly adds some messages to the history.

It is worth noticing that CONSTRUCT jobs enabled us to remodel the RDF replica according to the SPICE Ontology Network (SON) developed in the context of the task T6.3. The SPICE Ontology Network is described in detail in deliverable D6.5. In fact, the RDF transformation operated by SPARQL Anything acts on syntactic level only, without considering the semantics of data. By means of the CONSTRUCT queries the dataset owner can reshape data in accordance with SON. No strict ontological commitment has been enforced on data during the initial stage of data gathering in the project. However, the CONSTRUCT jobs enable us to *project* data into the ontological model thus accessing and validating them through the lenses of the SON. This solution ensures that data can be exchanged in a very flexible manner without losing its semantic characteristics that can be used when needed.

The interface component also gives users the option rebuild specific graphs or to rebuild the entire RDF replica of the original dataset. Similar to the CONSTRUCT jobs, REBUILD jobs are then batch processed by the RDF Uploader component.

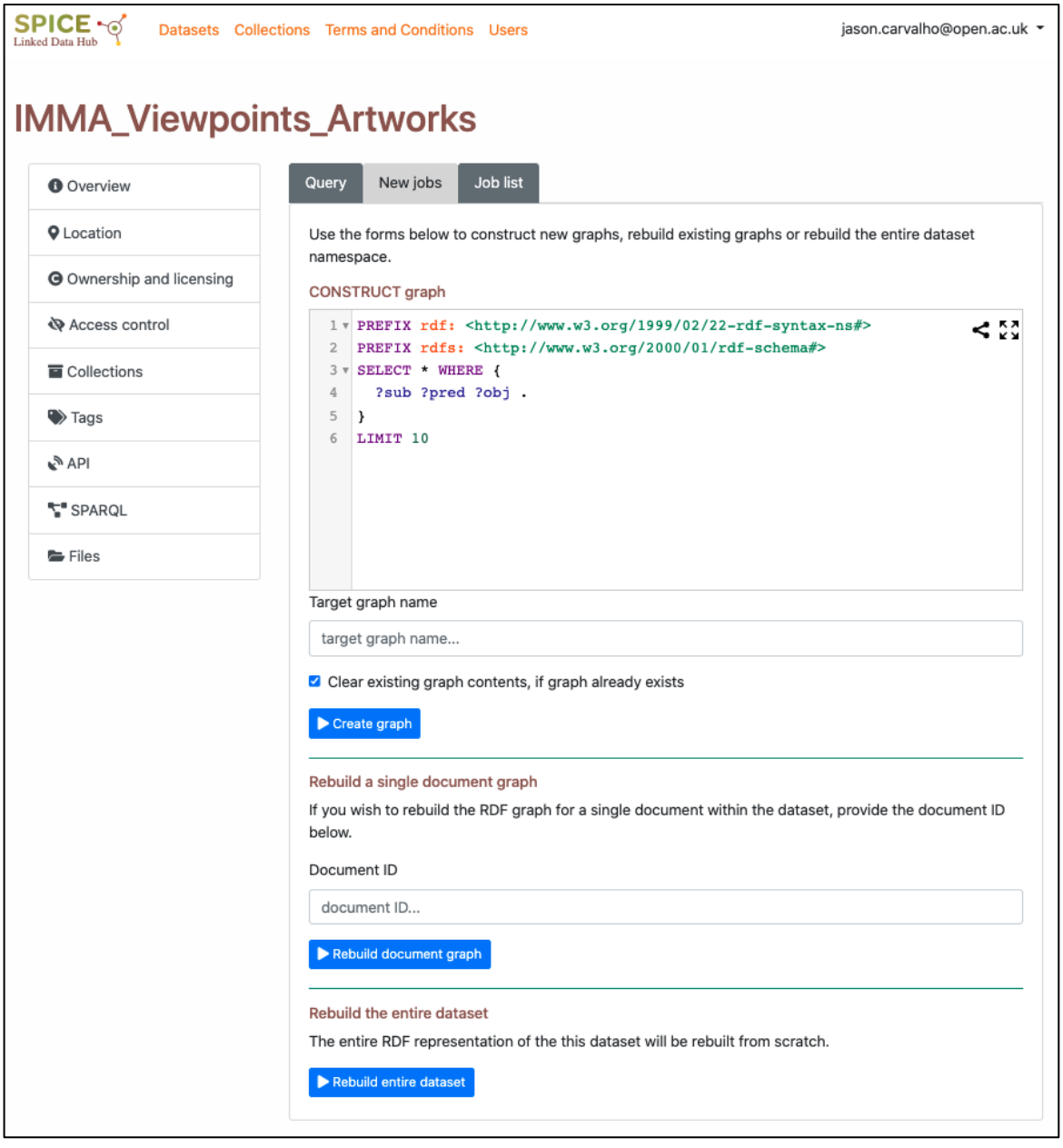

Figure 2.2.4.1. SPARQL and RDF jobs interface

Since the jobs are not processed in real-time, SPARQL queries on newly constructed or rebuilt graphs are not possible until the appropriate job has been batch processed. Typical batch processing frequency is very short, usually less than a minute. In order to provide clarity on this, users are able to inspect and monitor the status of SPARQL jobs via the Job list panel.

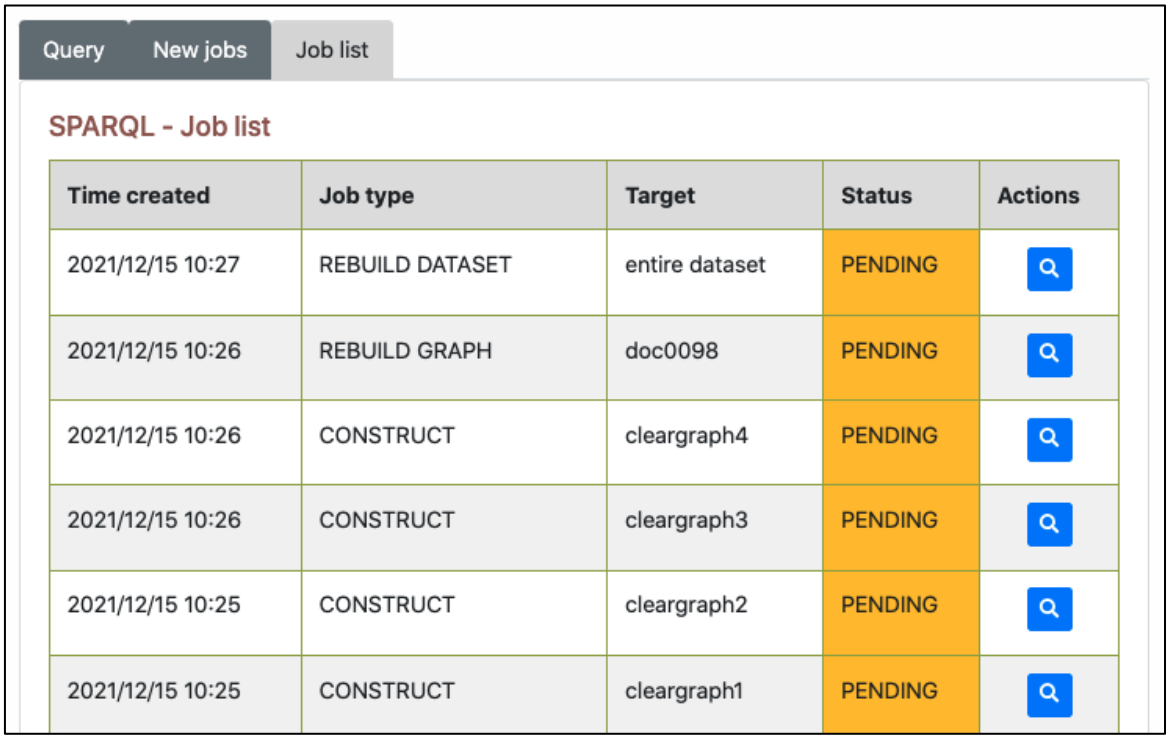

Figure 2.2.4.2. SPARQL and RDF jobs status

This work to materialise views on data as alternatively structured RDF and also to issue rebuild commands to parts or all of datasets satisfies WP4.2 requirement 30.

3 Status update in relation to the requirements

Table 3.1. WP4.2 requirements

	Nickname	Role	Action	Target	D4.1 Status	D4.2 Status
1	[AnalyseUsage]	custodian/owner	analyse	access and usage of my data	50%	80%
2	[BackupContent]	data manager	backup/restore	my data to support recovery in the case of a loss event.	30%	
3	[BrowseIndex]	builder	browse	an index of the resources I have access to	30%	80%
4	[BrowseMarketplace]	custodian/owner/builder	browse	a marketplace of offers of digital assets	0%	
5	[ControlMetadata]	owner/custodian	control	the metadata production in the ingestion process	0%	100%
6	[DetectPII]	custodian/builder	detect	personally identifiable information (PII) included in user-generated content	0%	

7	[ExpressCopyright]	custodian/owner	express	the copyright associated with digital assets in my collection	30%	
8	[ExpressExemptions]	custodian/owner	express	exemptions and characterize them	0%	
9	[ExpressFees]	owner	express	fees as duties associated to the permissions granted	0%	
10	[ExpressOffers]	owner	express	offers with relation to the assets I own.	50%	
11	[ExpressPermissions]	owner	express	permissions, prohibitions, constraints and duties	30%	
12	[ExpressPolicies]	custodian/owner	express	usage policies in relation to my data	0%	
13	[ExpressQualityFeatures]	custodian/builder	express	the quality of the asset and their features	0%	
14	[ExpressTimeConstraint]	owner/custodian	express	time limitations to permissions I grant	0%	
15	[ExternalAccessData]	builder	access	data from an external application	100%	
16	[FilterSensitiveContent]	custodian/builder	filter	sensitive content for specific target groups	0%	
17	[GrantCheck]	builder/custodian/owner	verify	lawful access to a collection metadata or digital asset	30%	
18	[GrantRecovery]	owner/custodian/builder	view	terms of use granted	0%	
19	[InappropriateContent]	custodian/builder	identify/filter	user-generated content that can be inappropriate	0%	
20	[InspectIngestionProcess]	owner/custodian	inspect	the metadata production in the ingestion process	30%	100%
21	[ManageAccess]	data manager/custodian/owner	manage	access control to the data	100%	
22	[ManageVisibility]	data manager	manage	visibility of my registered data sources	100%	
23	[MonitorAccess]	data manager/custodian/owner	monitor	access to my data	30%	80%
24	[MultipleRightsAspects]	custodian	express	that multiple subjects hold copyrights on different aspects of the digital asset	0%	
25	[NegotiateRights]	custodian	negotiate	rights on behalf of the owner	0%	
26	[NominateDelegate]	custodian	nominate	an external entity to negotiate rights on behalf of a copyright owner	0%	30%
27	[ObtainCredentials]	builder	obtain	credential details (e.g., API Keys) to data	100%	
28	[OrganiseCollections]	custodian/builder	organise	resources I have access to into customized collections	10%	
29	[ProduceLD]	data manager	produce	linked data from existing non-LD resources	100%	
30	[PublishLD]	data manager	publish	linked data with alternative Linked Data vocabularies (Viewpoints)	0%	100%
31	[ReadData]	builder	read	data from a dataset –e.g., via a (secured) Web API	100%	
32	[RecognisedAuthor]	owner	be_recognised	as author of the picture of the artwork	0%	

33	[RegisterSources]	data manager	register	existing Linked Data sources	0%	50%
34	[RequestAccess]	builder	request	access to data	0%	
35	[RequestPermission]	builder	request	permission to use a digital asset under specific terms	0%	
36	[RevokeRights]	owner/custodian	revoke	usage permissions I granted in the past	30%	100%
37	[SecureStack]	data manager	secure	the content against malicious attacks	100%	
38	[SetupRepository]	data manager	setup	a data repository	100%	
39	[ShareCollections]	custodian/builder	share	my customized collections as linked data	0%	
40	[UploadDataset]	data manager/owner/custodian	upload	data to my dataset	100%	
41	[UsagePolicyGrant]	owner/custodian	grant	permission to use a digital asset under requested terms	0%	
42	[WriteData]	data manager/builder	write	data to a dataset –e.g., via a (secured) Web API	100%	

Note that requirements 12, 14, 18, 24 and 25 relate to policy management and are covered in more detail in deliverable D4.5.

4 Strategies for Integrating Museums' Data

In this section we report on an analysis of possible strategies for integrating data from museum collections with the SPICE infrastructure. These findings derive partly from our experience with open data published by museums in the SPICE consortium, and partly from selecting content management systems that are adopted by the sector (including SPICE museums).

We look at this issue from two, complementary perspectives. Initially, we focus on museums' content management systems and discuss how customised integrations could be developed. Crucially, we discuss how the heterogenous landscape of data sharing methods does not allow for a one-size-fits-all, off-the-shelf solution. Therefore, we discuss how linked data transformers can be developed to ingest (open and non-open) data published on the Web.

4.1 Integrating (open) data from existing content management systems

In a recent survey developed by the SPICE consortium and published on the ACM Journal of Computing and Cultural Heritage (JOCCH), we discussed how the requirements for citizen curation go beyond the capabilities of typical content management systems [Daga et al, 2021]. In the article, we reviewed technologies used in the sector, including typical content management solutions. Reporting in details the technical features of each one of them is not necessarily useful to our analysis. Instead, we consider here one content management system, Omeka-S, which has features that are exemplary of most of the other solutions, in detail. We refer to the mentioned article for an extensive analysis.

Released under GNU Licence, Omeka S[2] is a web publishing platform designed for digital collections, extensible and strongly oriented to universities and research institutions. Through Omeka S, it is possible to collect, publish and share data with Linked Open Data. The core of Omeka S is a relational database, but records can be ingested and accessed by third parties via a REST API. The format of the records in Omeka S is provided by a set of templates, which define the record metadata scheme based on RDF vocabularies. A set of RDF vocabularies are pre-loaded in the standard installation, namely, the Bibliographic Ontology

[2] https://omeka.org/s/

vocabulary (BIBO) for the bibliographic sources, the Dublin Core Metadata Element Set (DC-MES) for the provenance and description of data, and the Friend Of A Friend vocabulary (FOAF) for the social identities, but other dictionaries can be added to meet the needs of specific domains. Thanks to its modular architecture, new components can be plugged onto the basic installation, including additional functions, such as a IIIF server and client for publishing and visualizing images, videos and 3D objects, and a set of adapters to connect the installation to external repositories. This situation is open to two different types of solutions for the integration with SPICE Linked Data Hub. A loose integration can be obtained via Omeka S REST API, which can be exploited to ingest data to and from Omeka S, provided that SPICE vocabularies are added to the Omeka S basic set; this solution can scale up to a tighter integration by programming some type of unidirectional or bidirectional synchronization between the two endpoints. As a more stable alternative, then, Omeka S PHP API can be exploited to program a dedicated SPICE module for configuring and managing the synchronization, similarly to existing Omeka S modules for other repository systems such as Fedora or DSpace[3]. Omeka S can be considered an exemplar CMS for museum collections metadata. The main take-away message is that CMSs may include some RDF/Semantic management approach, and that they share their data through rich Web APIs.

However, data is shared in a variety of methods, besides exposing databases of content management platforms via Web APIs. In recent years, thanks to the increasing availability of dedicated software suites for digitalization and analysis of cultural items, the number of digital objects related to cultural heritage has increased dramatically, often as a by-product of the conduction of large projects in which digital objects are accumulated for a later stage in which they will be selected and published in some structured form. For this reason, resorting to cloud repositories, more or less research oriented (such as Github, Zenodo, or Dropbox), to store large quantities of cultural data – from digital equivalents such as images and 3D models to processing outputs such as transcripts and spectrographs – has become more and more common. While most cloud-based solutions provide APIs for accessing and managing data (consider for example, Google Docs API), an obstacle to integration is that cloud-based archives don't share the conceptual model underlying standard collection managed systems, which revolves on the notion of collection; however, they all acknowledge some folder-based hierarchical organization of data files, thus proving a partial solution for mapping resources onto SPICE LDH, despite the relevant differences in conceptual organization.

A more complex issues with cloud-based repositories is that they do not pose constraints on metadata for data description and encoding, making it impossible to devise solutions that generalise over single repositories. This problem is alleviated if the publishing of data is carried out via FAIR-oriented platforms, such as Dataverse, which is acquiring growing popularity in research institutions. Dataverse is an open-source repository software for publishing research data[4]. Launched in 2017 by Harvard University, its flexibility has paved the way to its adoption by research entities worldwide, at local and national level. Its system of roles and nested repositories allow enforcing well defined data curation, preservation and publishing workflows, in support of the needs of project- and community-specific requirements on metadata and data encoding. The integration with specific Dataverse repositories, in principle, could be addressed in a similar fashion to the integration with Omeka discussed above, by leveraging Dataverse API in conjunction with the integration of SPICE vocabularies in for representing data and metadata in Dataverse.

4.2 Integrating (open) data published on the Web with SPARQL Anything

From our analysis, it is obvious that pursuing the objective of a unifying method to connect such variety of museum open data is an unrealistic goal. Instead, the Linked Data Hub incorporates technologies that allows to configure adapters to a variety of sources in a flexible way.

We consider open data published on the Web, which include: (a) downloadable files, (b) data incorporated in HTML websites, and (c) Web APIs. We already described in D4.1 how Linked Data Transformers can be developed using the novel system SPARQL Anything, developed in SPICE, to support extracting information

[3] https://omeka.org/s/modules/
[4] ttps://dataverse.org/

from those sources, specifically, considering data in CSV, JSON, and HTML formats. Here, we give a brief summary of the features of SPARQL Anything, expanding on the ones that have been recently developed.

The main features of SPARQL Anything are the following:

- Query files in plain SPARQL 1.1, via the SERVICE <x-sparql-anything:> and build knowledge graphs with CONSTRUCT queries (see examples from D4.1).
- New supported formats: Excel, Text, Binary, EXIF, File System, Zip/Tar, Markdown, YAML, Bibtex, DOCx.
- Transforms files (local or published on the Web), inline content, or the output of a program executed in the host machine
- A full-fledged HTTP client allows to query Web APIs of content management systems, tailoring the specificity of the API design, including customized HTTP headers, authentication, and all HTTP/REST methods supported.
- An extended set of functions have been added to perform operations on RDF sequences, strings, hashes, easy entity building
- A novel, iterator-based execution style allows for transforming very large files without significant memory requirements (supported for CSV, JSON, and XML formats).

Table 4.2.1 reports on the HTTP configuration parameters within SPARQL Anything for querying third-party Web APIs.

Table 4.2.1. HTTP options in SPARQL Anything

Option name	Description	Valid Values	Default Value
http.client.*	Calls methods on the HTTPClient Java object. E.g. http.client.useSystemProperties=falsemeans to avoid inheriting Java system properties (Default 'yes')		
http.client.useSystem Properties	Use Java System Properties to configure the HTTP Client.	true/false	true
http.header.*	To add headers to the HTTP request. E.g. http.header.accept=application/json		
http.query.*	To add parameters to the query string. E.g. http.query.var=value or http.query.var.1=value to add more variable of the same name		
http.form.*	To add parameters to the POST content. E.g. http.form.var=value or http.form.var.1=value to add more variable of the same name		
http.method	HTTP Method	GET,POST,...	GET
http.payload	Sets the payload of the request		
http.protocol	Protocol	0.9,1.0,1.1	1.1
http.auth.user	Authentication: user name		
http.auth.password	Authentication: password		
http.redirect	Follow redirect?	true/false	true

We already demonstrated in D4.1 how we applied LD transformers for capturing catalogue metadata from IMMA, GAM, and other open data (such as the Tate gallery data). Here, we only show how the new functionalities allow for connecting to Web APIs, including providing security credentials. We show one example of SPARQL Anything query that leverages the external *DigitaltMuseum* API[5], a well-known museum open data aggregator for cultural heritage resources of Norway and Sweden[6]:

```
PREFIX owl:  <http://www.w3.org/2002/07/owl#>
PREFIX rdf:  <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX fx:   <http://sparql.xyz/facade-x/ns/>
PREFIX xyz:  <http://sparql.xyz/facade-x/data/>

SELECT ?title ?owner
WHERE {
SERVICE <x-sparql-anything:> {
 fx:properties fx:location "http://api.dimu.org/api/solr/select" ;
 fx:http.query.q "artifact.event.place:(oslo)" ;
 fx:http.query.wt "json" ;
 fx:http.query.api.key "demo" ;
 fx:media-type "application/json"
 .
 [] xyz:artifact%2Eingress%2Etitle ?title ;
   xyz:identifier%2Eowner ?owner .
     }
}
```

In the above query, the system interrogates the Web API at http://api.dimu.org/api/solr/select with an HTTP GET and the query parameters “q”, “wt”, and “api_key”, then interprets the result as application/json re-engineering the output as Façade-X/RDF. Such RDF is then queried using the basic graph pattern in the clause, selecting title and owner of the resulting cultural heritage artifacts.

As next step, we will explore how to use this feature to integrate data from Web APIs of the CMSs of SPICE museums.

5 SPICE pilots: applications and feedback

This section of the report details the SPICE pilot applications that have been developed so far that are making use of the linked data server technology that is described in this deliverable. For each pilot application, we outline its its background and purpose, its technical requirements and the workflow of the application with respect to the linked data server technology. Finally, for each application, feedback is given on how the integration with the LDH has gone so far and considerations for moving forward with further development.

5.1 IMMA – Deep Viewpoints

5.1.1 Background

The IMMA case study is concerned with developing tools to support Citizen Curation in the form of two distinct processes:

i. *mediation*, in which museum visitors select one or more artworks, associate them with a theme and develop a script comprising a sequence of stages of contextual information and questions that guide visitor interpretation
ii. *interpretation*, in which museum visitors select a script associated with a theme and develop and share their own interpretation by following the stages of the script

Two applications working with the LDH have been developed: IMMA Viewpoints and IMMA Deep Viewpoints.

IMMA Viewpoints was the first application developed and focussed on the process of interpretation. Built as a website intended for mobile use, IMMA Viewpoints can be loaded with a set of artworks and a set of

[5] https://dok.digitaltmuseum.org/en/api
[6] https://www.programmableweb.com/news/top-10-apis-museums/brief/2020/12/26

questions stored on the LDH. For a trial, Viewpoints was loaded with a set of outdoor IMMA sculptures and a set of questions as used in a Slow Looking activity (e.g., "What's going on in this artwork?"). A visitor, generally when standing in front of the sculpture with a smart phone, could view information about the sculpture and respond to a randomly selected question or reload a new question. The responses were queued for moderation by IMMA. The moderated responses were shown on an "Other People" page where the visitor could find out how others had responded to the artwork. All responses were stored in the LDH[7]. IMMA Viewpoints was used by general visitors to the IMMA grounds from July to October 2021.

The IMMA Viewpoints application is available online at: https://spice.kmi.open.ac.uk/demos/imma-viewpoints

Full source code for the application is available : https://github.com/spice-h2020/imma-viewpoints

IMMA Deep Viewpoints (later referred to as *IMMA Slow Looking*) was subsequently developed to support both the mediation and interpretation processes. The mediator (who could be a member of IMMA staff or citizen community group) could create a script that guided the visitor through a set of stages containing statements, questions and IMMA artworks. The interpreter could develop and share their interpretation by proceeding through the stages of the script. Script owners or IMMA staff could moderate the responses to any script. Deep Viewpoints used IMMA artwork metadata added to LDH using SPARQL Anything and also stored scripts and interpretations on the LDH[8]. Since November 2021, Deep Viewpoints has been used in workshops with community groups including: Migrant Women - Opportunities for Work (Mi-WOW) via New Communities Partnership, Black and Irish, Black Queer Book Club, Asylum seekers and staff and students from Dublin City University as part of the MELLIE programme, and young people at Oberstown Children's Detention Campus.

5.1.2 Requirements

The two applications had the following data requirements:

- Storage and access to the IMMA collection metadata to support the inclusion of IMMA artworks within Viewpoints and the scripts of Deep Viewpoints
- Storage and access of contributions. These took the form of questions, answers and artwork selections in Viewpoints and scripts, themes, artwork selections and interpretations in Deep Viewpoints.
- User accounts and their management by an IMMA moderator were required in Deep Viewpoints. This enabled contributions (e.g., scripts, interpretations) to be associated with an author who could retain edit and delete rights to their contribution.

5.1.3 Application Workflow

A typical workflow in Deep Viewpoints is as follows:

1. A community member logs in Deep Viewpoints using their username and password. Login attempts are checked against the user accounts stored on the LDH.
2. They browse the existing themes and if they wish add themes to be associated with their scripts. Themes are stored on the LDH.
3. They search the IMMA Collection and select artworks for their community artwork collection to be used in their scripts. The IMMA collection is stored in a separate LDH dataset and queried via SPARQL.
4. They create a script comprising a set of stages containing statements and questions. An artwork from their community collection is selected as the homepage artwork for the script. Artworks from their collection can be added to any stage of the script. The script and its stages are stored on the LDH.
5. Once the script is ready, the author (or IMMA) can set the script as public (displayed on the Deep Viewpoints homepage) and open to contributions. They can also determine whether interpretations

[7] https://spice.kmi.open.ac.uk/dataset/details/47
[8] https://spice.kmi.open.ac.uk/collection/details/4

produced using the script are moderated before appearing on the Other People page. Script status information is stored on the LDH.

6. Any user of Deep Viewpoints can now access the script and create an interpretation. Interpretations can be created by either logged in or anonymous users. Logged in users retain edit/delete rights over their contributions. Interpretations are stored on the LDH.
7. Once the response has been moderated (if moderation was required) the interpretation is live on the Other People page. The author of the interpretation, the author of the script and the IMMA user account all retain long-term edit/delete rights over the contribution. Status and rights information associated with the interpretation are stored on the LDH.
8. At any time, the script author can close the script to new contributions. A closed script can still be kept public meaning it appears on the Other People page but is no longer featured as an available activity on the Deep Viewpoints home page.

5.1.4 Feedback

LDH has been responsive and reliable throughout both the Viewpoints trial (July to October 2021) and the Deep Viewpoints trial (since November 2021).

The artwork image files used within Deep Viewpoints are held in the IMMA online collection rather than in LDH. This means broken links can occasionally occur when artwork locations change in the IMMA online collection. This is resolved by refreshing the artwork links in the LDH. Caching the artwork images on the LDH would resolve this problem. The LDH has the capacity to do this. However, rights issues need to be resolved in order for this to happen.

Deep Viewpoints (and also Viewpoints) are Angular apps that communicate asynchronously with the LDH. This enables a seamless, responsive interaction with the content. However, this can create occasional design problems. For example, if multiple users are simultaneously adding, removing and reordering a shared set of themes, it is difficult to ensure consistent content across multiple clients. In a standard relational database driven design, this would probably be handled server side, using for example autoincrement fields and primary keys. This is difficult to manage purely client side. This may not necessarily indicate a missing LDH feature but rather the need to adopt appropriate design patterns when consistency is required across multiple clients.

5.2 Hecht Museum – Student Experience with Relevance of History and other Views

5.2.1 Background

School students before, during and after a museum school trip at the Hecht Museum learn about their country's history and at the same time learn about the diversity of opinions regarding historical and national issues. Students learn to interpret museum artifacts according to their own personal views, reflect upon other students' opinions, connect their opinions with tangible artifacts at the museum, and perform citizen curation activities. Teachers, educators and museum curators have been involved in the design of the activities and application, with teachers providing information and feedback on the educational goals of the system. We have designed and built a web application that supports students in the classroom, during the museum visit (on handheld devices), and after the visit. In addition, we have an application which supports the teachers, curators and researchers to evaluate an and analyse the data. The data consists of content generated by the users (photos, descriptions and tags), user answers to surveys, users' responses to questions, user model and history of user interaction with the system.

5.2.2 Requirements

The pilot needs to manage 5 major objects with basic CRUD operation: Users, User History, User Model Properties, User Generated Content and Sourced Content.

5.2.3 Application Workflow

The user interacts with a page/form where each field is a part of a property-value pair of the User History. When the form is submitted the fields are entered into the User History dataset. Additionally, if it is User Generated Content, it is sent to the Semantic Analyzer and stored in the corresponding LDH dataset. The

results from the Semantic Analyzer and fields from the User history are analysed and used to create property-value pairs for the User Model. The User Model info is then used to create communities by the Community Model. This info will be used by the recommender system.

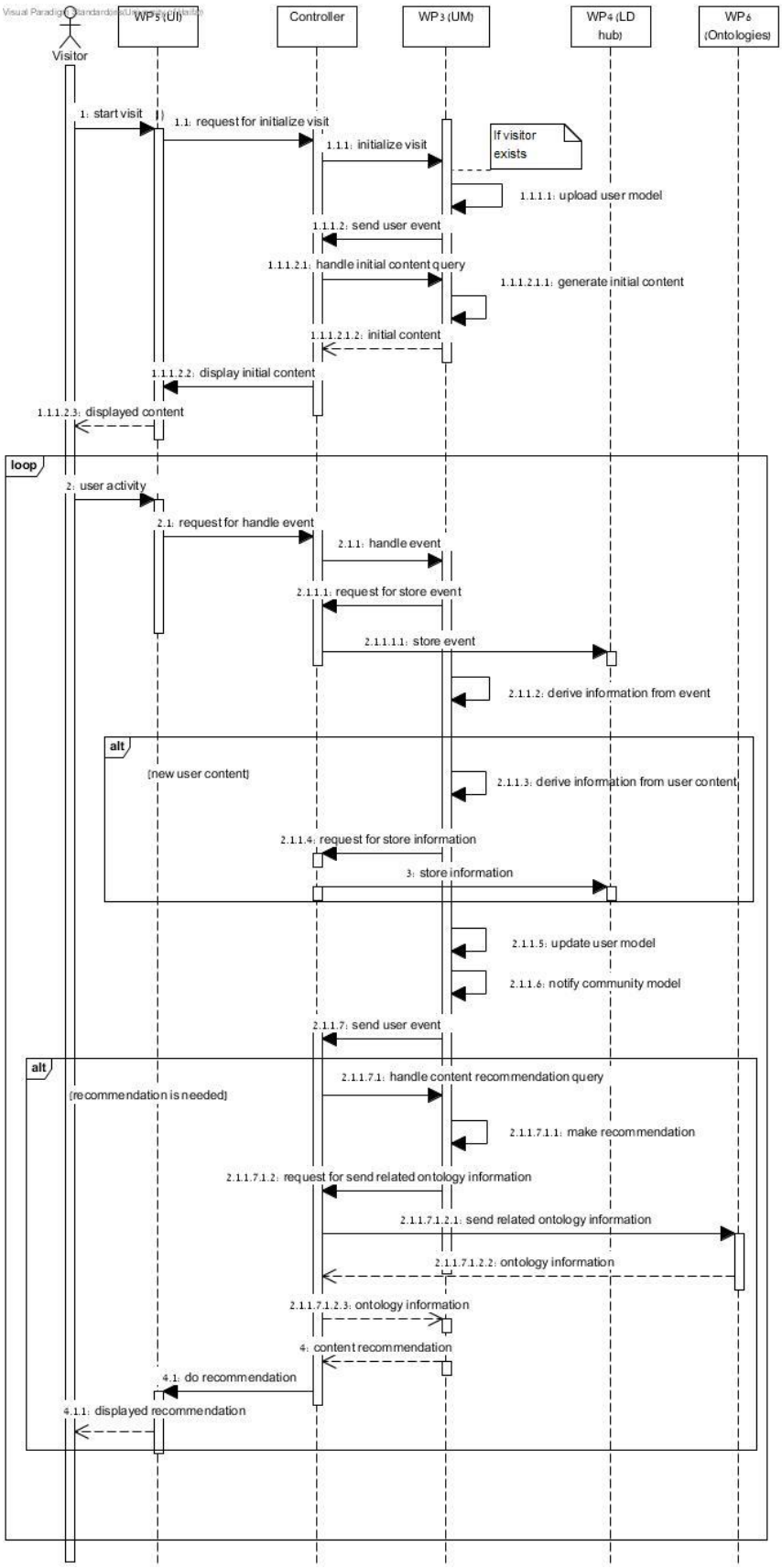

Figure 5.2.3.1. Application workflow

5.2.4 Feedback

The pilot makes use of the LDH indirectly through the User Model. The LDH covers all necessary functionality. The pilot application was developed with Java Spring Boot, making use of JPA (Java Persistence API) and a MYSQL database. The LDH most recent updates cover essential functionalities of this pilot (e.g. files repository), and efforts for integration are currently ongoing. The migration would be easier if the LDH could have integrated with the JPA architecture. However, Web APIs are currently the state of the art approach for distributed systems and, therefore, integration will follow this method.

5.3 Design Museum Helsinki – Pop-up VR Museum

5.3.1 Background

The focus of the Design Museum Helsinki (DMH) Case Study is on developing the citizen curation methods by first gathering interpretations of DMH collection objects in workshops with selected end-user communities, namely senior citizens, remote dwellers, and asylum seekers (D7.3, pg33). An application known as the the *Pop-up VR Museum* will be designed and accessible to audiences via portable VR headsets. Its users can access, interact, and engage with Design Museum Helsinki's collections. The target audience of the pilot include three (3) different end-user communities, namely senior citizens, rural communities, and asylum seekers. In this process, the mediators play an important role in aiding the application's development and guiding its users; these mediators are often comprised of curators, researchers, and members of institutions such as senior care centres.

Some prominent features of the Pop-up VR Museum for its end-users include:

i. interacting with DMH artefacts (**3D models .FBX** in a virtual environment) such as picking it up, rotating, and other kinds of tactile interactions
ii. selecting and listen to stories (**audio recordings .WAV/.MP3 and text**) emanating from the artefacts
iii. experiencing a generative immersive virtual environment (comprised of **3D models .FBX**, colour and lighting of the virtual scene dictated by themes, emotions, and values obtained from the narratives, i.e., **textual data** in the LDH)
iv. progressing within the experience through gameplay that involves solving quests/puzzles, reflecting on these stories with own contributions such as drawing (**3D models .FBX**), painting (**3D models .FBX**), narration (**audio recordings .WAV/.MP3 and text**), and comments (**text**).

Features for mediators and other contributors include:

i. curating several contributions via selection and editing
ii. adding new or missing artefacts to the catalogue of the Pop-up VR Museum

Through this dynamic process we aim towards inclusivity, enhancing participation in citizen curation via DMH collection, and closing the gap between museums and technology.

5.3.2 Requirements

The experience of the Pop-up VR Museum is bound to be generative and dictated by a dynamic online repository of artefacts (3D models) and narratives (audio recordings and textual data) stored in the LDH. Mediators such as DMH staff, researchers, and members of affiliated institutions would add artefact ontologies and narratives collected from end-user contributors to the LDH.

5.3.3 Application Workflow

The link below demonstrates the application overflow wherein a user belonging to two different end-user communities contributing a story that is experienced in the Pop-up VR Museum by a member of a different end-user community:

https://app.mural.co/t/spiceuxmapping5715/m/spiceuxmapping5715/1637004854396/69290182fd75d518b5bd2a0a81693b5c27188325?sender=ub0102f60708b02c681790953

Data regarding items that are color-coded in the legend are to be stored in the LDH.

5.3.4 Feedback

It would be useful for DMH to leverage recent functionality developments within the LDH to enable curators, mediators, and researchers to add datasets about artefacts and narratives. Specifically, the new API features supporting binary file storage and retrieval would support multimedia resources such as 3D models and audio descriptions of artefacts to be exchanged alongside traditional textual information using the existing JSON API.

Furthermore, recent developments to the RDF replication agent, particularly the ability to create custom graphs as materialised views on collections of existing linked data, may open up new user interface possibilities.

As we move forwards, we are looking at the process of integrating the LDH API with the Pop-up VR Museum to support the seamless dynamic storage, editing, and retrieval of these resources using the SPICE linked data server technology.

5.4 GAM Game

5.4.1 Background

The case study of the Gallery of Modern Art (GAM) in Turin, which addresses the inclusion of deaf people as target community, revolves around the notion of storytelling. Through the web app, called GAM Game[9], users can create short stories by collecting and sequencing the artworks from the museum collection, and add a personal response to each of the artworks in the story. Stories can be created at any time before, during and after the museum visit.

Personal responses can consist of simple *tags*, text *comments* (created from a catalogue of templates: [the artwork] makes me feel …, [the artwork] reminds me of …, etc.) and *emojis*. More than one element per each category (or none) can be added to each artwork in the story. The user can also upload a single image from her/his device as a response to an artwork.

Stories can be saved in the user's personal space associated to her/his account and later deleted (but not edited). In order to identify them, and also as part of the creative process, users are prompted to give a title to each story as a precondition to saving it.

After creating each story, the user receives a set of recommendations of artworks based on the emotions assigned by the DEGARI (see deliverable D6.3) reasoning service to the artworks in the story (in turn, these associations are inferred by DEGARI from the text content which accompanies the artwork: artwork description from the collection record, user comments, etc.). Recommendations are of two types: they concern artworks associated with the same or similar emotions, and artworks associated with emotions of opposite polarity. If accepted, a recommended artwork will be attached to the story.

Once shared, stories are associated with the artworks in the museum collections, and can be browsed by the other users, who can express their appreciation to them through likes, in the style of social media.

5.4.2 Requirements

The LDH infrastructure needs to manage three main entities in the interaction with the client:

- User id and data, which include the links to the stories created by the user;

[9] http://spice.padaonegames.com/gam-game/consumer/visit/6227632cc632154a7316e211/home

- Stories, each including its own properties (date, title, etc.) and the list of links to artworks by which it is composed, and the user responses (stored in textual form) associated to each artwork in the story;
- Artefacts, each accompanied by its metadata, which include both the ones extracted from the collection catalogues and the ones added by the sensemaking components (associated emotions, values, themes, etc.)

5.4.3 Application Workflow

Stories, after the user registration, are created by the users during the visit in the GAM museum. Later, they are able to send the stories (point 6 in the sequence diagram shown in Figure 5.4.3.1) (JSON array is composed by artefacts characterised by a set of attributes) to the LDH. After, User/Actors can request to LDH the JSON file (https://spice.kmi.open.ac.uk/dataset/stream/details/62) which contains the story by specifying <story ID>.

In any time, users can retrieve their stories by using a simple dashboard or create another story.

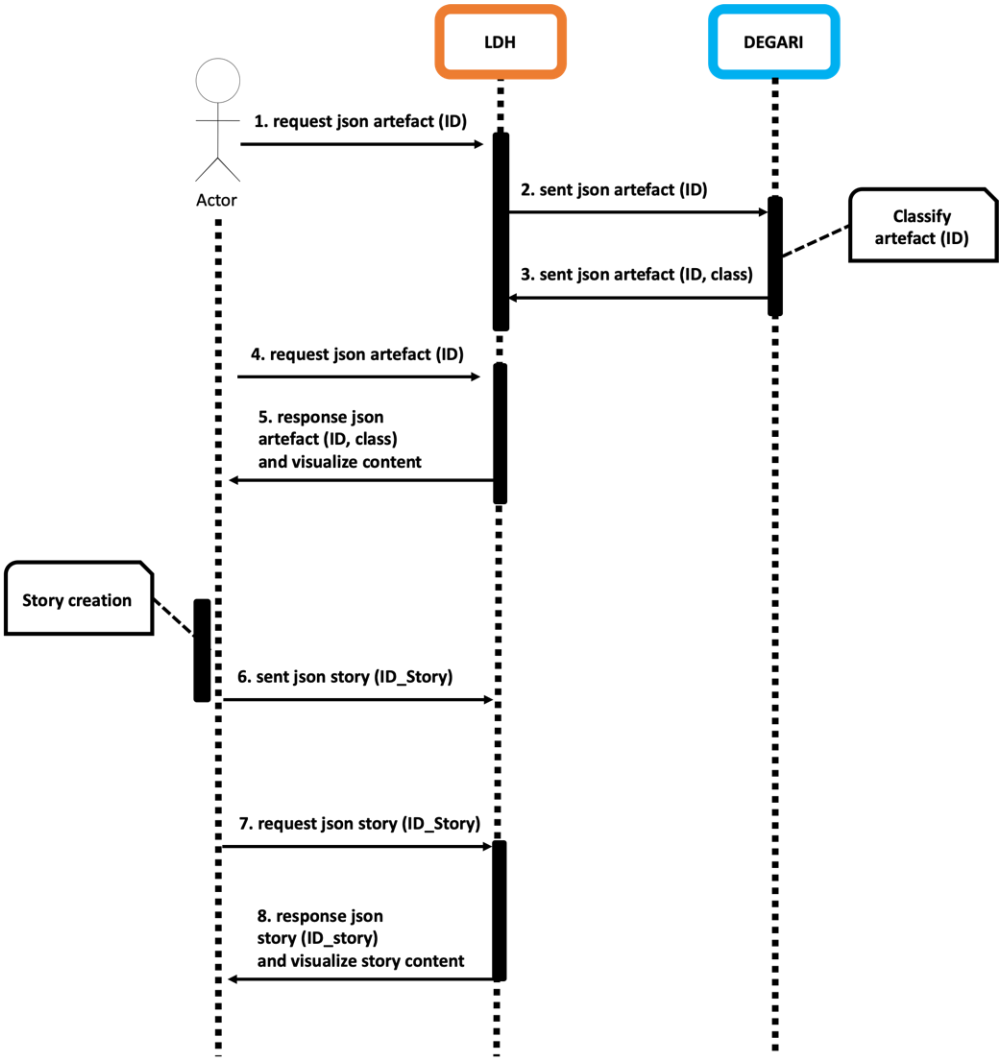

Figure 5.4.3.1: Workflow with LDH and DEGARI for GAM-artefact collection

The whole pipeline of the LDH and DEGARI, service is sketched in *Figure 5.4.3.1*, and relies on the following working as follows:

1. Users/Actors (via a client call initiated by a web app) can request to LDH a JSON file artefact by using SPARQL RDF-query method. This JSON file contains the description of a particular artefact from GAM Museum collection
2. DEGARI get from LDH a JSON file with ID and then executes the emotional reclassification of the JSON artefact
3. After the reclassification of the artefact, DEGARI automatically updates the LDH dataset by sending the same JSON artefact ID, with its specific emotional class-label.
4. Users/Actors (via a client call initiated by a web app) can request to LDH a JSON file artefact, by using SPARQL RDF-query method, in order to get the classification result for its JSON artefact (sent at step 1).
5. User/Actors can visualize JSON file artefact which has been enriched with the emotional class label by DEGARI <artefact ID, emotional label>.
6. Users/Actors (via a client call initiated by a web app) can create their personal story and send a JSON file story to LDH a JSON by using SPARQL RDF-query method.
7. After, User/Actors can request to LDH the JSON file which contains the story by specifying <story ID>.

8. LDH responds to the Users/Actors with the JSON file containing the previously requested story.

5.4.4 Feedback

Below we summarise general comments regarding potential changes/improvements that we would like to see included in the LDH and that are relevant for the GAM game.

The connection of the LDH data with the ontological reasoning services has scope for improvement. Currently, the only way to have this direct connection to the LDH is through an API service (currently under construction) that makes LDH dataset activity available to dataset owners. This service should be monitored regularly to observe updates to dataset content based on user activity and the generation of user-content. This architecture causes difficulties for near real-time use of the ontology-based and sense-making services provided in WP6. A push-based architecture, where updates flow in the direction from the LDH to the ontological reasoning service, and not the other way around, would be better suited to this scenario. In addition, it would be useful to ask the reasoning services to reason only on the part of the dataset that has changed, avoiding a re-run of the whole sense making service on the entire dataset when only small changes have occurred.

5.5 Madrid - Treasure Hunt

5.5.1 Background

The case study of the Natural Musem of Natural Sciences of Madrid (MNCN) revolves around treasure hunts in the museum. A treasure hunt consists of a series of searches guided by clues describing the object in the collection to be found. Once the object is found, the game provides relevant information related to it and may pose related questions.

The activity is designed for a group of schoolchildren led by their teacher. The teacher has the possibility to design a specific treasure hunt for her students, and thus choose the theme of the game, the selection of objects in the collection, the information to be shown and the questions to be asked.

The game itself is an application developed in Unity 3D that runs on a tablet. The definition of the content of the treasure hunt is done through a web application where the teacher can create the content and where she also has access to other treasure hunts created previously.

The answers to the questions posed during the quest are collected and provided to the teacher so that she can analyse the results of the activity.

The pilot application is used for both the building and creation of treasure hunt activities and also the execution of the treasure hunts and collection of user responses and interaction.

In what follows, we describe the application's technical requirements from the point of view of the SPICE linked data server technology, an overview of the application workflow and how the LDH is used within the context of the application and finally we give feedback on issues to consider when planning future LDH development.

The pilot makes extensive use of the InSpice framework developed in WP5 and documented in deliverable D5.2.

5.5.2 Requirements

According to the preceding background, the LDH infrastructure needs to manage a series of entities in its interaction with the client:

- On the one hand, it is necessary to store a user id alongside its associated user information, without actually keeping sensitive details such as passwords or personal emails within the LDH, the latter to be stored in the InSpice persistence layer itself. All content that references a user, such as answers

generated for given questions or treasure hunts created by a teacher or museum curator, will reference the associated user by means of their unique id in the LDH user collection.

- On the other hand, a collection of treasure hunt definitions over the museum artifacts needs to be maintained, each containing a set of common basic properties (availability in terms of dates, description, target audience, title, author id, whether the treasure hunt is public, or tags to facilitate its subsequent search and categorization, amongst others), and a sequence of phases to be executed within the Unity 3D application.
- Each of the phases adheres to a specific format determined by its type field (multiple-choice question, object search phase, multimedia information phase, etc.). As an example, an artifact search phase has the id of the museum artifact that the student must find when reaching this stage, as well as a list of clues written by the teacher to guide the search process, among other properties, as unique fields. The case of a multiple-choice question phase, to name a second example, is defined by the text associated with the question, a list of answers also in textual format, and the maximum and minimum number of options that the user can select.
- The artifacts referenced in the treasure hunts, together with their associated metadata and images, will also be contained in a dedicated LDH collection, and will be referenced by id in any subsequent use of them in application documents.
- User responses and interactions with each treasure hunt will be persisted in collections of interaction events, each with information regarding the session in which such interactions were recorded (time and date, unique session id, device, school or associated school group, etc), and with its data payload (response content, dependent on the type of phase in which the response is collected).

5.5.3 Application Workflow

The communication flow between the different parts involved in the creation of a treasure hunt by an educator or a museum curator is schematically represented in *Figure 5.5.3.1*. It should be noted here that prior to the start of action 1 (start the creation of a treasure hunt), the user has to perform a login action such as the one represented in *Figure 5.5.3.3.* Below we summarize the points included in this initial flow diagram.

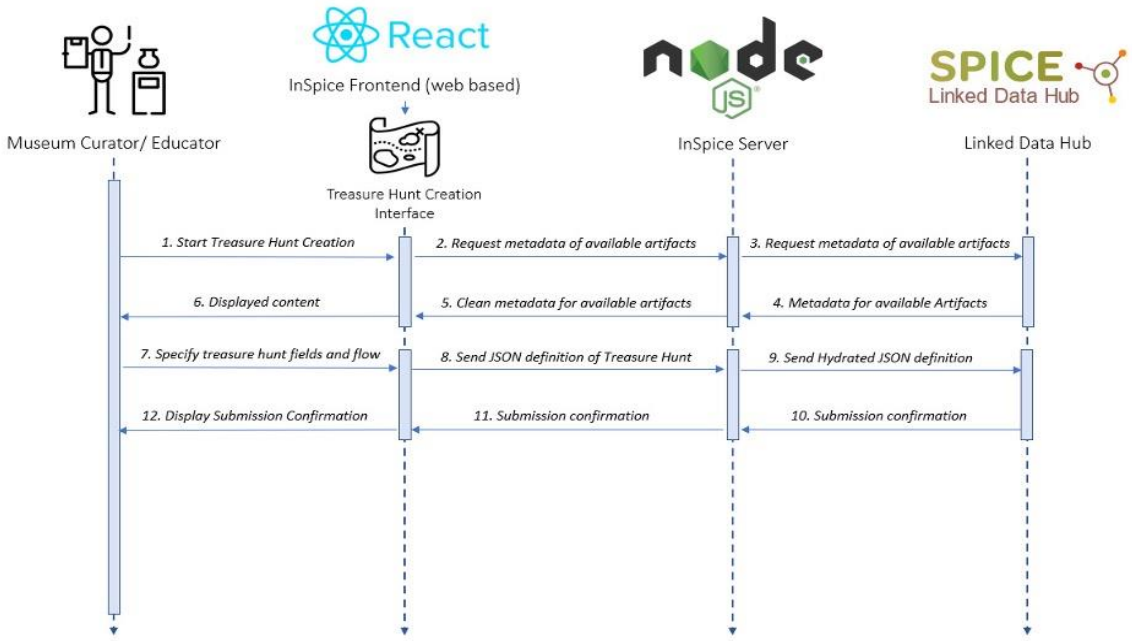

Figure 5.5.3.1: Treasure Hunt Creation Flow Overview

1. The user enters the web application and decides to access the flow for creating a treasure hunt after having logged in at a previous stage (otherwise, he/she will be redirected to the relevant flow before being allowed to create content).

2. The web application requests a list of available pieces of works from the InSpice server in the context of a treasure hunt as a step preceding the loading of the creation interface. This list of works corresponds to a JSON file containing an array of objects with the fields exposed within the museum's collection of artifacts stored in the LDH.

3. The InSpice server requests a complete list of works from the LDH (or a single page in the collection if this is expected to be of large enough size to be unmanageable).

4. The LDH returns an array of objects in JSON format with the information associated with each of the artifacts in the collection.

5. The InSpice server returns a list of JSON objects with the artifact information to the web application after a potential intermediate cleanup step and mapping of the data to the possibly more simplified data type used on the client side.

6. With this information, the web application updates the view to show the works included in the collection, as well as the complete creation flow of the activity, in the form of a web formulary that enables a definition to be constructed in a way that is convenient for the curator.

7. The application user fills in the necessary fields to define his/her treasure hunt, and proceeds to define phases of different types, selecting pieces from the collection whenever an artifact search type phase is generated. Once the user has completed the definition, he/she proceeds to perform a submission action.

8. The web application builds a suitable definition in JSON format from the data entered by the user and sends it to the InSpice server.

9. The server receives the definition and hydrates it with additional fields such as the id of the user who is designing the activity, if necessary. After this, the hydrated JSON file is sent to the LDH for storage and further consumption.

10, 11, 12. The LDH receives and stores the new definition file, incorporating possible additional data (time, date, etc), and consecutive confirmation messages are sent until reaching the client side where an upload confirmation screen is displayed.

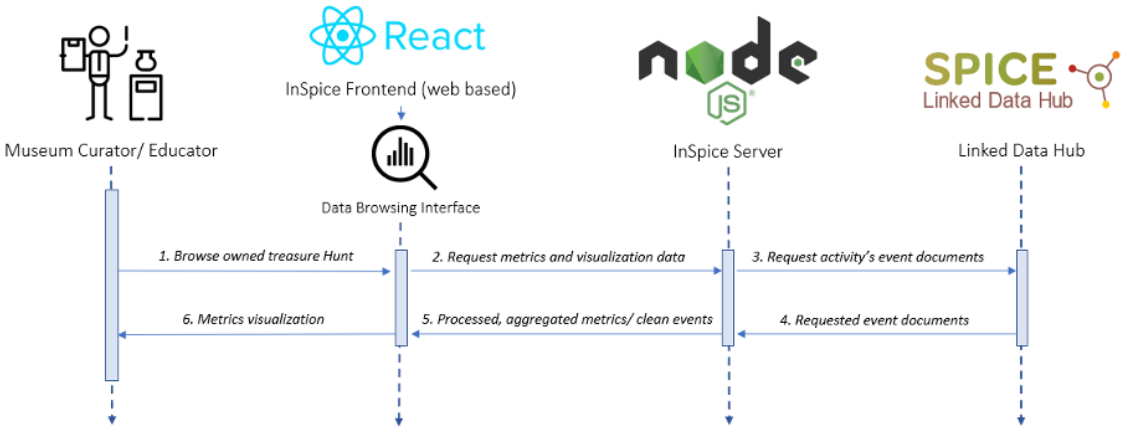

Figure 5.5.3.2: Treasure Hunt Metrics Visualization Flow Overview

The data visualization flow of the interactions with a treasure hunt created by a particular user is summarized in *Figure 5.5.3.2*. In this case, upon entering the data browsing interface, the web application requests the necessary metrics and/or raw events for the corresponding user request (e.g., distribution of answers to a given question by users associated to a specified class). The InSpice server, on its side, retrieves the required events from the LDH and uses them to compute an aggregation that satisfies the requirements of the original request. The result is then sent in JSON format to the web application, which in turn uses it to display an appropriate diagram/table or file to the user.

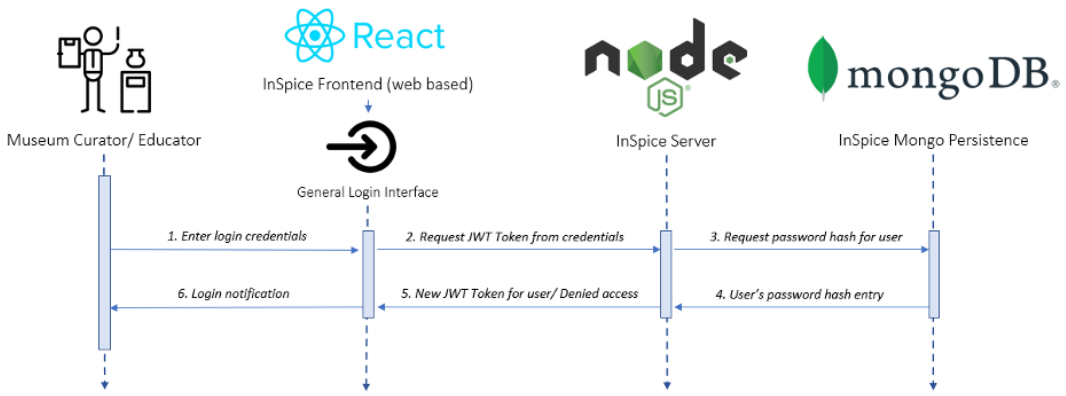

Figure 5.5.3.3: General Login Flow for any authenticated InSpice tasks

The login process (*Figure 5.5.3.3*), on the other hand, does not make any direct use of the LDH, since the validation of user credentials occurs entirely on the InSpice server side and its persistence (which is where the user password hashes are stored). Thus, when the user makes a login request, the web application sends the credentials in JSON format via a secure connection to the server, which in turn is responsible for validating the credentials and, in case of success, issuing a JWT token with a limited validity with API access rights to the corresponding user. This token is sent back to the client, who then stores it into local storage and can use it in future requests to perform any type of constrained action on the server.

It is worth mentioning here that the LDH is in fact used in the user registration part, during which two parallel documents are created both in the InSpice persistence and in a user data collection within the LDH without sensitive information. This ensures a separation between users' sensitive information and their general interest data in the LDH.

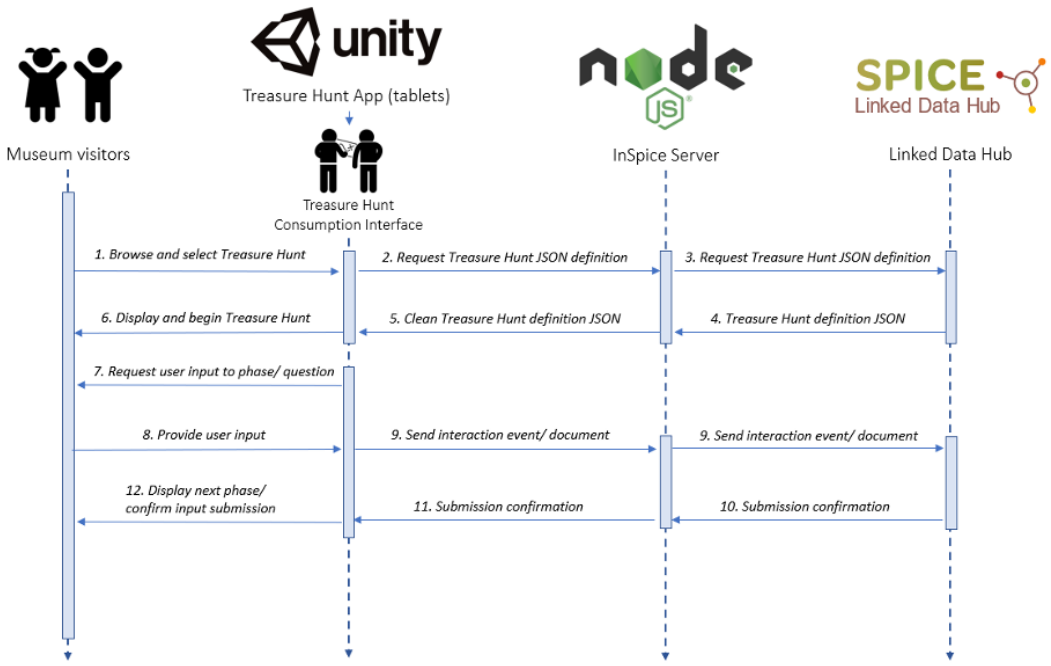

Figure 5.5.3.4: Treasure Hunt Consumption Flow Overview (Museum tablets)

Finally, *Figure 5.5.3.4* shows the general consumption flow of a treasure hunt from the perspective of a user of the Unity application on museum tablets. In this context, the app user, usually guided by an educator or by a museum curator who has been involved in the creation of the treasure hunt, proceeds to search by name and/or by unique identifier for the activity to be undertaken in an interface enabled for that purpose at the start of the Unity app (1). The Unity application then makes a request to the InSpice server to retrieve the JSON file with the activity definition (2). The InSpice server makes a subsequent request to the LDH (3), from which it receives this document (4) and finally, after a possible cleanup and mapping step to a format that can be understood by the Unity application, sends it to the latter (5). Upon receiving this information, the application is ready to launch the treasure hunt (6).

At each stage of the definition, the application will request some kind of user input (7, 8) (in the form of questions, artifact searches, etc.). The user interaction information at each phase is hydrated here with session data (school group to which the student belongs, unique session id, etc) and sent to the InSpice server for subsequent storage in the LDH (9, 10, 11, 12). This process is repeated until the end of the activity on the user side is reached.

5.5.4 Feedback

Below we summarize some general comments regarding potential changes/improvements that we would like to see included in the LDH:

1. One of the typical use cases within our applications is the manual introduction of new data objects or definitions within a given collection into the LDH (artifacts, test treasure hunts definitions, etc), as well as the browsing of such objects. Currently, the only way to have this direct connection to the LDH is through the API enabled for that purpose, but this is not particularly convenient and would greatly benefit from some sort of web interface within the LDH page itself to be able to do manual browsing and editing, deleting, or uploading of objects in the collections.

2. On the other hand, another common scenario we have encountered is that artifacts often go hand in hand with some kind of associated multimedia content (images, videos, audio files), which is currently a very limited functionality within the LDH. While we understand that this type of persistence makes more sense in the context of a service dedicated to that purpose, it would be very useful to count with some guidelines or

general recommendations on how to integrate these services within the infrastructure of an application that uses jointly the LDH and persistence of multimedia files (file referencing, ids, recommended technologies, etc).

6 Integrating the feedback from SPICE pilots

The feedback gained from the pilot applications described in this report are encouraging and suggest that we are moving in the right direction with the development of the SPICE Linked Data Hub. Indeed, many of the requests for new functionality are in fact features that have only recently been released and others are those that are on the development roadmap for the coming year.

A common theme emerging from several pilots was the ability to manage not just JSON documents via the API but also a range of other multimedia resources. The new LDH file management features are a welcomed addition to the platform and we see them being used heavily as the pilots progress. The integration of the file management features with the graph database querying features will also open up opportunities for application developers to use SPARQL to explore these resources alongside existing textual data and metadata.

Where these file storage features are being used to keep cached copies of resources that originate elsewhere, it is important that we look into potential associated copyright and licensing issues. It may be the case that the solution to this varies from application to application, but LDH functionality to should be put in place that could support this.

It is clear that, while the LDH provides access to its resources in graph database form, individual pilots may have RDF data requirements that lie beyond the scope of this work package. In these cases, a fine-grained publish-subscribe model that enables pilot application developers to maintain their own graph database for specific analysis and processing would be beneficial.

Moving to the third year of the project, the linked data server technology developed so far is in a good place for addressing some of the final requirements of WP4. Many of the recent developments to the LDH have not brought about large functionality changes, but have been put in place to provide the foundations for upcoming feature development.

Upcoming work on requirements 33 and 34 (*Table 3.1*), requesting and granting access to resources, will build on much of the recent work that has gone into finer grained control and visibility of dataset access for both owners/custodians and users/builders.

The development of the RDF replication functions and more sophisticated LDH activity logging also now gives a good platform for opening up access to the LDH activity log to dataset owners. Combined with the proposed implementation of a publish-subscribe model, this will give application developers the ability to maintain their own replicated graph databases that will support near real time semantic reasoning and analysis.

7 Conclusions

This report presented the deliverable of WP4 of the SPICE project, focusing on developments to the SPICE Linked Data server technology. It also provided a report of progress on SPICE pilot applications that make use of the Linked Data Hub, offering feedback on how the integration has gone so far and what some of the main areas for development should be with respect to the LDH as we move into the third year of the project.

8 References

[Daga et al, 2021] Daga, Enrico; Asprino, Luigi; Damiano, Rossana; Daquino, Marilena; Agudo, Belen Diaz; Gangemi, Aldo; Kuflik, Tsvi; Lieto, Antonio; Maguire, Mark; Marras, Anna Maria; Pandiani, Delfina Martinez; Mulholland, Paul; Peroni, Silvio; Pescarin, Sofia and Wecker, Alan (2022). Integrating Citizen Experiences in

Cultural Heritage Archives: Requirements, State of the Art, and Challenges. Journal on Computing and Cultural Heritage, 15(1) pp. 1–35. DOI: https://doi.org/10.1145/3477599