Europeana Space – Spaces of possibility for the creative reuse of Europeana's content

CIP Best practice network - project number 621037

Deliverable number	D2.3
Title	The Europeana Space Infrastructure

Due date	Month 20
Actual date of delivery to EC	28 January 2016

Included (indicate as appropriate)	Executive Summary	☒	Abstract	☐	Table of Contents	☒

Project Coordinator:

Coventry University

Professor Sarah Whatley

Priority Street, Coventry CV1 5FB, UK

+44 (0) 797 4984304

E-mail: S.Whatley@coventry.ac.uk

Project Website address: http://www.europeana-space.eu

Context:

Partner responsible for deliverable	NTUA
Deliverable author(s)	Nasos Drosopoulos
Deliverable version number	3.1

Dissemination Level	
Public	☒

History:

Change log			
Version	**Date**	**Author**	**Reason for change**
0.1	23-07-15	Nasos Drosopoulos	Outline & Table of Content
0.2	25-09-15	Nasos Drosopoulos, Eirini Kaldeli, Giorgos Marinellis, Maria Ralli, Michael Giazitzoglou	First draft
1.0	23-10-15	Nasos Drosopoulos	Second draft – Internal peer review (NTUA)
1.1	30-10-15	Nasos Drosopoulos	Corrections from peer review
2.0	30-11-15	Nasos Drosopoulos	Add current developments for UI and data model
2.3	07-12-15	Nasos Drosopoulos	Peer review version by IN2, Promoter and COVUNI
3.0	13-01-16	Nasos Drosopoulos	Final version
3.1	28-01-16	Tim Hammerton	Minor additions

Release approval			
Version	**Date**	**Name & organisation**	**Role**
3.1	28-01-16	Tim Hammerton, COVUNI	Project Manager

Statement of originality:

This deliverable contains original unpublished work except where clearly indicated otherwise. Acknowledgement of previously published material and of the work of others has been made through appropriate citation, quotation or both.

TABLE OF CONTENTS

1 EXECUTIVE SUMMARY

The Europeana Space project aims to increase and enhance the creative industries' use of Europeana and other online collections of digital cultural content, by delivering a range of resources to support their engagement. The project addresses all sectors of the creative industries, from content providers to producers, exhibitors, artists and makers of cultural/creative content, publishers, broadcasters, telecoms and distributors of digital content.

The Technical Space will be available for cultural institutions and organizations, professional users and third party developers in order to easily search for the cultural resources that meet their retrieval criteria so as to collect, use and re-use them to promote innovation and demonstrate the social and economic value of cultural content. This is achieved through the delivery of APIs that facilitate the development of applications based on cultural content and is validated through the realisation of the six pilot projects. The latter serve as the basis for continued experimentation and innovation in a series of dedicated hackathons and workshops that are expected to produce new applications and services based on Europeana's resources. The interaction with this Innovation Space produces additional requirements that are evaluated and addressed by the Technical Space, reflecting a 'real-world' approach that can be make developments immediately useful.

This document reports on the first official release of the platform, describing its software components and services, the workflows and interfaces for engaging and interacting with the users and, the processes that drive its ongoing development and fine-tuning to align with emerging requirements. The aim is to support the project's activities by providing a reference document that addresses the needs of the different levels of users involved (content providers, curators, developers and end-users), while setting a framework for systematically gathering requirements for the development in following iterations.

2 INTRODUCTION

2.1 EUROPEANA SPACE INFRASTRUCTURE CONCEPT AND COMPONENTS

Europeana Space introduces the Technical Space in its effort to support and promote the re-use of digital cultural heritage resources. The Technical Space is a platform for storing, accessing and processing content and metadata. It is designed and developed in alignment with complementary services used and produced in the Europeana ecosystem, and informed by the design of respective infrastructures being developed such as Europeana Labs[1], the Europeana Cloud[2] and LoCloud[3], and of more specialized applications targeting cultural heritage content visualization and re-use, such as the tools and pilots of AthenaPlus[4] and EUscreen[5]. Feedback from pilots, and the Content and Innovation Spaces have also been considered.

At a high level, its functionalities can be summarized in the following list:

- Aggregate multiple sources of cultural heritage content.
- Create and curate collections of digital resources.
- Add your own metadata and content to the search base.
- Maintain interoperability with data models and standards using the services of the metadata processing unit (MPU, see D2.2 – *The Metadata processing unit* - for specifications and documentation).
- Store metadata in several formats and serialisations; support for widely used domain models.
- Serve collections as specific back-ends for specialized front-end applications.

When reaching out to potential user groups and stakeholders, the methods of engagement can be broken down to:

- Discover (leveraging APIs to cultural content from around the world).
- Aggregate (upload metadata, map, transform and publish to external repositories).
- Create (collections, exhibitions and stories).
- Annotate (link and enrich using SKOS thesauri and linked data repositories).
- Participate (share, tag and rate collections, follow users & spaces and join user groups).
- Build (use the API to access data and services).

1 http://labs.europeana.eu/

2 http://pro.europeana.eu/project/europeana-cloud

3 http://locloud.eu

4 http://www.athenaplus.eu/

5 http://blog.euscreen.eu/euscreenxl

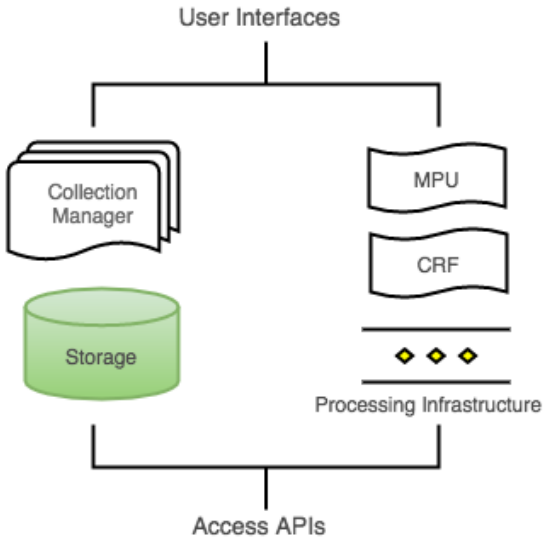


Figure 1. *Software components*

The platform consists of the following components, illustrated in the diagram of Figure 1:

- The storage layer for metadata and content.
- The workflow engine for managing and publishing collections of digital resources.
- The Content Reuse Framework and the Metadata Processing Unit.
- The Processing Infrastructure and the services deployment and integration layer.
- The User Interface to access functionalities and visualize data.
- The Access APIs for the platform's data and services.

It has become clear through the interaction with the project's pilots, as well as from the requirements analysis that was also informed by Europeana, networks and projects in the domain that a key point to making this infrastructure fulfil its objective is to facilitate content sourcing while respecting intellectual property and highlighting available rights clearance procedures. By streamlining this, often prohibiting, part of the process the project can then reach out to potential stakeholders and users offering clear workflows that will trigger their interest, enable the participation of the creative sector and present incentives for re-using the work of cultural heritage institutions.

2.2 REQUIREMENTS COLLECTION AND ANALYSIS

Deliverable 2.1 - *Requirements for the creative use of Europeana Cultural Resources* - reports on the approach and outcome of the collection of requirements from all sectors of the creative industries, from content providers to producers, exhibitors, artists and makers of cultural/creative content, publishers, broadcasters, telecoms and distributors of digital content. This analysis was possible primarily through the interaction with the project's pilot teams that explore scenarios of content re-use in six thematic areas, including the concept of the Protected Space. This also considers the concept of the Protected Space, which is further detailed in D3.2/4 – *Final Report on the Legal Aspects and the Content Space.* In a first iteration the teams demonstrated prototypes that exploit cultural heritage repositories, followed by the organization of co-creation and co-development events, such as hackathons, that aim to facilitate experimentation and streamline the development of further applications.

Furthermore, the analysis is informed by evolutions in the Europeana ecosystem, especially in relevant projects such as Europeana Creative[6], LoCloud, Europeana Food & Drink[7], EUScreen and Europeana Sounds[8]. In the framework of the last two in particular NTUA has been developing pilot applications based on the Technical Space platform, helping to further identify and address requirements of domain aggregators and, to set up and promote thematic user groups within the Technical Space platform, as those are described in the respective section (3.3.1).

Participation in Europeana's Projects Group Assembly and related technical workshops as well as in the EuropeanaTech[9] R&D community and its respective task forces has also assisted in gathering requirements from researchers, developers and experts in the network. The result was a more detailed specification of the Technical Space's architecture (reported in D2.1) and the identification of potential synergies and cooperation with existing and under development systems, in various technical levels such as storage infrastructure or third party services for developers and users. Interoperability with the Europeana ecosystem is a significant aspect of the architecture in order to inform technical choices with existing system capabilities and requirements. These are all ongoing interactions and collaborations that enable important testing scenarios for the platform, while providing new requirements and use cases to the agile development process.

The precise collection and analysis requirements for the Technical Space were unclear during the initial months of the project. During the negotiation phase, the requirement to use the Europeana Labs infrastructure as a repository was added to the DoW, but details of what this was did not become clear for some time and ultimately when they did, Labs had developed in a different direction to that required by the Europeana Space project therefore requiring the re-instatement of the original plans. This period without clarity had an impact upon the pilots that had to develop their prototypes without access to what would become the WITH platform and API, with the technical infrastructure having to fit at a later date.

2.3 STRUCTURE OF THE DOCUMENT

Chapter 3 presents the Technical Space platform, starting from an overview of modules in 3.1 and a detailed presentation of the Data Infrastructure (3.2) and the Collection Manager (3.2), which correspond to the back-end and front-end of the system. Chapter 4 reports on the evolution of the platform and its architecture through the interaction with the project's pilots and events (4.1) and WP2's evaluation activities (4.2). The last three sections present ongoing and future development plans for significant parts of the infrastructure, namely its user interface (4.3), its data model (4.2) and the services for annotating, linking and enriching data using the principles of the semantic web. Finally, Chapter 5 summarizes the report and the work package's ongoing and expected activities related to the development and deployment of the Technical Space.

6 http://europeanacreative.eu

7 http://foodanddrinkeurope.eu/

8 http://www.europeanasounds.eu/

9 http://pro.europeana.eu/structure/europeana-tech

3 THE TECHNICAL SPACE

3.1 OVERVIEW OF SOFTWARE MODULES AND SERVICES

The Technical Space is built around a storage layer for content and metadata that are either uploaded/ingested or collected from one of the supported sources of cultural heritage material. Figure 2 illustrates this architecture, offering a high level overview of the platform's services and interactions.

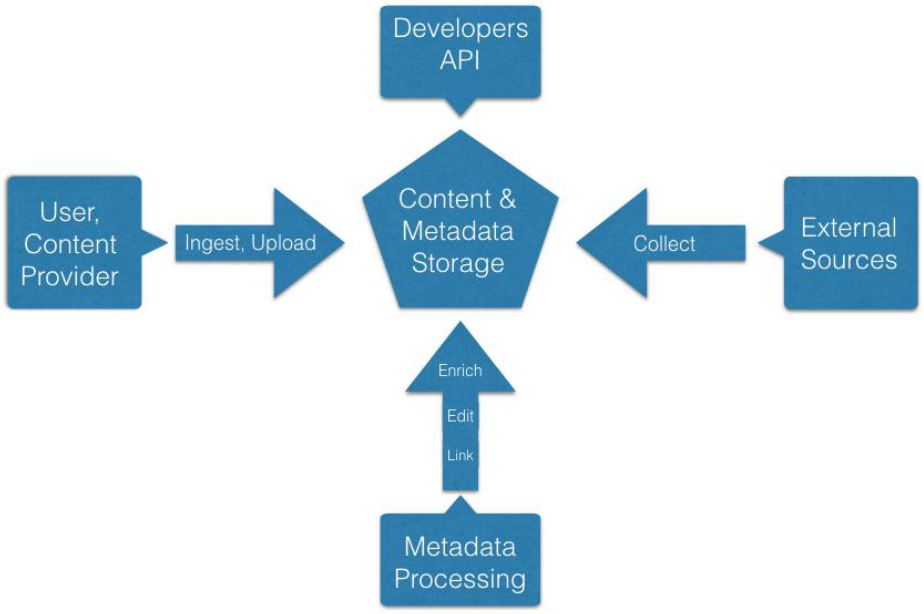


Figure 2. *Services and interactions with the Technical Space repositories*

A normal or specialized (e.g. Content Provider) user can either upload content and fill in related metadata from the platform's user interface or, use the Metadata Processing Unit (deployed using the MINT tool[10]) to harvest or ingest datasets containing metadata records for digital CH resources that are served online in an institution's portal or repository. The MPU, along with newly introduced services built on top of the Technical Space, also allow editing and enrichment of metadata records as well as linking between CH resources. The results of these procedures are also stored in the platform, keeping all required information such as provenance, the type of service employed, a confidence metric for the result and so on. Users also collect resources using the federated search for external APIs offering access to CH content that can be copied and stored locally for different purposes depending on the associated rights statements.

10 http://mint-projects.image.ntua.gr/espace

Finally, the platform's own API is used to access collections created by the users on the Technical Space and use them as a backend for applications developed by third parties.

There are various building blocks for the Technical Space; existing, such as the MINT platform that implements the MPU, the Europeana API and various 3rd party APIs and services and, newly designed and developed such as the storage layer, the processing infrastructure and the UI. The new platform that integrates the aforementioned blocks and implements the Technical Space backend and UI is based on WITH, a software for data aggregation and publication developed by NTUA. On a high level, these building blocks are split in three distinct sets of software tools:

- The Data infrastructure that includes the storage layers for content and metadata from available sources, and the ingestion interfaces that enable the collection of resources.
- The Processing module that incorporates available services for the management and manipulation of content and metadata resources.
- The Access module that includes the user interface of the platform and the set of APIs made available to professional users and third party developers for the creation of applications that use and re-use digital cultural heritage resources.

This deliverable focuses on presenting the backend and the user interfaces of the platform, documenting the workflows, and planning for future iterations of the software. The APIs of the platform are also documented, mostly from an architectural perspective as D2.4 – *Access APIs* - elaborates in more detail on the programmatic interfaces offered by the Technical Space.

3.2 DATA INFRASTRUCTURE

The main objective of task T2.2 is the creation of the data infrastructure, on which the Metadata Processing unit and the Access APIs will be based. It will be populated with metadata and content that users search&collect or upload, enabling access through APIs and offering services to associate content in an interoperable and semantically rich way.

The core of the data infrastructure consists of:

- The content storage and retrieval system that supports content sourcing for consumption and re-use by applications,
- A content analysis suite, based on Europeana Creative's Media File Checker[11], for detection and extraction of content's technical metadata,
- A repository to ingest, collect and manage metadata and a semantic repository for the respective serialisation of resources and their linking with those extracted from other sources and,
- The indexing and processing engines, which support the infrastructure modules, such as the MPU, with search&retrieval, XML processing and statistics services.

11 https://github.com/europeana/contrib/tree/master/ntua/mediachecker

3.2.1 Content storage

The Technical Space provides storage and HTTP access to medium and high quality content uploaded by users. The types of content supported are image (formats supported by ImageMagick[12]) and text files (txt, epub, pdf, xml, rtf). For content stored in typical web-based repositories (e.g. Flickr), as well as for the storage of videos (e.g. YouTube), API integration is implemented in the collection manager, described in Section 3.3. To upload offline content a user also needs to ingest the respective metadata records, using the MPU and its mapping tool in order to include in the record the URLs issued by the content repository along with a license or a rights statement to define the conditions for the re-use of that view of the content. There are evolving discussions and developments for the project's Content Space (WP3) and in particular regarding the implementation of different access rules for content according to usage scenarios and identified users. As far as the technical infrastructure is concerned, it controls access to content based on its license, while also allowing for employing different licensing and access strategies that project partner and developers expect to follow for the same content according to its intended audience.

There are two different ways for uploading and storing content: one for single images and one for batch uploading of files. Single file uploading is straightforward and is performed through the collection manager UI. It requires the creation of a collection for storing the image into and filling a minimal list of metadata (at the moment only the image title and a short description but eventually a full record, e.g. EDM). Batch upload is more complex but offers greater flexibility for metadata processing. It is built as a two step process, documented in D2.2; content is uploaded to the Technical Space using the FTP protocol (users register through a web interface[13]), and then the MPU is used to upload and align the metadata to the content. Using this methodology, it allows for uploading of content regardless of type (image, video, audio or text files, depending on the capabilities of the media server that will in turn host it). More so, it allows for post-processors to be created and offer additional functionality in the form of content analysis that will be elaborated in the following section. For example, using the Media File Checker in a post-processing step, the records that correspond to content files can be enriched with technical metadata (i.e. width, height, color space, color palette, file size, bitrate, frame rate etc.).

3.2.2 Content analysis

The inspiration for this module is Europeana's Content Reuse Framework[14]; this is a storage and access infrastructure developed by Europeana Creative & Europeana Cloud to allow interaction between content providers and creative industries based on the Extended Europeana Licensing Framework. The latter enables access to high quality content based on respective rights statements. The Media File Checker is a software agent developed, with the participation of NTUA, in the framework of the Europeana Creative project as part of the Content Re-use Framework.

12 http://www.imagemagick.org/

13 http://space.image.ntua.gr/

14 http://pro.europeana.eu/files/Europeana_Professional/Projects/Project_list/Europeana_Creative/Deliverables/eCreative_D3.3_KL_v1.0.pdf

It performs automated extraction and detection of technical properties for content that is related (as edm:WebResources) to a cultural heritage object (edm:ProvidedCHO) in the Europeana Data Model and more specifically can:

- Recognise an evolving set of identified file formats of interest (currently: jpg, png, tiff, MP4, mov, WEBM, MP3, FLAC, WAV, APE, pdf, txt, epub, xml and rtf).
- Determine the resolution of still images and video files.
- Determine the sample rate and bit depth of audio.
- Determine if a text file can be fully searched.

This data can be stored as metadata (enrichments) related to the edm:WebResource. The current version provides a collection of static functions that wrap around some of the best media analysis libraries available. These include ImageMagick for images, FFMPEG for audio and video and, iTextPDF for PDF files. The service updates metadata resources that link to the content to include information on dimensions, MIME types, color spaces or palettes and quality.

The second version that is currently being designed and will be developed in the framework of Europeana Space and Europeana Sounds introduces a Content Analyzer to perform classification of content and enable the use of more specialized analysis tools. NTUA is also planning to introduce content-based image analysis tools that can implement respective search engines. This has been tested in the Europeana Fashion[15] project through the detection of dominant colours and shapes in fashion catwalk images. With this type of analysis users will eventually be able to pose visual queries for specific classes of objects (buildings, people, faces and so on).

3.2.3 Metadata storage, models and serializations

The storage layer of the Metadata Processing Unit offers an XML repository where metadata taken from Europeana or from Content Providers' repositories, as well as the versions produced after operations by the MPU services can be stored (for detailed documentation, see D2.2 – *The Metadata processing unit*). The MPU groups metadata records that refer to content into datasets that can be accessed individually. A dataset corresponds to a set of records that are represented using the same data model. Any data model is allowed when ingesting, while aggregation and publication can be performed using EDM and available profiles, or more domain specific standards such as LIDO. Record serialization can be CSV, XML or JSON with preview interfaces for raw data and HTML renderings where available (e.g. LIDO XML2HTML transformation). Access types include HTTP upload and download, OAI-PMH protocol and HTTP API for the storage layer of the Technical Space. For supported, formal data models (such as EDM and profiles, domain models like LIDO) the system offers schema validation services based on the XSD and/or schematron rules. Datasets can have different versions that are stored and accessed individually. New versions of a dataset are mainly produced either by transformations to other data models or through the application of data manipulation services (e.g. enrichment, linking). Other types of dataset transformations that produce new versions include the application of services for ID generation, normalization and dereferencing.

15 http://www.europeanafashion.eu/

Uploading a dataset, performing schema validation and transformations to different data models are the main processes provided through the MPU. The user can export appropriate representations of a dataset's records to the Technical Space through the “export to WITH” option that connects the MPUs interface to the Technical Space. An XSD representing the WITH XML schema has been imported in MINT along with an XSLT mapping between EDM and WITH XML schemata. The user is requested to type the Name of the collection and also mark the new collection as publicly accessible or private. After this step the browser transfers the user to a new Tab, showing the log in screen for the Technical Space, to complete the export to WITH process. This creates a new collection in the Technical Space containing WITH-compliant representations of the imported dataset's records. It is also possible to transfer the original and/or the EDM versions of the dataset to the Technical Space, to be exposed through the API.

The MPU already stores all versions of a dataset used in a workflow in XML serialization (or CSV for original submitted data), along with formal transformations (XSLT) between them. For XML and JSON files, a NoSQL system has been integrated (MongoDB[16], used for XML files based on which MINT’s OAI-PMH server is implemented). The Technical Space uses the same infrastructure to store and serve through OAI-PMH XML metadata imported from the MPU, created by a user during the upload of content or, metadata collected using the capability to search and retrieve digitized items from a number of different online repositories. The federated search functionality (described in detail in the next chapter) allows for simultaneous search of multiple external sources such as Europeana, the Digital Public Library of America (DPLA), and DigitalNZ, which provide access to digital objects along with metadata describing the actual content. Other online collections of digital culture-related objects, from sources such as museum repositories to more generic like YouTube, can also be part of the federated search service offered by the platform, and the metadata they provide can be stored in the data infrastructure.

Items coming from different sources are stored as *Records* into the data infrastructure. A record comprises the content of the item and the metadata which describe it in a well-defined JSON format. The WITH JSON includes a number of important fields, such as “title”, “description”, “dataProvider”, “rights” etc, which are extracted from the original item metadata (see Appendix I for the current full WITH JSON specification). Custom mappers from a number of supported schemata to the WITH-JSON can be easily added to the system, while for datasets imported from MPU, the mapping editor can be used. In all cases, the serialization(s) of the full metadata representation(s) of the original item are retrieved and saved in the WITH JSON under the “content” field. A number of different schemata, either in XML or JSON format, are supported, like Europeana's JSON-EDM and JSON-LD, EDM XML or LIDO XML.

A record saved in the WITH database can only exist as part of a *Collection,* which is the first-class data structure in WITH, and corresponds to a set of records along with a set of metadata describing the collection itself (currently these include the fields title and description, a set that will evolve in next releases to address the requirements that are being defined in a dedicated task force set up within Europeana Space). Users of the platform can search for records they are interested in (both from the external data pools and the WITH database), and make a selection to create their own collections. Users can manage their collections by adding records, and editing their title and description.

16 https://www.mongodb.org/

They can also share their collections with other users or user groups, by passing them READ or WRITE access (more information on access policies and rights management are provided in Section 3.3.1).

An *Exhibition* is a type of collection that can contains additional user annotations for each of its members in the form of i) caption/description that serves as the story narration and ii) attached video from external repositories e.g. YouTube. There is also a dedicated play-out that visualizes an exhibition on the front-end. Via the exhibition editor, users can select records from their collections and enrich them with additional description and videos to "tell a story" (see documentation in Section 3.3). Other types of collections and ways to visualize them will be supported in the future to serve the needs of different types of users, e.g. preparing a presentation for educational purposes.

Records, collections and exhibitions are saved as Mongo documents in the document-based MongoDB maintained by the WITH platform. The Jackson[17] JSON processor is used to convert the JSON representations into Java entities, and the Morphia library[18] to map them in turn into the Mongo DB and back in a typesafe way. Morphia's Query API is used to search in the database, which supports filter criteria, offset specification, and limiting of the number of results.

The platform maintains a distinction between the concept of an "external item", i.e. the original object and its metadata as it is in the external source it comes from (e.g., Europeana items, a MINT dataset, or user contributed content), and the copies of the item saved in the database as part of a specific collection. Each time an item is "collected" by a user (i.e. added to a collection), a new object of type Record is saved in the WITH database. If the item selected by the user does not already exist in the WITH database (i.e. is the result of a search on an external resource), the full metadata of the item – if available – are retrieved from the external source, and parsed to extract the WITH JSON fields. When a record referring to the same external item that already exists in the local database, then a new Record instance is created by copying the WITH JSON fields that refer to the common "external item" metadata. The record instance contains fields that associate it with the collection it belongs to and its position in it. This way, the WITH JSON contains "versions" or "copies" of the same "external item", each of which follows its own "lifecycle" from the point it becomes part of a collection: users may put extra annotations on the record object (e.g. tags, extra descriptions, attach videos to it etc), edit annotations added by other users if they have the necessary access level (e.g. collaborative management of a collection), re-arrange its position in an Exhibition etc. A richer annotation model, supporting e.g. comments by different users and relations to other similar items is designed and will be implemented in the next versions of the platform.

Published datasets can also be available through a semantic store, an industrial-strength repository following W3C recommendations for RDF serializations and the SPARQL 1.1 Query Language. EDM is the data model used to serialize datasets, and eventually collections and exhibitions, in RDF. Having reviewed and tested several solutions for RDF storage such as graph databases and triple-stores, among them *Bigdata*, *OWLIM*, *4Store, Neo4j, SHARD, Dydra* and *Sesame*, the MPU and Technical Space use the *Apache Jena* TDB store and *Fuseki* SPARQL server.

17 http://wiki.fasterxml.com/JacksonHome

18 https://mongodb.github.io/morphia/

The setup provides a high performance RDF store and a server that offers REST-style SPARQL HTTP Update, SPARQL Query, and SPARQL Update. It uses the SOH (SPARQL Over HTTP) set of scripts for working with SPARQL 1.1. The transformation between XML and semantic web serializations for datasets that are represented using RDF vocabularies are performed by the system, transparently to the user.

3.2.4 Indexing and statistics

The Technical Space implements an indexing service to facilitate querying required for services developed (e.g. metadata cleaning and enrichment) as well as for accessing the aggregated repository through the UI and API. The indexing mechanism – followed by the statistics mechanism – acts upon WITH records and WITH collections. Both structures have an internal (WITH) JSON representation as discussed in the previous section. In the case of collections the current WITH JSON consists of metadata about the collection (like title, created etc.), the first records of the collection, some application specific information like the owner's id, and the rights for this collection as set by the owner. For records, WITH JSON is built using the original metadata record (XML, JSON, JSON-LD etc) and consists of some application specific fields like collectionId, totalLikes etc, some critical fields extracted from the original item gathered from the external sources APIs (like title, provider, isShownBy etc.), the actual content of the external item (the full XML or JSON response) and finally some annotated metadata added from the user (e.g. tags). From Collection structures, all fields are indexed except the list of first records, which can be retrieved directly from the database with the same cost. For records, all the fields, except the full record of the original item, are indexed; depending on the input source and serialization, the full record can provide more information that can be indexed if necessary.

Indexed structures are searchable through the Collection Manager, the web interface for the Technical Space that is described in the next chapter. The search interface provides search based on all the above fields, full-text search and also some filtering factors. Finally, the user can search for collections and records indexed in the indices, of course taking into account his/her user rights.

WITH uses ElasticSearch[19] as its indexing and search engine. It is a solution for distributed, scalable, and highly available indexing, while offering the ability for real-time search using a large and fully customizable REST API. The combination with the MongoDB database is also straightforward and takes advantage of their scalability and high availability features. The implementation allows for searching both for a phrase and the parts that compose it. Using ElasticSearch's nested object type allows for maintaining the correlations between elements in the JSON document, such as userID and rights.

To avoid duplicate results of items belonging to different collections, or in case of items from external sources that are already collected in WITH, a structure for merging these records for indexing has been introduced while maintaining the platform specific attributes that differentiate them outside search results (such as the user that collected it, the collection they belong to, extra annotations added through the Technical Space etc.). In that way only the original item will appear in the results while the user will be able to find information about its usage in the platform and extra information that has been provided by users in the form of tags, annotations etc.

19 https://www.elastic.co/products/elasticsearch

It is envisioned that in a later release users will be able to filter user generated data according to their relationship with others (e.g. always see data from friends, trust a user-group's annotations etc. See also section 4.5 discussing management of user contributions).

The indexing engine is also used to provide informative statistics about external sources, collection details, records usage and ratings by users. They are computed using the engine's high performance features (aggregations, filters and caching) and can be available on demand. The Technical Space also records and provides statistics around its operation, such as number of users and their activity, size features of the repository, UI and API usage. The statistics package will be deployed in a later release of the platform when the data model is stable (see section 4.4)

3.2.5 Application Programming Interfaces

The WITH back-end uses a REST API to communicate with the front-end and provide a way for developers of other applications to use the data and services that the Technical Space provides. WITH is a complex tool oriented towards both users and developers. Therefore all calls, schemas and responses have been documented in detail and are provided in the WITH developers page. This documentation is encoded in the Swagger schema (detailed in *http://swagger.io/specification/*) and provided in two visualizations, the default Swagger UI and the Swagger UI responsive theme (*https://github.com/jensoleg/swagger-ui*). These pages can be accessed through links in the landing page of the Technical Space. In this section the following topics are presented:

- A brief description of REST APIs and a short discussion of their philosophy and technical specifications.
- A categorization of all available calls to the WITH back-end and the general functionality of each category. Lists of these calls are provided in their respective categories.
- A description of Swagger and an example of the visual interfaces.
- Authentication and API keys.

RESTful APIs

REST stands for Representational State Transfer. It represents a set of constraints to the design of software components that communicate over HTTP. RESTful systems like web browsers use a constrained vocabulary of expressions to connect to APIs and retrieve web resources or send data to remote servers. In a similar fashion, different web services can exchange data through APIs.

A RESTful API is defined from its base URI, the media type it uses for data communication, the HTTP methods it uses, and the hypertext links to reference states and reference-related resources. In the case of WITH, the base URI is “http://with.image.ntua.gr/” and the media type is JSON. The most usual HTTP methods are GET, PUT, POST and DELETE. These are standard verbs that operate on identifiers, like /users/login, which appear after the base URI. Along with these requests a number of parameters can be passed to facilitate data exchange.

Calls to the WITH API.

WITH offers various services for searching in external and internal sources, collecting and creating records, collections and exhibitions as well as sharing them with other users, groups, organizations or projects. All these services are accessible through its API and therefore, the calls provided are organized according to their functionality in each respective service. This deliverable does not go into detail for each call, but refer to the parts of this document which do so. A more elaborate documentation of the Technical Space's APIs can be found in D2.4 – *Access APIs*. Even more detail about each call can be found in the APIs online documentation, described in a following section, which includes any updates in the API as development continues. Appendix II lists all available calls that have been tested. Finally, it is important to note that most of these calls are used by the Technical Space's front-end, therefore the inspector console (especially the network tab) of all major modern browsers is a great source of examples of how these calls are meant to be used.

Search

Search can refer to internal or external resources. In the API two search calls are provided, a general search call that only returns an array of records from external sources, and an advanced search that also returns a list of filters to facilitate faceted search among both internal and external records. To be precise, filter properties are present in both search results in objects within the record objects of the response arrays, however in the advanced search case the filters array contains aggregated filters for all sources, instead of individual sources or records.

Records

The records category refers to calls for actions among internal records, i.e. records within the Technical Space database. Three calls using the same path are used to retrieve a record's meta-data, update a record or delete it. The record is specified by a parameter in the call path, the record's ID value. External search results are not kept in any database, however any records added to collections are stored in the database.

Collections

Calls to collections are the main way users can search the internal records of the WITH database. Since user access to collections is controlled by their owners through access rights, it is a consequence that any search performed should follow those rights. Therefore two calls are provided that list all collections available to a logged in user. List collections shows all the collections that the user has at least read access to, and list shared collections shows all collections that have been explicitly shared with the user, i.e. not public collections. These two calls can have many different filter options passed as parameters in the request, details about which is available in the API documentation page of the application.

Other calls concerning collections offer basic functionalities such as create, edit or remove a collection, add or remove a record to a collection, list all the records in a collection and list all the collections of a specified user. Collections and exhibitions are treated in a similar fashion in the database, therefore calls to collections can refer to exhibitions as well.

Users and user groups

Users refer to individual user accounts that have registered in WITH. User groups on the other hand can refer to either groups, organizations or projects. The calls to organizations or projects are, in essence, the same with calls to groups, however different paths have been provided in the API for simplicity. On the other hand, some calls that refer to common elements of users and groups, like the profile id for example, are the same, so the same path is used.

Media

WITH, along its other services, works to provide users with content. The calls for media refer to images stored in the WITH database. Calls provide functionalities for creation, access, editing meta-data and deletion of media.

Other

Other API calls exist, but mainly provide technical functionalities of the front-end. Therefore those are not covered in the API documentation, however some of them are provided here for good measure. Such calls refer to auto-complete functions, mapping static resources like files to the URL path etc.

Swagger

Swagger is a framework for representing a RESTful API through a formal specification and then using this with a set of tools it offers. It can automatically extract the API schema from many programming languages, create interactive documentations and offers many other functionalities. In the WITH case, the API schema is specified top down using the provided editor which uses the YAML language to script the definitions. This definition is then translated to a JSON structure automatically from the editor and used in two different user interfaces in the developer's page of the application.

Swagger UI

The main interface is using the Swagger UI responsive theme (Figure 3). This was chosen as the main interface as the design has been found to be more intuitive to an unfamiliar user, and also because of the good overview the left hand menu gives for the whole API options. The secondary API-LITE (Figure 4) interface is the standard Swagger UI that is provided in the swagger website. Both of those have some custom modifications, such as buttons to request an API key. The biggest functionality these interfaces offer is the option to test the calls with sample requests directly from the documentation page. However, depending on the parameters a call may require, this may need additional configuration.

An example of a simple search call is presented and its representation in the Swagger user interfaces. The following image shows the main documentation page, and the one after shows the lite version. Note that these pages offer the same information and functionalities, however the first one has a better layout for exploring the API documentation, whereas the second is faster for testing out calls.

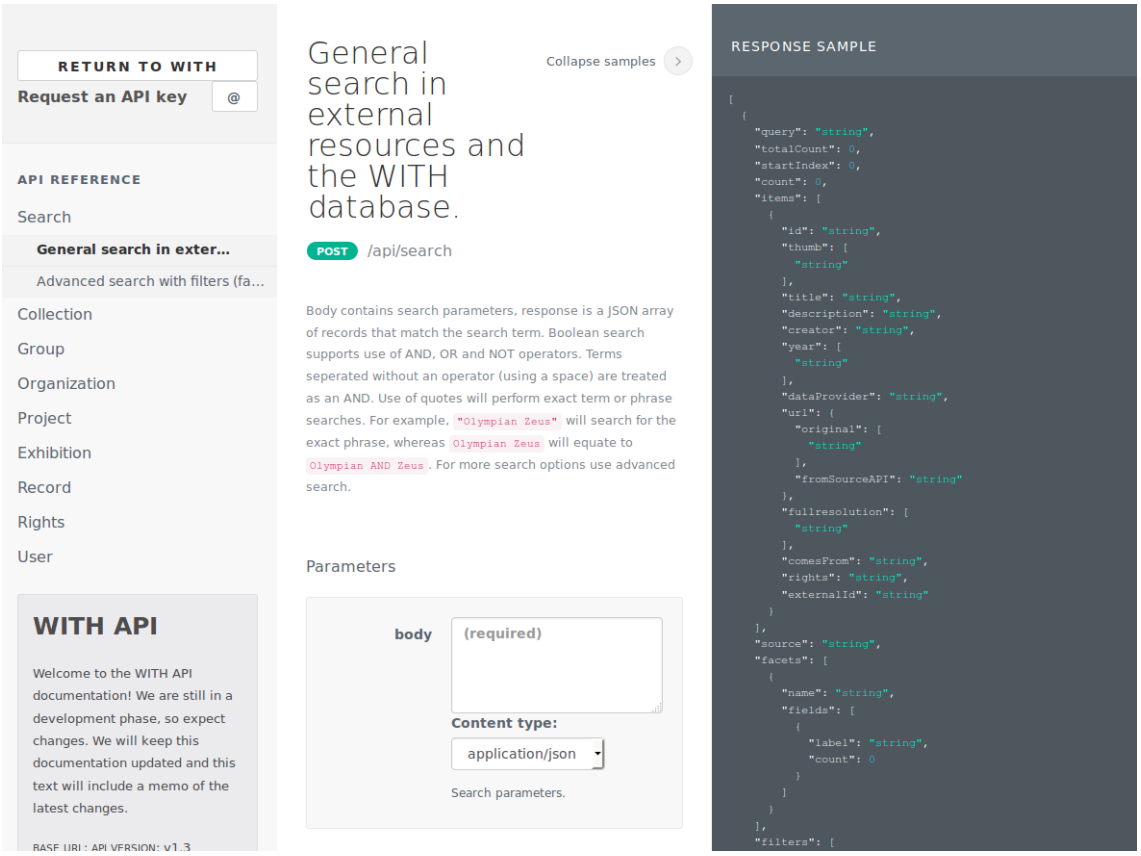


Figure 3. *API swagger documentation*

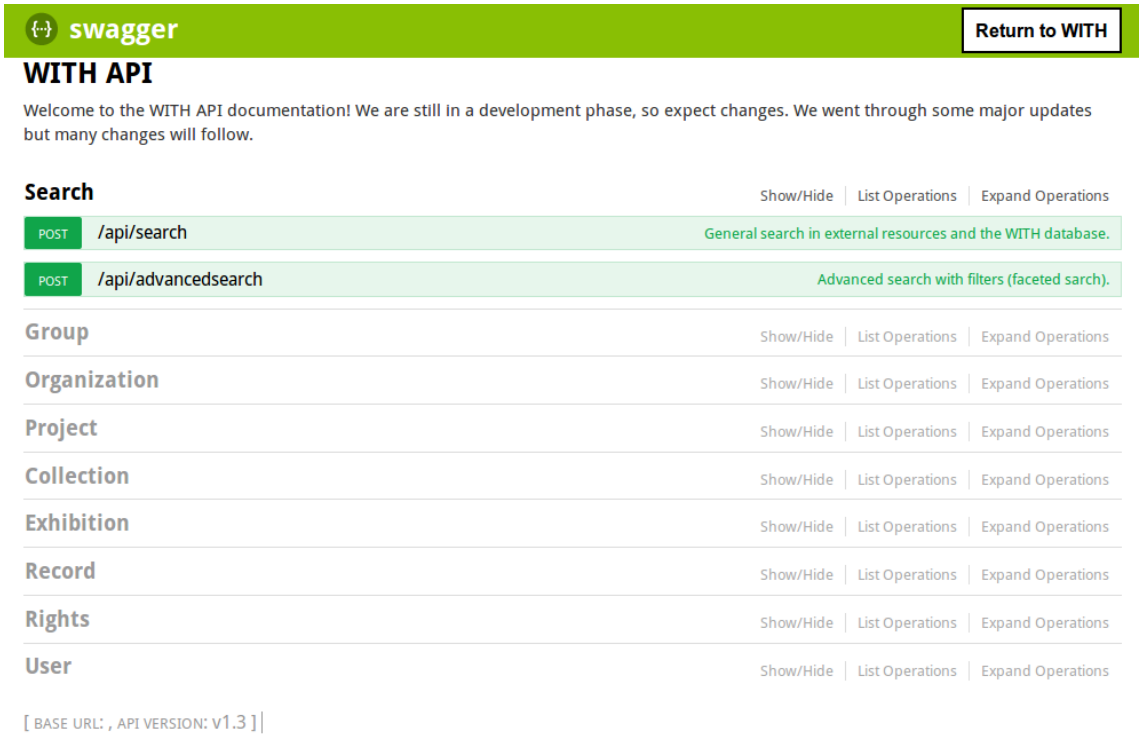


Figure 4. *API Swagger documentation – Lite view*

In the main documentation page, on the left hand side there is a menu where all API calls are sorted by category. Calls that cover two categories, for example the one that adds records to a collection, can be duplicate. By clicking on a category and selecting a call, the middle of the page displays a description of the call and all the path or header parameters it can take. By clicking on the "Show samples" link in the top right, the right hand menu (Figure 5) displays four options to show the following:

- Response sample: The JSON object of the response to this call.
- Response schema: Details about the response object and its fields.
- Body sample: The JSON body of the call, if it has one.
- Body schema: Details about the body object and its fields.

Figure 5. *Swagger "Show Samples" sidebar*

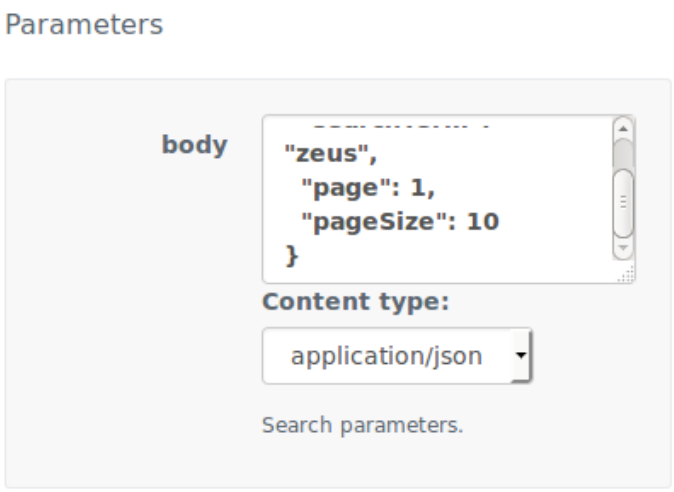


Figure 6. *Swagger parameters box for a call*

By clicking on the "Body sample" container, the parameters box (Figure 6) in the bottom middle of the page fills with this sample, as seen in the following images. After setting the values, the call can be tested and the results will show in a pop up container in the page (Figure 7). In the Lite API page this will also work, however due to the different layout all this information is shown right under the call in the same menu, the description, the models, the schema, even the test call response (Figure 8).

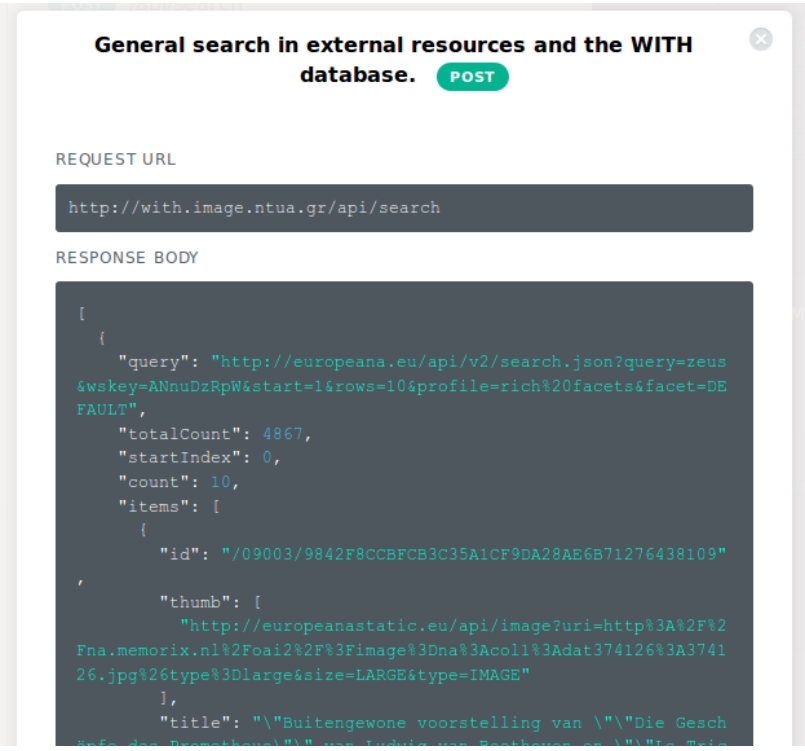


Figure 7. *Swagger API call results from test*

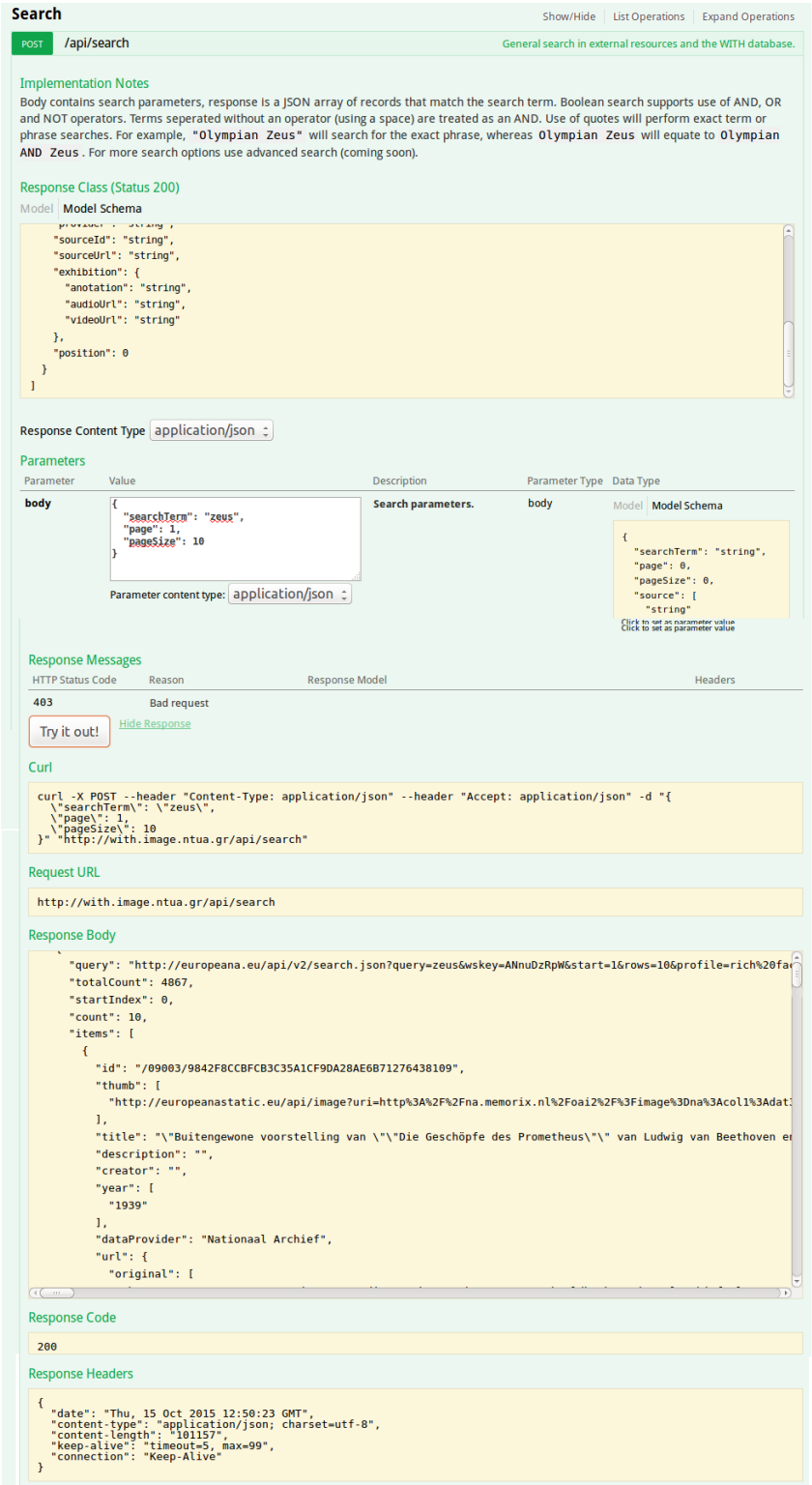


Figure 8. *Testing API calls in Swagger Lite UI*

API Keys

API keys are special parameters in the requests that authenticate the origin of the call, therefore each call to the WITH API needs an API key. Access to the API is open, so that any registered user can request a key from the primary Swagger UI developer's page. Clicking on the "Request an api key" button will result in an email being sent with the API key. If a user already has an API key but has forgotten it, they will need to send an email in order to get a new one.

API keys can be provided to the API in three ways. Firstly, they can be provided with a simple "apikey" parameter in each request. Secondly, they can be stored in a session cookie which the back-end sees and extracts from. These methods can be used by developers or for back-end service communications. Neither of these methods is very secure however, so the WITH front-end uses a different way to communicate with the back-end.

The WITH application (its UI and subsequently the Technical Space front-end), connects using an API key to the back-end. This key should not really be made available publicly, so a custom transmission scheme is employed that encodes this key and obfuscates its extraction. The back-end decodes the key, authenticates it, and then re-encodes it in a different obfuscation scheme. Developers that wish to create third party web applications to be distributed to users, will have access to certain JavaScript functions that perform this encoding, upon request and verification.

3.3 COLLECTION MANAGER

In the previous chapter the back-end for the Technical Space was presented. The Data Infrastructure together with the platform's APIs can implement all the content sourcing and delivery scenarios that were identified during requirement's analysis. A complete application can be developed using this infrastructure and its capabilities for user management, access to cultural heritage repositories, creation and manipulation of collections of cultural heritage content, metadata processing and search, retrieval and publication mechanisms.

To complete the project's infrastructure and help fulfil its objectives, a dedicated front-end is developed, implementing the methods described and offering multidisciplinary teams a full suite of tools for pilot projects development, ranging from a UI for content curators, a data infrastructure for storing, indexing and accessing content and metadata, to a set of APIs for reusing the data and services of a pilot for rapid prototyping and development of new applications. This chapter presents the Collection Manager front-end (Figure 9), deployed at http://with.image.ntua.gr/, highlighting the platform's authentication and rights management services, the UIs for content sourcing, creation and management of collections, the federated search that accesses an evolving set of cultural heritage repositories and the platform's social dimensions that promote collaboration, dissemination and appropriate re-use and repurposing of available content.

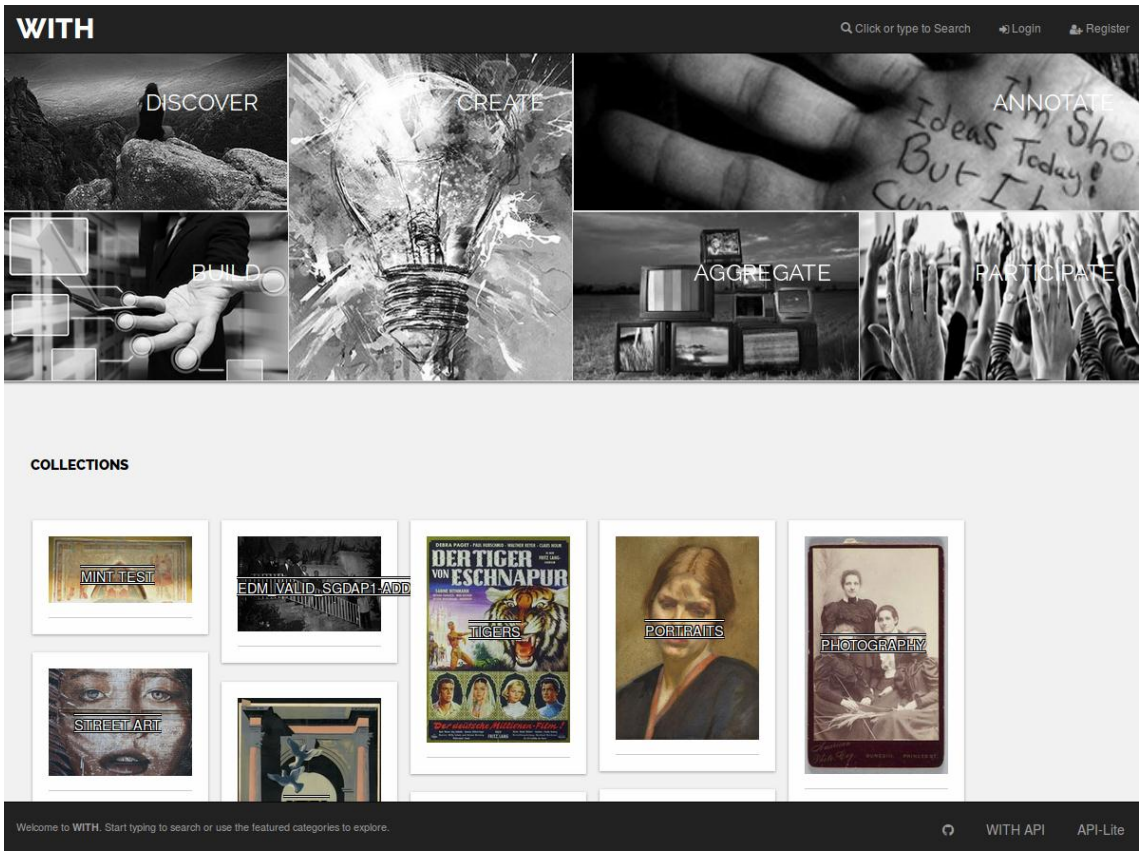


Figure 9. *Collection manager landing page*

3.3.1 Identification, Authentication, Rights management

The platform employs a typical user management system to authenticate access by identifying a user according to details of his account. It also implements user groups e.g. an organization or an interest group, for which access rights are configured and assigned collectively. The Technical Space uses access rights to control access to specific areas of the repository. This is accomplished by assigning privileges to either allow or deny access to a resource (dataset, content or service). Access rights must be able to represent IPR restrictions according to rights statements for resources. According to the developments of the project's Content and Innovation Spaces, the Technical Space is able to implement the access rights as those will be defined to allow proper re-use of content.

The Terms and Conditions Governing the use of the Technical Space are outlined within D3.2/4 – *Final Report on Legal Aspects and the Content Space.*

User Groups

Every user group contains a list of users along with some metadata for the group itself. Moreover, it contains its creator, a list of administrators and a list of parent groups. The creator has the role of "superuser" for the group and the rights to edit the metadata of the group, add/remove administrators or users of the group and even delete the group. The administrators of the group is a list of users. Their main job is to add and remove users of the group.

There are basically three type of user groups:

- Interest groups (e.g. Tango lovers): general purpose groups that do not correspond to a specific organization or project. For example a group of users that admire a specific art form (e.g. wall paintings).
- Organizations (e.g. NTUA, INA): Organizations are an extension to the basic User Groups containing more metadata (e.g. address) for their description (Figure 11). They may be museums, universities, research institutes etc.
- Projects (e.g. EuscreenXL): Projects are an extension to the basic User Groups (as Organizations) which usually have the organizations as children (Figure 10 shows the Europeana' Space project landing page).

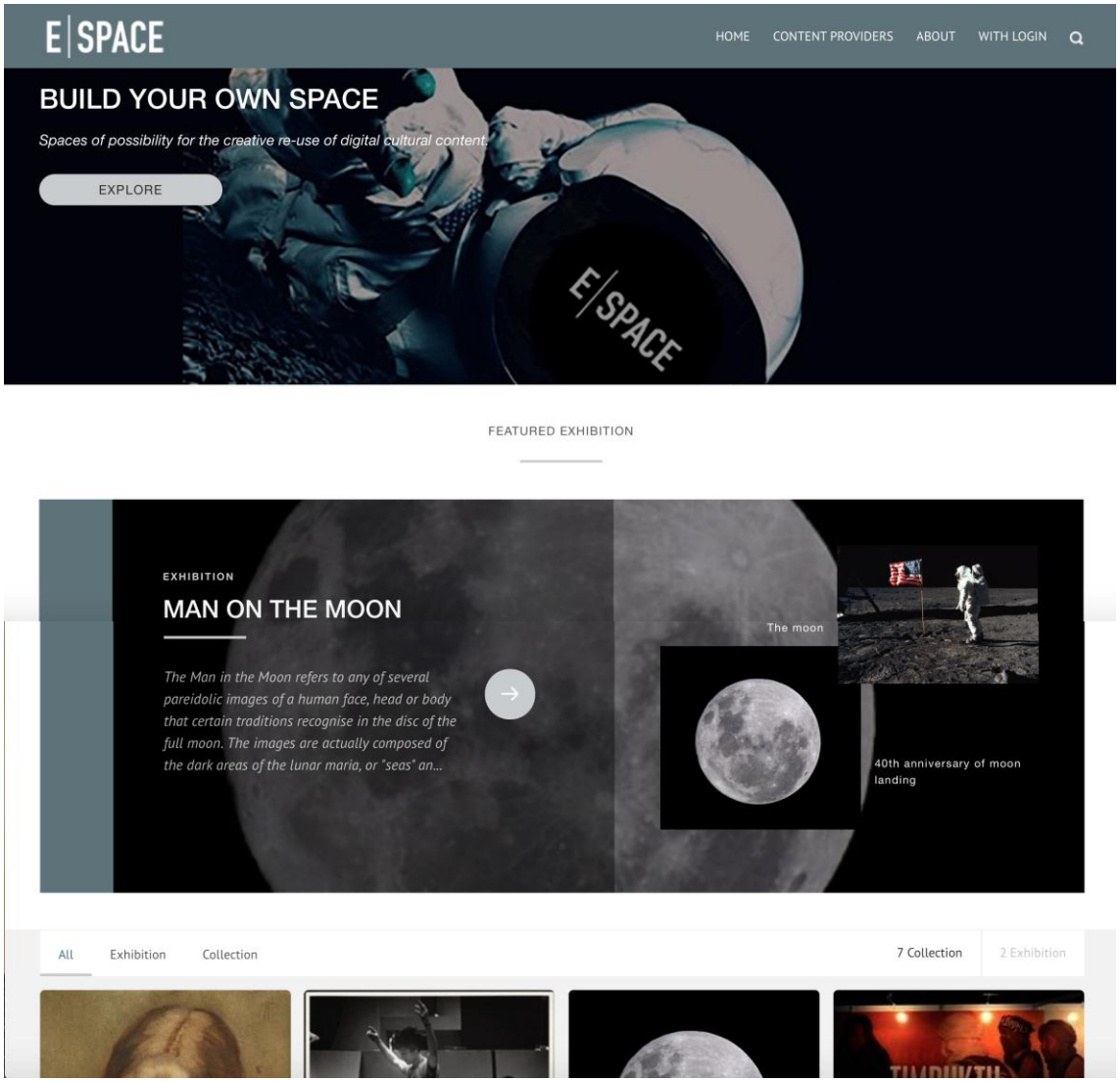


Figure 10. *The E|Space project space*

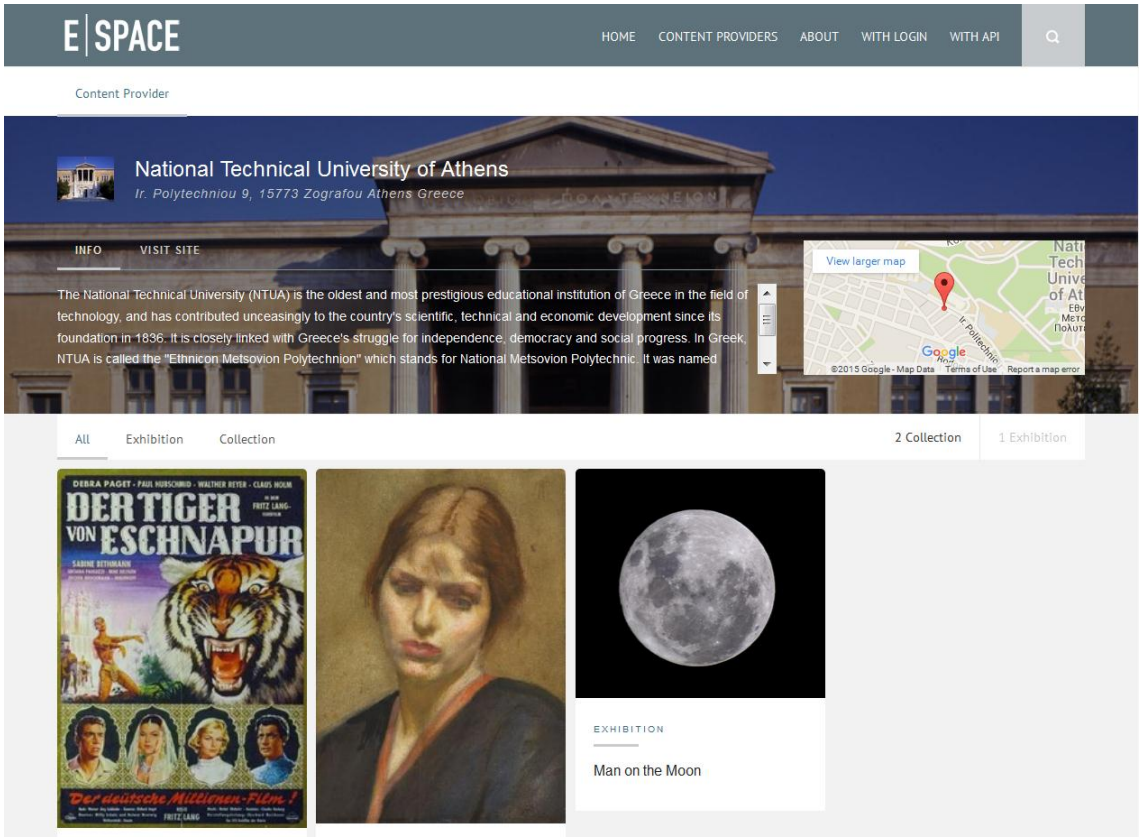


Figure 11. *User/Organization provider page*

Rights

Rights are used mostly for collection access as well as for their records. There are four basic rights that a user group or a simple user can have for a collection:

- *NONE*: The user/group has no right for the collection.
- *READ:* The user/group can only see the collection with its records.
- *WRITE:* The user/group can edit the metadata of the collection and add/remove records.
- *OWN:* The user/group can edit, delete the collection, add/remove records and give or take away the rights of a user/group regarding this collection.

There is also an option to make the collection public during its creation or editing. This means that every user or group has read right to that collection.

The rights of a group are given to the users who belong to it and they are inherited from the parent groups to their children groups. An example to make this clear: The project "EuscreenXL" is parent of the organization "NTUA" which contains the User Group "Modern Art Group". The collection "Salvador Dali" is private and the following table shows the changes of rights of all groups after their setting for one of them (it is assumed that before each case the rights are NONE for every group):

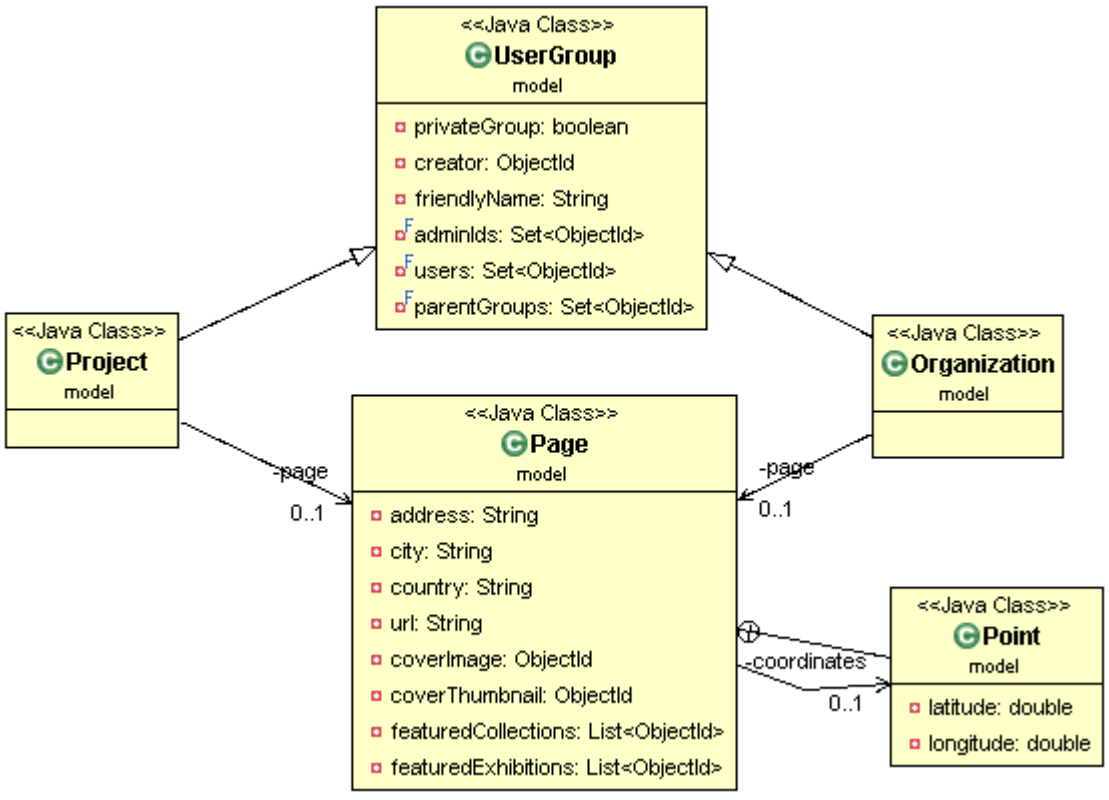
Action	"EuscreenXL"	"NTUA"	"Modern Art Group"
Give WRITE access to "EuscreenXL"	WRITE	WRITE	WRITE
Give WRITE access to "NTUA"	NONE	WRITE	WRITE
Give READ access to "Modern Art Group"	NONE	NONE	READ
1) Give WRITE access to "EuscreenXL" 2) Set NONE access to "NTUA"	WRITE WRITE	WRITE WRITE	WRITE WRITE
1) Give WRITE access to "EuscreenXL" 2) Give READ access to "NTUA" 3) Set NONE access to "EuscreenXL"	WRITE WRITE NONE	WRITE WRITE READ	WRITE WRITE READ

Model

As mentioned at the description, there are three basic entities regarding the groups, the "UserGroup", the "Organization" and the "Project". The class diagram in Figure 13 shows the fields of these entities along with their hierarchies.

Figure 12. *Class diagram for user groups*

One can see that the entities "Project" and "Organization" are an extension to "UserGroup" and both include one extra field which is the group "Page". As it is implemented now the "Project" does not differentiate from the entity "Organization" but there will be a differentiation in the future when one or more fields will be added to these entities.



3.3.2 Federated Search

The main idea behind the platform's content sourcing approach is the combination of third-party APIs for accessing several cultural heritage repositories through a unified interface (and subsequently data model). The need to consume information from many data sources is ever present today in web development and in the cultural domain specifically, with the proliferation of repositories hosting important information in the form of metadata records, thesauri, authority files, linked data resources etc. WITH implements a federated search that can configure, integrate and access various sources through their exposed APIs. The different results of the common search are processed by the system before being presented to the user. This processing corresponds mainly to two categories, translation and analysis. The former corresponds to mapping, formatting and normalization of the API responses so that they can be listed in an aggregation format, compared and re-used in a common way. The latter corresponds to comparing the different responses and information and allowing for common handling of important data such as the rights statements and licensing information.

The federated search of the Technical Space currently integrates the following cultural heritage data sources:

- Europeana
- Digital Public Library of America
- National Library of Australia
- Digital New Zealand
- Rijksmuseum
- British Library collections on Flickr Commons
- Europeana Fashion
- YouTube

Most expose metadata records and content files for cultural heritage items with a different range in terms of geographical coverage (from continental aggregators to individual institution repositories) and types of content (Image, Text, Audio and Video etc.) while others offer more specialized resources such as people, places and thesaurus concepts, or focus on exposing higher quality content.

The process of federated searching consists of:

- transforming a query and broadcasting it to a group of disparate databases or other web resources, with the appropriate syntax,
- merging the results collected from the databases,
- presenting them in a succinct and unified format with minimal duplication,
- providing a means, performed either automatically or by the user, to sort the merged result set.

It combines the potentials of the different APIs to provide a single powerful new service that gives access to a huge set of heterogeneous information (images, videos and records of different metadata schemata etc.)

A user can search using terms and phrases while receiving autocomplete suggestions from respective services of individual APIs (currently Europeana and YouTube). Results can be refined using an aggregated set of filters that are computed by processing the individual results. Most of the filters are common across the sources (e.g type of content) but also some important, source-specific ones are included (e.g for Europeana). For the former, common interpretations are adopted and the respective translations from the different data sources are implemented such as in the case of rights, where the CC approach is adopted and all license information and rights statements that fit are mapped to four main categories (Allow re-use, Allow re-use and modifications, Allow re-use for commercial, Rights Reserved) with the rest being listed individually or falling in more generic (e.g. Unknown) or specific categories (e.g. http://www.europeana.eu/rights/out-of-copyright-non-commercial/).

An example for a typical search workflow would start with a user making a query for a term e.g. 'zeus'. The autocomplete services may offer suggestions for single terms or phrases (Figure 13) and when eventually the user selects the terms and searches then a call to the advanced search API is issued.

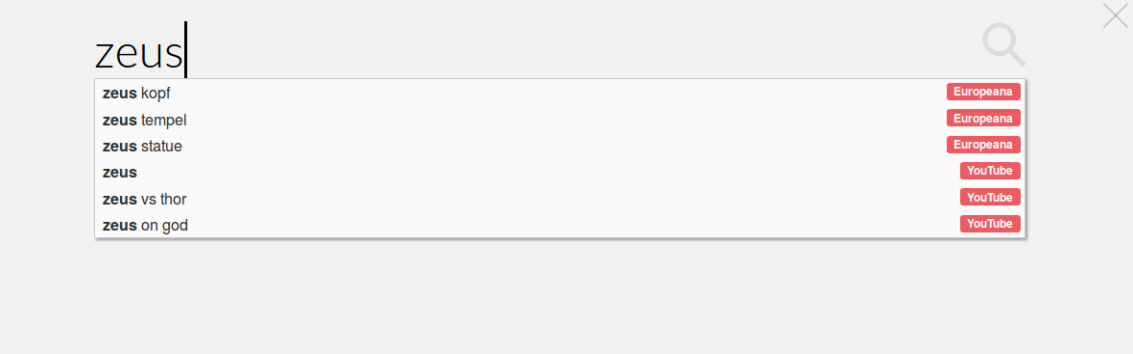


Figure 13. *Autocomplete suggestions using two engines*

The body of the call contains information regarding the sources that will be searched and the search details themselves e.g.

{"searchTerm":"dancing","page":1,"pageSize":20,"source":["Europeana","DPLA","DigitalNZ","Rijksmuseum","WITHin","The British Library"],"filters":[]}

The backend API then issues calls to external repositories, e.g.:

http://api.digitalnz.org/v3/records.json?text=%20dancing&api_key=XXXXXX&page=1&per_page=20&facets=year,creator,category,usage

http://europeana.eu/api/v2/search.json?query=%20dancing&wskey=XXXXXX&start=1&rows=20&profile=rich%20facets&facet=DEFAULT

and then processes the returned results to present them in a common interface, prevent duplicate results (the same content in different sources, e.g. an institution and an aggregator), and produce a common set of filters to apply for fine-tuning the search.

The Filters (Figure 14) currently supported are:

- Sources, that correspond to the APIs that this search will be propagated to
- Type, of content
- Provider
- Data Provider
- Content Usage, as described above

- Creator, of the cultural heritage resource, if applicable
- Country
- Year
- Contributor

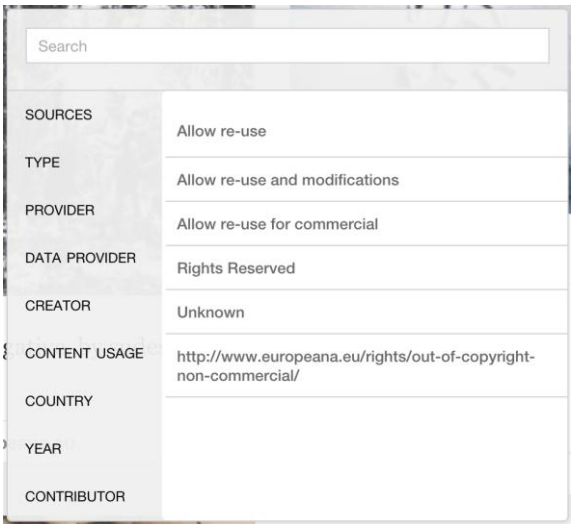


Figure 14. *Available aggregated filters*

There are two views for the results, one that lists each source individually (Figure 15) and a masonry for a combined view (Figure 16).

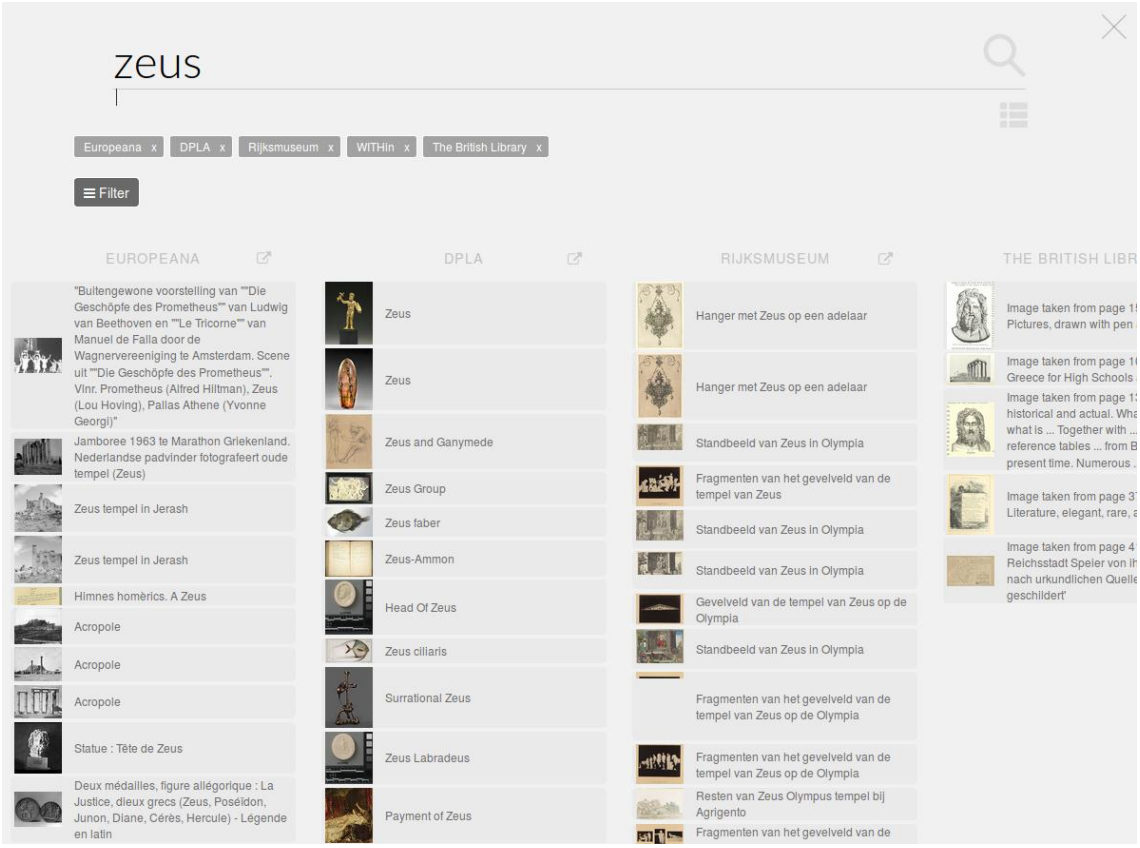


Figure 15. *Listing of results per source view*

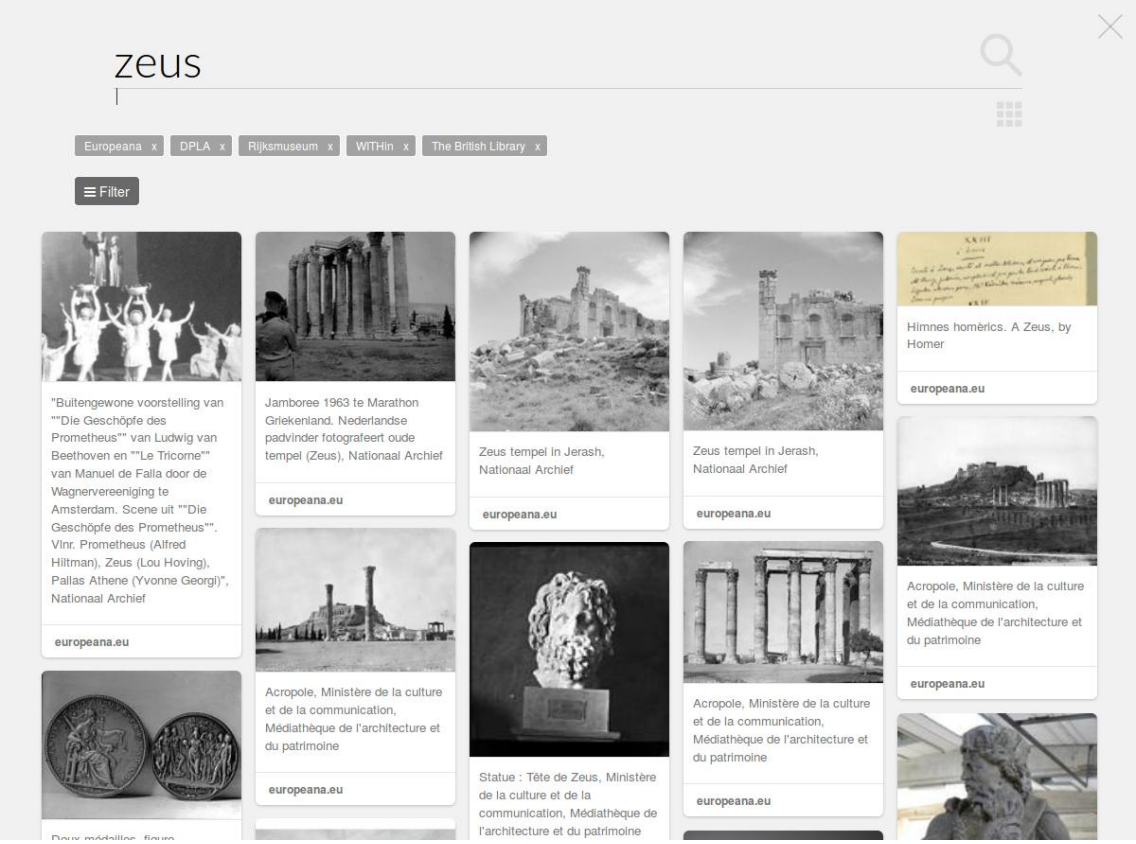


Figure 16. *Mosaic view of results*

3.3.3 User interface and visualizations

The Technical Space introduces a collection management front end to leverage the capabilities that its Data Infrastructure and APIs offer for content sourcing, editing and curation and, to set the space for collaborative creation and re-use of cultural heritage content. The user interface has been implemented as a Single Page Application (SPA), with the aim to offer a fluid and responsive app-like experience to the user, without the need for constant page reloads. The SPA experience is powered on top of the Play[20] web framework for communicating with the REST API back-end. The Play framework provides a lightweight and scalable architecture for building web applications following the model–view–controller (MVC) architectural pattern.

The model object layer consists of a set of Java classes that represent the data structures and operations on which the application operates. The controller is the intermediate layer between the model and the user interactions with the browser, and is responsible for responding to HTTP requests by invoking actions that access or modify the data model. A clear RESTful model for designing an API allows the connection between HTTP methods and the respective calls to the methods that perform the necessary actions on the backend. The view layer renders the data model into the form of user interface. The knockout[21] javascript library

20 https://www.playframework.com/

21 http://knockoutjs.com/

is used to build the interfaces that correspond to the underlying data objects, through declarative bindings. This way, whenever the data model's state changes, e.g., a new collection has been created or deleted, the UI updates automatically.

Users can register at the platform or use their Facebook and g+ accounts to log in. They have a personal space for datasets they are managing, in the different forms offered such as collections (Figure 17) and exhibitions. The landing page showcases featured collections, exhibitions and spaces together with a list of public datasets. Using the MPU, the direct upload functionality or the federated search that was described in the previous section, a user can create and manage collections, visualize and eventually share them publicly or to a specific user group or space.

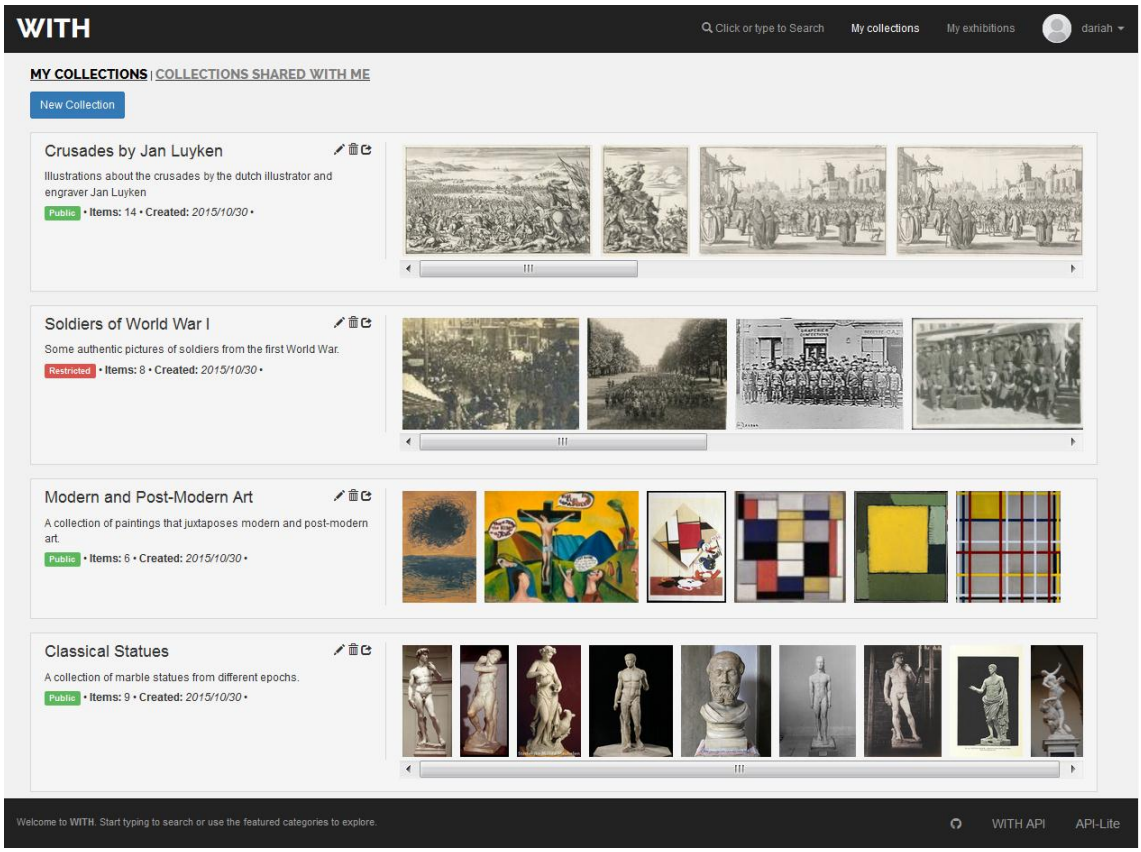


Figure 17. *My Collections screen*

Users can upload their own content and provide metadata for it using the functionality of the front-end or, for batch upload in the case of content providers, use the MPU to ingest metadata records, align them to the repository using the mapping editor, transform and publish them as collections. Such datasets can be processed further using services such as the group-edit suite of the MPU in order to repurpose or enrich them (see D2.2 for full documentation). The Data Infrastructure is the centrepiece of the platform, being in charge of storing all required versions of a dataset and exposing them through the API according to license and terms of use. The interaction with the stored content happens through the UI for normal users and the APIs for developers. A user can discover content from external sources as was illustrated in the previous section and manage it using the front-end. The results of the federated search can be browsed individually (Figure 18) and collected (Figure 19).

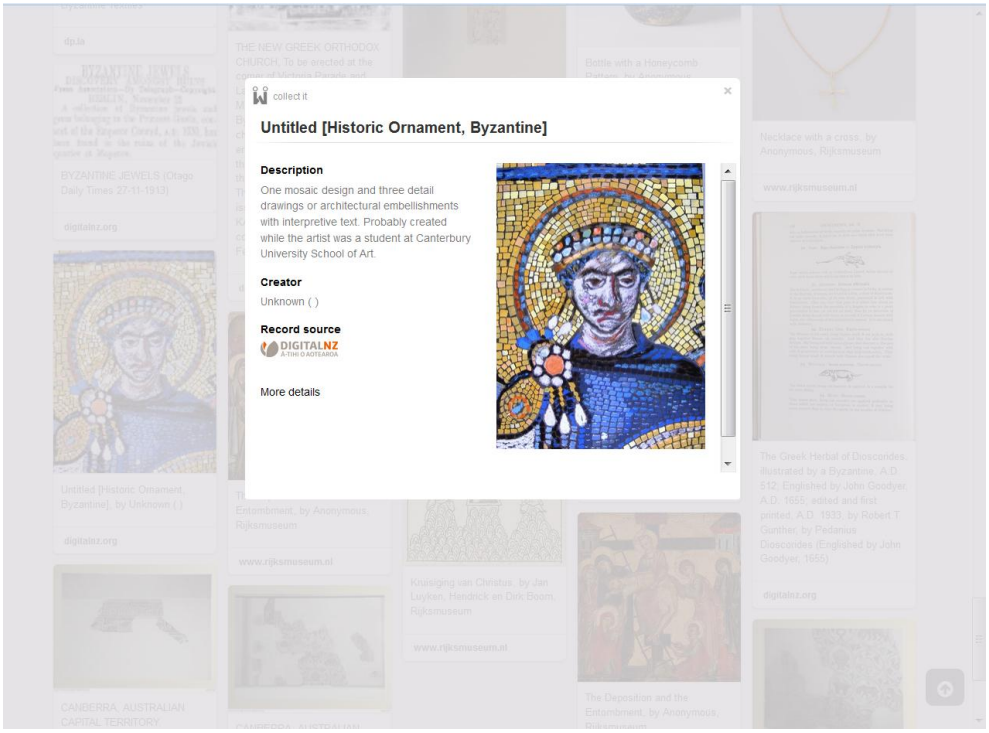


Figure 18. *Viewing details for a resource from an external repository*

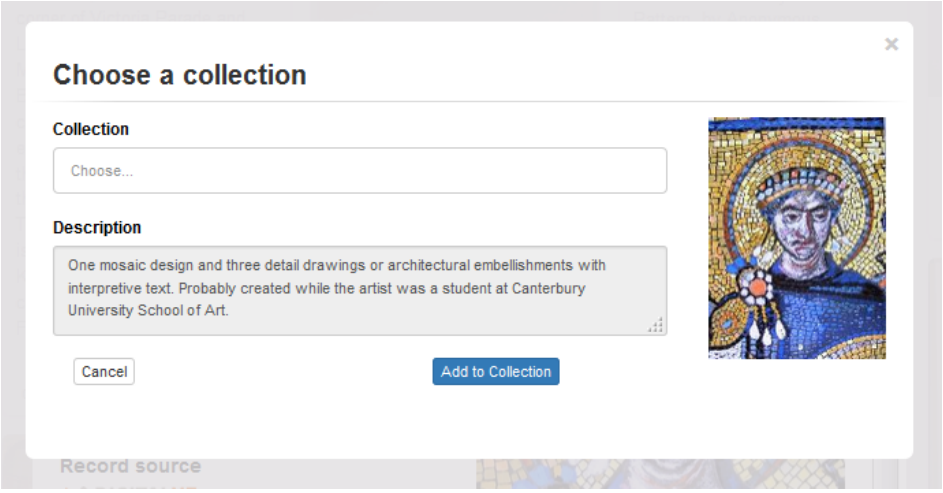


Figure 19. *Collecting a resource from an external repository*

Other visualization and organization options for datasets are being explored that help contextualize a dataset and visualize it with the purpose of sharing a personal experience around the content that consists it e.g. telling a story. The exhibitions UI is a step towards this direction, allowing users to add information (captions, descriptions and embedded multimedia) for each resource in a collection (Figure 21) and use its play out method to present it in a more engaging manner (Figure 22). Similarly organization and visualization schemes such as 3D galleries, user or content provider pages and spaces are being further investigated.

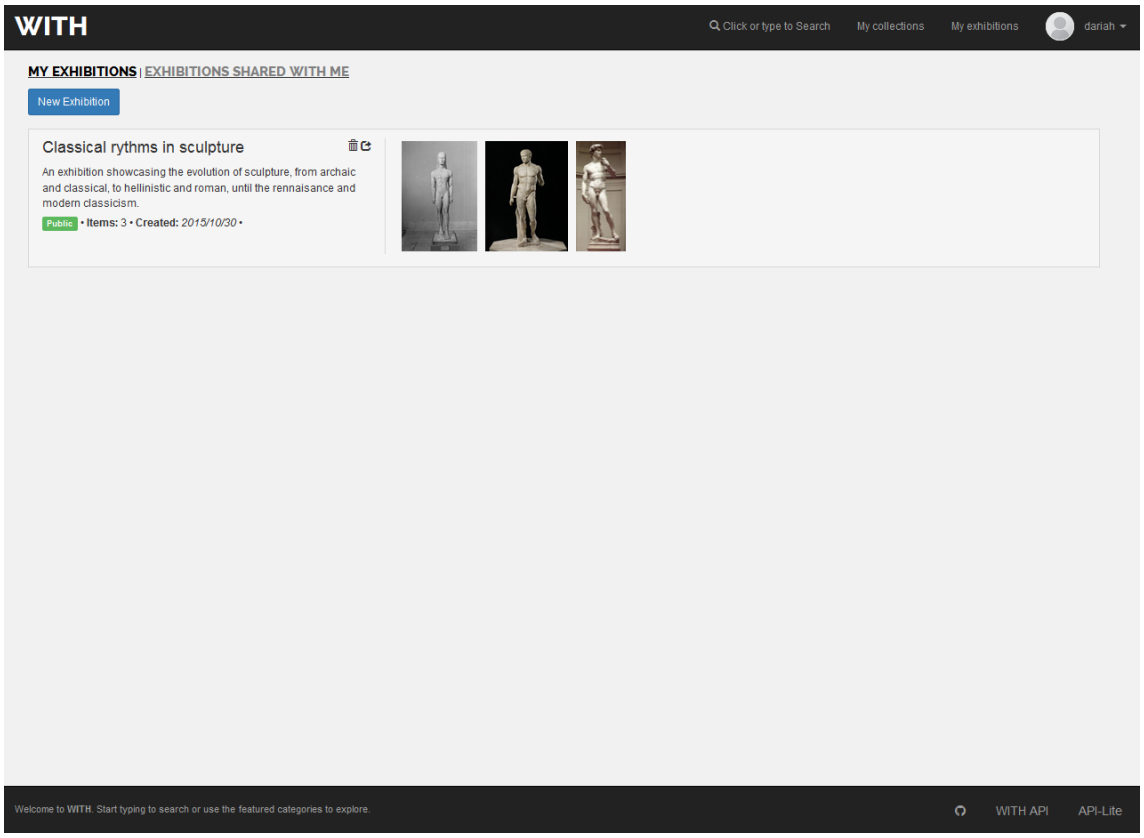


Figure 20. *My Exhibitions screen*

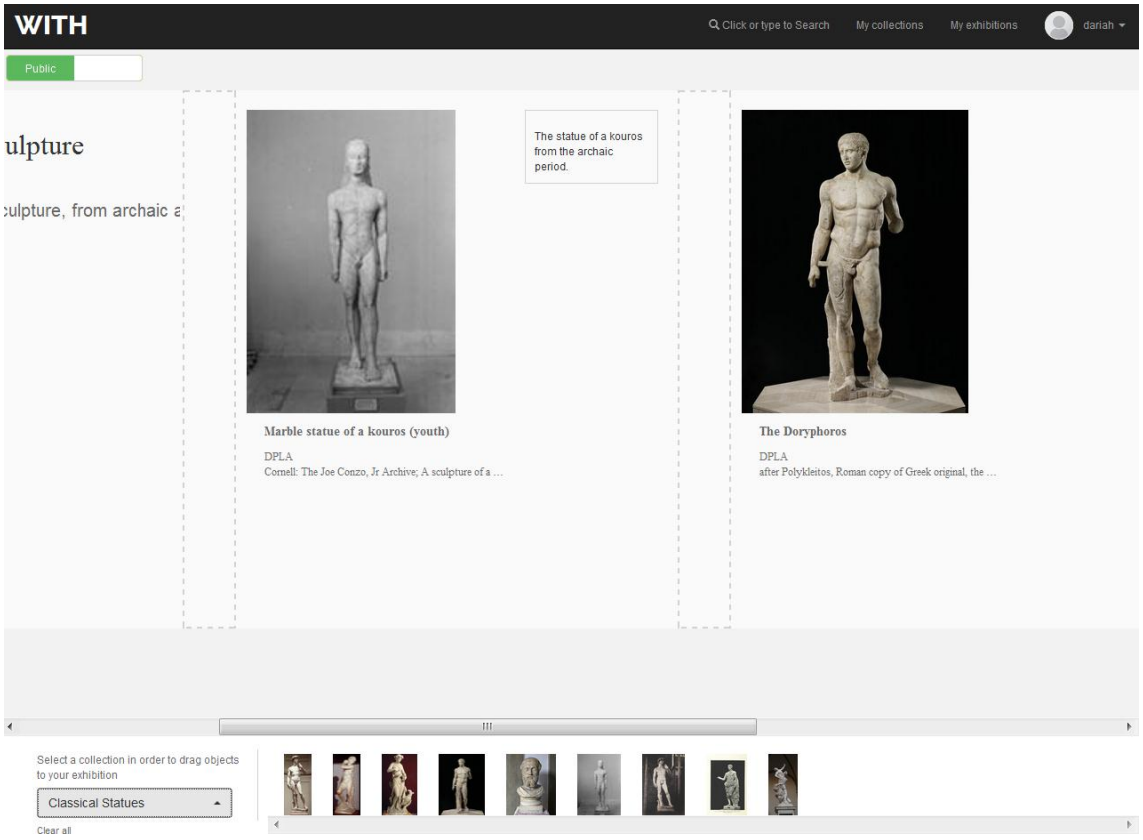


Figure 21. *Exhibition editor*

Spaces is a concept that was introduced with user groups and, in practice, corresponds to specific, access-based views of stored data. A user group has access to collections shared by its members and can be moderated. The Technical Space offers the ability to visualize user groups (e.g. see Figure 10 for projects) individually and allows customization of the front end (descriptive texts, images, CSS). The content (collections and exhibitions etc.) is limited to those that the user group can access and the scope of the search engine can be customized as well, for example to exclude some sources or, to only search for video resources etc (Figures 23 & 24).

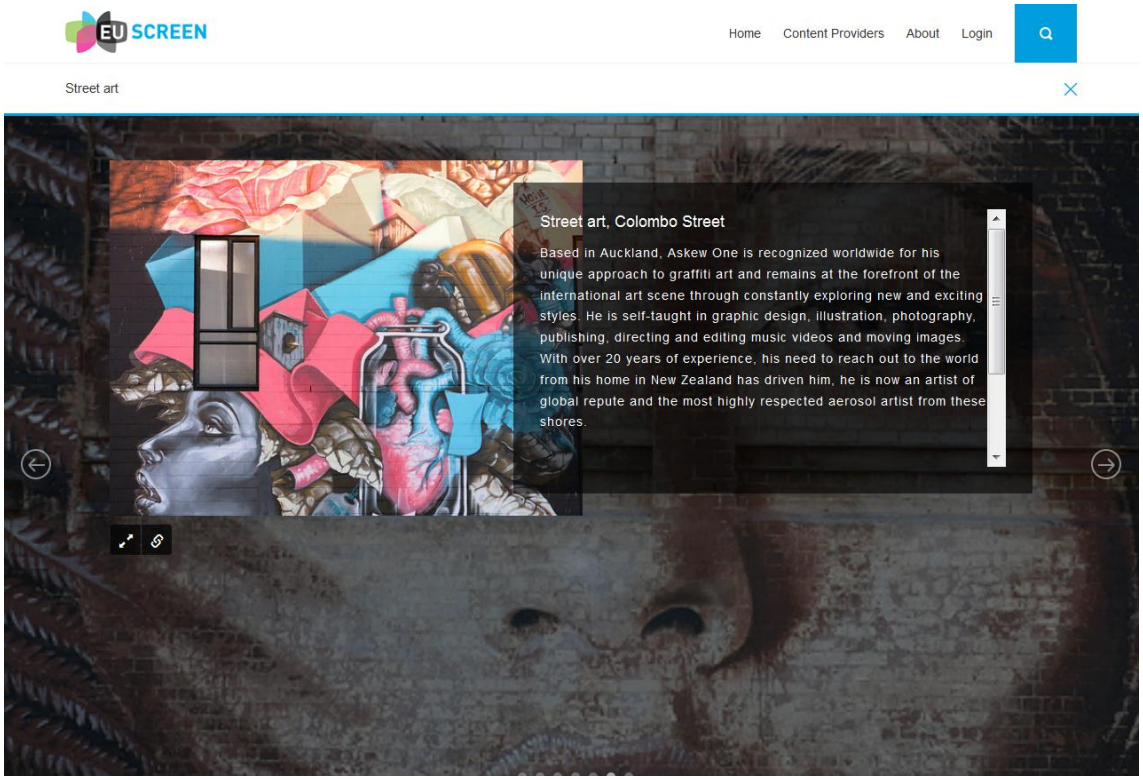


Figure 22. *An exhibition slide in the play out*

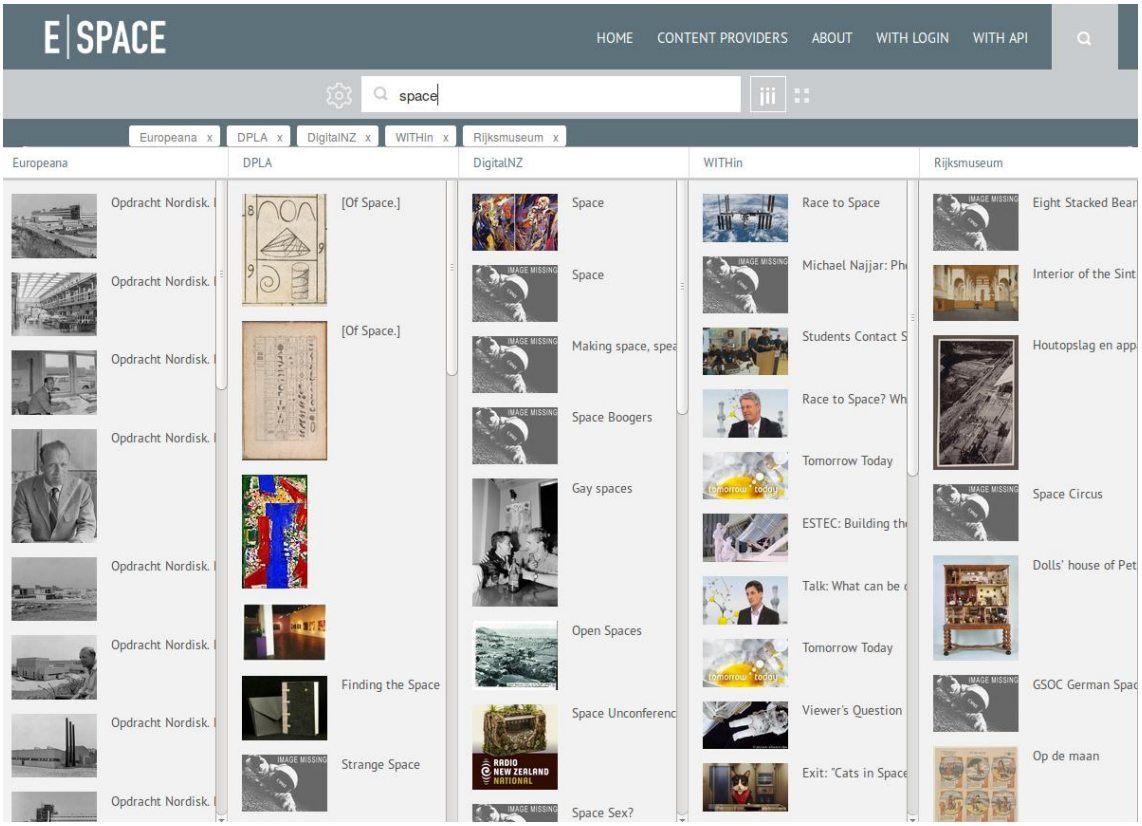


Figure 23. *The search page for a custom space*

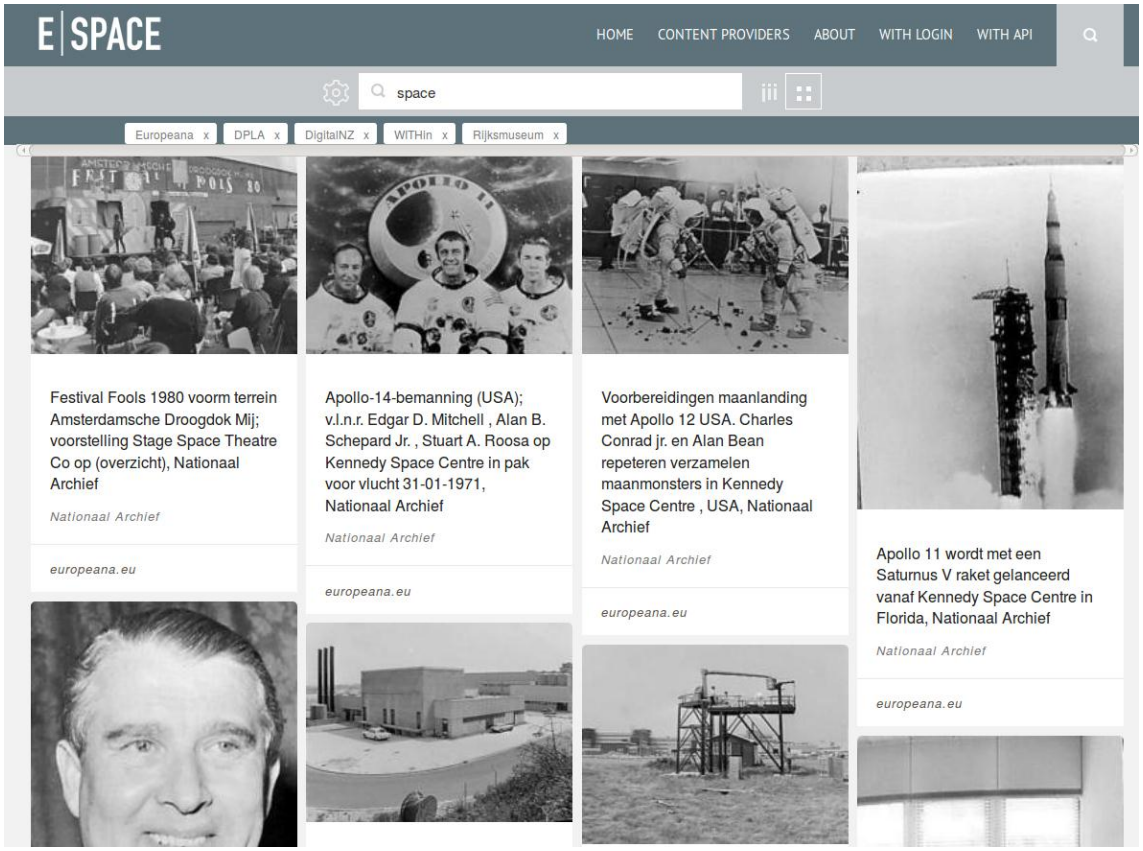


Figure 24. *Mosaic view of the results of federated search for the E|Space space*

3.3.4 Social Dimension

There are several features that introduce and reinforce the social dimension of the platform, focusing on collaborative creation and offering the ability to easily share and disseminate collections and their updates. A user can make a collection public for other users of the platform or join and share it with a user group, as those were defined in the previous chapter (Figure 25). A basic rating mechanism in the form of favourites/likes for platform resources (items, collections and user groups etc.) is also being tested to provide the basis for a ranking engine that will improve search results and allow the search engine to adapt to user preferences. Finally, a user can join or follow user group or project spaces and other users, with a notification mechanism (Figure 26) that informs her of updates, pending requests or invitations.

3.3.5 Workflows

The Technical Space is addressing the needs of a wide spectrum of users that engage with it in different scenarios. It can be used by the public, domain scholars or content providers, being members of a network, a project, or participating in events like editathons or hackathons. Depending on the type of user and the intended usage there are several workflows that can be followed for populating the Technical Space with content, creating and managing collections, and consuming the produced work. Following is a list of the typical workflows that are executed by different users in the framework of the project, with brief descriptions:

- Create and manage a collection; a user can register (section 3.3.1), use the federated search to collect material (3.3.2), edit the collection or create an exhibition (3.3.3) and promote or share it (3.3.4).
- Publish content and metadata (see D2.2); a provider can upload content (3.2.1), ingest, map, transform and edit metadata, export the dataset as a collection in the Technical Space or publish to Europeana (3.2.3).
- Develop using the back-end and APIs; using the APIs' documentation a third party developer can issue an API key and use it to access data and services available in the Technical Space (3.2.5 and D2.4) from an application.
- Set up and manage a user group; a user can create a group, connect it with a dedicated space, customize the latter and create dedicated pages for its members (3.3.3)
- Prepare and support a hackathon; Pilot teams create the event's space (previous workflow, 3.3.3), create and add collections to it (3.3.4), invite users (3.3.4), set up dedicated access scenarios with corresponding terms of use and content licenses (see 4.1) and use the space to inform, guide and assist developers in creating applications that properly re-use the available CH material.

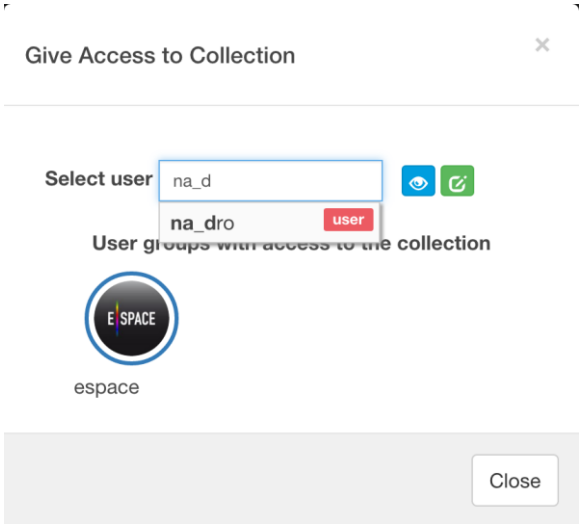


Figure 25. *Sharing a collection with a user or group*

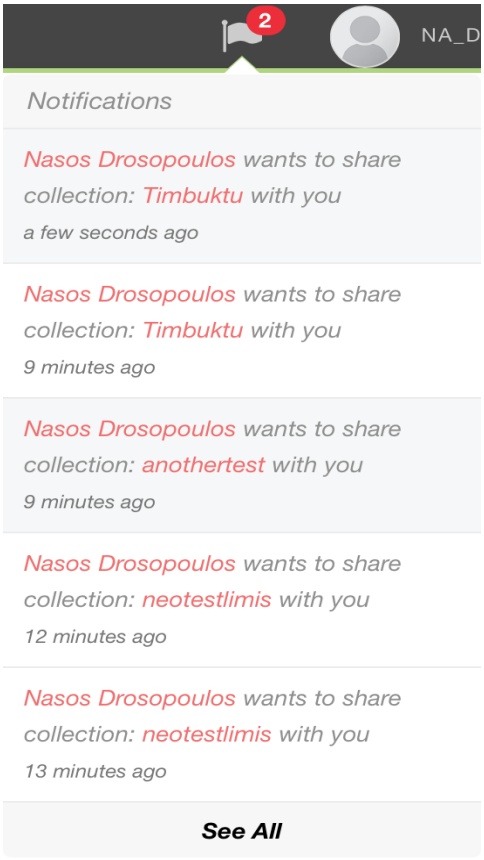


Figure 26. *Notifications*

4 ONGOING AND FUTURE DEVELOPMENT

4.1 AGILE DEVELOPMENT AND USER-CENTRED DESIGN

Development of the Technical Space follows the Agile philosophy, mainly using the Scrum process, and injecting it with Extreme Programming (XP) iterations when there is a need for faster development cycles with strict prioritization. The need for a platform for creative re-use of CH material has been identified and analysed extensively throughout the long history of research and development in the domain – and especially in the process of developing and sustaining Europeana, with the participation of both stakeholders and end-users. NTUA participated in Europeana Creative, a project that paved the way towards creative re-use of CH, where an extensive user-based research took place to involve end-users in the design process.

The target audience of the Technical Space includes development teams, e.g. pilots using it as a backend for their application, and end-users that may simply experience the platform's functionality or use it specifically for the creation of content that they will in turn consume using another application (e.g. create collection and load it in a game). The different types of usage and the continuous interaction of the development team with both users and 'customers' enabled the incorporation of user-centred design principles in the development process. Developers take active part in the design phase of the different modules of the platform, starting from the basic content sourcing functionality and access APIs that were addressing pilot and third-party development teams, up to the ideation of later features targeting end-users, such as visualizations and social networking. This allows for better planning of development sprints and ensures the technical viability of the design given the restrictions in time and funding that an EC-funded project imposes. In turn, those participating in the design process also assist the development process, reviewing and verifying the software against the design and even contributing tasks to the development backlog.

In the framework of Task 2.1 – *Requirements for the creative use of Europeana Cultural Resources* - a series of workshops took place (two dedicated events in Brussels – video recording available[22] – and Florence, and two during the General Assembly meetings of the project in Venice and Coventry) in order to define and analyse requirements for the Technical Space. Designs and prototypes of the platform were evaluated and used to identify the functionality and workflows required for the pilot's hackathon events. For the first two hackathons the E|Space usergroup was used to collect and visualize appropriate material that was prepared for participants in order to use the API for re-using cultural heritage material in their applications. It was then decided to implement separate sub-spaces, with their own visual identity and sets of collections, for each of the pilots that correspond to the six thematic areas of cultural heritage.

The capability of the Technical Space to implement the different access scenarios that are described in the project's Content Space, such as the availability of content for limited time period, will be evaluated during the remaining hackathons. Such scenarios can be performed with the use of user-groups, where content providers can share datasets with a specific set of users and enforce a specialized set of rules regarding content use and re-use.

22 http://www.europeana-space.eu/thematic-workshops/23-24-march-2015-europeana-space-technical-workshop/

A typical example is a hackathon user-group, where providers share content with a license that allows re-use only for the duration of the event; users adhere to the terms of use drafted specifically for the user-group, and they can proceed to experiment and prototype with high quality content that would otherwise be unavailable to them in order to reinforce their applications and business models. This in turn helps the content provider to fully understand the functionality and possibilities of an application before proceeding to the rights clearance processes required.

The requirements set by WP4 and WP5, and the actual usage of the platform and APIs during events like the hackathons are also used to inform the development process. A dedicated tracking system[23] has been setup at http://bugzilla.with.image.ntua.gr/ where bugs and feature requests are reported and the process and actions to resolve them are recorded. Performance monitoring modules are implemented to collect data about server usage, availability and responsiveness. This will become more important as more users start engaging with the platform and more content is added, in order to make sure that the platform can perform as expected in a scalable manner.

Finally, the platform is being used by several projects in the domain where NTUA participates in order to test its applicability in other use cases and environments. One example is EUScreen[24], an audiovisual aggregator project that created its own space using the platform. It is used to showcase important collections published by its content providers as well as to implement a different way of browsing and visualizing the project's content. Customization of space and the user/content provider pages were of particular interest in this scenario that has driven the development of these parts of the platform. In another example, Europeana Space and Preforma[25] have recently started collaborating in the integration of the latter's conformance checkers for media files in the Technical Space. That will provide users with additional search options based on the information extracted, ensure that the files stored in the Technical Space are in a format that can be accessed in the future and allow to establish and conforms with specific policies that are set by Europeana Space content providers.

23 https://www.bugzilla.org/

24 http://with.image.ntua.gr/custom/euspace/index.html

25 www.preforma-project.eu

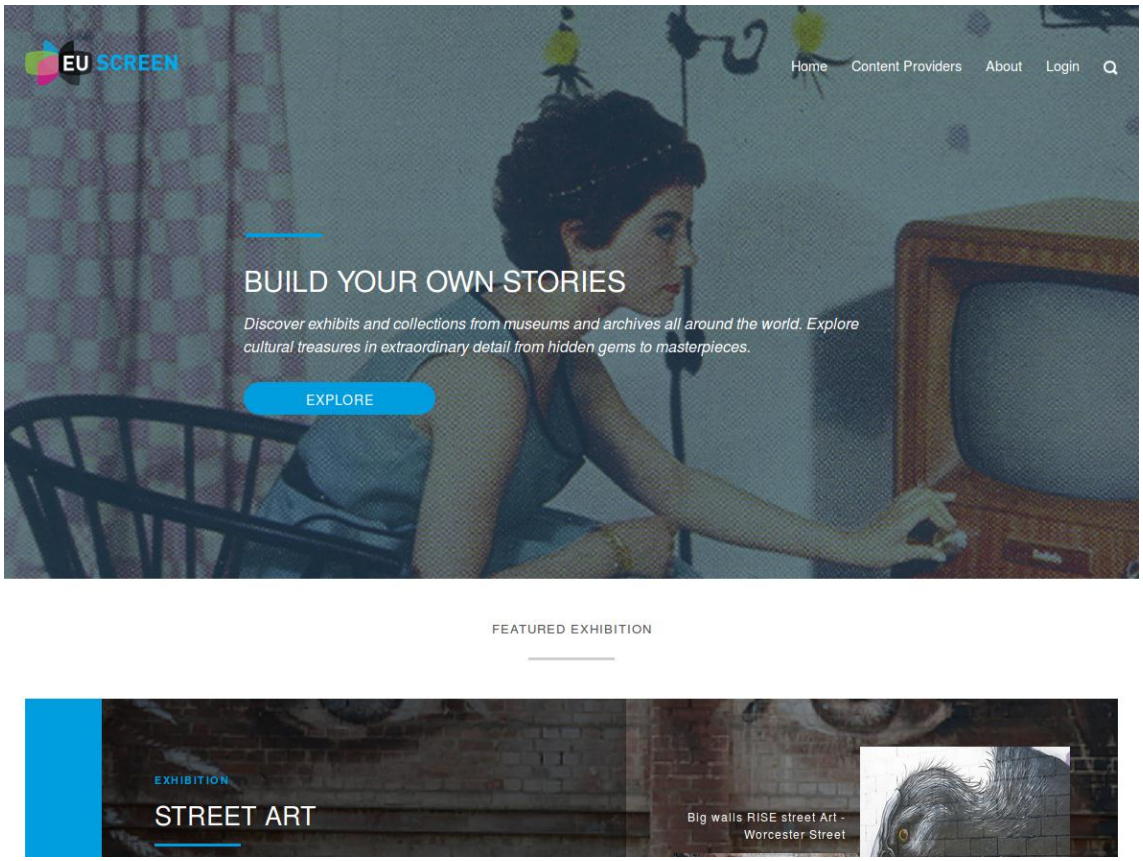


Figure 27. *The EUScreen space landing page*

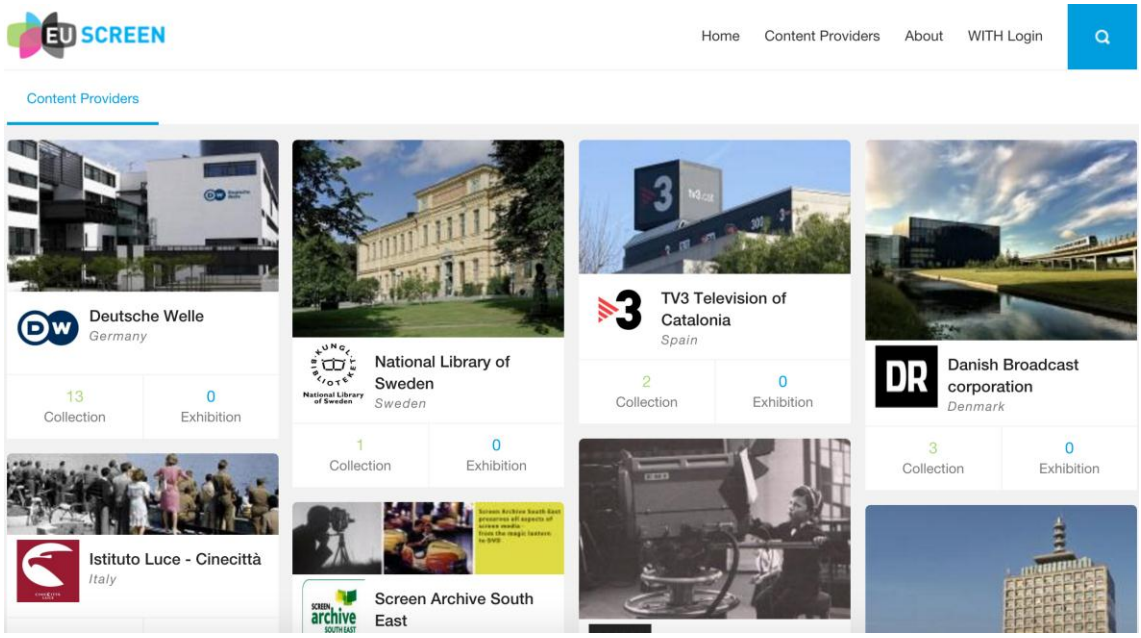


Figure 28. *List of content providers in the EUScreen space*

4.2 NEW DESIGN FOR THE USER INTERFACE

The interaction with the different usage scenarios and events is also guiding the evolution of the user interface, from prototyping to a full blown design. The team is in the process of deploying a new user interface that was designed taking into account the results and feedback from the processes described in the previous section, project events and the issues reported in the tracking system. The introduction of a dashboard (Figure 29) for logged in users is the major modification, together with a lot of aesthetic changes (CSS, reorganization of screens etc.) This will finalize the first stable release of the system that will be used for the rest of the hackathons and introduced to the public.

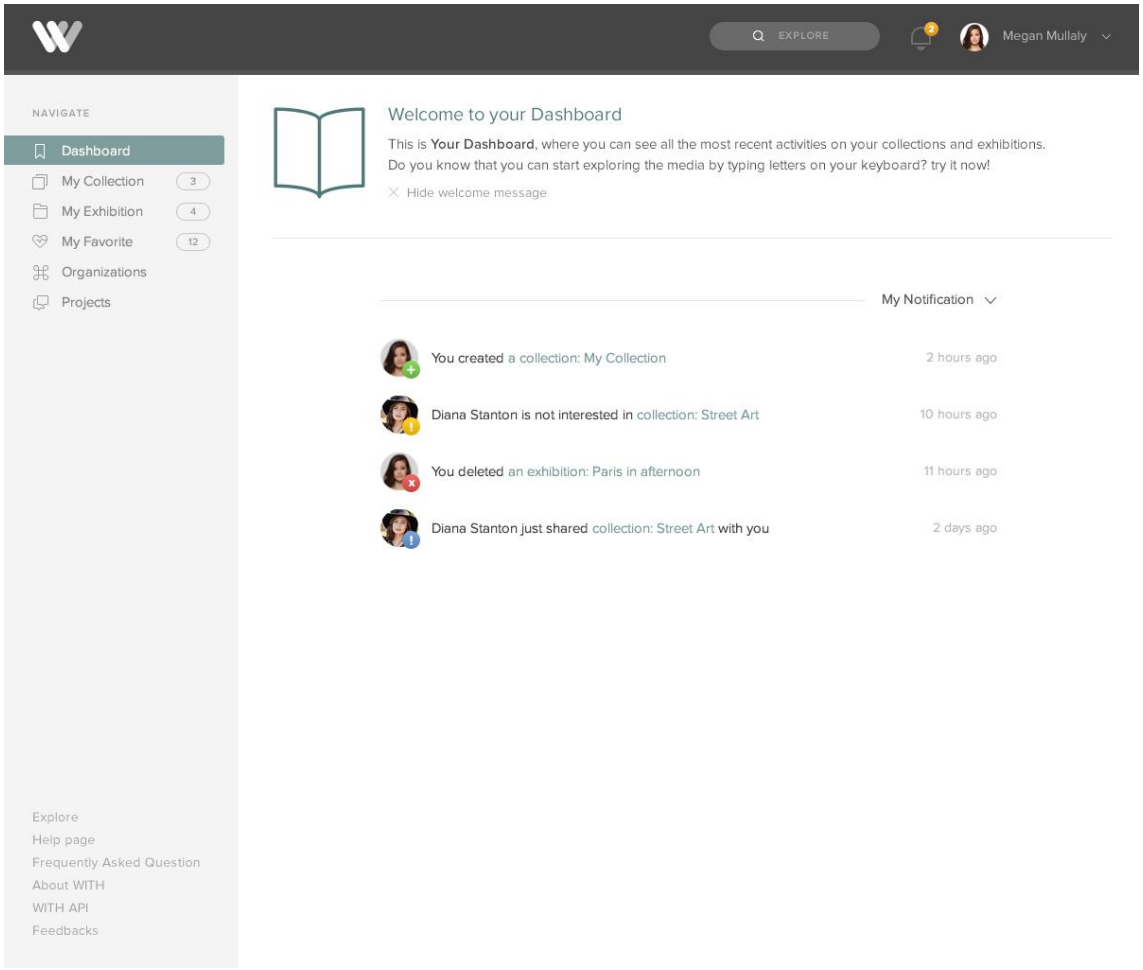


Figure 29. *Design and prototype of the platform's dashboard*

4.3 WITH DATA MODEL REVISITED

One of the parts of the platform that has evolved the most is the data model for representing collected items from external repositories. The architectural choices for the platform specifically allowed for an evolving data model, as the modelling activities in the cultural heritage domain are numerous, with new schemas frequently introduced, existing standards updated, thesauri and authority files evolving and, the level and expressivity of their semantics being a parameter that is still evaluated and fine-tuned. The establishment of interoperability between the Data Infrastructure and the external sources, through the implementation of formal mappings between them, allows for updating the data model and extracting any additional information or updating the crosswalk to correspond to the improved alignment. In that way the platform's aggregation data model can continue developing without interrupting or disrupting the platform's usage.

The currently process involves implementing a new data model for the platform's next iteration, which will allow big step towards a more powerful and expressive system. There are several updates that focus on enabling better ways of combining sources and applying common filters and criteria when searching for cultural heritage material in the evolving set of external sources. Only one of the main aspects of this update is described here, the decision to collect resources instead of what is traditionally termed as 'items'. Currently, Europeana, DPLA and such aggregation repositories, collect and expose metadata records about cultural heritage objects or items.

In the record CH objects are connected to other resources such as the media files that depict them and the Agents (people or organizations), Places, Timespans or Concepts that appear in their documentation. There are also several very important sources that only expose those resources not in connection with an item, such as thesauri of concepts, datasets describing famous artists, geographical databases and so on. The new data model allows for searching and collecting all kinds of resources instead of only CH objects, giving the ability to create collections that are not just sets of records but a group of resources that are semantically linked in an expressive way that can provide an advantage to the applications that consume them. In that sense a user that is interested, for example in Italian Renaissance painting, can collect famous paintings (items from CH aggregators) together with resources that represent the painters themselves (e.g. DBpedia topics or VIAF authority files), the locations where they were working or exhibiting (e.g. GeoNames entries), a relevant time period (http://dbpedia.org/page/Category:Renaissance), a set of concepts that describe the period or movement and so on.

This extension can take more advantage of the semantic store and engine of the Data Infrastructure – both to perform reasoning and to expose more expressive data through its endpoint – and serve as a basis for development of applications that aim at being part of the semantic web and/or linked data cloud. It also strengthens the impact of the Technical Space as a platform for re-organizing, re-purposing and re-using cultural heritage by moving away from the notion of item repositories towards a network of connected resources. In the same sense the data model will also implement a structure for recording events (combinations of objects, agents, places and timespans, as defined by CIDOC-CRM and used by the LIDO schema) in order to properly represent and visualize information about cultural heritage resources.

4.4 EXTENDING THE MPU LINKING SERVICES, INTRODUCING ANNOTATIONS

Following the developments and discussions of the previous section is the natural evolution of the linking services provided by the MPU and the user annotation services to be introduced in the Technical Space. Although one can already edit, append and enrich metadata collected or uploaded in the platform, these are tasks that are meant to be performed 'officially' by the owner of the content. When other users modify content then there is no structured approach for this information to be recorded, visible and usable by any other than the user himself, or in the case where simple edits may be allowed (e.g. user free-text or controlled tagging) it is hard for a user to select, configure and evaluate the information presented.

Using W3C's Web Annotation Data Model the platform will allow users to add or edit metadata for resources using free text, controlled vocabularies and linked data sources, to comment and rate existing information, to create more links between resources or describe existing ones using more expressive vocabularies, and eventually share those annotations in an environment that promotes the evolution of knowledge in a formal, machine understandable way. Adding provenance information for all the actions described here enables the evolution of the repository towards allowing different views of its data based on trust, i.e. a user will see original metadata and content for a cultural heritage resource together with annotations from other users whose contributions he decided to trust.

5 CONCLUSION

The Technical infrastructure plays an important role within the project, as a platform for storing, accessing and processing content and metadata. It is aligned with the requirements of pilot products and applications based upon regular project dialogue and has a key function to play within the hackathons. More than that, it is a platform that is built with knowledge of the Europeana ecosystem, learning from what has gone before and complementing other ongoing initiatives. It will also be available for the project target groups comprising cultural institutions, creative industries and members of the public to develop collections and/or for the creative re-use of content, as well as provide content for Europeana.

A lot of planning took place to identify the requirements of aggregation, curation and metadata interoperability, within a platform that can also operate as the back-end for future applications. It is true that the project's pilots could have benefitted from earlier access to the WITH platform, but it was important to ensure its functionality before rolling it out to minimise disruption. (Work also started later than originally planned due to the assumption that Europeana Labs might fulfil this project requirement.) Pilots instantly benefitted from an API that draws upon multiple cultural heritage data repositories, which is a positive indication that it will also be welcomed by members of the project's target groups.

The front-end collection management system provides a user friendly interface to help target users discover, aggregate, create, annotate, participate and build collections. Within this, a user management system authenticates access to defined user access rights; a feature that is important to the project's innovative trial of a protected space, with content that is only available for a particular time period, such as during a hackathon.

Following two years of design, analysis and development, a technical platform with user-orientated front and back-end operability is available which will provide great opportunities for cultural institutions, creative industries and members of the public to build collection and re-use content, as well to develop further innovative applications that can draw upon the wide range of source content and metadata available.

APPENDIX I – RECORD WITH JSON FIELDS

The Record WITH JSON data model is described in the following table. Fields such as title, description, creator, provider, itemRights are extracted from the original item metadata (content) based on the custom mapper implemented for the respective format and schema (e.g. JSONLD_EDM and JSON_EDM which are the formats returned by Europeana) etc. The mapping may also involve grouping of possible values which refer to the same concept in different schemata, e.g., “Sound” and “Audio” in the field indicating the item type are both mapped to “Sound”, itemRights values, which are often urls, are mapped to one of the enumerated values expected by the WITH JSON model etc. Fields such as tags, and the embedded exhibitionRecord object are filled in by user-provided data. The collection a record belongs to, its position, the number of times it has been liked are also fields whose values are calculated based on the user's behaviour. There is also a list of fields whose values are assigned and controlled automatically by the WITH system, e.g., the dbId assigned by the WITH database, the creation date etc.

The access level rights dictating what kind of actions a user is allowed to perform on an object on the WITH database (read, edit, or delete) are attached at the level of the collections (see section about rights management). The rights associated with a record are inherited by the rights attached to the collection this record is a member of.

Field	Datatype	Description
dbId	String (required)	Unique id to retrieve the record from the WITH database.
externalId	String (required)	A hash value of the url that characterizes the original item the record refers to, and is used to determine whether two records in the WITH database correspond to the same object. For Europeana externalId=hashValue(isShowBy?null:isShownAt). Extracted from the original item metadata.
isShownAt	String	url to the provider web page for that record. Extracted from the original item metadata.
isShownBy	String (at least one of isShownAt and isShownBy is required)	url to the (full resolultion) content. Extracted from the original item metadata.
thumbnailUrl	String	url to the thumbnail of the object. Extracted from the original item metadata.
title	String (required)	A name by which the item is known. Extracted from the original item metadata.

description	String	A description of the digital item. Extracted from the original item metadata.
creator	String	An entity primarily responsible for making the resource. This may be a person, organisation or a service. Extracted from the original item metadata.
dataProvider	String	The name or identifier of the data provider of the object (i.e. the organisation providing data to an aggregator). Extracted from the original item metadata.
provider	String	The name or identifier of the provider of the object (e.g. the project providing the item). Extracted from the original item metadata.
contributor	Array<String>	Entities responsible for making contributions to the resource. Extracted from the original item metadata.
year	Array<String>	A list of years associated with important events in the life of the item. Usually reprsents the year in which the original item was created. Extracted from the original item metadata.
type	Enumeration	Values are one of: “Image”, “Video”, “Sound”, “Text”. Extracted from the original item metadata.
itemRights	Enumeration	The rights statement that applies to the item. Values are one of: "Attribution alone"; "Restricted"; "Permission"; "Allow re-use and modifications"; "Allow re-use for commercial"; "use for commercial purposes modify, adapt, or build upon"; "NOT Comercial";"NOT Modify"; "not modify, adapt, or build upon, not for commercial purposes"; "share alike"; "use by attribution"; "Allow re-use"; "Rights Reserved";"Rights Reserved - Paid Access"; "Rights Reserved - Restricted Access"; "Rights Reserved - Free Access"; "Unknown"; Extracted from the original item metadata.
content	Map<Enumeration, String>	A map containing all known metadata serializations of the record. The keys indicate the format and schema of the metadata

		representation, and can be one of the following values: JSON_UNKNOWN, XML_UNKNOWN, JSON_EDM, JSONLD_EDM, XML_EDM, JSONLD_DPLA, JSON_NLA, XML_NLA, JSON_DNZ, XML_DNZ, JSON_YOUTUBE. The values of the map are the respective serializations.
source	Enumeration	The external source from which the item was imported. Values are one of: Europeana", "DPLA", "DigitalNZ", "Rijksmuseum", "The British Library", "UploadedByUser". Other sources, like Youtube, NLA, can also be supported.
sourceId	String	The id of the record in the source from which it has been imported.
sourceUrl	String	A url to retrieve the record from its source, e.g. Europeana's guid value, or http://dp.la/item/recordId.
tags	Array<String>	List of tags added by a user, which characterize the record.
collectionId	String (required)	Id of the collection (or exhibition) to which the record belongs to.
position	Integer (required)	Index of record's position in collection or exhibition.
created	Date (required)	The date on which the record was added in the WITH database.
exhibitionRecord	JSON	{annotation: String, videoUrl: String, audioUrl:String}
totalLikes	Integer	Number of total likes that the original object (i.e. records with the same externalId) has received by users

APPENDIX II – LIST OF WITH'S API CALLS

Call	Path	Categories
GET	/collection/list	Collections
GET	/collection/listShared	Collections
GET	/collection/featured	Collections
POST	/collection/create	Collections
POST	/collection/listByUser	Collections
POST	/collection/:id/addRecord	Collections
DELETE	/collection/:id/removeRecord	Collections
GET	/collection/:id/list	Collections
GET	/collection/favorites	Collections
POST	/collection/liked	Collections
DETELE	/collection/unliked/:recId	Collections
GET	/collection/favoriteCollection	Collections
GET	/collection/:id/listUsers	Collections
GET, POST, DELETE	/collection/:id	Collections
POST	/api/advancedsearch	Search
POST	/api/search	Search
GET	/record/merged/:externalId	Record
GET	/record/similar/:externalId	Record
GET, PUT, DELETE	/record/:id	Record
POST	/group/create	Group
GET, PUT, DELETE	/group/:id	Group
PUT	/group/addUserOrGroup/:groupId	Group, User
PUT	/group/removeUserOrGroup/:groupId	Group, User

GET	/group/descendantGroups/:groupId	Group
GET	/group/descendantOrganizations/:groupId	Organization
GET	/group/descendantProjects/:groupId	Project
POST	/organization/create	Organization
POST	/project/create	Project
GET	/user/findByUserOrGroupNameOrEmai	Group, User
GET	/user/listNames	Group, User
GET	/user/:id/photo	Group, User
GET, PUT, DELETE	/user/:id	User
GET	/user/resetPassword/:emailOrUserName	User
POST	/user/changePassword	User
GET	/user/apikey/create	User
POST	/user/register	User
POST	/user/login	User
GET	/user/logout	User
GET	/user/emailAvailable	User
GET	/user/token	User
GET	/user/loginWithToken	User
GET	/api/autocompleteExt	Auto-complete
GET	/media/create	Media
GET, POST, DELETE	/media/:id	Media

APPENDIX III – API CALLS & JSON SCHEMA FOR USER-GROUPS

API Calls

The list of the API calls regarding the User Groups are the following:

Method	Call	Parameters/return	Description
POST	/group/create	adminId: the administrator id adminUsername: the administrator username return: the JSON of the new group	Creates a "UserGroup" with the specified user as administrator and with the given body as JSON. The name of the group must be unique. If the administrator is not provided as a parameter the administrator of the group becomes the user who made the call.
POST	/organization/create	(Same as /group/create)	Creates an Organization with the same way as a UserGroup.
POST	/project/create	(Same as /group/create)	Creates a Project with the same way as a UserGroup.
PUT	/group/:id	return: the updated group metadata	Edits group metadata and updates them according to the POST body. Only the creator of the

			group has the right to edit the group.
DELETE	/group/:id	return: success message	Deletes a group from the database. The users who participate are not deleted as well. Only creator has the right to delete the group.
GET	/group/:id	return: the JSON of the group	Gets the group.
PUT	/group/addUserOrGroup/:groupId	id: the user or group id return: success message	Adds the user/group with id "id" to the group with id "groupId".
PUT	/group/removeUserOrGroup/:grouping	id: the user or group id return: success message	Removes the user/group with id "id" from the group with id "groupId".
GET	/group/findByGroupName	name: the group name collectionId: the collection id return: the JSON of the group	Gets the UserGroup with the specific name along with the rights of the group for the collection defined. (If collectionId is null the rights are not returned).
GET	/group/descendantGroups/:groupId	direct: (default true) collectionHits: (default false)	Gets the children groups of the specified group. If the direct flag is

		return: a JSON array with all the groups.	true then only the direct children are returned, else all the descendant groups of the group are returned. If the flag collectionHits is true then “totalCollections" and "totalExhibitions" are returned along with every group.
GET	/group/descendantOrganizations/:groupId	(same as /group/descendantGroups/:groupId)	Gets the children Organizations of the specified group.
GET	/group/descendantProjects/:groupId	(same as /group/descendantGroups/:groupId)	Gets the children Projects of the specified group.

JSON schema

The JSON format of the entities “Organization”(the “Project” has the same) is shown at the following example:


```
{
  "username": "ntua",
  "about": "The National Technical University (NTUA) is the oldest and most prestigious educational institution of Greece in the field of technology, and has contributed unceasingly to the country's scientific, technical and economic development since its foundation in 1836. It is closely linked with Greece's struggle for independence, democracy and social progress. In Greek, NTUA is called the \"Ethnicon Metsovion Polytechnion\" which stands for National Metsovion Polytechnic. It was named \"Metsovion\" to honor the donors and benefactors Nikolaos Stournaris, Eleni Tositsa, Michail Tositsas and Georgios Averof, all from Metsovo, a small town in the region of Epirus, who made substantial donations in the last half of the 19th century.",
  "privateGroup": false,
  "friendlyName": "National Technical University of Athens",
  "page": {
    "address": "Ir. Polytechniou 9, 15773 Zografou",
    "coordinates": {
      "latitude": 37.977541,
      "longitude": 23.776029900000026
    },
  "city": "Athens",
  "country": "Greece",
  "url": "http://www.ntua.gr",
  "coverImage": "560d3d59d4c6f6ed34039a40",
  "coverThumbnail": null,
  "featuredCollections": null,
  "featuredExhibitions": null
  },
  "dbId": "560bd5b277c8f93860e22e37",
  "thumbnail": "56127edfd4c6b55f34f7c29e",
  "adminIds": "[5609307a77c88da66373ee83]",
  "users": "[5609307a77c88da66373ee83]",
  "parentGroups": "[560d3277e4b06044750e5a53]"
}
```