D3.4 Content Management Services

Deliverable number	**D3.4**
Deliverable title	**Content Management Services**
Nature[1]	**OTHER**
Dissemination Level [2]	**PU**
Author (email) Institution	**Peter Smatana (peter.smatana@tuke.sk), Jan Hreno (jan.hreno@tuke.sk), Tomas Sabol (tomas.sabol@tuke.sk) TUK**
Editor (email) Institution	**Nikos Frangakis (nikos.frangakis@iccs.gr)** **ICCS**
Leading partner	**TUK**
Participating partners	**TUK, ICCS, UMA, CLIO, Xteam,**
Official submission date:	**31/05/2019**
Actual submission date:	**15/07/2019**

[1] **R**=Document, report; **DEM**=Demonstrator, pilot, prototype; **DEC**=website, patent filings, videos, etc.; **OTHER**=other

[2] **PU**=Public, **CO**=Confidential, only for members of the consortium (including the Commission Services), **CI**=Classified, as referred to in Commission Decision 2001/844/EC

Modifications index	
8/9/2017	Initial Release
29/9/2017	ICCS review comments
31/5/2018	Updated for MS3.8
03/07/2019	Updated to current state for D3.4
15/06/2019	Proof reading

This work is a part of the PLUGGY project. PLUGGY has received funding from the European Union’s Horizon 2020 research & innovation programme under grant agreement no 726765. Content reflects only the authors’ view and European Commission is not responsible for any use that may be made of the information it contains.

Table of Contents

Executive summary

The base architecture of PLUGGY includes three main modules: A Social Platform, a Curatorial Tool and a Content Management Services. The D3.4 deliverable describes the content management services and is delivered in 3 phases

- MS3.4 Conceptual prototype (M10),
- MS3.8 Technical prototype (M18)
- and the final D3.4 (M30).

As all of these are SW deliverables. This written accompanying document will briefly describe the usage and main functionalities of the SW. This final version of the document describes the online documentation and source code of the implementation. Services implemented within D3.4 are based on requirements raised from D2.3 (Social Platform design and specification), D3.2 (Social Platform), D2.4 (Curatorial Tool design and specification), D3.3 (Curatorial Tool) and all applications developed within WP4. Future application developers can create user friendly applications using publicly available APIs described in the online documentation and in this document. Developer guidelines, lesson learned and deeper description of integration of 3rd party application into the PLUGGY platform will be described in D4.5 Guidelines and Instructions for PLUGGY Apps.

1 Introduction

This is the third and the final version of the D3.4 “Content Management Services” of WP3 “Social Platform and Curatorial Tool implementation”. It is delivered at M30 as the accompanying report of deliverable D3.4 “Content Management Services”. This document summarises information related to the Content Management Services. Full documentation of the services is available online (links to the online documentation is introduced later in this deliverable). The document itself can be also used as a starting point to explore PLUGGY’s core services. The current technical prototype contains the REST API, description of installation and usage information. Table 1. shows important notes from DoA (Description of Actions) related to D3.4.

> The aim of the task is to develop Content Management Services the main functionality of which is to provide unified access to the content managed by the PLUGGY platform internally or integrated from the existing digital archives or social media sites. The main part of the Content Management Services will be service layer supporting content search, recommendation and automatic management of content metadata including the tracking of content provenance and enforcing of the IPR management strategies are
>
> The second part of the services will consist of integration layer for connecting to the existing digital collections and social media sites. The content integration layer will have a modular architecture composed of connectors connected to external sources unifying the access to the content and associated metadata. Content Management Services will also provide local storage for the content supplied by users through the Social platform or by curators through the Curatorial tool managed internally by PLUGGY.

Table 1: Notes from the DoA

1.1 Implementation

We have decided to use MongoDB[3] as backend storage due to a lot of advantages for the project, like flexible data model, high scalability, performance, expressive query language etc. MongoDB provides great possibilities for storing and querying user content. All the functionalities provided by Content Management Services are the result from the users’ requirements described within WP2 and WP4.

An agile development approach was selected including weekly sprint meetings of PLUGGY developers to create and hone all PLUGGY SW components including the Content Management Services.

[3] https://www.mongodb.com/

We have also used an issue tracker system that tracks user issues and requests for the Content Management Services on https://isense-gitlab.iccs.gr/PLUGGY/PLUGGY-core/issues.

1.2 WebSocket/RealTime communication

Notification services use a real-time communication based on the WebSocket protocol. Thanks to this, applications connected to Content Management Services can receive notifications when new content is created, or an existing content has new followers or some other events related to currently logged user has happened. Endpoint service is available via special URL: wss://develop.PLUGGY.eu/ Example source codes and deeper description will be covered in D4.5.

1.3 Security

As D3.1 describes, all the communication is handled via HTTPS protocol and all the requests for the protected content can be processed only in case that users have a valid token with appropriate ownership to the modified content. A valid token is issued using our OAuth2 service. PLUGGY platform does not store unnecessary information. Passwords stored in the platform are hashed and salted. All content created via the platform has its owner and only an owner with a valid token can modify and delete such content. Private content is visible only by its owners.

1.4 3rd party Applications

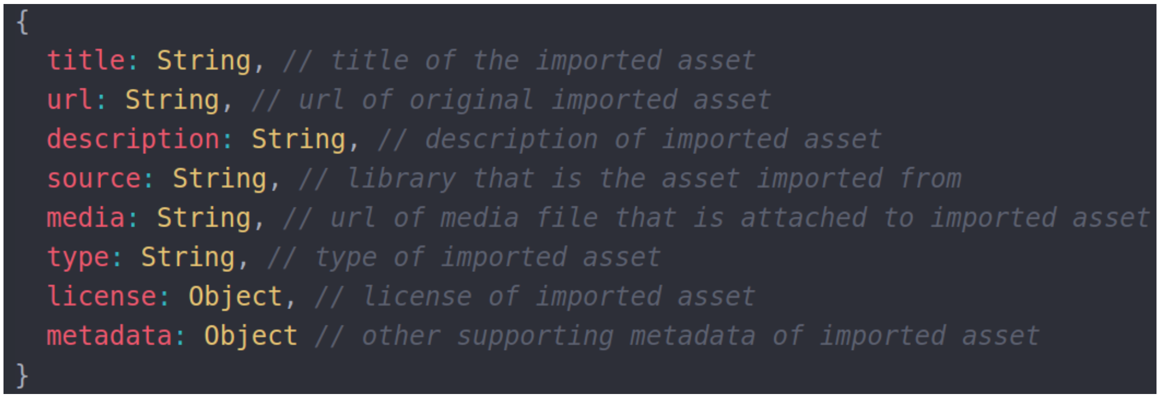
All developers can create their own application which has to be registered to the platform and they receive their own credentials that will be used to communicate with the platform. The registration process is done via the Developer tools integrated into the social platform. More information will be covered in D4.5.

1.5 Plugins External Library

PLUGGY allows the integration of external libraries as a source of new assets. An external library can be integrated when it has a search API available and its content can be mapped to our import format (see Figure 4). Currently we have integrated three external libraries:

- https://www.wikipedia.org/,
- https://www.europeana.eu,
- https://www.britishmuseum.org/.

All the content from external libraries that the interaction is made with, is automatically cached in the PLUGGY platform with respect to the original licence.



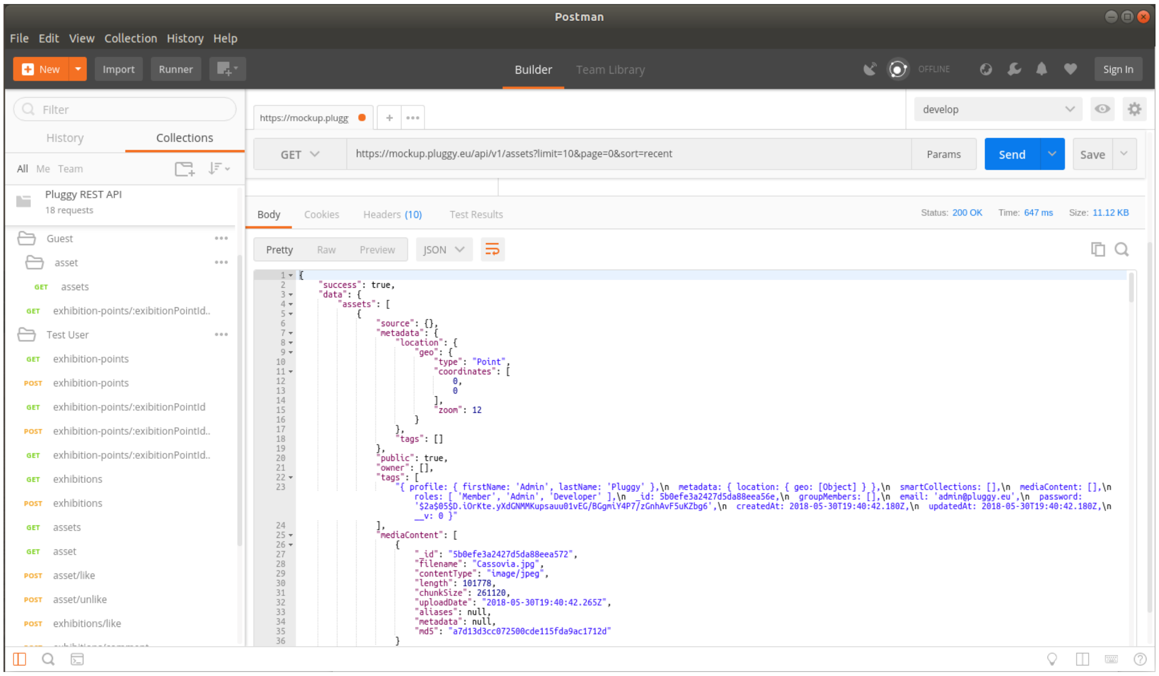
```
{
  title: String, // title of the imported asset
  url: String, // url of original imported asset
  description: String, // description of imported asset
  source: String, // library that is the asset imported from
  media: String, // url of media file that is attached to imported asset
  type: String, // type of imported asset
  license: Object, // license of imported asset
  metadata: Object // other supporting metadata of imported asset
}
```

Figure 1: imported asset data model

2 PLUGGY Content Management Services REST API

PLUGGY Content Management Services is a REST API providing services for saving and retrieving of assets, exhibitions, receiving recommendations, hints and do a search on these entities. Additionally, the Content Management Services provides services for users and teams registration and authentication.

PLUGGY's REST API is accessible via HTTP over TLS. The best way to start with PLUGGY REST API is to install some tools like the Postman Tool that helps explore all the PLUGGY services (https://www.getpostman.com/postman). Developer could also use our online Swagger documentation (https://beta.PLUGGY.eu/api/doc/) which is publicly available and where API calls can also be performed.



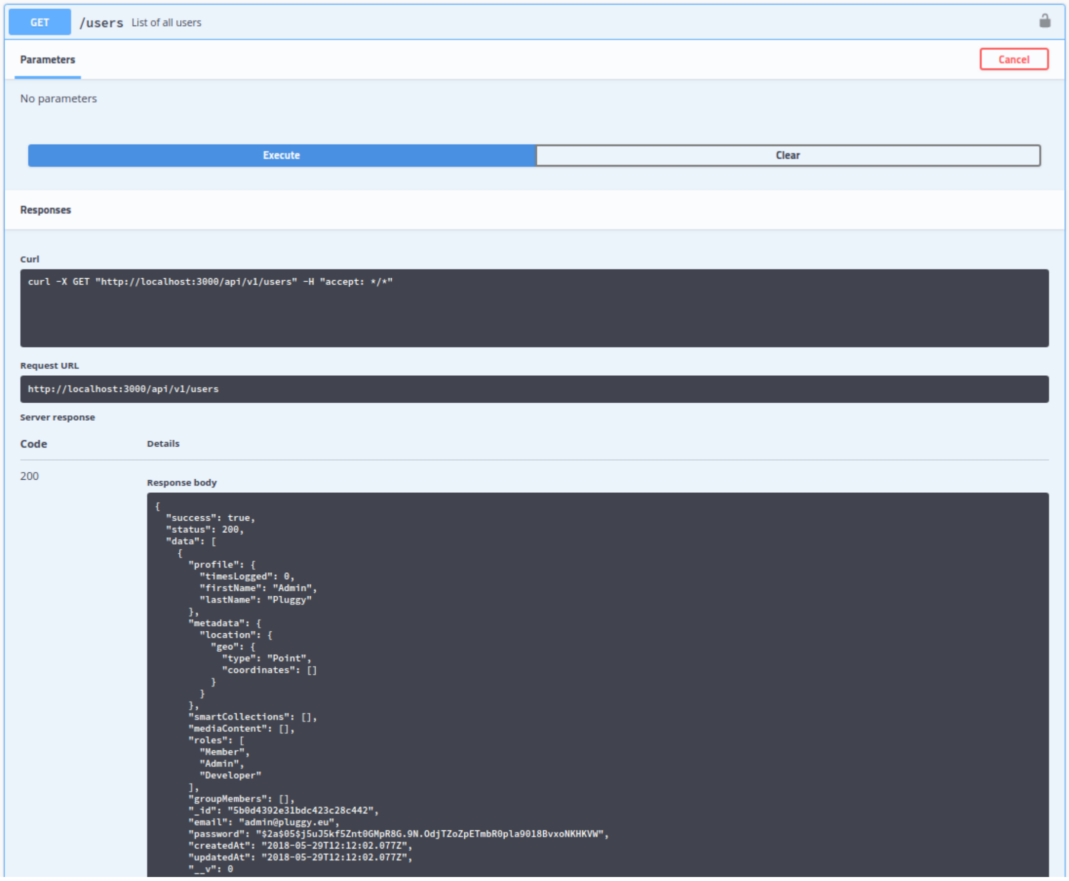
GET https://mockup.pluggy.eu/api/v1/assets?limit=10&page=0&sort=recent

Status: 200 OK Time: 647 ms Size: 11.12 KB

JSON

```
{
    "success": true,
    "data": {
        "assets": [
            {
                "source": {},
                "metadata": {
                    "location": {
                        "geo": {
                            "type": "Point",
                            "coordinates": [
                                0,
                                0
                            ],
                            "zoom": 12
                        }
                    },
                    "tags": []
                },
                "public": true,
                "owner": [],
                "tags": [
                    "{ profile: { firstName: 'Admin', lastName: 'Pluggy' },\n  metadata: { location: { geo: [Object] } },\n  smartCollections: [],\n  mediaContent: [],\n
                        roles: [ 'Member', 'Admin', 'Developer' ],\n  _id: 5b0efe3a2427d5da88eea56e,\n  groupMembers: [],\n  email: 'admin@pluggy.eu',\n  password:
                        '$2a$05$D.iOrKte.yXdGNMMKupsauu01vEG/BGgmiY4P7/zGnhAvF5uKZbg6',\n  createdAt: 2018-05-30T19:40:42.180Z,\n  updatedAt: 2018-05-30T19:40:42.180Z,\n
                        __v: 0 }"
                ],
                "mediaContent": [
                    {
                        "_id": "5b0efe3a2427d5da88eea572",
                        "filename": "Cassovia.jpg",
                        "contentType": "image/jpeg",
                        "length": 101778,
                        "chunkSize": 261120,
                        "uploadDate": "2018-05-30T19:40:42.265Z",
                        "aliases": null,
                        "metadata": null,
                        "md5": "a7d13d3cc072500cde115fda9ac1712d"
                    }
```

Figure 2: Postman used to call PLUGGY REST API



GET /users List of all users

Parameters

No parameters

Responses

Curl

```
curl -X GET "http://localhost:3000/api/v1/users" -H "accept: */*"
```

Request URL

```
http://localhost:3000/api/v1/users
```

Server response

Code: 200

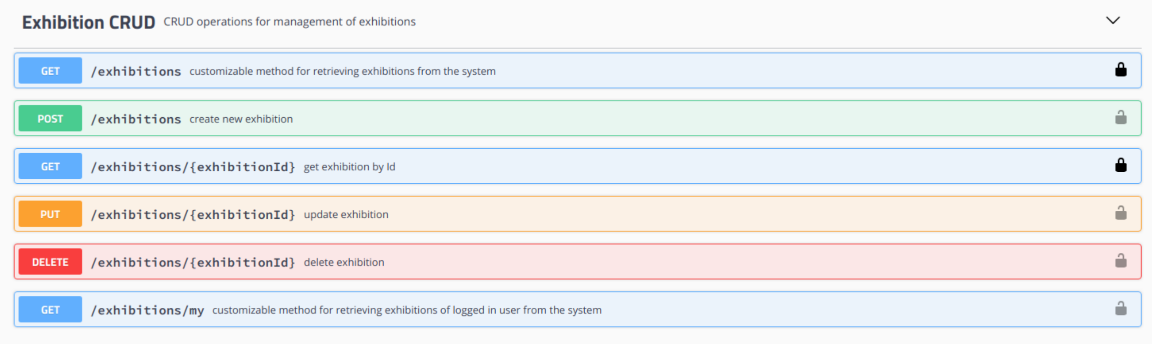
Details

Response body

```
{
  "success": true,
  "status": 200,
  "data": [
    {
      "profile": {
        "timesLogged": 0,
        "firstName": "Admin",
        "lastName": "Pluggy"
      },
      "metadata": {
        "location": {
          "geo": {
            "type": "Point",
            "coordinates": []
          }
        }
      },
      "smartCollections": [],
      "mediaContent": [],
      "roles": [
        "Member",
        "Admin",
        "Developer"
      ],
      "groupMembers": [],
      "_id": "5b0d4392e31bdc423c28c442",
      "email": "admin@pluggy.eu",
      "password": "$2a$05$j5uJ5kf5Znt0GMpR8G.9N.OdjTZoZpETmbR0pla9018BvxoNKHKVW",
      "createdAt": "2018-05-29T12:12:02.077Z",
      "updatedAt": "2018-05-29T12:12:02.077Z",
      "__v": 0
```

Figure 3: PLUGGY REST API documentation - direct REST API call

Figure 4 shows an example of available services for exhibitions. (CRUD meaning Create, Read, Update, Delete)



Exhibition CRUD CRUD operations for management of exhibitions

- GET /exhibitions customizable method for retrieving exhibitions from the system
- POST /exhibitions create new exhibition
- GET /exhibitions/{exhibitionId} get exhibition by Id
- PUT /exhibitions/{exhibitionId} update exhibition
- DELETE /exhibitions/{exhibitionId} delete exhibition
- GET /exhibitions/my customizable method for retrieving exhibitions of logged in user from the system

Figure 4: Part of online REST API documentation

In the following sub-chapters, we briefly describe all available operations for the different entities of the PLUGGY platform.

2.1 Assets

Using the PLUGGY API, Assets can be retrieved via a customisable method for retrieving assets from the system. New assets can be created and existing assets can be imported from an external library. Assets can also be updated anddeleted.

Extra services to retrieve, create, delete media files related to the assets like the cover images and thumbnails for the asset are available.

For the social functionality, services for liking, commenting and reporting of the asset are available.

2.2 Exhibitions

Similar to assets, exhibitions can be created, modified, deleted and retrieved via available API services. Thumbnail, cover and original images related to the exhibition and liking, commenting and reporting of the exhibition is possible via corresponding API service.

2.3 Exhibition points

Set of services for the exhibition points is the same as for assets and exhibitions.

2.4 Users

In the current model we have three types of users: individual user, team user (group, organisation, institution) and application user. Users can be created, updated, deleted and retrieved. Users can be linked to a thumbnail and/or image of their choice', can join teams, have followers, follow others. Folders related to users can be created containing some content. Notifications for the user can be retrieved and marked as read.

2.5 Teams

Similar operations can be made with team entities as with user entities. Membership of users in the team can be managed additionally.

2.6 Applications

Applications list can be retrieved from the API. A new application record can be created, updated and deleted. The development status of the application can be changed, a list of application members can be set, and roles of users in a developer team can be updated. Applications can be represented by thumbnail, and full-size image art.

2.7 Events

Events can be created, updated, deleted, retrieved.

2.8 Search

Customizable search that allows combination of different search parameters to search by full-text, geo location, owner, licences, tags, title, content type within all content in the PLUGGY platform. It allows to search within assets, exhibitions, applications, teams, users or search within integrated external digital libraries.

2.9 Hinting

Hinting services for tags, places, titles and users are provided

2.10 GDPR

To comply with theGDPR directive of the EU, our API provides services that cover the following main points:

- Right to data accessed – user can ask admin to export all the information that are stored within the PLUGGY platform regarding him or her.
- Right to be forgotten – user can ask admin to delete all the information regarding him or her from the platform.
- Data portability – exported format of data is standardised JSON format. Assets and Exhibitions are exported in standardised EDM (Europeana Data Model) format, could be imported to different platforms that supports EDM.
- Privacy by design – system architecture and data security are implemented to conform with current best practices and the current state- of-the-art in the field.

2.11 Licences

Service that helps users to choose the best licence for an exhibition based on licences of assets that are part of the exhibition was developed. Furthermore, a recommendation service for a licence that fulfils the required usage of the licenced content has been developed.

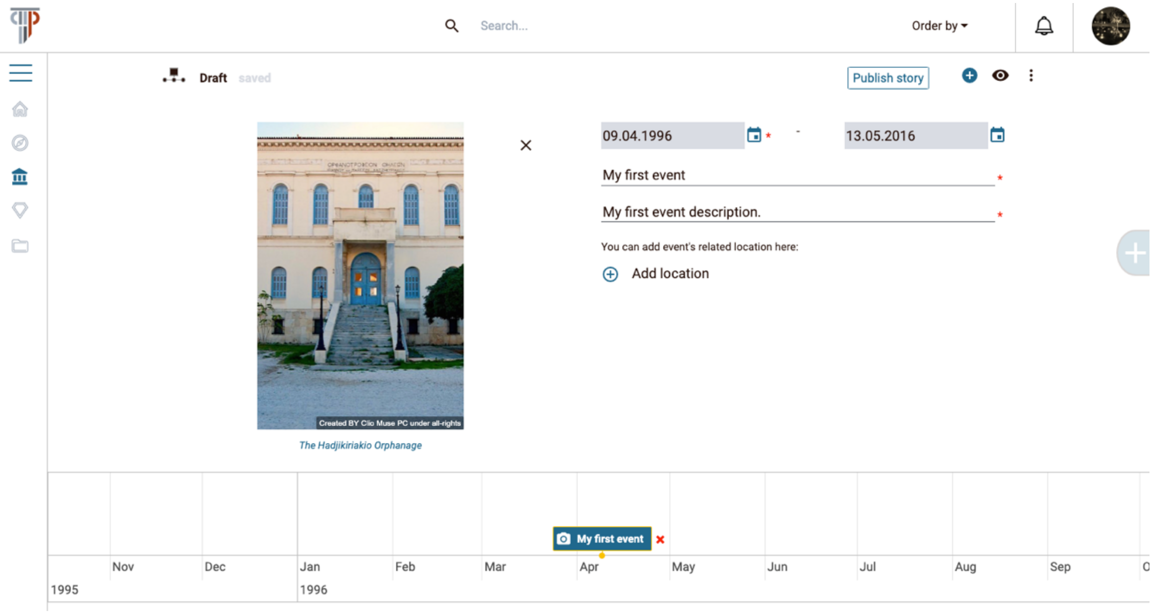
2.12 Statistics

A service was developed as a need of the WP6 for evaluation. This service returns statistical information related to content and users.

3 PLUGGY Web UI Integration

REST API is already integrated with other software tools developed within WP3. Namely “Social Platform” and “Curatorial Tool” which are used as user interface and commonly named the PLUGGY Web UI. The running Web UI demo is available publicly at: https://beta.PLUGGY.eu/

A screenshot of the integrated prototype of the Curatorial Tool is shown in Figure 5, and Figure 6 shows the integrated version of Social Platform.



09.04.1996 * - 13.05.2016

My first event *

My first event description. *

You can add event's related location here:

⊕ Add location

The Hadjikiriakio Orphanage

Figure 5: Curatorial Tool integrated with REST API

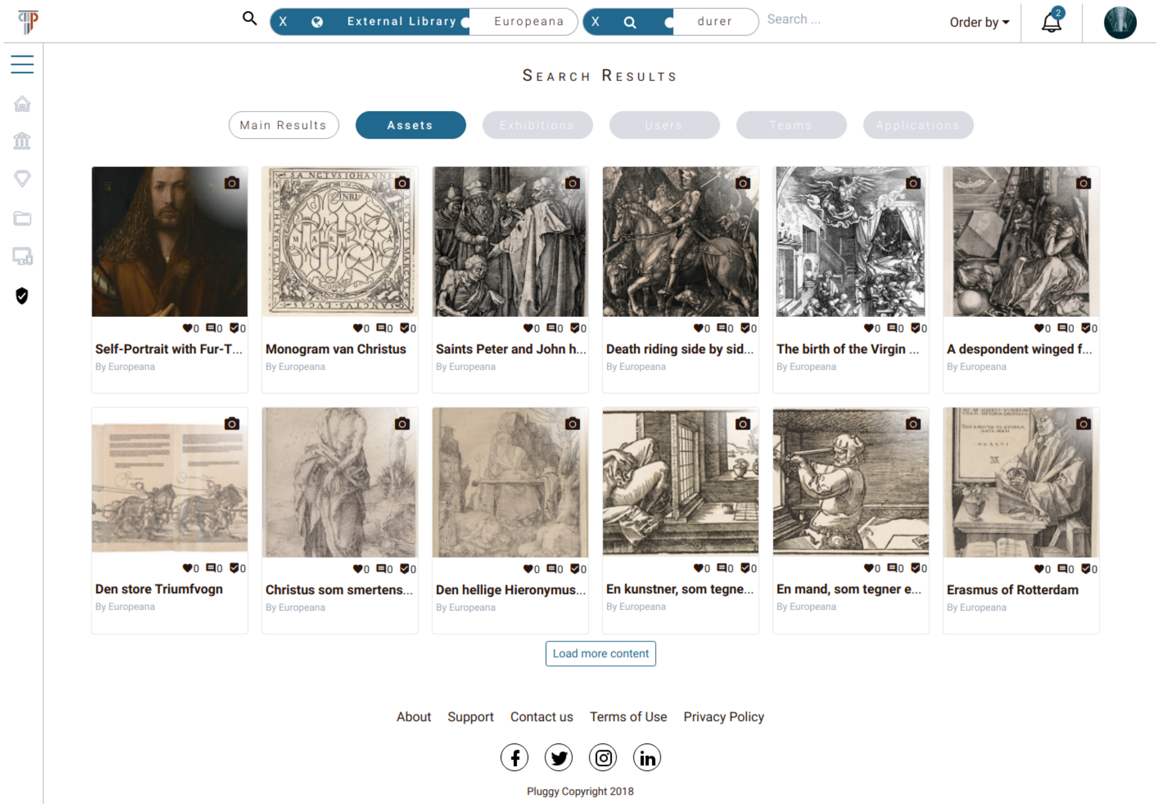


Figure 6: Social Platform with integrated REST API

4 Installation

Here we describe in simple steps how the technical prototype can be installed for local testing or development and production deployment. These instructions currently work for project partners only, but now that the final solution for SW licences (see D2.5) has been agreed on, we will soon be ready to move the SW from the consortium only GitLab versioning system to a public GitHub server.

1. Install Node.js for your platform:

https://nodejs.org/en/download/

2. Install mongo DB for your platform:

https://www.mongodb.com/download-center#community

3. Clone or fork source code from our git repository (currently works for PLUGGY members only. Will be available publicly after SW licences between partners are agreed):

https://isense-gitlab.iccs.gr/PLUGGY/PLUGGY-core

4. Run “npm. install”

5. Load example data to database if needed: “npm run init”

6. Start server using *“npm run start”* (PLUGGY-core is running locally on http://localhost:3000

In case that developer has more specific requirements he or she can modify the configuration file (“/src/api/config.js”) to fulfil their needs.

5 Conclusions

Within D3.4 we have delivered Content Management Services as a software prototype which is already integrated with the PLUGGY Web user interface and applications developed within WP4. It is available both as a running prototype service and will be available for download as an open source software.

The delivered software solution fulfills the requested functionality and was also tested by multiple teams of developers, that integrate their applications with PLUGGY Content Management Services.