D3.1 Architecture specification

Deliverable number	**D3.1**
Deliverable title	**Architecture specification**
Nature[1]	**R**
Dissemination Level [2]	**PU**
Author (email) Institution	**Jan Hreno (jan.hreno@tuke.sk), TUK**
Editor (email) Institution	**Daniel Gonzales Toledo (dgonzalezt@uma.es), UMA; Spyros Bolierakis (spyros.bolierakis@iccs.gr), ICCS**
Leading partner	**TUK**
Participating partners	**ICCS, UMA, VIA, ICL**
Estimated person/months per leading and participating partners	**TUK (3pm), ICCS (1pm), UMA (1pm), VIA (1pm), ICL (1pm)**
Official submission date:	**30/11/2018**
Actual submission date:	

Modifications index	
15/02/2017	Initial Release
31/05/2017	MS3.1 M6 version ready
31/01/2018	MS3.5 M14 Second version of the architecture ready
30/11/2018	D3.1 final version ready

This work is a part of the PLUGGY project. PLUGGY has received funding from the European Union's Horizon 2020 research & innovation program 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.

[1] **R**=Document, report; **DEM**=Demonstrator, pilot, prototype; **DEC**=website, patent fillings, 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

Table of Contents

1 EXECUTIVE SUMMARY

This is a final version of the D3.1 Architecture specification.
PLUGGY Architecture addresses the Obj2 objective of the project:
Obj2 - Design an architecture of the social platform to allow the easy integration of applications, the scalability of the platform and the support of specialized devices (AR/VR/trackers etc.).

Deliverable D3.1 is delivered as a part of the *T3.1 Architecture design task*. The goal of this task is to specify technical details for the system architecture, i.e. identify main system components, their interactions and specify integration interfaces. Input into this task was the overall functional specification and user requirements identified in Work Package 2. Part of the work in this task was also to survey existing applications, tools, frameworks and libraries and identify which of the existing technologies can be reused for the implementation of the PLUGGY platform components. The main output of this task is the detailed technical specification of platform components and implementation notes.

This deliverable is fed by the *"WP2 Social Interaction design and specifications"* results. The deliverable results will be used in the following tasks of the same work package *as well as* in *"WP4: Applications design and implementation"* and *"WP6 Evaluation and validation of usability and utility"*. There are two intermediary milestones *"MS3.1 Conceptual version of the architecture ready (M6)*" and *"MS3.5 Second version of the architecture ready (M14)"* related to this deliverable. Work package interdependency graph with tasks and work packages dependencies can be seen below

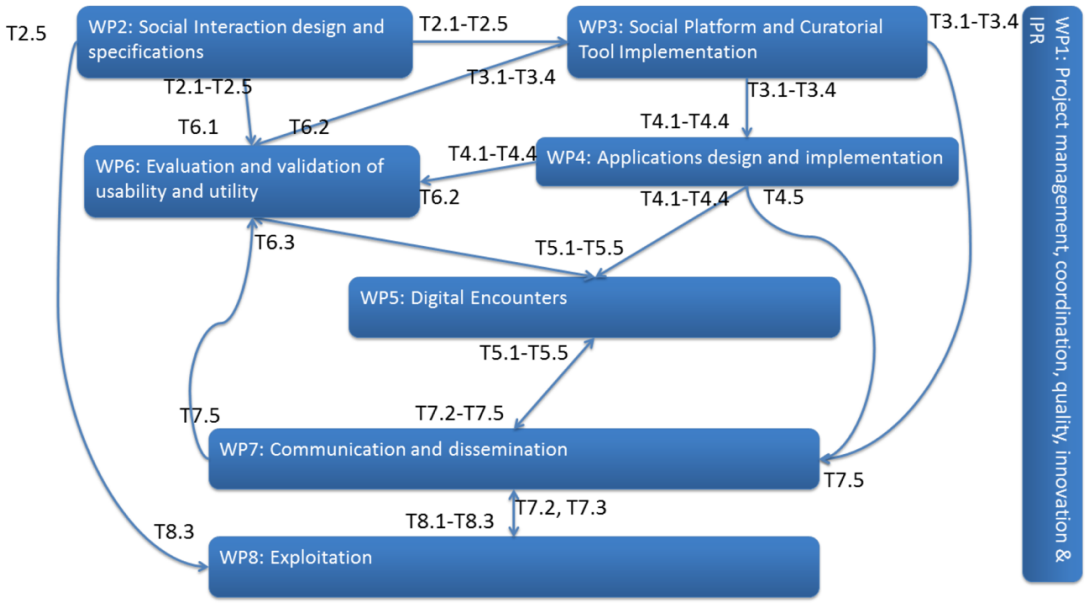


Figure 1: Work package interdependencies

This deliverable does not describe all the parts of the system in full detail, as several system components have their own design deliverables. In the following table these additional deliverables are listed.

Table 1: List of deliverables to contain detailed information on architectural components and responsibilities of PLUGGY partners

Component	**Owner**	**Main contributors**	**Design Deliverable**	**Implementation Deliverable**
Curatorial tools	ICCS	*ICCS, TUK, UMA, VIA, ICL, XTS, ESM, CLIO*	D2.4 Curatorial Tool design and specification (ICCS, R, PU, M18)	D3.3 Curatorial Tool (ICCS, OTHER, PU, M30)
Social Platform	UMA	*TUK, UMA, VIA, ICL, XTS, ESM, CLIO*	D2.3 Social Platform design and specification (UMA, R, PU, M18)	D3.2 Social Platform (UMA, OTHER, PU, M30)
Content Management Services	TUK	*ICCS, TUK, ICL*		D3.4 Content Management Services (TUK, OTHER, PU, M30)
AR app	UMA	ICCS, TUK, UMA	D4.1 AR App (UMA, OTHER, PU, M32)	D4.1 AR App (UMA, OTHER, PU, M32)
Collaborative Game	XTS	ICCS, TUK, VIA, XTS	D4.2 Collaborative Game App (XTS, OTHER, PU, M32)	D4.2 Collaborative Game App (XTS, OTHER, PU, M32)
Geolocation App	CLIO	ICCS, CLIO	D4.3 Geolocation App (CLIO, OTHER, PU, M32)	D4.3 Geolocation App (CLIO, OTHER, PU, M32)
Sonic App	ICL	ICL	D4.4 3D Sonic App (ICL, OTHER, PU, M32)	D4.4 3D Sonic App (ICL, OTHER, PU, M32)
Future apps	UMA	ICCS, TUK, UMA	D4.5 Guidelines and Instructions for PLUGGY Apps (UMA, R, PU, M36)	

2 Introduction

Figure 2: PLUGGY's Social Platform and pluggable architecture

PLUGGY will provide the necessary architecture, structure and interfaces for the creation of pluggable applications, allowing for beyond-the-project, not yet imagined ways to utilize the content on the social platform, while focusing on the design of the social interaction, helping to build new virtual heritage communities.
PLUGGY's Social Platform will be designed based on a modular architecture, allowing web and mobile applications to be plugged to the social platform and utilize its content. During the action's lifetime 4 applications will be designed and implemented using PLUGGY's architecture: a) an Augmented Reality application focusing on indoor applications, based on optical markers, ideally suited for museums and heritage sites, where AR tracking can work very well b) a geolocation mobile application, focused on outdoor activities, ideally suited for large outdoor heritage sites and cities, which will also be very useful for daily usage in order to discover everyday landscape c) a 3D sonic narratives application suited for three dimensional stories that focus mainly on audio, utilizing advanced 3D sound techniques and iv) a collaborative game, targeted for younger generations applicable to any museum or heritage site.
Moreover, the architecture will allow the development of any kind of application utilizing the social platform's content. The applications will be able to focus on specific aspects of the user experience, like improved access, interactive experiences or personalized content. Users, prior to their visit, will be able to download the applications and experience the advanced interaction they will offer, prior during or after the visit.

2.1 STRUCTURE AND CONTENTS OF THIS DELIVERABLE

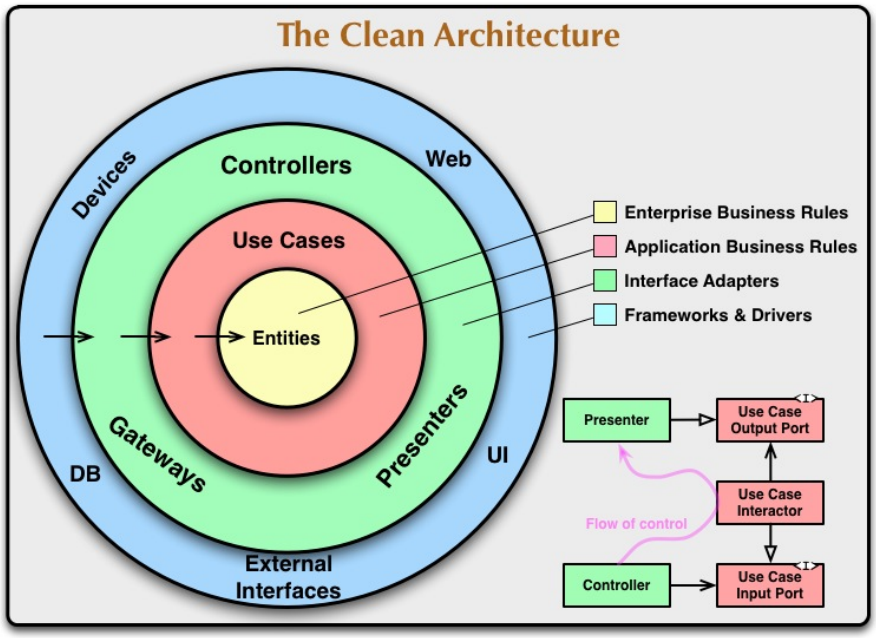
This deliverable describes the architecture of the PLUGGY Platform. Chapter 2 is the PLUGGY platform introduction. In Chapter 3 a well-defined methodology for SW architecture description based on views and perspectives is described. Chapter 4 uses Architectural views to describe different aspects of the PLUGGY Architecture. In annexes we present a full set of requirements, components description, list of identified and implemented API functions and a brief glossary of the project.

3 Software Architecture Introduction

We are going to use some well described and widely used principles to describe the PLUGGY platform architecture. One of these will be the Clean Architecture per components/applications as described in Uncle Bob's blog[3] and the other one is the Viewpoints and Perspectives architecture description as proposed in the Software System Architecture book from Rozanski and Woods[4]. The later will be used for description of the whole platform.

3.1 CLEAN ARCHITECTURE

The main property of the clean architecture is the independence of different layers from each other. The purpose is the separation of concerns by keeping the business rules not knowing anything at all about the outside world, thus, they can be tested without any dependency to any external element. This should speed up different application development process.



Source: https://8thlight.com/blog/uncle-bob/2012/08/13/the-clean-architecture.html

Figure 3: Simple Architecture

On the basic schema of the Simple Architecture on Figure 3 we see few main concepts of the Clean Architecture:

- Entities: are business objects of the application.
- Use Cases: orchestrate the flow of data to and from the entities.
- Controllers, Presenters, Gateways: convert data from the format most convenient for the use cases and entities.
- External Interfaces, UI, Web, Devices, DBs: This is where all the application details are specified.

[3]https://8thlight.com/blog/uncle-bob/2012/08/13/the-clean-architecture.html

[4]Software Systems Architecture: Working with Stakeholders Using Viewpoints and Perspectives, Nick Rozanski, Eóin Woods, Addison-Wesley, 2011

The idea here is that all four layers are using its own data models, so these can be tested separately.

3.2 VIEWPOINTS AND PERSPECTIVES

We will not go deep into the detail of the methodology described in Software System Architecture by Rozanski & Woods. Rather we pick the most important concepts of their approach.
The Viewpoints and Perspectives software architecture focuses on Viewpoints and Perspectives to describe many different aspects of the proposed software system while keeping separation in different concerns, developer focus, managing complexity, and eases communication with stakeholder groups.
A ***viewpoint*** is a collection of patterns, templates, and conventions for constructing one type of view. It defines the stakeholders whose concerns are reflected in the viewpoint and the guidelines, principles, and template models for constructing its views.
An ***architectural perspective*** is a collection of activities, tactics, and guidelines that are used to ensure that a system exhibits a particular set of related quality properties that require consideration across a number of the system's architectural views. Architecture views can be: Accessibility, Availability, Internationalization, Performance, Scalability, Security, Usability.

Based on Rozanski and Woods we propose a list of architectural views that will be used to describe the PLUGGY platform. For every view, we try to describe them from different perspectives if possible. The list is presented in the Table 2.

Table 2: PLUGGY architecture views

Context	Describes the relationships, dependencies, and interactions between the system and its environment (the people, systems, and external entities with which it interacts).
Logical, Functional	Describes the system's functional elements, their responsibilities, interfaces, and primary interactions.
User interaction	Describes the way user interacts with the system. User interfaces mock ups can be prepared here
Information	Describes the way that the architecture stores, manipulates, manages, and distributes information. Develops a complete but high-level view of static data structure and information flow.
Development	Describes the architecture that supports the software development process. Development views communicate the aspects of the architecture of interest to those stakeholders involved in building, testing, maintaining, and enhancing the system.
Infrastructure, Deployment	Describes the environment into which the system will be deployed, including capturing the dependencies the system has on its runtime environment.
Operational	Describes how the system will be operated, administered, and supported when it is running in its production environment.

4 PLUGGY Architecture

4.1 CONTEXT VIEW

Eliciting the different use cases, stakeholders and the requirements is used to sketch a basic PLUGGY functionality.

4.1.1 Use cases

The use cases are the basic functions of the system from the user's point of view. This UML diagram in Figure 4 shows the basic use cases for generic users. More detailed, stakeholder-dependent diagrams and specifications can be found in deliverables D2.3, D2.4.

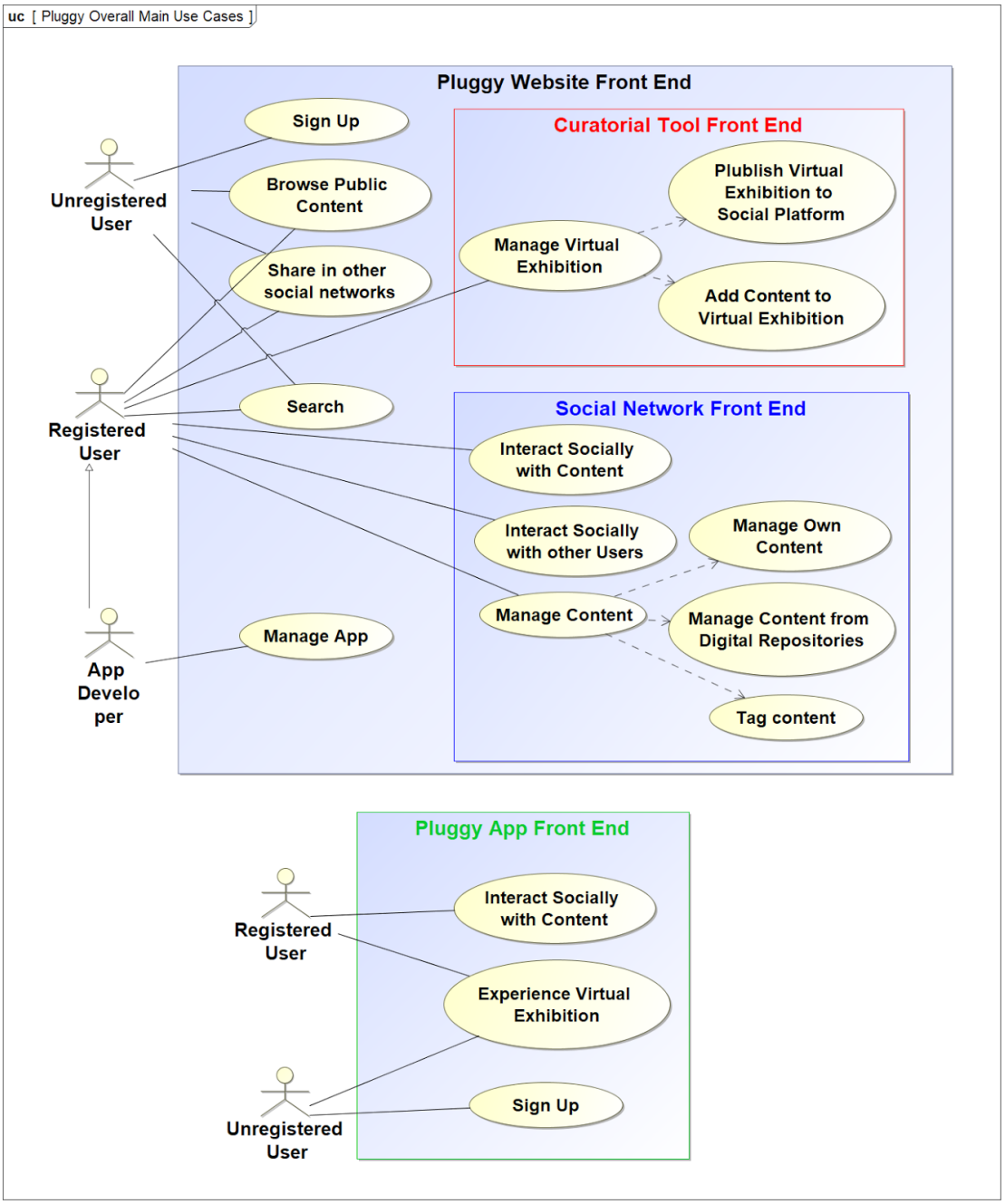


Figure 4: Basic Use cases in PLUGGY

Actors have been divided into two categories: the ones that are registered in the PLUGGY platform and the ones that are not registered. Within the registered users, the diagram also includes another subcategory to classify the developers of the pluggable applications. Every actor can browse and search public content, share links of the PLUGGY content in other social platforms and experience Virtual Exhibitions through a PLUGGY Application. Additionally, the registered user is allowed to: manage the content of the virtual exhibitions, interact socially with other users, manage and interact with the content of the social network content (his/her own content and content from digital repositories) and interact with the content of the PLUGGY Applications. The application developer is in charge of managing everything related with his/her PLUGGY Application.

4.1.2 User requirements

This subsection aims to describe briefly the methodology followed to collect the requirements and the requirements themselves. A Technical Requirements Specification is a document that communicates the requirements of the users to the technical community who will specify and build the system. The collection of requirements that constitutes the specification and its representation acts as the bridge between the two groups and must be understandable by both the users and the technical community. This type of communication generally requires different formalisms and languages.
The requirements elicitation process started along two main lines in the WP2 work package. The first was a questionnaire, which was filled in by all the stakeholders and experts. The second line was based on studying the FARO convention documents, which resulted in the Faro Guidelines for PLUGGY. From the analysis of all the data collected, a set of end user requirements is presented below. According to these very high-level user requirements, a set of system requirements was described. From the technical point of view, this is a more specific set of requirements which arise to meet the user requirements.

4.1.2.1 End user requirements

The requirements presented in this section describe the needs that users require from the system. They describe what the user does or need with/from the system. As it was mentioned before, this set of requirements are based on questionnaires and Faro guidelines and they are written using an accessible language. Resulting end user requirements can be seen in the Annex I of this deliverable.

4.1.2.2 System user requirements

This section presents a set of system requirements, which arise from the user requirements presented above, more specifically from the user social interaction needs. The set of system requirements is listed in the Annex II of this deliverable.

This set of requirements s considered as the building blocks that developers will use to build the system. For this reason, they have been described using specific formalisms, this is, using the traditional "shall" statements that describe what the system "shall do". Furthermore, in order to avoid the ambiguity or complexity, requirements are organized on different hierarchical levels. In this way, there are very general requirements which have rather the status of a heading contain more specific, more detailed requirements. This hierarchies are shown through the requirement ID.

4.2 LOGICAL VIEW

Based on the initial set of requirements and the use cases we propose the PLUGGY architecture as depicted on Figure 5. The main idea is to provide a Content Management Services system available via a well-defined REST API to all possible user tools. In our case example user tools will be the Green boxes in the figure:

- Social Platform
- Curatorial Tools
- Applications (AR, Geo, 3D)
- Games

All these user tools will be able to use any of the REST API interfaces. That for example means, that even if the Curatorial Tools are the main tool to create new stories, Games or other Applications can also publish their stories.

The Web Application Platform is a modular platform that allows usage of common functionality, like logging and security, for the Social Platform, Curatorial Tool and Application Developer Tool.

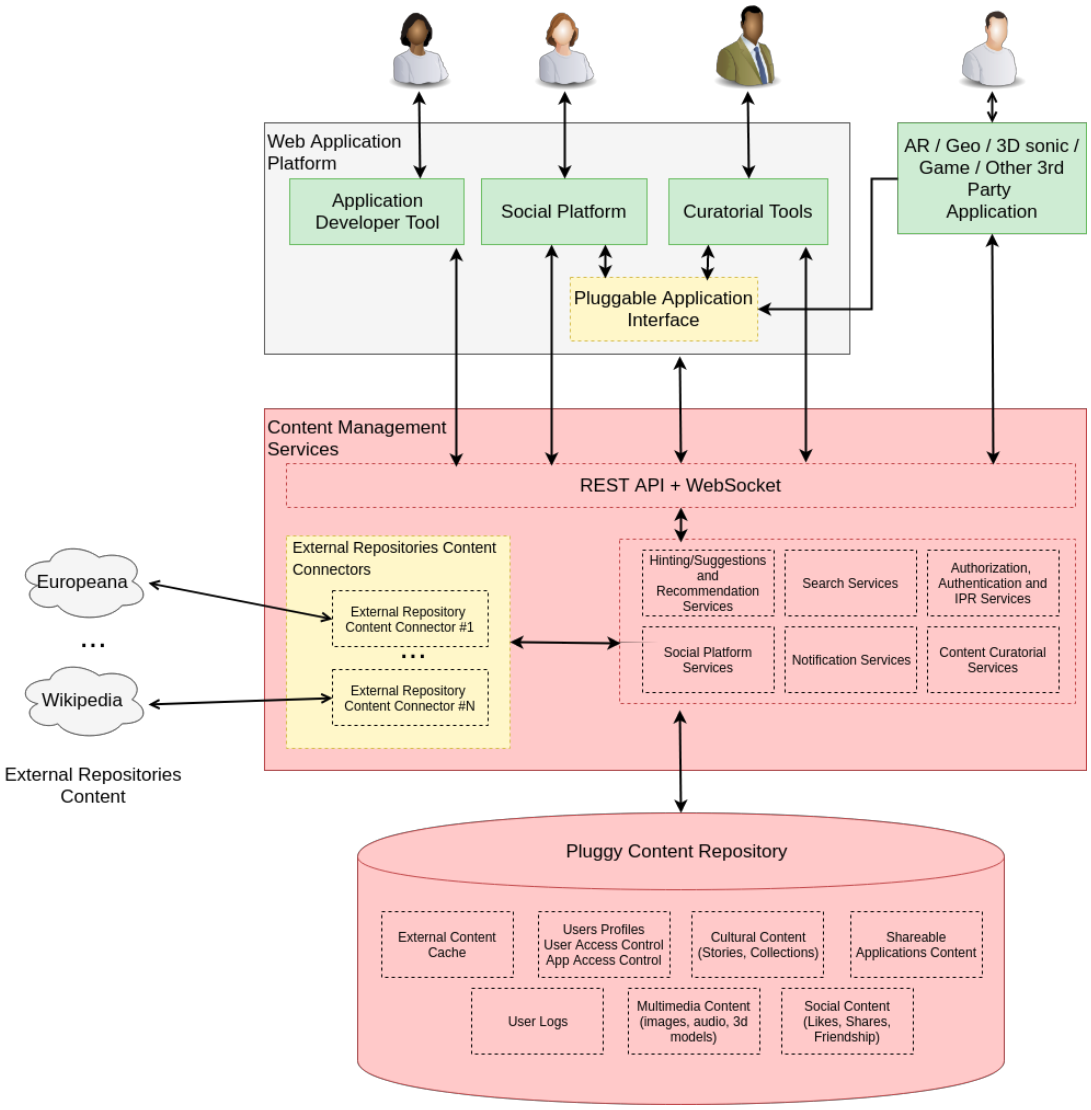


Figure 5 PLUGGY Logical Architecture

As described in the introduction of this deliverable, these green boxed user tools will be described mainly in their corresponding deliverables. However, the red boxes need to be clarified here. Let's call these red boxes the "Core PLUGGY Components". From the Figure 5 it is obvious that there are 2 main modules namely:

- Content Management Services
- PLUGGY Content Repository

Content Management Services will contain all needed software components to process, store and serve information to the external user tools via the REST API. These components are:

- External Repository Content Connectors
- Hinting/Suggestions and Recommendation Services
- Search Services
- Authorization,
- Authentication and IPR Services
- Social Platform Services
- Notification Services
- Content Services

All of these components access the PLUGGY Content Repositories for all possible information described later in the Information View section of this deliverable. Basic functions of all different components are described in the following text
Yellow boxes present two basic principles of extensibility of PLUGGY platform:

- Pluggable Application Interface - allow users to explore and create content of different types, that core PLUGGY application does not support
- External Repositories Content Connectors - allow users to reuse content from existing external content repositories

4.2.1 PLUGGY components

In the following table, we list all the components identified so far and their corresponding owner (consortium partner to implement main parts of it)

Table 3: PLUGGY components

Component name	Component owner
Web Application Platform	TUK
Social Platform	UMA
Curatorial Tool	ICCS
Application Developer Tool	TUK
AR App	UMA
Collaborative Game App	XTS
Geolocation App	CLIO
3D Sonic App	ICL
Content Management Services	TUK
PLUGGY Content Repository	TUK

In separate tables in Annex III of this deliverable, we briefly describe every component (including components described in more details in other deliverables) to get the overall picture of their functionality.

4.2.2 RESTful API

We have started developing the first version of the Content Management Services RESTful API. The API is described using the swagger tool. Swagger is the world's largest framework of API developer tools for the OpenAPI Specification (OAS), enabling development across the entire API lifecycle, from design and documentation, to test and deployment. In Swagger, users can see the documentation of the API as well as users can try API calls directly from the web page including a preview of responses. Proposed version of the API is depicted in **Error! Reference source not found.**. The API covers an initial set of interfaces used in our user interface mock-ups (described in the User Interaction view chapter of this deliverable). Current version of the API is available online https://develop.pluggy.eu/api/doc.

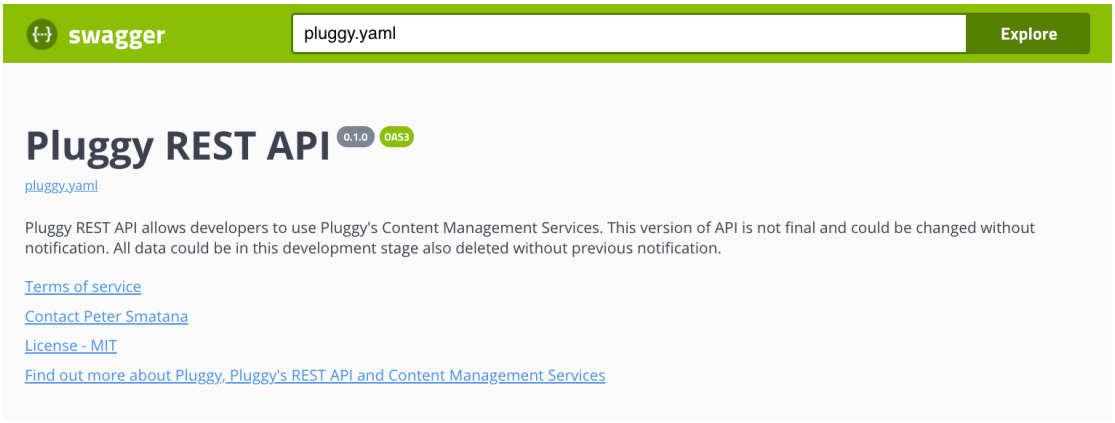


Figure 6 API documentation online

The main idea of the API is to provide a simple but powerful set of interfaces to interact with the PLUGGY. Interfaces mirror the basic component services of the PLUGGY Management Services. List of available interfaces is listed in the Annex IV of this deliverable.

4.2.3 Security

Few basic security concepts to secure user data were identified, which have to be implemented:

- Deployment server would be hosted on Unix-like OS, and therefore we are adopting some security best practices[5].
- PLUGGY is a pluggable platform, multiple applications could access users' content. For this functionality we are adopting the OAuth2[6]. concept. Users should enter credentials only via trusted PLUGGY's authentication interface and every application which want to access user's content has to be approved. PLUGGY will be compatible with other platforms, that support OAuth2 such as Google and Facebook. User could use their existing credentials for access to the PLUGGY platform.
- We are adopting the HTTPS protocol[7] () to prevent Man-In-The-Middle attacks

[5] https://wiki.centos.org/HowTos/OS_Protection.
[6] https://tools.ietf.org/html/rfc6749
[7] https://tools.ietf.org/html/rfc2818

4.2.4 WebSocket

The purpose of the integration of WebSockets[8] to PLUGGY is to provide better user experience. It allows the platform to provide real time notification to users about changes or updates in their favourite Stories. WebSockets also allow us to handle real time communication between users (chat).

4.3 USER INTERACTION VIEW

User interface mock-ups are presented in deliverables *D2.3 Social Platform Design and Specification* and *D2.4 Curatorial Tool Design and Specification*. Changes have been also made based on evaluation provided within WP6 specifically in D6.2.
These mock-ups were created to better communicate developer ideas to user partners of the project and to prove the basic architectural decisions. We call the current web interface the User Dashboard (which, for demonstration purposes, is a merge between the future curatorial and social platform tools). The page provides an overview of recently created stories for logged in user, his/her bookmarked stories and a collection of user's Virtual Exhibitions (as depicted on Figure 7). There is also a search bar and a button to create a new story.

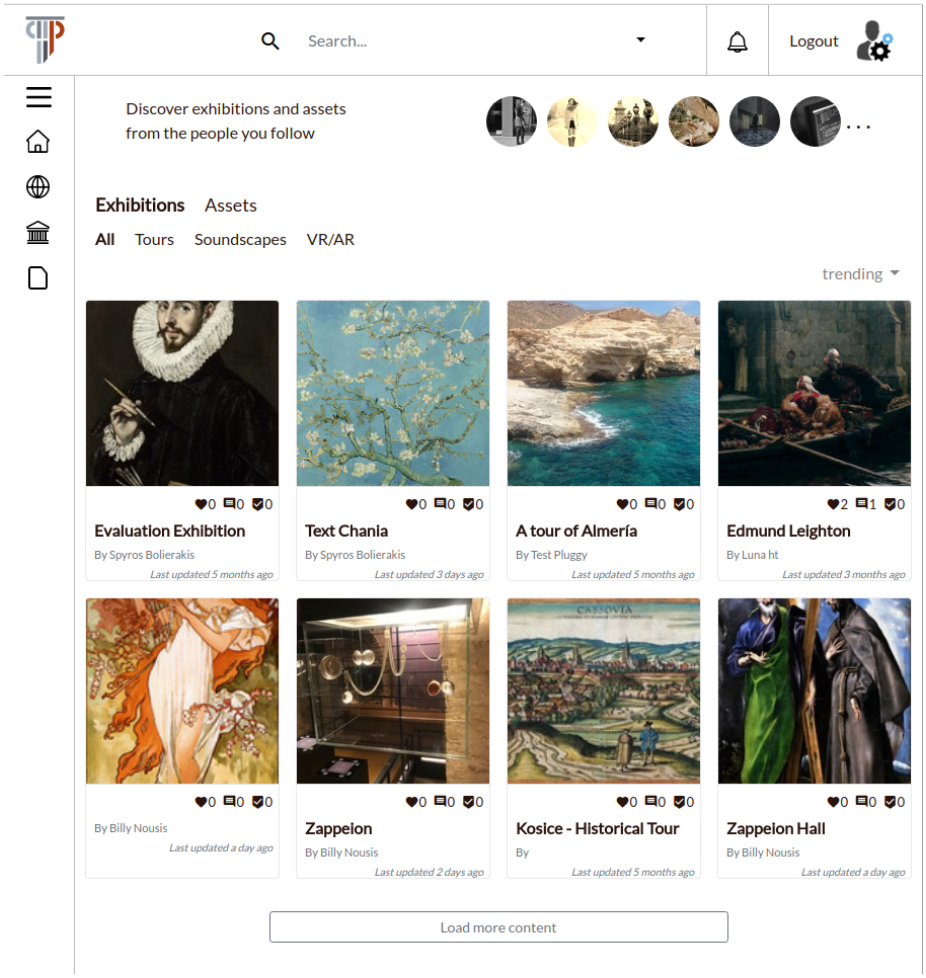


Figure 7: User's dashboard proposal

[8] : https://tools.ietf.org/html/rfc6455

4.4 INFORMATION VIEW

In this section, we are describing the information model depicting information we store, the relations between different pieces of information and some basic scenarios on how it is used to represent our needs. The following figures show the proposal of our information model. Individual class diagrams present different perspectives of our model, like content, social interaction and security.

Figure 8 shows the class diagram from content and user perspective. It also shows a security part of PLUGGY's data model. OAuth2 is introduced to secure app development. Users have to register their application, and subsequently they are able to use the REST API for their needs. The Approval, RefreshToken and Application classes are designed to support OAuth2. The data model contains class Developer which is inherited from the User. User with a Developer's role are allowed to integrate their own application (native or web) into PLUGGY platform if they fulfil API requirements (will be specified in WP4 developer guidelines).

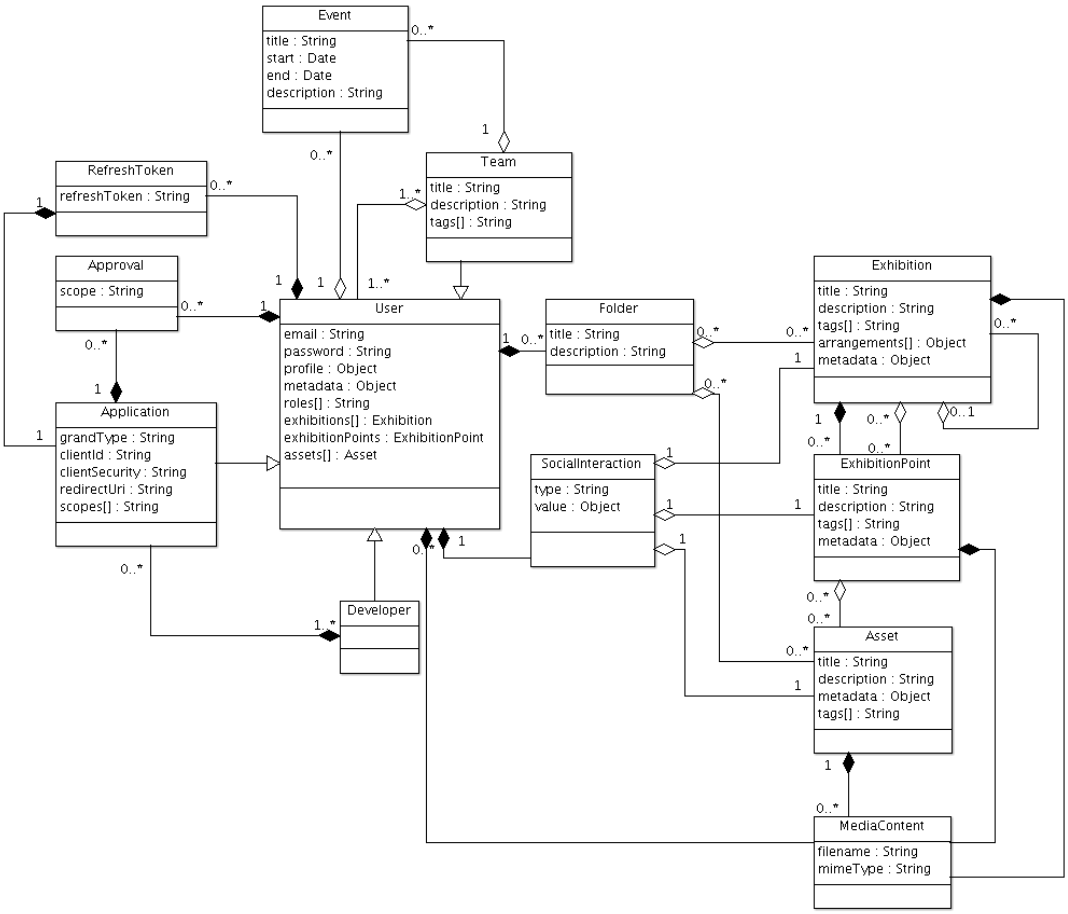


Figure 8: UML Class Diagram of content and user part of data model

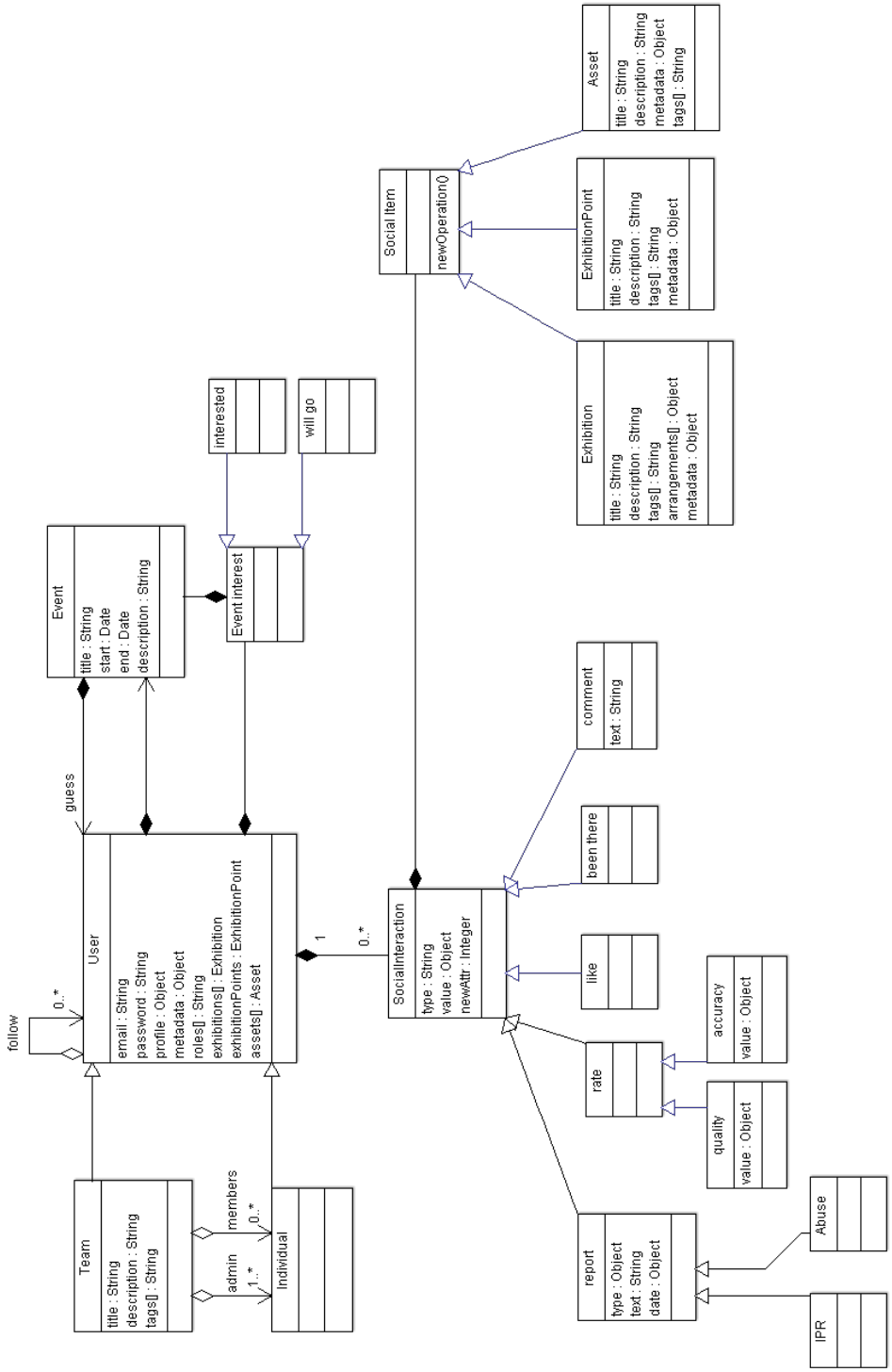


Figure 9: UML Class Diagram of social interaction part of data model

In Figure 9 we show different social interactions that can be performed on different user's content and events.

Here is a short description of each class of the proposed data model:

MediaContent: MediaContent cannot exist separately, it always should be part of an asset or an exhibition or and exhibition point, which can contain multiple MediaContent instances.

Asset: Asset is an elementary unit of content, that a user can interact with. The Metadata property can contain various types of information like geolocation, license, author, etc. Every asset can be a referenced by multiple Exhibition Points (see below).

Exhibition Points: Exhibition Point puts Assets into context. It represents a point of interest that aggregates multiple assets. It can be a part of a story that is sticking together a few assets. A set of thematic Exhibition Points can be part of an Exhibition. An Exhibition Point cannot be part of multiple Exhibitions. Reusability of an Exhibition Point can be made by its duplication. An Exhibition Point has to be a part of an Exhibition.

Exhibitions: Generally, an Exhibition is a set of Exhibition Points and Exhibitions arranged into some data structure. The simplest Exhibition is an ordered list of Exhibition Points (in this case no arrangement has to be defined). For the more complex exhibitions we have property arrangements which could contain multiple arrangements for exhibition data (e.g., blind people friendly exhibition for a geolocation application; or map definition for the game application). Please note that Exhibitions can be infinitely nested, which is not a problem, as search in Exhibitions could be handled by cache.

User: Every registered user is able to create his own Exhibitions, Exhibition Points, Assets and upload Media Content. For clarity, these associations in UML diagram are not shown. Users' content is organized by tags which are assigned to these entities. Users' content is organized by tags for better accessibility.

Folders: Every user could organize his/her content into folders also with interesting content from others. Thus, Folders work also as bookmarks.

Social Interaction: Users are able to like, rate, indicate that they have 'been there', comment and report Assets, Exhibition Points and Exhibitions.

Team: User could be a member of multiple teams.When creating content, a User can choose if the content belongs to them or to one of their Teams. A Team is just a special type of users, which means that Teams can interact socially with other Users.

Event: User or Team is able to create events that could be presented to PLUGGY users.

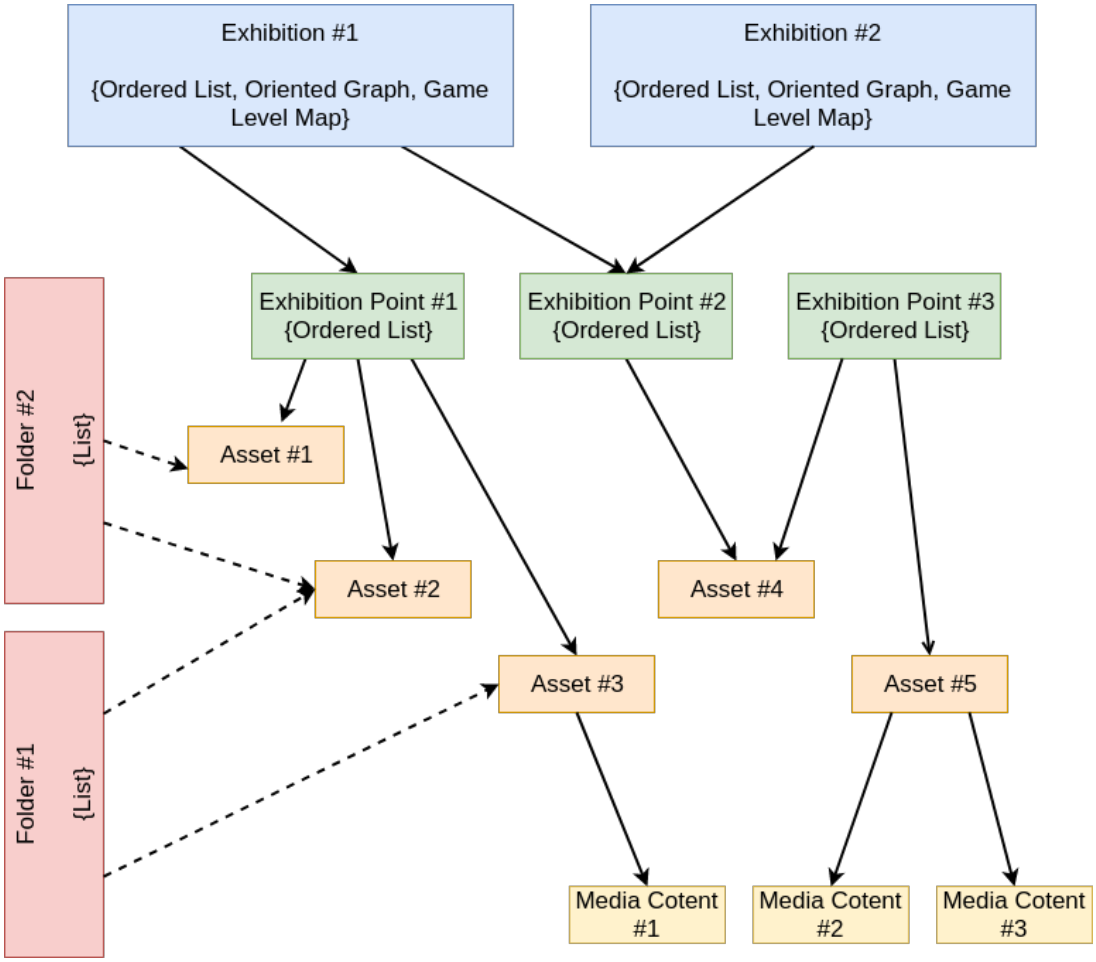


Figure 10: Example of instances

In Figure 10 we see several instances of Exhibition Point linked with different Assets and contained in different Exhibitions.

The following diagrams try to model instances of the part of The Chimneys Tour. First one shows all stored data in repository and the next are related to specific applications like Geolocation App, Augmented Reality App, etc.

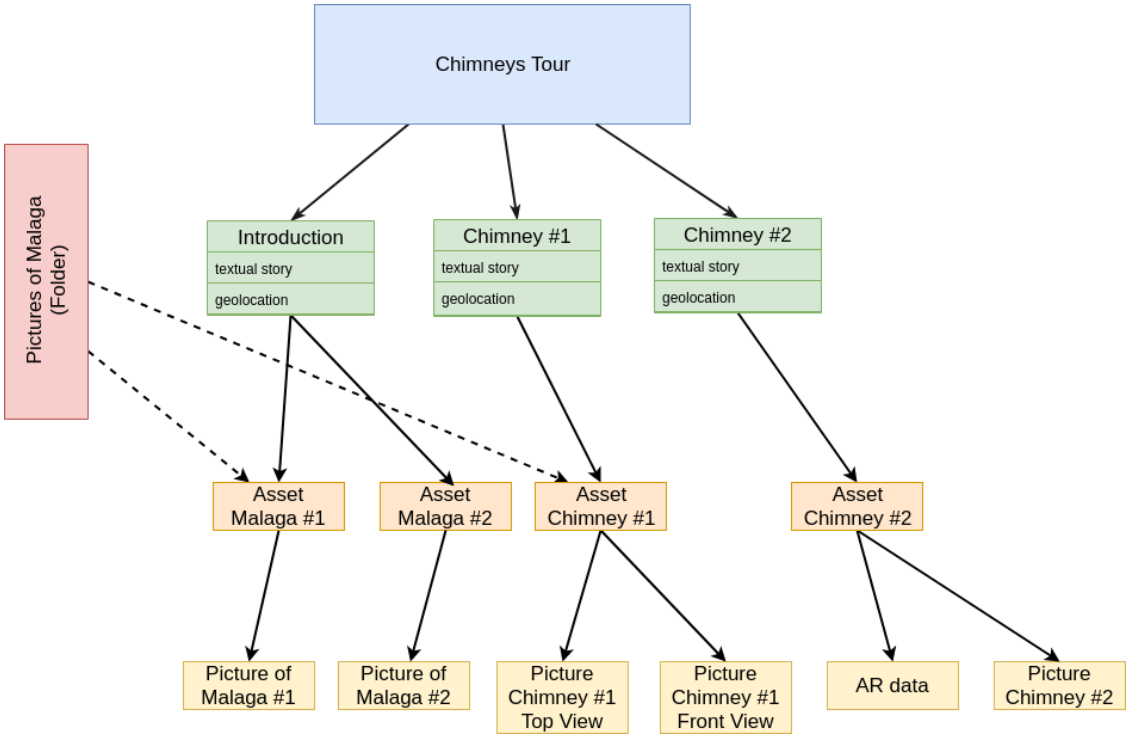


Figure 11: Chimneys Tour – stored data

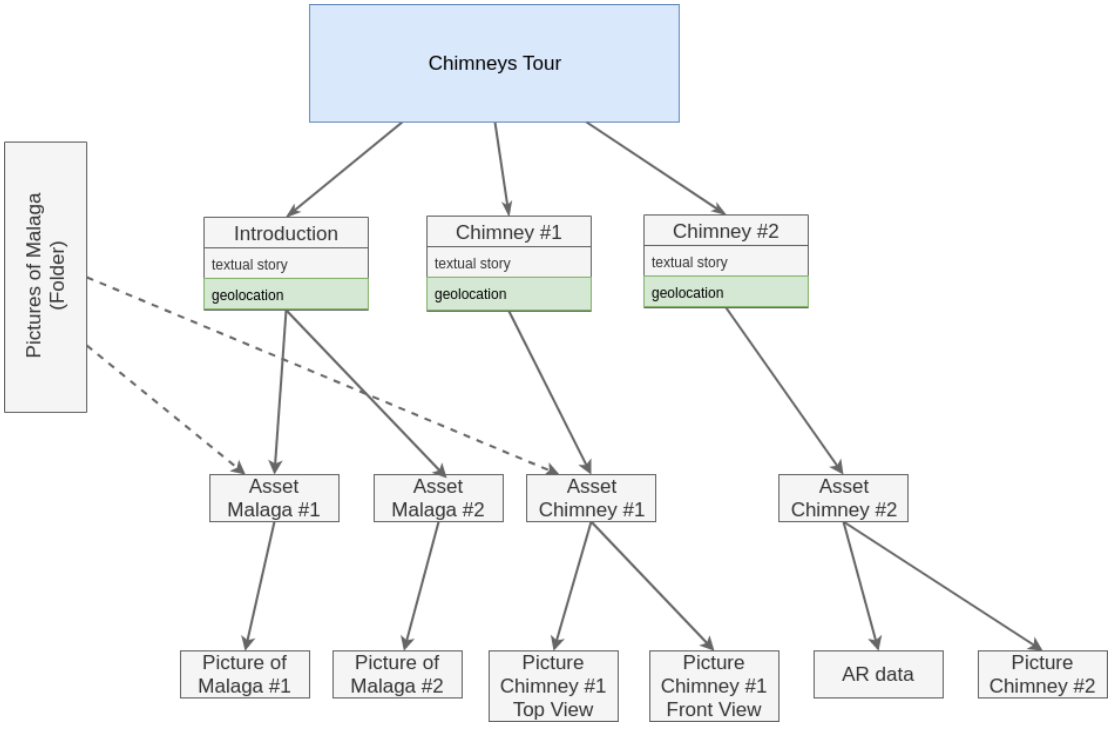


Figure 12: Chimneys Tour - Geolocation App

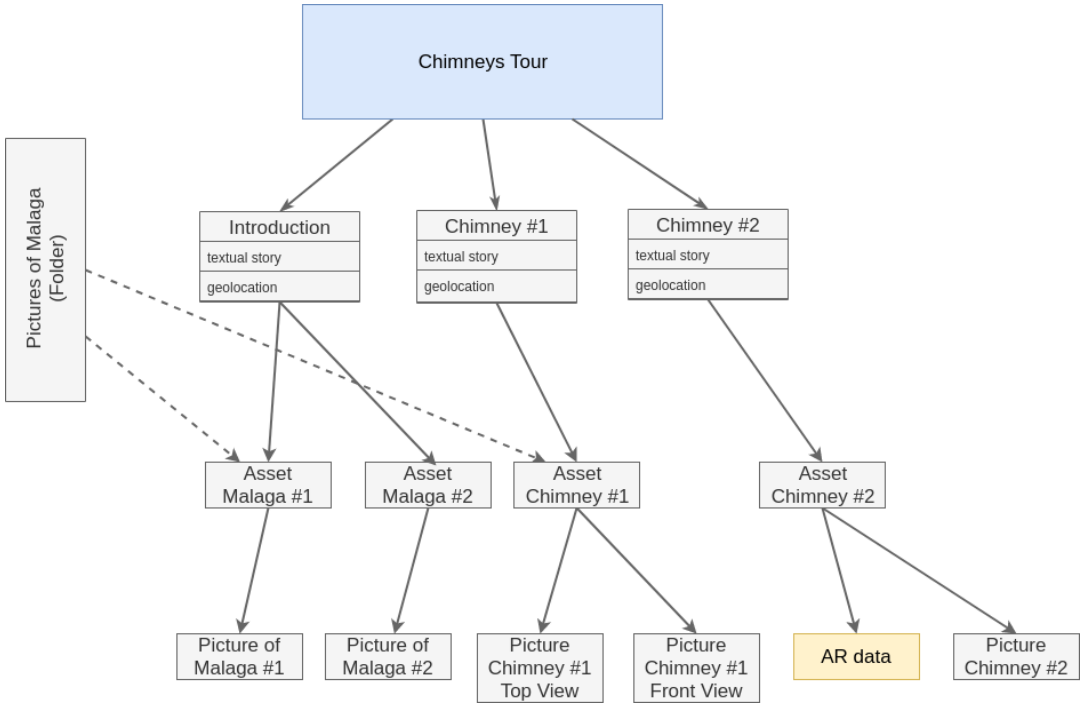


Figure 13: Chimneys Tour - Augmented Reality App

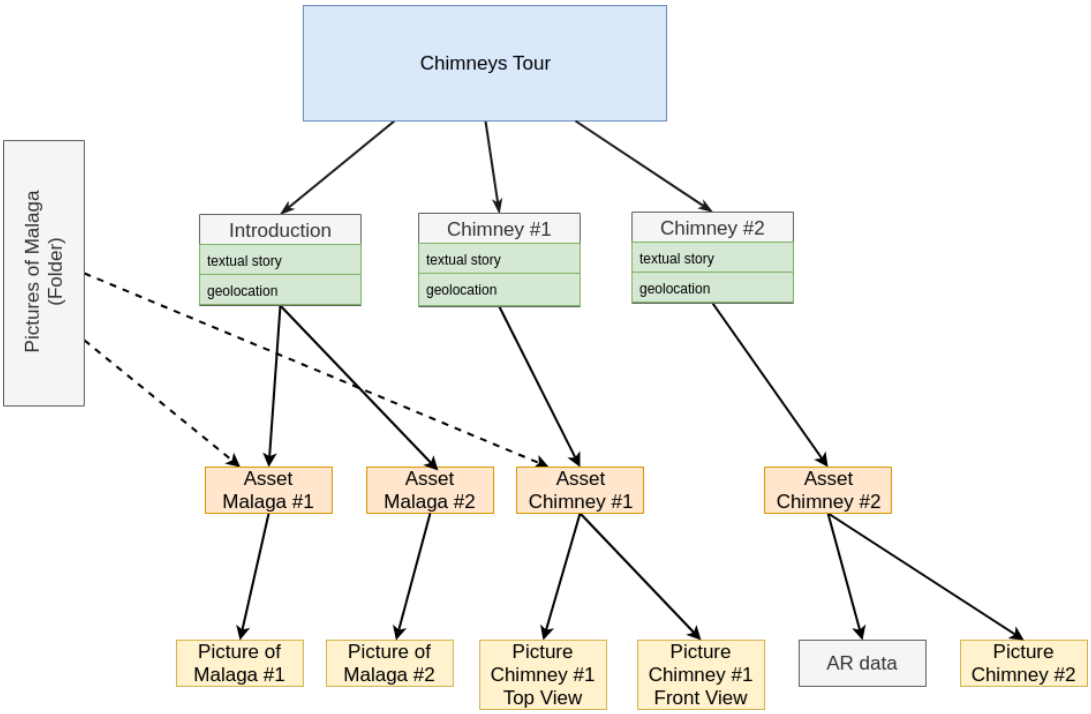


Figure 14: Chimneys Tour - Virtual Exhibition on Social Platform

4.5 INFRASTRUCTURE, DEPLOYMENT VIEW

The server to run the software will be cloud based but, on partners premises during the project. Figure 15 shows single server deployment of PLUGGY platform. This deployment is minimal and can be easily extended to the scalable solution displayed on Figure 16. This minimal solution is deployed onto one single HW server where all individual components of PLUGGY run and Apache HTTP server creates a unified entry point for the clients. This deployment is planned also for the final demonstration of the project (see T7.5). The software is designed to be scalable and very easily deployed onto multiple HW servers. The extended deployment architecture shown on Figure 16 protects the PLUGGY platform from a single point of failure, as all components of the system could be duplicated and in case of failure of the component. Its functionality will be automatically handled, in real time, by other components with the same functionality.

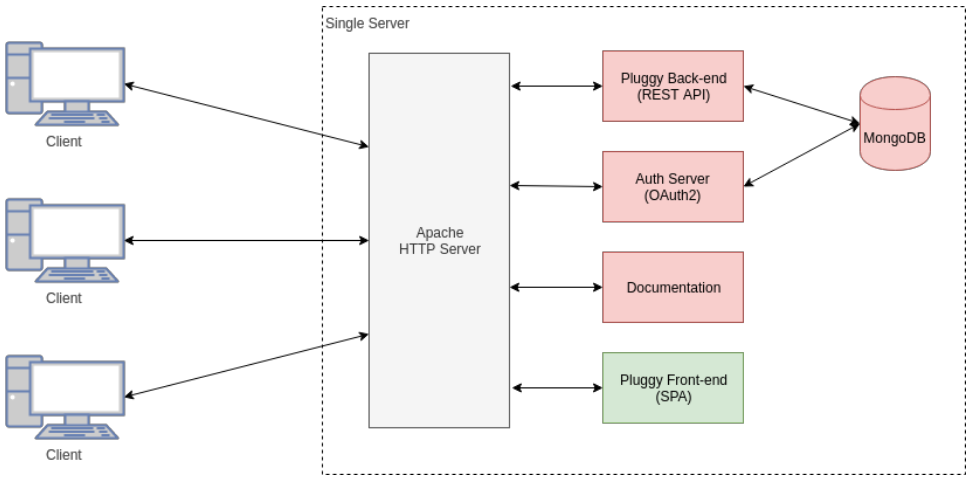


Figure 15: Single Server Deployment

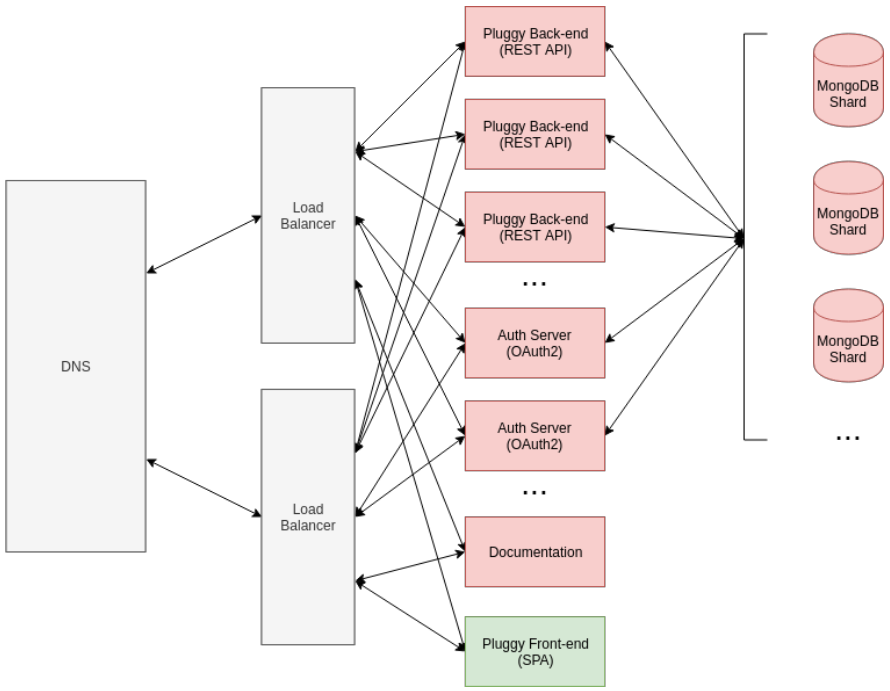


Figure 16: Scalable Deployment Architecture

Deployment process of individual components is described in PLUGGY's source code repositories:

- PLUGGY Back-end + Auth Server + Documentation: https://isense-gitlab.iccs.gr/PLUGGY/pluggy-core
- PLUGGY Front-end: https://isense-gitlab.iccs.gr/PLUGGY/pluggy-web

Individual SW components are platform independent, and can be deployed onto servers with different OS (MS Windows, Unix based OS, MacOS)

4.6 DEVELOPMENT VIEW

4.6.1 Development environment

Every user from PLUGGY's development team is using git[9] – a version control system. Currently we are using private ICCS version of the git server (https://isense-gitlab.iccs.gr/). Public release is awaiting a final decision on the software license (see Deliverables emerging from WP8). Once this decision is made, PLUGGY's git repository will be upgraded to a fully public repository.

The Git platform not only provides version control system, but also multiple tools to support the development process:

- continuous integration
- issue tracking
- wiki documentation

There are currently two running instances of PLUGGY, deployed on a virtual Ubuntu server:

- http://develop.pluggy.eu/ - the latest unit tested build from develop branch
- http://mockup.pluggy.eu/ - the latest integrated version of PLUGGY platform ready for user testing

4.6.2 Integration

The PLUGGY system is being developed in a distributed development environment, i.e. a development environment in which different teams in different locations are involved, each of them responsible for developing a part of the system. In this context, it is helpful to define an integration process, which defines a workflow that aims to avoid integration issues by facilitating frequent, partial integrations of the different parts as development progresses.
The integration model is such that when a developer uploads code changes to a development branch of the front-end or back-end repositories, these changes are automatically deployed to a development server, without manual intervention. In more detail:

- A developer works in a develop branch of their local code repository. They are able to see and test code changes thanks to their development environment and

[9] https://git-scm.com/

their local development web server to which they can continuously deploy and test.

- Once they are satisfied with the code changes, they push these changes to the develop branch of the PLUGGY source control version server (develop branch of the Gitlab repository).
- The PLUGGY Continuous Integration Server, detects these changes in the develop branch, and subsequently:
  - Builds the code.
  - Makes automated unit tests.
  - Notifies fail or success to the developers.
  - Publishes the new version into the develop web server.
- Much less frequently, and manually, TUK merge the code onto the master branch the PLUGGY repository and deploy new versions onto the PLUGGY public production web server.

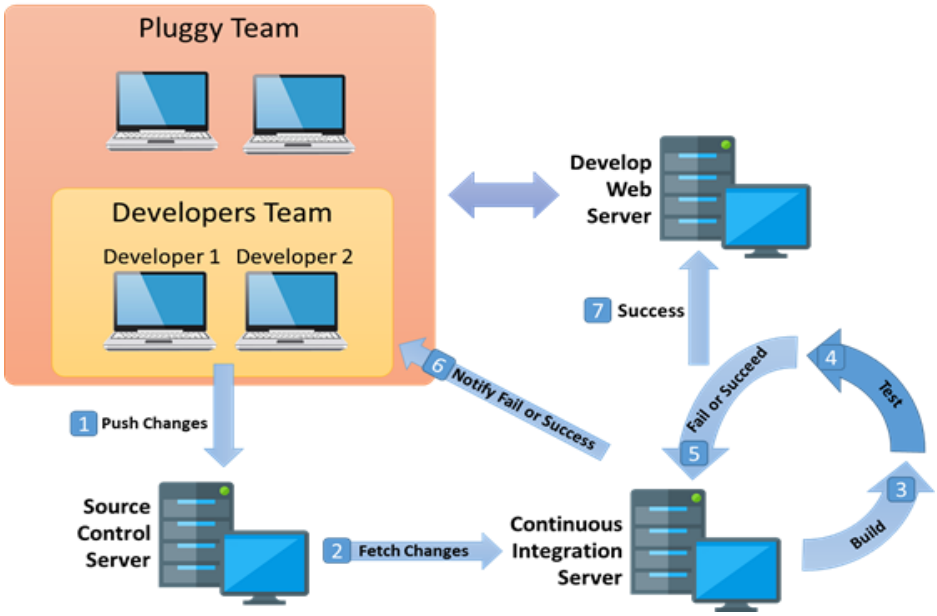


Figure 17: PLUGGY integration model diagram

4.6.3 Progress tracking

The development of PLUGGY systems is carried out through a joint effort of different organizations. For this reason, it was necessary to establish methodologies and tools that allow us to know the state of development of each of the elements and to discuss the necessary refinements.

The methodology to follow establishes a lifecycle to track the implementation requirements, assigning each requirement to a particular stage of this lifecycle. A requirement will progress forward through the stages if everything goes according to plan, but is also allowed to jump backwards due to re-definition. As can be seen in the:

- *Back-log:* this is the first stage of a requirement. Implementation of requirements in the Back-log has not yet started, so these requirements are therefore subject to important changes if necessary.

- *Discuss:* At this stage, a conversation is opened between the different stakeholders to clarify how the implementation of a requirement will be, before starting its development.
- *ToDo*: the requirements at this stage are ready to begin their development. When a requirement has started to be developed, it goes into *Doing* state.
- When implementation has finished, tentatively, it has to be evaluated (stages *Ready for Evaluation* and *Evaluation*). After the evaluation of a requirement, it may need changes which have to be implemented, or it will go into the *closed* state if its development is terminated.

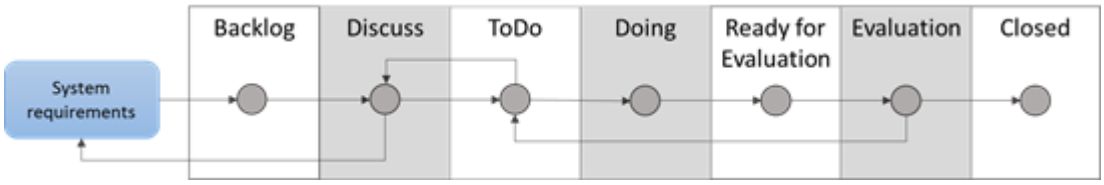


Figure 18: Stages in the requirement implementation life-cycle

The labels of the GitLab Issue system are used to express the stage at which each requirement is at a given moment. The Issue Board allows to have a bird's eye view of the current state of the project.

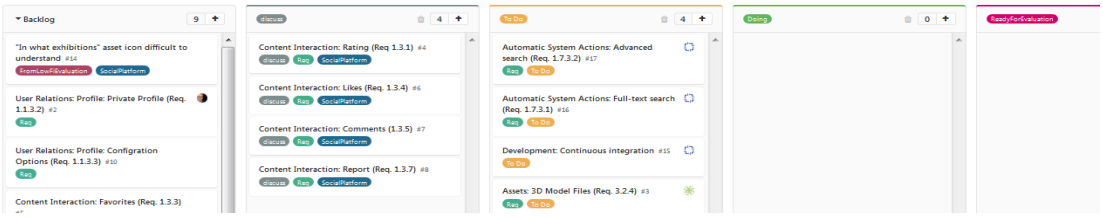


Figure 19: GitLab issue board

In order to bootstrap the methodology described above, we will create an issue for each one of the system low-level requirements. These issues or requirements will go through the different stages by using the label system of the tool, just moving the issues (drag and drop) from one stage to other.

GitLab issues offer other functions like commenting, attaching of images and other multimedia content associated with each issue. This allows you to carry out the collaborative process of refining each of the requirements, before its development, and later to fully track their life cycle.

4.6.4 Open Source Licensing

Software licensing is described in the deliverable D2.5 IPR Report and D8.3 Exploitation Plan.

4.7 OPERATIONAL VIEW

PLUGGY platform should be installed based on recommendations from section 3.5 (Infrastructure, Deployment View). When system is correctly installed all the functionality could be managed from admin's zone accessible via web browser. Main functions accessible via admin's zone are:

- User management – managing users in GDPR compliance way; export of user's data; banning of trolling users; etc.
- Content management – managing reported content due to IPR or inappropriate content.
- Application management – management of 3rd party applications; banning of inappropriate applications; managing of trustworthy applications
- System management - managing of automatic backups; exporting of PLUGGY's data; viewing basic usage statistics based on user logs.

Selected users will have access to administration zone and will be educated to correctly manage content published to PLUGGY platform with strong focus on GDPR and IPR issues.

PLUGGY platform is opened to all developers who can integrate their applications into it. Developers' zone in web application is prepared for them to simplify the development and integration process. Detailed information will be provided within D4.5 Guidelines and Instructions for PLUGGY Apps.

5 Conclusions

The PLUGGY Platform is a pluggable platform allowing any future applications to use its services for sharing cultural heritage. Thus, we proposed in this deliverable the architecture for the system that defines a clear set of RESTful APIs of a central content management services to allow it. The platform is being developed currently based on this architecture and will be deployed for trial installations soon.

Annex 1 - End user requirements

Id	Name	Descriptions	Source Rationale
22	Privacy	The privacy of all stakeholders should be assured	Questionnaire
23	Quick access to my files	Users should be able to quickly access content uploaded to the system	Questionnaire
27	Collect and enrich heritage	Stakeholders should be allowed to collect and enrich local and national heritage and cultural vitality	Questionnaire
28	Asset portraits	Stakeholders may desire to make a "portrait" of their favorite cultural heritage asset	Questionnaire
29	Revive and celebrate	Some stakeholders may need to collect, conserve and celebrate their history and revive the heritage of their place	Questionnaire
31	Identity	Some stakeholders may have the desire to validate or provide a greater sense of a community's heritage or identity	Questionnaire
32	Community narratives	Stakeholders may have the desire to dive into a community's narratives and diverse culture	Questionnaire

34	Traditions	Some stakeholders might have the desire to connect to certain cultural traditions, social values, beliefs, religions and customs	Questionnaire
36	Awareness of threats dangers	Stakeholders interested in heritage need help to raise awareness of threats and potential dangers in cultural heritage and assets	Questionnaire
38	Time evolution	Ability to adjust the content (or a view to an asset) as time progresses	FARO Guidelines for PLUGGY
39	Accessibility to all	The platform and the curatorial tool must be open to all who wish to become members of the heritage community without any exclusions (with the caveat, elaborated further on, that participants must abide by the rule of recognizing the right of other participants to promote their values and aspects of cultural heritage).	FARO Guidelines for PLUGGY
40	Forums	The system shall support forums where discussion is actively encouraged	FARO Guidelines for PLUGGY
41	Support a savoir-vivre	Certain restrictions are unavoidable in terms of material that may be presented in the platform. The civility and decorum expected for participation in a public setting should be encouraged and enforced either through the use of moderators or self-reporting mechanisms.	FARO Guidelines for PLUGGY
42	Cooperation between public authorities and heritage actors.	The Platform should allow all heritage actors (i.e. public authorities, experts, owners, investors, businesses, NGO's and members of the civil society) to come together and exchange knowledge, good practices and expectations.	FARO Guidelines for PLUGGY
43	Enable the design or complement educational material	The Curatorial Tool should have an educational role and allow educators and students to take advantage of digital technologies to design or complement educational material.	FARO Guidelines for PLUGGY
44	Promote and encourage the adherence to the highest standards of content quality	It should promote and encourage the adherence to the highest standards of content quality.	FARO Guidelines for PLUGGY
45	Combat illicit trafficking in cultural property	It should employ methods to combat illicit trafficking in cultural property. Co-operation with databases that attempt to curb the trade in illicit cultural goods seems essential.	FARO Guidelines for PLUGGY

46	Available in the greatest diversity of languages	It is essential that both the Social Platform and the Curatorial Tool are available in the greatest diversity of languages that is feasible, to combat the linguistic hegemony that accompanies globalization and that often leads to the marginalization of entire groups of heritage communities.	FARO Guidelines for PLUGGY
50	Community Content review	All created content could be reviewed by PLUGGY community	

Annex II. - System user requirements

Id	Name	Description	Refined by use case
1	Social Interaction	The system shall implement features to promote social contacts/relations between users and interaction between users and contents. Such as friendship, user groups, recommendations, notifications, favourites, and item rating.	
1.1	User Relations	The system shall allow the relations between users and group of users	Interact socially with other users
1.1.1	Subscriptions	The system shall allow users to make subscriptions to groups or events	
1.1.2	Friends	The system shall allow users to make virtual relation of friendship with other users	
1.1.3	Profile	The system shall allow users to manage its user profile information. This profile information will be divided in, public information (which is visible for everyone), private information (which is visible for nobody) and options of configuration (such he/she wants to receive notifications or not).	
1.1.3.1	Public Profile	The system shall keep visible to all users of PLUGGY, logged or not, certain user profile information, such as nick name, picture.	
1.1.3.2	Private Profile	The system shall keep certain user profile information private, such as email address, aged or date or birth.	
1.1.3.3	Configuration Options	The system shall allow users to configure certain settings of the system. For example, if they want to receive notifications of events or not, or if they only want to receive it when it has been generated by your friends or groups to which they are subscribed.	

1.1.4	Groups	The system shall allow users to create and participate in groups with common interest (e.g. Participate to a group of people of similar cultural background)	
1.2	User Interaction	The system shall implement mechanism to allow and promote interaction between users with the same or different profiles. The system shall allow user to deal with other users, sending messages and events.	Interact socially with content Interact socially with other users
1.2.1	Reputation	The system shall implement mechanism to measure and show the user overall position in terms of visibility, loyalty and reliability.	
1.2.2	Certifications	The system shall implement mechanism to allow user to certify and validate another user content.	
1.3	Content Interaction	The system shall implement mechanism to allow and promote interaction between users and the social platform content. The content interaction requirement category includes all the requirements associated with the management and socialization of contents, such as uploading, rating, tagging, and so on.	Manage content
1.3.1	Rating	The system shall allow users to assign a rate to a content, according to a predefined scale of values.	
1.3.3	Favourites	The system shall allow user to tag content as preferred, so as to see it highlighted or in special section.	
1.3.4	Likes	The system shall allow users to indicate that they like specific content	
1.3.5	Comments	The system shall allow users to add comments related to specific content	
1.3.7	Report	The system shall allow user to report inappropriate content (IPR or other)	
1.3.8	Keywords	The system shall allow users to assign keywords to content, either from a predefined set of keywords or as free text.	
1.3.9	Stats	The system shall provide mechanism for informing users about statistics related to social interaction of both users and content. That is, likes, views, visits, etc.	
1.5	Content Access Rights	The system shall implement mechanism to assign access rights to content published on the social network. The granularity can be at the user level, at the group level, or at the user category level (e.g., owner, friends, others).	Manage Content

1.6	Events	The system shall allow users to share cultural events and schedules with friends or within groups.	Interact socially with other users
1.7	Automatic System Actions	The system shall provide recommendations, notifications, statistics, and search capabilities.	
1.7.2	Recommendations	The user shall provide mechanism to make recommendations about content that could be of interest to users. These recommendations shall be based on the user profile, preferences, and tastes.	
1.7.4	Notifications	The system shall send notification messages to users who have enabled so in their profile. These notifications will inform users about news and events.	
1.8	Search	The system shall allow users to search content by a specific topic (e.g. Search for previous generations and the history of where you come from; explore the heritage and history of a place or destination)	Search
1.8.1	Full-text search	User shall be able to search stories/collections/events/users via full-text.	
1.8.2	Advanced search	User is able to create more complicated queries related to metadata attributes (creation period/type of artefact/author/location/...) of stories/collections/events/users.	
1.9	Extension Support	The system shall implement mechanism to manage plugins, import/export features, and location-based services.	Manage App
1.9.1	Plugins	The system shall allow the increase of the features of the social network by adding new widgets, plugins or components	
1.9.2	External Social Synchronization	The system shall synchronize the social content from different platforms or machines (e.g. Users could share events/stories/collections directly to Facebook, Instagram, Twitter...)	
1.9.3	Location	The system shall allow users to add geolocalisation information to their content	
2	Virtual Exhibitions	Users shall be able to manage virtual exhibitions which make use of their assets and other users' assets.	
2.2	Assets Usage	The Virtual Exhibitions shall be directed graphs. The nodes in these graphs are Asset Usages. Each Asset Usage points to one or several assets and contains other metadata (see sub requirements).	Add content to virtual exhibition
2.2.1	Narrative Stories	Users shall be able to create/delete/modify their stories.	

2.2.2	Geolocation	The Asset Usages shall contain a Geolocation attribute.	
2.2.3	App Metadata	The Asset Usages shall have customizable attributes that can be used by Apps for experiencing Virtual Exhibitions (see 2.6).	
2.2.4	Timestamp	The Asset Usages shall contain a Timestamp attribute.	
2.3	Virtual Exhibition Types	The PLUGGY system shall support natively a predefined set of Virtual Exhibitions (see sub requirements).	Manage virtual exhibitions
2.3.1	Timeline Exhibition	The PLUGGY system will allow creation and management of Virtual Exhibitions containing a succession of timestamped exhibition points.	
2.3.2	Map Exhibition	The PLUGGY system will allow creation and management of Virtual Exhibitions containing an arrangement of geolocated asset usages/exhibition points.	
2.3.3	Media Exhibition	The PLUGGY system will allow creation and management of Virtual Exhibitions containing text and basic media such as images and video.	
2.3.4	Game Exhibition	The PLUGGY System will allow the creation and management of basic video games.	
2.3.5	Soundscape Exhibition	The PLUGGY system will allow creation and management of exhibitions containing spatialised 3d audio.	
2.3.6	VR/AR Exhibition	The PLUGGY system will allow creation and management of exhibitions displaying 3D content.	
2.4	Crowdsourced Virtual Exhibition	The system shall provide mechanisms to allow for crowdsourcing of virtual exhibitions (e.g. group ownership of exhibitions, nested exhibitions).	Manage virtual exhibition
2.5	Curatorial Tool	The system shall offer the users a centralized tool to curate virtual exhibitions, called the Curatorial Tool	
2.5.1	Suggest Virtual Exhibition Types	The Curatorial Tool shall suggest to users the types of virtual exhibitions that can be created within PLUGGY.	
2.5.2	Suggest Suitable Assets	The Curatorial Tool shall suggest what assets (types, ownership) to use in the Virtual Exhibition	
2.5.3	Edit Narrative Stories	The Curatorial Tool shall allow users to manage narrative stories associated to Asset Usages	
2.5.4	Edit Virtual Exhibition Types	The Curatorial Tool shall allow full creation (natively) of some types of Virtual Exhibitions (see sub requirements)	
2.6	Experience of Virtual Exhibitions	The Virtual Exhibitions shall be experienced on the PLUGGY Website and through the Pluggable Apps.	

2.6.1	Preview	The Virtual Exhibitions shall be previewed on the PLUGGY Website.	
2.6.2	Full Experience	The Virtual Exhibitions can be experienced fully, depending on type, by the Pluggable Apps.	
2.7	Virtual Exhibition Access	The system shall allow shared ownership of a virtual exhibition so that several users can create collaboratively a virtual exhibition, sharing private drafts of the virtual exhibition before it is published.	
3	Assets	The system shall support creation, deletion and modification of units of content called assets. Each asset has an owner (the user or group that creates the asset), title, description, metadata and media files (see sub requirements). The asset is the smallest, searchable unit of content in PLUGGY.	
3.1.1	Asset Metadata	The assets shall include a number of Metadata.	Manage content
3.1.2	Geolocation	The system will allow the user to introduce a geolocation associated with an asset.	
3.1.3	Keywords	The system shall allow the users to introduce a set of keywords (tags) that allow classification of a given asset. The front-end will suggest existing keywords to use.	
3.1.4	Standardizatio n	To Be Described	
3.2	Asset Media	The system shall allow for each asset to have more than one media files. A number of media types will be natively handled by the PLUGGY website (see sub requirements)	Manage content
3.2.1	Image Files	The system shall allow uploading and rendering of image files in formats JPG, PNG, GIF and BMP.	
3.2.2	Audio Files	The system shall allow rendering of audio files in various formats	
3.2.3	Video Files	The system shall allow rendering (play and pause) of video formats TBD	
3.2.4	3D Model Files	The system shall allow rendering of 3D models of OBJ format, including the associated texture and material files	
3.2.5	Text Files	The system shall allow rendering of text files of PDF Format	
3.2.6	Media Links	TBD	
3.3	External Asset Repositories	The system shall allow import of assets from existing Asset Repositories (see sub requirements)	Manage content from Digital Repositor ies

3.3.1	Europeana	The system shall allow import of assets from Europeana (see https://pro.europeana.eu/resources/apis)	
3.3.2	IAPH	The system shall allow import of assets from IAPH (see http://repositorio.iaph.es/)	
3.3.3	History Pin	The system shall allow import of assets from History Pin (see http://www.historypin.org/resources/docs/api/site/index.html)	
3.3.4	Wikipedia	The system shall allow import of assets from Wikipedia (see https://www.mediawiki.org/wiki/API:Main_page)	
3.4	Asset Access	The system shall allow setting the access level of assets to public/private	Manage content
4	Smart Collections	The system shall allow users to tag their assets or virtual exhibitions with any word. All the items (assets or exhibitions) that share a tag are said to "belong" to a Smart Collection. A Smart Collection shall be identified by a tag (or combination of tags). However, an authenticated user sees two such smart collections: the smart collection formed by the assets he/she owns with such a tag (or combination) AND the smart collection formed by all the assets in PLUGGY with such a tag (or combination).	Tag content
19	Accessibility for low vision	The system shall support high contrast visibility for vision impaired	
49	Intellectual Property Right	The system shall make clear on with what conditions of IPR the user is sharing content	
49.1	External content intellectual property	The system shall ensure the intellectual property of the content of external networks	
49.2	User contribution intellectual property	The system shall ensure the intellectual property of the users' contribution	
51	Support for Multilanguage	The system will support several languages to display and enter content.	
51.1	Multilingual Website	The system must display its web interface in the language requested/desired by the user.	
51.2	Support for Multilingual Content Edition	The system must allow the user to enter content in different languages.	

Annex III. - PLUGGY components

Table 4: Web Application Platform

Component name:	Web Application Platform
Description:	The Web Application Platform will: ● Provide common modular platform for Social platform, Curatorial Tool and Application Developer Tools ● Provide security middleware ● Provide web components shareable between different modules ● Provide common PLUGGY layouts ● Allow modules to use Pluggable Applications Interface
Dependencies:	Content Management Services
Technologies:	● Angular5 ● HTML5 ● CSS ● TypeScript
Data (in/out/storing):	IN - user credentials, access token, user activity OUT - access token, application usage logs STORING - access token (client browser local storage)
Main Technical Use Cases	TBD

Table 5: Social Platform component

Component name:	Social Platform
Description:	Specified in D2.3 The Social Platform will: ● Allow users to access to the PLUGGY content ● Provide a visual representation of the virtual assets ● Allow users to introduce media content ● Allow users to access Virtual Exhibitions ● Allow users to search in PLUGGY overall content ● Allow users to create social networks and communities ● Allow to link with other social platforms Support content exchange with the content management services
Dependencies:	Content Management Services
Technologies:	Specified in D2.3 ● Angular 5 ● HTML 5 ● CSS ● JavaScript ● Typescript 3D content viewer based on Unity3D and webgl.
Data (in/out/storing):	IN : user credentials, access token, user activity, contents (assets), social stats, notifications

	OUT : user credentials, new content, content tags, content metadata, social activity, etc
Main Technical Use Cases	See Deliverables D2.3-4

Table 6: Application Developer Tool Component

Component name:	Application Developer Tool
Description:	The Application Developer Tool will: ● Provide user interface for administrator to manage pluggable applications ● Provide reporting mechanism for the users of PLUGGY to report issues with pluggable applications ● Enable developers of applications to register their applications to PLUGGY ● Provide guidelines and set of examples for application developers
Dependencies:	Web Application Platform Content Management Services
Technologies:	● Angular5 ● HTML5 ● CSS ● TypeScript
Data (in/out/storing):	IN - list of registered applications OUT - registered application metadata, users' reported issues STORING - -

Table 7: Geo App component

Component name:	Geo Apps
Description:	Specified in D4.5 The Geo App will enable users to: ● Add individual pins ● Connect existing pins in routes ● Record, upload & assign audio stories to pins ● Record new routes ● Follow routes ● Download / save routes on device ● Download audio files Share stories, points, routes
Dependencies:	Content Management Services
Technologies:	Specified in D4.5 Currently, the most solid & mature technologies for our case appear to be: ● SERVER: PHP Laravel Framework, MySQL ● MOBILE: Cordova Framework, Onsen UI ● ONLINE MAPS: Google Maps API ● OFFLINE MAPS: OpenStreetMaps API, LeafletJS Framework ● STORING FILES: audio + map tiles in device memory via Cordova file plugin, images in

	SQLite via Cordova SQLite plugin USER LOCATION: via GPS by Cordova Geolocation plugin
Data (in/out/storing):	IN/OUT/STORING ● geolocation ● routes ● user credentials ● audio files

Table 8: AR App component

Component name:	AR App
Description:	Specified in D4.5
Dependencies:	Content Management Services
Technologies:	Specified in D4.5 ● Unity ● Google ARCore Apple ARKit
Data (in/out/storing):	IN/OUT/STORING ● User credentials ● Virtual Exhibitions ● Assets (3D models) Metadata

Table 9: 3D Sonic App component

Component name:	3D Sonic App
Description:	Specified in D4.5. The 3D Sonic App will: ● Allow users to process and edit audio material ● Automatically generate/select audio material from non-audio content ● Support 3D audio features (using the 3D Tune-In Toolkit) Support basic audio features in the other PLUGGY modules
Dependencies:	Content Management Services
Technologies:	Specified in D4.5 Binaural audio (3D Tune-In Toolkit)
Data (in/out/storing):	IN - audio + metadata (e.g. positioning, tracking, additional information, etc.) OUT - audio (possibly synchronised with non-audio content) STORING - audio + metadata
Main Technical Use Cases	

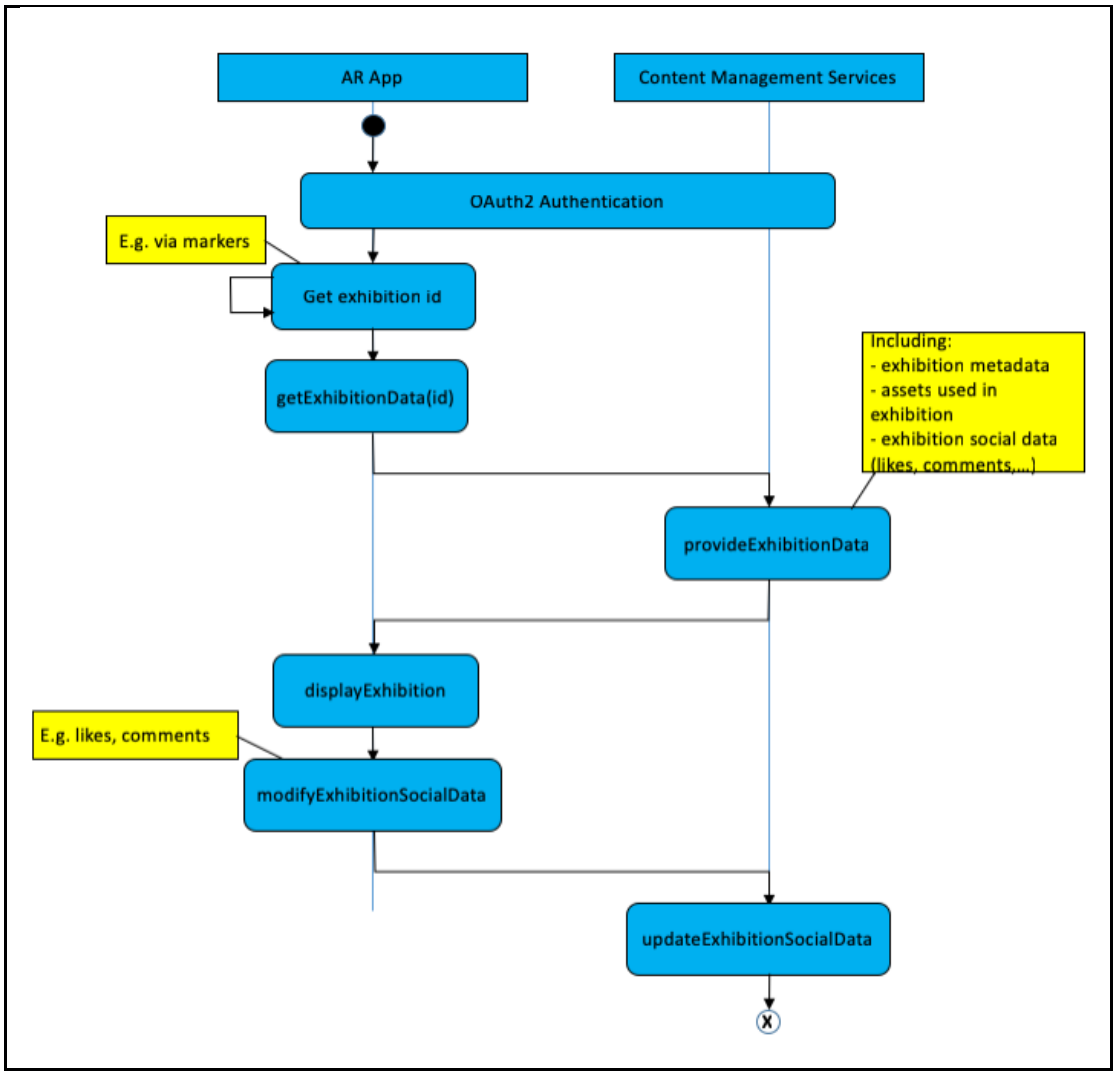


Table 10: Game App component

Component name:	Game App
Description:	Will be specified in D4.2 The Game App will: ● Drag and Drop interface for assets. ● Map editor ● Event management game flow design. ● Database management for character and other game resource handling.
Dependencies:	Content Management Services
Technologies:	Will be specified in D4.2 JavaScript
Data (in/out/storing):	IN ● assets (graphics, sounds, stories, …) ● metadata OUT: ● gaming

	STORING: ● score/badge/achievement

Table 11: Curatorial Tool component

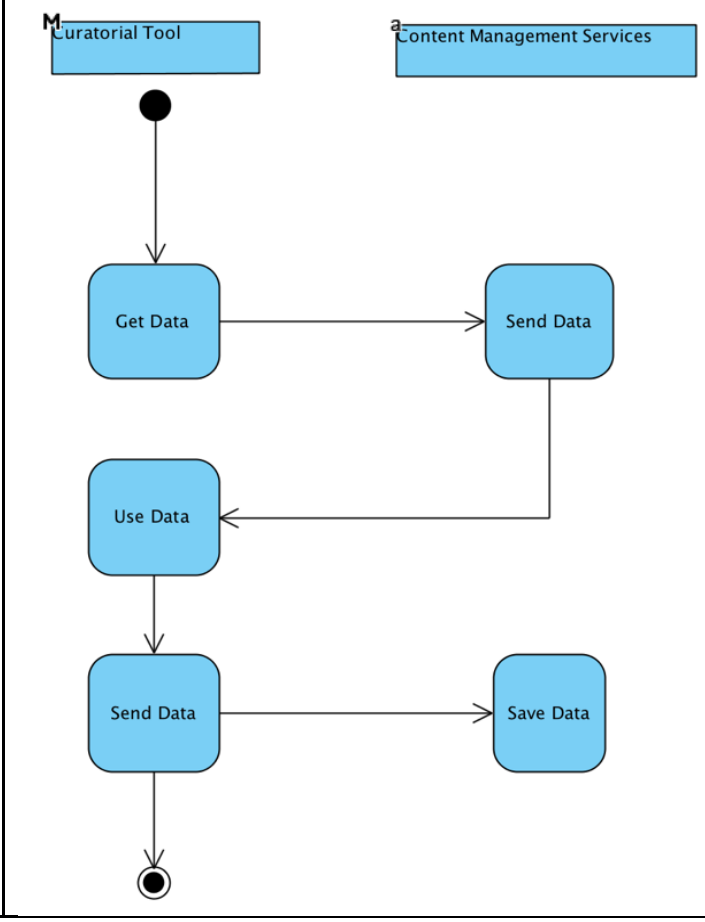
Component name:	Curatorial Tool
Description:	Will be specified in D2.4. The PLUGGY Curatorial Tool will: ● Provide a visual representation of the assets. ● Create content based on user inputs. Support content exchange with the content management services
Dependencies:	Content Management Services
Technologies:	Will be specified in D2.4 Content Tools Angular 6
Data (in/out/storing):	IN - contents (assets) of other components OUT - json data STORING - json data (Cultural Content)
Main Technical Use Cases	

Table 12: Content Management Services component

Component name:	Content Management Services
Description:	Set of services that could be used via RESTfull API or WebSocket by other application
Dependencies:	PLUGGY Content Repository
Technologies:	Node.js

Table 13: Social Platform Services component

Component name:	Content Management Services - Social Platform Services
Description:	Services used for social interaction of users: like a exhibition, make a friendship, etc. Primary user of these services will be Social Platform
Dependencies:	PLUGGY Content Repository
Technologies:	Node.js

Table 14: Search Services component

Component name:	Content Management Services - Search Services
Description:	Search services allow user to search content, that user has permissions to it.
Dependencies:	PLUGGY Content Repository
Technologies:	Node.js

Table 15: Hinting/Suggestions and Recommendation Services component

Component name:	Content Management Services - Hinting/Suggestions and Recommendation Services
Description:	Recommendations, suggestions and hinging services for better User Experience.
Dependencies:	PLUGGY Content Repository
Technologies:	Node.js

Table 16: Authorization, Authentication and IPR Services component

Component name:	Content Management Services - Authorization, Authentication and IPR Services
Description:	Security services to protect users and their content.
Dependencies:	PLUGGY Content Repository
Technologies:	Node.js

Table 17: Content Services component

Component name:	Content Management Services - Content Services
Description:	Create/Read/Update/Delete operation for exhibitions, exhibition points, assets, media content and user collections
Dependencies:	PLUGGY Content Repository
Technologies:	Node.js

Table 18: Notification Services component

Component name:	Content Management Services - Notification Services
Description:	WebSocket server used for real time notification of the users about activities on PLUGGY platform.
Dependencies:	PLUGGY Content Repository
Technologies:	Node.js

Table 19: External Repository Content Connectors component

Component name:	Content Management Services - External Repository Content Connectors

Description:	Each connector is designed for specific external repository (i.e. Europeana, Wikipedia) and allows to PLUGGY users to search and reuse content from these repositories in their stories and virtual exhibitions.
Dependencies:	PLUGGY Content Repository
Technologies:	Node.js

Table 20: PLUGGY Content Repository component

Component name:	PLUGGY Content Repository
Description:	All content and data produced by the social platform and curatorial tool will be stored in this repository. Third party application could also use this repository to persist their data.
Dependencies:	-
Technologies:	MongoDB/ElasticSearch

Annex IV. – Basic API functions

- CRUD operations for management of assets

GET/assets
- customizable method for retrieving assets from the system

POST/assets
- create new asset

POST/assets/import
- import asset from external library

GET/assets/{assetId}
- get asset by Id

PUT/assets/{assetId}
- update asset

DELETE/assets/{assetId}
- delete asset

POST/assets/{assetId}/media
- create new media for the asset

GET/assets/{assetId}/media/{mediaId}
- get media of the asset by Id

DELETE/assets/{assetId}/media/{mediaId}
- delete media of the asset by Id

GET/assets/{assetId}/media/{mediaId}/thumbnail
- get thumbnail of the media of the asset by Id

GET/assets/my
- customizable method for retrieving assets of logged in user from the system

POST/assets/{assetId}/like
- update asset

POST/assets/{assetId}/unlike
- unlike asset

POST/assets/{assetId}/comment
- comment on asset

POST/assets/{assetId}/report
- report asset

exhibitions
- 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

POST/exhibitions/{exhibitionId}/media
- create new media for the exhibition

GET/exhibitions/{exhibitionId}/media/{mediaId}
- get media of the exhibition by Id

DELETE/exhibitions/{exhibitionId}/media/{mediaId}
- delete media of the exhibition by Id

GET/exhibitions/{exhibitionId}/media/{mediaId}/thumbnail
- get thumbnail of the media of the exhibition by Id

GET/exhibitions/my
- customizable method for retrieving exhibitions of logged in user from the system

POST/exhibitions/{exhibitionId}/like
- update exhibition

POST/exhibitions/{exhibitionId}/unlike
- unlike exhibition

POST/exhibitions/{exhibitionId}/comment
- comment on exhibition

POST/exhibitions/{exhibitionId}/report
- report exhibition

applications
- CRUD operations for management of applications

GET/applications
- customizable method for retrieving applications from the system

POST/applications
- create new application

GET/applications/{applicationId}
- get application by Id

PUT/applications/{applicationId}
- update application

DELETE/applications/{applicationId}
- delete application

users
- CRUD operations for management of users. In current model we have 3 types of the users: individual user; team user (group, organization, instiution); application user.

GET/users
- retrieving users from the system

POST/users
- create new user

GET/users/{userId}
- get user by Id

PUT/users/{userId}
- update user

DELETE/users/{userId}
- delete user

POST/users/{userId}/media
- create new media for the user

GET/users/{userId}/media/{mediaId}
- get media of the user by Id

DELETE/users/{userId}/media/{mediaId}
- delete media of the user by Id

GET/users/{userId}/media/{mediaId}/thumbnail
- get thumbnail of the media of the user by Id

search
- search functionalities

GET/search
- full-text and geo location search with external repository search integration

hinting
- hinting functionalities

GET/hinting/tags
- hinting service for tags

GET/hinting/places
- hinting service for places - provided by google maps API

GET/hinting/places/{placeId}
- retrieve information for specific place - provided by google maps API

GET/hinting/title
- hinting service for title

Annex V. – PLUGGY Glossary

Term	Definition
Story	Narrative text (usually short) and/or set of assets provided by curator about the specific cultural heritage subject or by user describing her/his personal experience.
Exhibition	Ordered sequence or oriented graph of combination of exhibition points and exhibitions, one exhibition point can be part of the exhibitions. Exhibition has associated title and description properties.
Exhibition Point	Exhibition Point is a set of assets that has something in common and are organized and described as point of interest for the interested person.
Asset	Set of media content related to some artefact.
Media Content	Image/audio/application/video files. Must be a part of asset.

Favourites Collection	List of assets, exhibition points or exhibitions specified by their enumeration of by search query. Favourites Collection has associated title and description properties.
Artefact	Any physical or non-physical object related to the cultural heritage, which is represented in the system as a meta-data record with properties such as type, author, time period, location etc.
Social Platform	Platform that enables users to be part of virtual society. Users could be passive (reader) or active (content creator - discussions, posts/content evaluator - like) participants.
Curatorial Tool	Tool that enables users to create virtual exhibitions, curate stories, interlink information, multimedia or objects from different cultural collections, etc, in order to present cultural content through novel interfaces and techniques.
Sonic Narrative	A story (account of connected events) which is mainly delivered through sound. Sonic narrative could also be referred to as sounds and music which accompany a spoken, written, and/or visual narrative.