MediaVerse

A universe of media assets
and co-creation opportunities

D6.2

Development, Integration and Testing of MediaVerse Components v1

Project Title	MediaVerse
Contract No.	957252
Instrument	Innovation Action
Thematic Priority	ICT-44-2020 Next Generation Media
Start of Project	1 October 2020
Duration	36 months

Deliverable title	Development, Integration and Testing of MediaVerse Components v1
Deliverable number	D6.2
Deliverable version	V1.0
Previous version(s)	N/A
Contractual Date of delivery	30.06.2022
Actual Date of delivery	29.06.2022
Nature of deliverable	Demonstrator
Dissemination level	Public
Partner Responsible	ATC
Author(s)	Tasos Lampropoulos (ATC), Spyros Papafragkos (ATC), Stratos Tzoannos (ATC)
Reviewer(s)	Stephan Gensch (VRAG), Sara De Luca (LINKS), Manos Schinas (CERTH)
EC Project Officer	Alberto Rabbachin

Abstract	D6.2 accompanies the implemented, tested, and integrated MediaVerse platform-demonstrator in its first release, and provides information on the software structure, deployment and usage.

Keywords	MediaVerse software, platform, user guide, components, architecture, deployment

Copyright

MediaVerse is an H2020 Innovation Project co-financed by the EC under Grant Agreement ID: 957252.

Revision History

Version	Date	Modified By	Comments
V0.1	12/05/2022	Tasos Lampropoulos, Spyros Papafragkos, Stratos Tzoannos (ATC)	First Draft Table of Contents
V0.2	18/05/2022	Tasos Lampropoulos, Spyros Papafragkos, Stratos Tzoannos (ATC)	Final Table of Contents
V0.3	27/05/2022	Tasos Lampropoulos (ATC)	First Draft
V0.4	03/06/2022	Spyros Papafragkos (ATC)	Second Draft
V0.5	08/06/2022	Stratos Tzoannos (ATC)	Updated Second Draft
V0.6	09/06/2022	Spyros Papafragkos (ATC)	Final Second Draft Review
V0.7	13/06/2022	Stephan Gensch (VRAG)	Review
V0.8	14/06/2022	Tasos Lampropoulos, Spyros Papafragkos, Stratos Tzoannos (ATC)	Addressing comments
V0.9	16/06/2022	Sara De Luca (LINKS)	Review
V1.0	28/06/2022	Manos Schinas (CERTH), Evangelia Kartsoundiou (CERTH), Symeon Papadopoulos (CERTH), Tasos Lampropoulos, Spyros Papafragkos (ATC), Stephan Gensch (VRAG)	Final Review and quality control

Glossary

Abbreviation	Meaning
DAM	Digital Asset Management
DB	Database
GDPR	General Data Protection Regulation
IPFS	InterPlanetary File System
MV	MediaVerse
SC	Smart Contract
SLC	Smart Legal Contract
UI	User Interface
WP	Work Package

Table of Contents

Revision History 3
Glossary 3
Index of Figures 5
Index of Tables 5
Executive Summary 6
1 Introduction 7
1.1 Integration Status 7
1.1.1 KONG-GATEWAY – Gateway Service 8
1.1.2 MV-DAM-API – Digital Asset Management 8
1.1.3 APACHE SOLR – Index of content metadata 9
1.1.4 MONGO – Data Layer 9
1.1.5 IPR-SERVICE – Intellectual Property Rights Service 9
1.1.6 MV-SLC-ENGINE – Smart Legal Contracts Management 9
1.1.7 CICERO-SERVER – Smart Legal Contracts and Templating System 9
1.1.8 MV-BCSP - MV-BCSPEH - MV-ETH – Smart Contracts and Blockchain Management 9
1.1.9 DASHBOARD-UI 9
1.1.10 IPFS-HOST and IPFS-API – Federated Search Services 10
1.1.11 TRANSCODER and PUBLISHER – Content Adaptation Services 10
1.1.12 COPYRIGHT-NEGOTIATION – License Advisor Module 10
1.2 Implemented Features 10
1.3 Deployment Status 11
2 Installation Instructions 13
2.1 Basic Configuration 13
2.1.1 Mongo 13
2.1.2 DAM 14
2.1.3 Rights-Management Components 16
2.1.4 UI 17
2.1.5 IPFS API 18
2.2 Docker Execution 19
3 User Manual 20
4 Summary and Next Steps 27

Index of Figures

Figure 1. Docker Containers 8
Figure 2. List with the available functionalities (first page) 10
Figure 3. List with the available functionalities (second page) 11
Figure 4. Login webpage. 20
Figure 5. Create account page. 20
Figure 6. Dashboard page 21
Figure 7. Upload page 21
Figure 8. My Assets page 22
Figure 9. Assets details 23
Figure 10 License registration for an asset 24
Figure 11. Search page 24
Figure 12. Projects page 25
Figure 13. Fader webpage 25
Figure 14. Profile details 26

Index of Tables

Table 1. VMs characteristics and the corresponding URLs 12

Executive Summary

This document complements the implemented, tested, and integrated MediaVerse (MV) platform-demonstrator in its first release. It provides all the necessary details (Docker containers, etc.) concerning the status of the MV node v1.0. Specifically, it describes the integrated components, the main functionalities and an overview of the deployed nodes at the time of writing this report. It also includes specific installation instructions for helping any interested party to deploy their own MediaVerse node. Finally, it provides a user's guide along with screenshots that demonstrate the navigation in the platform and a set of key functionalities of this release.

1 Introduction

MediaVerse aims to offer a decentralized platform for media asset management. Towards this direction, a fully functional platform will be delivered at the end of the project as a result of the research and development activities carried out by the project partners. The conceptual architecture of the platform was presented in deliverable D2.2 - Conceptual Design of the MediaVerse Framework[1] of the project and several functionalities have been originally specified through the user requirements in deliverable D2.1 - Use Cases and User Requirements[2]. Research for the actual implementation of the several components has been conducted in the technical WPs of the project (i.e., WP3, WP4, WP5, and WP6).

The following sub-sections provide details about the status of the first MV platform release. More specifically, these include the presentation and a brief explanation of all the integrated components in this first release, the available functionalities of these components, and finally information on the deployed nodes so far.

1.1 Integration Status

A MediaVerse node consists of a set of Docker containers, as depicted in Figure 1. These contain core functionalities of each node developed in WP6, such as user authentication and media asset management, as well as services that have been developed in WP3, WP4 and WP5 and integrated in the MV node architecture (through WP6), such as federated search and retrieval, IPR management, and media asset publishing. Although, several functionalities are still at an early stage, this first release is an important step towards the fulfilment of the MediaVerse objectives.

The services developed in WP3 for media annotation are not part of the platform, but have been integrated as external services called by the MediaVerse Digital Asset management component (*mv-dam-api* in Figure 1) through an API gateway. In that way, the annotations for each uploaded asset are stored and indexed to facilitate the retrieval of assets. The work conducted in WP4 resulted in the different components under the IPR SERVICES group in Figure 1. The authoring tools part of WP5, are not part of the platform but the version described in this deliverable, has paved the way for their integration in the next release. More precisely, the current release provides all the needed mechanisms for user authentication, project management and asset management, retrieval, and publishing. In addition, all three core media types are supported (images, videos, 3D objects, and scenes), therefore authoring tools can communicate with MediaVerse nodes (through the *kong-gateway*) and retrieve content to be used in authoring procedures.

[1] https://mediaverse-project.eu/wp-content/uploads/2021/10/MediaVerse_D2.2_Conceptual-Design-of-the-MediaVerse-Framework.pdf
[2] https://mediaverse-project.eu/wp-content/uploads/2021/04/D2.1-V1.0.pdf

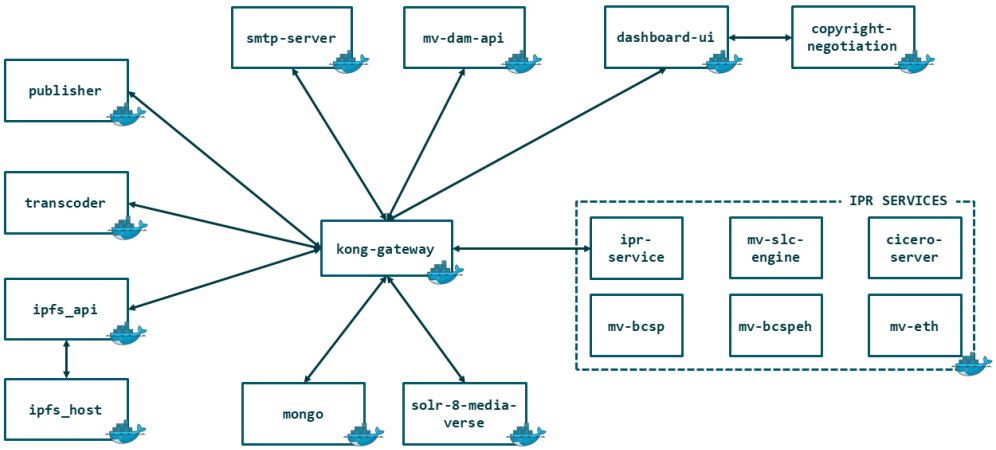

Figure 1. Docker Containers

1.1.1 KONG-GATEWAY – Gateway Service

The gateway, based on kong[3], manages the incoming/outgoing traffic between the end-user and the MV platform, but also the communication between the rest of the services in the platform.

1.1.2 MV-DAM-API – Digital Asset Management

The central element of the MediaVerse node, which participates in almost all interactions between the components of the platform. It is responsible for the implementation of the business logic.

- User management
  - Registration
  - Authentication
- Asset management
  - Upload
  - Update
  - Delete
  - Download
  - Search
  - Share images on Twitter
- Subtitle management
  - Create
  - Delete
  - View
- Project management
  - Creation
  - Update
  - Delete
  - Search
- Notification management
  - Creation
  - Update
  - Delete
  - Search

[3] https://konghq.com/kong

- IPR services management (SLC and SLC Template)
  - Creation
  - Update
  - Delete
  - Search

1.1.3 APACHE SOLR – Index of content metadata

The module is responsible for indexing asset details from different sources and making them searchable in near real-time. During the ingestion of an asset, the DAM will send all the available metadata to the Apache Solr instance in order for the latter to index it and make it available for search.

Apart from the metadata provided by the user during upload such as description, and metadata such as upload time and media type, AI-based annotations are also added to support visual content-based search. In particular, the AI-based annotations that are automatically detected in the image or video content of the assets and can be searched include the following: a) Celebrities (images, videos), b) Actions (images, videos), c) automatically generated caption (images), d) depicted objects (images, videos, 3D scenes), e) meme class (images), f) disturbing and NSFW flags (image, videos).

1.1.4 MONGO – Data Layer

This component is used by the DAM to persist users, assets, projects and all related data to support MV usage.

1.1.5 IPR-SERVICE – Intellectual Property Rights Service

This API handles the interactions with the Blockchain Service Provider and the Smart Legal Contract (SLC) Engine.

1.1.6 MV-SLC-ENGINE – Smart Legal Contracts Management

This engine handles all the run-time features related to SLCs (e.g., the creation of SLC instances, check for the trigger of specific SLC-related events).

1.1.7 CICERO-SERVER – Smart Legal Contracts and Templating System

This template engine handles the Smart Legal Contract Templates that conform to the Accord Project Template Specification.

1.1.8 MV-BCSP - MV-BCSPEH - MV-ETH – Smart Contracts and Blockchain Management

These services handle the deployment and the management of the Smart Contracts (SCs) and the interaction with the blockchain in general, ensuring the notarization of SLCs, and the implementation of the blockchain SCs counterparts of the rights on digital assets in a way that is transparent to the user.

1.1.9 DASHBOARD-UI

The web-based user interface enables users to access the different services of MediaVerse in an intuitive way. The dashboard functionality focuses on user, content and project management, offering support to image, audio, video, and 3D content.

1.1.10 IPFS-HOST and IPFS-API – Federated Search Services

These services allow users to search for content in the entire MediaVerse network without the need to know where and how many nodes are in the network. This service forwards the search queries and aggregates the search results across the network.

1.1.11 TRANSCODER and PUBLISHER – Content Adaptation Services

This service automatically generates preview formats from media files, such as GIF files, thumbnails, audio watermarks, etc. In accordance with user access rights, these previews are used during search and retrieval to facilitate content browsing and discovery from users that do not have access to the actual content but need a preview of the asset they intend to purchase. It also produces adaptive streaming content from video files.

1.1.12 COPYRIGHT-NEGOTIATION – License Advisor Module

The license advisor module is embedded in the UI to recommend appropriate licenses for asset media files.

1.2 Implemented Features

This section presents the available features/functionalities of the MV node. Figures 2 and 3 are screenshots from the tool used by the involved partners for issue monitoring. Further details on specific features can be found in the deliverable D2.1 - Use Cases and User Requirements.[4]

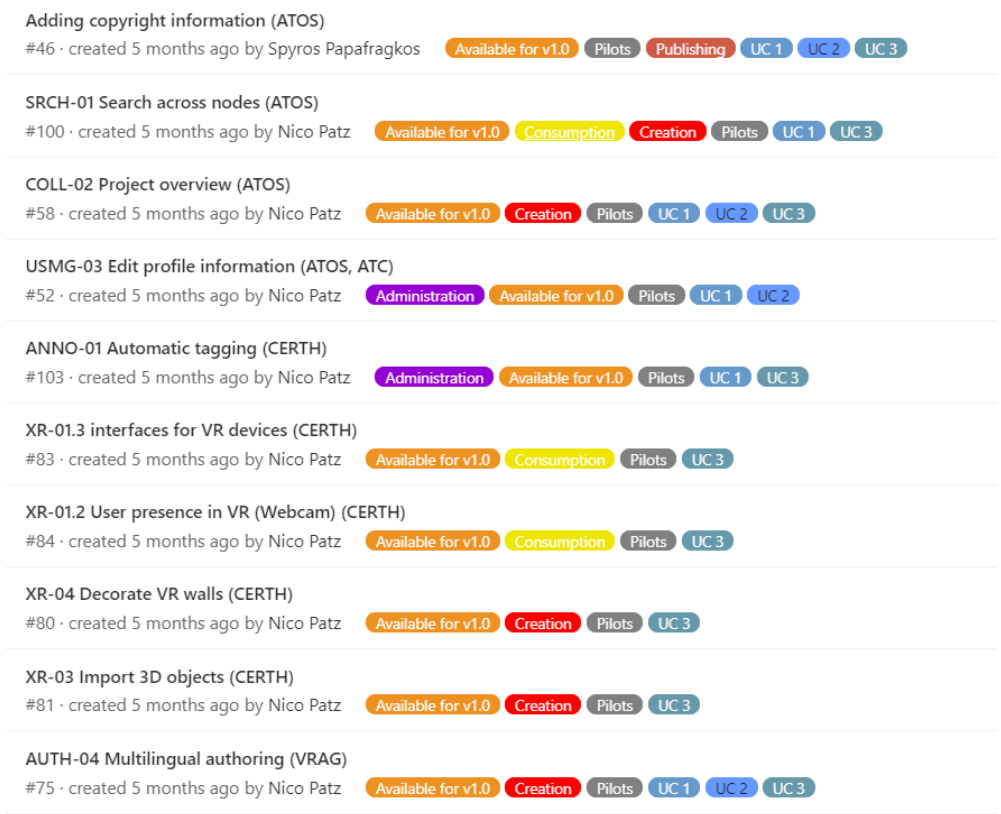

Figure 2. List with the available functionalities (first page)

[4] https://mediaverse-project.eu/wp-content/uploads/2021/04/D2.1-V1.0.pdf

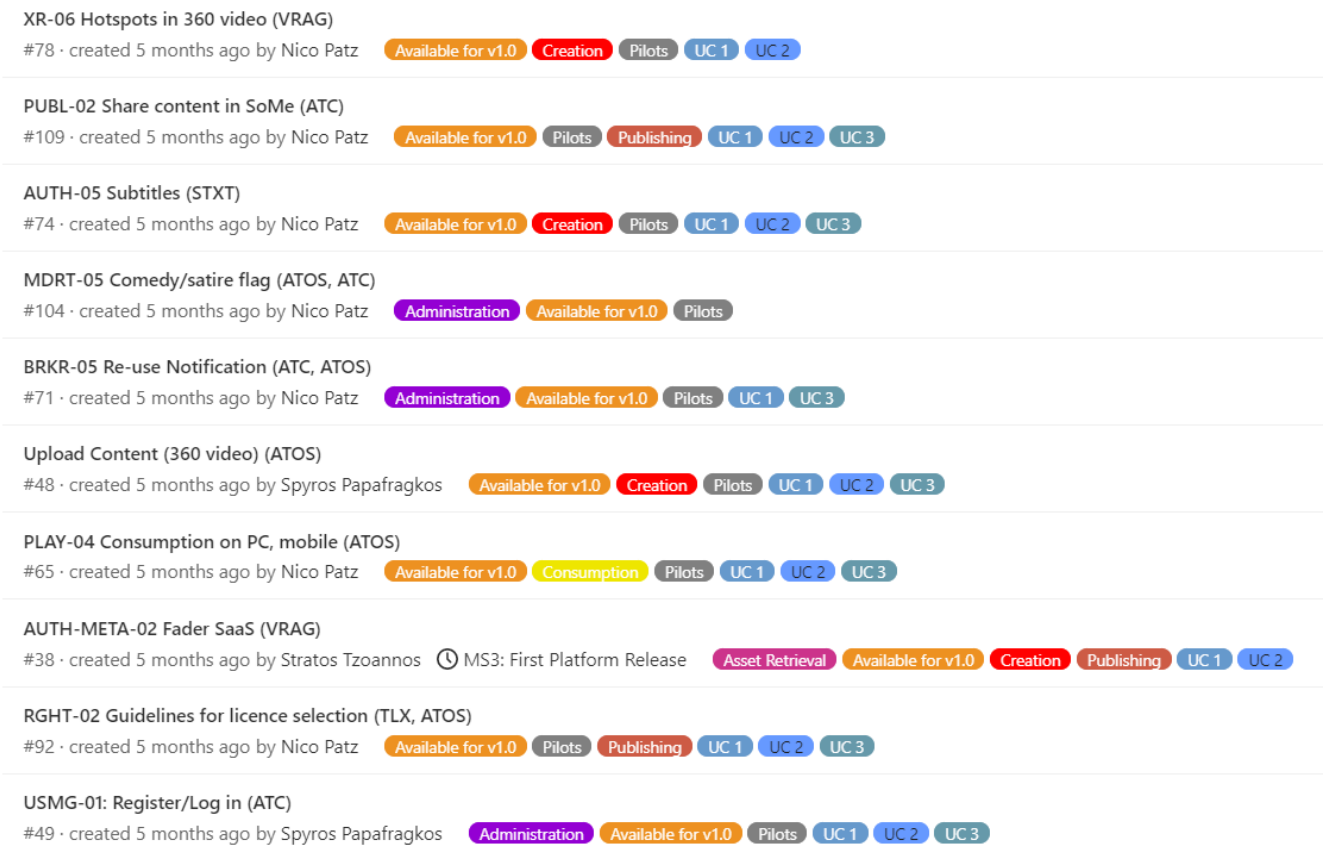

Figure 3. List with the available functionalities (second page)

1.3 Deployment Status

The MediaVerse software management is realized using a semantic versioning technique for each component that is integrated in the MV node. Every version of each module is stored in a dedicated docker hub repository as a docker image. The latest versions of the above images are then included in the docker compose configuration file that contains the instructions for creating all the available resources for the MediaVerse node.

Whenever a new version of a component is available, it is dockerized, uploaded in the docker hub and a new merge request in the respective Gitlab repo is triggered for the update of the docker compose file. In that way, the docker compose configuration is reviewed and it is available for cloning in the MediaVerse nodes.

The platform v1.0 has been deployed in three different nodes and specifically on the premises of ATC, CERTH and ATOS. Table 1 provides the deployment details.

Table 1. VMs characteristics and the corresponding URLs

NODE	SPECIFICATIONS			URL
	OS	CPU	RAM	
ATC	20.04.1-Ubuntu	Intel(R) Xeon(R) CPU E5-2620 v2 @ 2.10GHz (2 cores)	8 GB	https://dashboard.mediaverse.atc.gr/
CERTH	20.04.4-Ubuntu LTS	Intel(R) Xeon(R) CPU E5-2620 0 @ 2.00GHz (24 cores)	128 GB	Available on request (currently under reconfiguration from an IP-based URL to a user-friendly domain).
ATOS	20.04.3-Ubuntu LTS	Intel(R) Xeon(R) Gold 5118 CPU @ 2.30GHz (8 cores)	16 GB	http://mediaverse.ari-imet.eu:3000

2 Installation Instructions

Here, we detail all the necessary steps needed for the installation of the MV node.

2.1 Basic Configuration

The deployment structure consists of a **docker-compose** file, a **mongo init script** and a **config folder**. The docker-compose file, the init-mongo.js and the config folder must be placed in the root level.

1. The docker-compose file contains all the docker images, networks and volumes needed for the docker engine to deploy the MV node. It is predefined and it is common for all nodes, but node administrators can edit it accordingly to change the port in which each service is exposed.

2. The mongo init script will be called by the docker-compose file to properly initialize the DB. It is predefined and it is common for all the nodes.

3. The config folder has the following structure:

```
config
    ..dam
    ..ipfs_api
    ..ipfs_host
    ..mongo
    ..right-management
    ..solr
    ..ui
    ..ipfs_api
    ..ipfs_host
    ..solr
```

This folder includes all the necessary configuration files of the node.

All the following variables must be set before running the docker-compose file.

2.1.1 Mongo

`./config/mongo/.env`

These parameters define the properties needed by the MongoDB instance (e.g. admin user credentials).

```
MONGO_DB_NAME=root_db # It is the name of the DB
MONGO_ROOT_USERNAME=<not common> # It is the Admin username of the DB
MONGO_ROOT_PASSWORD=<not common> # It is the password of the Admin user
```

For example:

```
MONGO_DB_NAME=root_db
MONGO_ROOT_USERNAME=root
MONGO_ROOT_PASSWORD=**********
```

2.1.2 DAM

`./config/dam/.env`

These parameters are needed for the configuration of the DAM container and, among others, include all the properties required for the DAM to access other services of the node.

Set mongo DB parameters:

Mongo DB parameters are needed for the DAM to access the running mongo DB instance. These should be the same values as in 2.1.1.

```
MONGO_CONNECTION_URL=<not common>
MONGO_DB_NAME=<not common>
MONGO_ROOT_USERNAME=<not common>
MONGO_ROOT_PASSWORD=<not common>
```

The *MONGO_CONNECTION_URL* parameter has the following structure:

mongodb://MONGO_ROOT_USERNAME:MONGO_ROOT_PASSWORD@mongo:27017/MONGO_DB_NAME?authSource=admin&readPreference=primary&directConnection=true&ssl=false

where *MONGO_ROOT_USERNAME*, *MONGO_ROOT_PASSWORD*, *MONGO_DB_NAME* should be replaced with their actual values.

Set asset storage:

```
FILE_HOST_ENV=<not common>
```

Currently, the solution supports **local** and **S3**. **local** means that the storage used for the media assets will be based on the running machine's file system, while **S3** means that the files will be stored on Amazon.

For local storage add:

```
FILE_HOST_ENV=local
```

For external storage:

An active S3 storage must be created. Please check S3 documentation.

Then the following parameters must be included in the `.env`:

```
FILE_HOST_ENV=S3
S3_BUCKET_NAME=<not common>
AWS_ACCESS_KEY_ID=<not common>
AWS_SECRET_ACCESS_KEY=<not common>
AWS_REGION=<not common>
```

The domain or IP of the DAM has to be configured. This is used to generate proxy preview links and deep links.

```
DAM_DOMAIN=<not common> # It is the DOMAIN of the node
```

For example:

```
DAM_DOMAIN=https://xxx.xxx.xxx.xxx:5000
```

or

```
DAM_DOMAIN=https://my.domain.com:5000
```

with `xxx.xxx.xxx.xxx` being the IP of the server in which the service is deployed, and `5000` being the port in which DAM's API is exposed (5000 can itself be set to a different value).

A characteristic name for the node should be specified. This facilitates federated search as the retrieved assets are tagged with the name of the node, alongside its domain name or IP.

```
DAM_NAME=<not common>
# For example
DAM_NAME=atc-vm.gr
```

Finally, a Twitter dev account must be set to support sharing on that platform. For more details on the generation of the appropriate tokens please refer to Twitter's documentation.

```
TWITTER_OAUTH_CONSUMER_KEY=<not common - define the dev twitter key>
TWITTER_OAUTH_CONSUMER_SECRET=<not common - define the dev twitter secret>
```

To support near duplicate detection the following must be added.

```
# it is the URL of NDD service
NDD_URL=<not common> # for V1 it will be the same for all nodes

# For example: (CERTH can provide the URL upon request)
NDD_URL=*******
```

In the current release, NDD is a centralized external service, hosted by CERTH. However, in the next release, NDD will be an internal service and each node will include its own instance.

Similar to the NDD service, for the DAM to be able to communicate with the external media annotation service offered by the CERTH the following must be set:

```
# the URL of the media annotation service
GRPC_URL=<not common>  # for V1 it will be the same for all nodes,

# For example: (CERTH can provide the URL upon request)
GRPC_URL=*******
```

To support the automatic generation of subtitles for the uploaded videos, the RACU service must be enabled. This can be done by providing an authorization key to permit the interaction between the DAM and the external RACU service, hosted by STXT. If no key is provided, then this functionality is not supported.

```
# the token needed to authorize the calls to the RACU subtitling service

# For example: (STXT can provide the token key upon request)
RACU_KEY=*******
```

The following parameters can be left untouched:

```
SOLR_URL=http://solr:8983/solr # the same for all nodes.
IPR_URL=http://ipr-service:8081 # the same for all nodes
TRANSCODER_URL=http://transcoder:5000 # the same for all nodes

SERVER_PORT=8888 # the same for all nodes
```

2.1.3 Rights-Management Components

`./config/right-management/`

First, the *cicero-template-library* must be cloned from gitlab:

```
git clone https://gitlab.com/mediaverse/wp4/smart-legal-contracts/mv-cicero-template-library.git
```

Then put the contents to *path/to/mv-cicero-template-library* folder.

Parameters that shall be configured

Edit *mv-slc-engine* and *cicero-server* services in the docker-compose.yml. The *path/to/mv-cicero-template-library* in the docker-compose.yml shall be the path where the *mv-cicero-template-library* was cloned.

```
mv-slc-engine:
  volumes:
  - "path/to/mv-cicero-template-library:/mv-cicero-template-library"

cicero-server:
  volumes:
  - "path/to/mv-cicero-template-library:/mv-cicero-template-library"
```

Edit `./config/right-management/ipr-service/application.yml` to add *bearer token* of the IPR Service (to make requests to DAM):

```
server:
  bearer-token: XXXX
```

For v1, the following *bearer token* must be used:

```
server:
bearer-token:
eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJub2RlbmFtZSI6ImF0Yy12bS5nciIsInJvdXRpbmdJZCI6IjYyNDJhOGUyZTk0NWVlNzJkYTIwNGVmOUBhdGMtdm0uZ3IiLCJpZCI6IjYyNDJhOGUyZTk0NWVlNzJkYTIwNGVmOSIsImVtYWlsIjoiaXByLXNlcnZpY2VAbWVkaWF2ZXJzZS5hdGMuZ3IiLCJ1c2VybmFtZSI6Imlwci1zZXJ2aWNlQG1lZGlhdmVyc2UuYXRjLmdyIn0.CrAdaiqXYnqnS_khc5bKhMQmT4NXNhgj3Rl0WYXY9FY
```

Edit `./config/right-management/mv-bcsp/.env` to include private key for the MV Ethereum network

```
DEFAULT_MV_NODE_PRIVATE_KEY=XXXX
```

For that parameter, one of the private keys that are generated by the **mv-eth** container that can be retrieved by looking at the container logs.

For v1 all nodes can use the same key:

```
DEFAULT_MV_NODE_PRIVATE_KEY=4bfede256ea1a7de304081e223bb1db97c9ebd7cfe82d4e03bc4e583db759381
```

Parameters that can be left untouched

` ./config/right-management/ipr-service/application.yml`

```
server:
  port: 8081  #port on which ipr-service is listening
  error:
    include-message: always #enable error detailed description
mv-services-basepaths:
  mv-blockchain-service-provider: http://mv-bcsp:8082/ipr/bc  #endpoint of the BCSP
  mv-slc-engine: http://mv-slc-engine:8083  #endpoint of the MV SLC Engine
  mediaverse-node-backend: http://mv-dam-api:8888  #endpoint of the MV Node Backend
```

`./config/right-management/mv-slc-engine/application.yml`

```
server:
  port: 8083  #port on which mv-slc-engine is listening
  error:
    include-message: always #enable error detailed description
mv-services-basepaths:
  cicero-server: http://cicero-server:6001  #endpoint of the cicero-server
slc-templates:
  library-dir: /mv-cicero-template-library  #path of the SLC templates library (inside the container)
```

`./config/right-management/cicero-server/.env`

```
CICERO_PORT=6001  #port on which cicero-server is listening
CICERO_DIR=/mv-cicero-template-library  #path of the SLC templates library (inside the container)
```

`./config/right-management/mv-bcsp/config/config.json` shall be empty during first run

`./config/right-management/mv-bcsp/.env`

```
NODE_ENV="mv-eth" #name of the docker service container of the local blockchain deployment
```

`./config/right-management/mv-bcspeh/.env`

```
NODE_ENV=mv-eth #name of the docker service container of the local blockchain deployment
IPR_SERVICE_ENDPOINT=http://ipr-service:8081/ #endpoint of the IPR Service
UPDATE_API_ENDPOINT=event/update #path of the event update API of the IPR Service
```

2.1.4 UI

`./config/ui/.env`

The following URLs should be defined:

```
# It is the DOMAIN of the node. It is the same domain as b)
REACT_APP_API_URL=http://xxx.xxx.xxx.xxx:5000

# It is the DOMAIN of the node. It is the same domain as b)
REACT_APP_IPR_URL=http://xxx.xxx.xxx.xxx:5000

REACT_APP_METADATA_URL=http://xxx.xxx.xxx.xxx:3002
REACT_APP_IPFS_URL=http://xxx.xxx.xxx.xxx:5050/api
REACT_APP_TRANSCODING_URL=http://xxx.xxx.xxx.xxx:5500
REACT_APP_THUMBNAIL_URL=http://xxx.xxx.xxx.xxx:5501
REACT_APP_LICENSES_URL=http://xxx.xxx.xxx.xxx:3003

# It is the domain of the node plus the port of ipfs api (4040)
REACT_APP_FEDSE_URL=http://xxx.xxx.xxx.xxx:4040
```

All of these services point to the same IP or domain in which the services are deployed (`xxx.xxx.xxx.xxx`), with a different port according to the service.

A firebase account must be set, please refer to Firebase documentation.

```
REACT_APP_FIREBASE_AUTH_DOMAIN=<not commom>
REACT_APP_FIREBASE_API_KEY=<not common>
```

2.1.5 IPFS API

`./config/ipfs_api/.env`

The IPFS bootstrap node parameter helps IPFS to discover the rest of the nodes in the network.

```
IPFS_BOOTSTRAP_ADDR=<IPFS Bootstrap Node Identifier>
```

For v1, */ip4/83.149.101.53/tcp/4001/p2p/12D3KooWGgFA2ZV4Uy7JUFMzs6VLCFZesPNtEpePHa3FbaX6MGHf* will be used as bootstrap node for network discovery. To define it, the following shall be added in the .env file.

For example:

```
IPFS_BOOTSTRAP_ADDR=/ip4/83.149.101.53/tcp/4001/p2p/12D3KooWGgFA2ZV4Uy7JUFMzs6VLCFZesPNtEpe
PHa3FbaX6MGHf
```

Local DAM address should also be included, for the federated search component to be able to use the local search functionality offered by the DAM. It should be the same domain of the DAM defined in 2.1.2:

```
DAM_ADDR=http://xxx.xxx.xxx.xxx:5000
```

The following parameters can be left untouched:

```
IPFS_NODE_IP=ipfs_host # Container name of the IPFS Host service. Default: ipfs_host
IPFS_NODE_PORT=5001 # Port of the IPFS Host service. Default: 5001
IPFS_NODE_TIMEOUT=10
FSEARCH_RESULT_TIMEOUT=30
```

2.2 Docker Execution

Run the application, from the root level order:

```
docker-compose pull # pull the images
docker-compose up -d # run the docker-compose
```

3 User Manual

This section provides an overview of the user experience and implemented features through screenshots. This overview can serve as a user guide for the MediaVerse node.

Figure 4 depicts the Login/Create account page. In case a user already has an account, he/she is logging in with his/her credentials (email, password) otherwise the option to create an account should be selected.

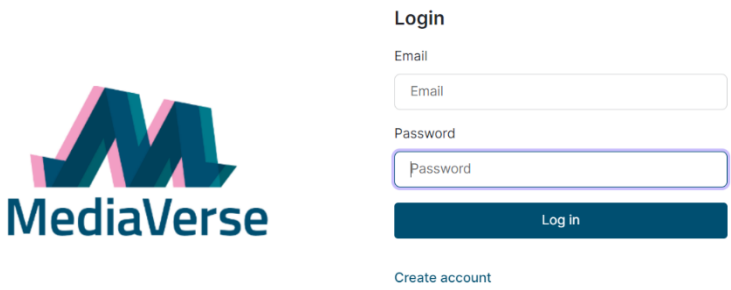

Figure 4. Login webpage.

The "create account" option leads to the corresponding page. Users must fill in the required details (email, username, etc.), accept the terms and conditions and the privacy policy after careful reading them, and then create their account (see details in Figure 5).

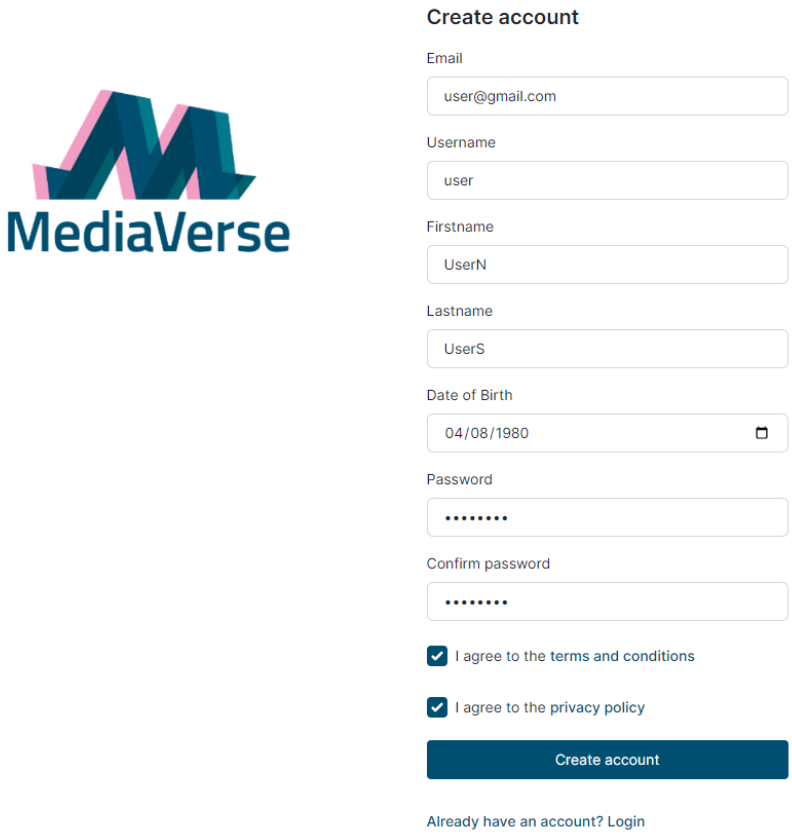

Figure 5. Create account page.

Once a user's account is created the homepage page appears (Figure 6). There, one can see the main menu on the left with the *Dashboard* page selected by default and a secondary menu with a help button, the language option and the profile details on the upper right of the page. Any notifications related to the user will appear on the dashboard page, and there is also a possibility to send invitations to other users.

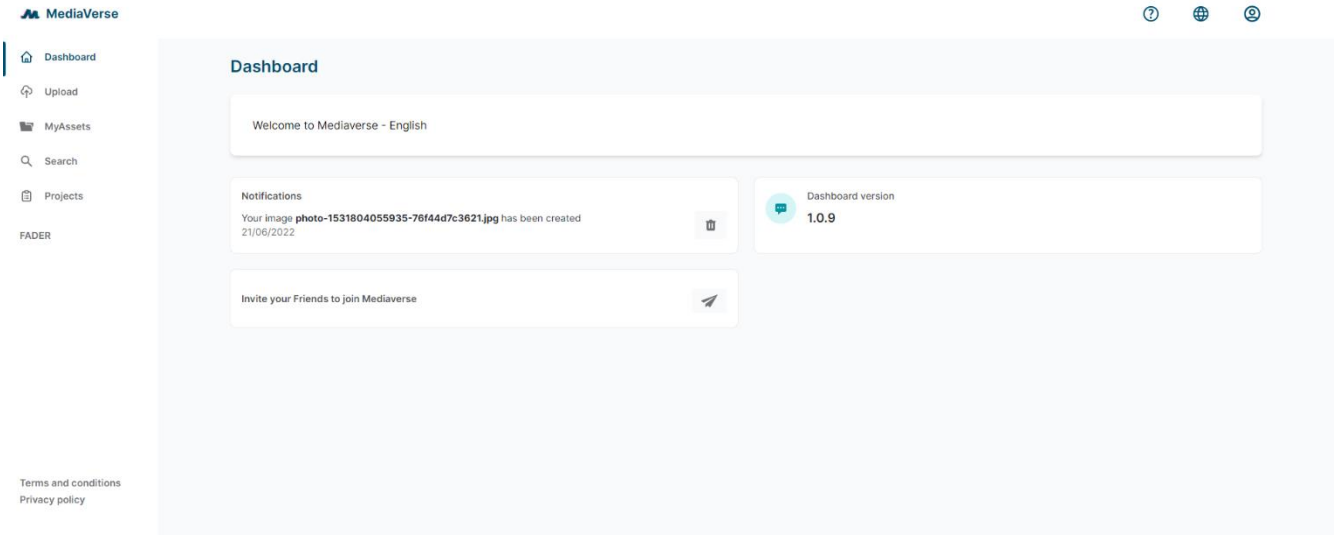

Figure 6. Dashboard page

The next item in the main menu is the *Upload* functionality (Figure 7). On this page, content guidelines are available to users, which define what kind of content is not allowed to be uploaded in MediaVerse. The **Drop your file here** button is the main functionality of this page (to be enriched in later releases) where one can upload the desired content (audio, images, videos, and 3D videos).

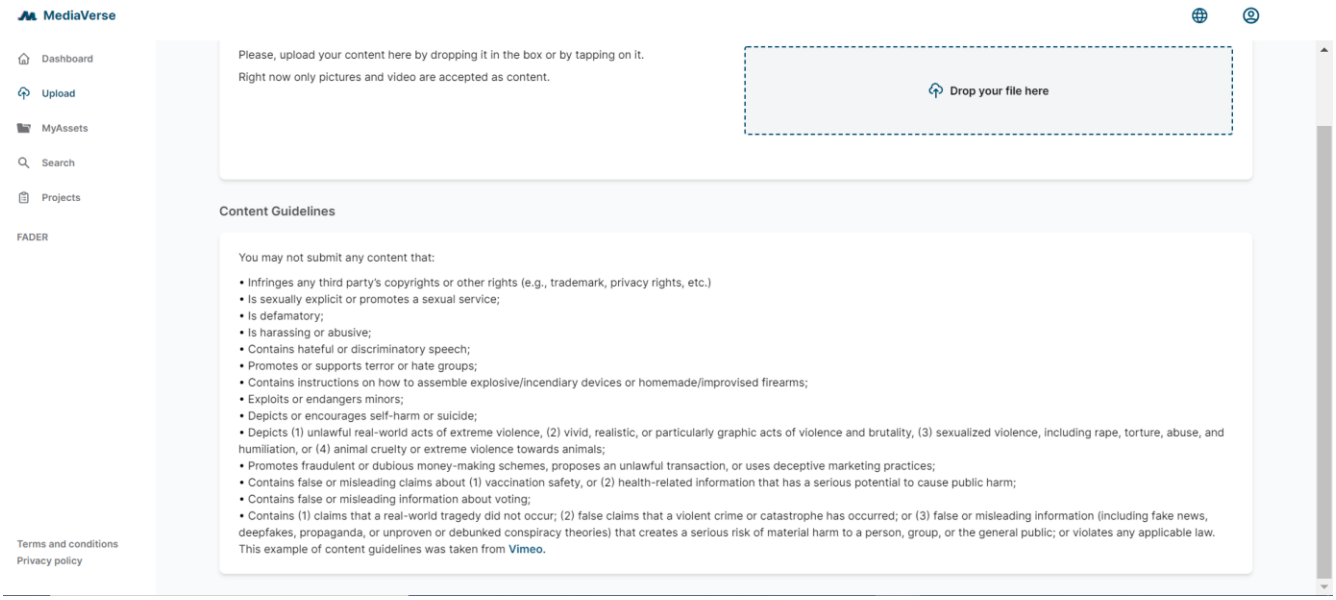

Figure 7. Upload page

In the “MyAssets” item in the main menu, all the available content (e.g., previous projects, videos, etc.) of the specific user will be displayed along with options of filtering the displayed results (Figure 8).

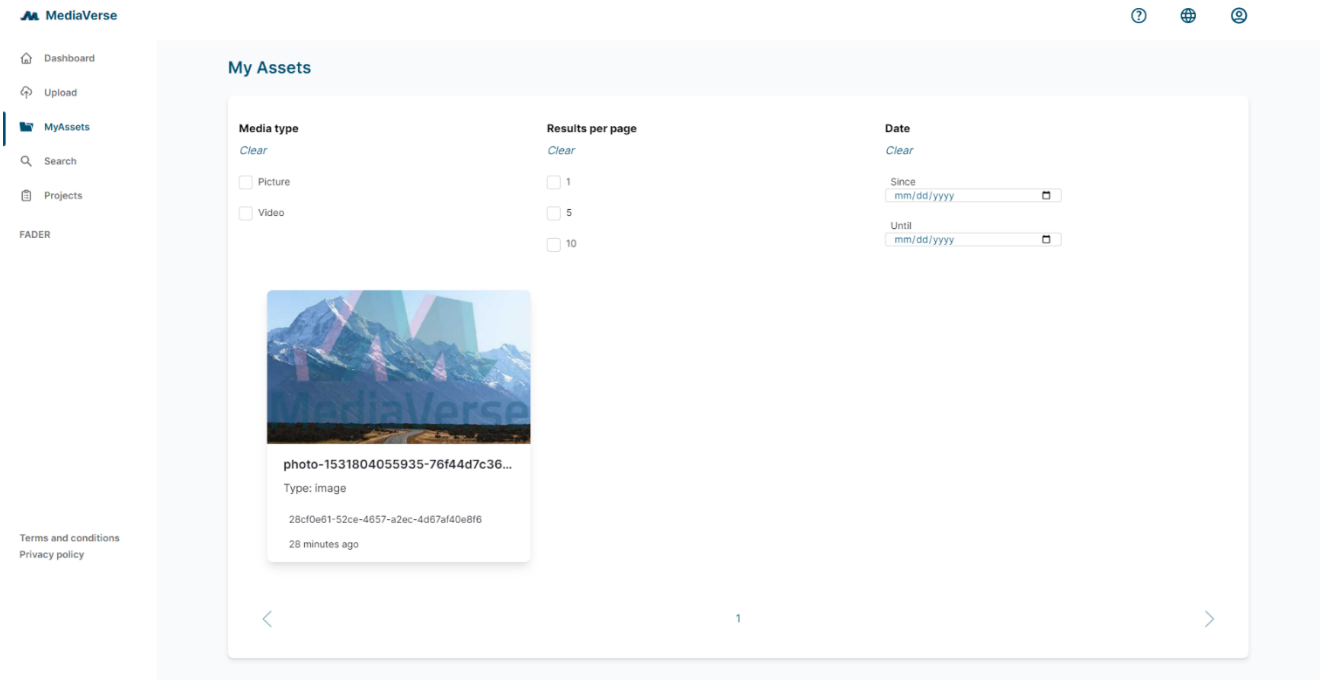

Figure 8. My Assets page

Next, when an asset is selected a page will be displayed (Figure 9), showing all the available details of a specific asset (e.g., description, content creation date, etc.).

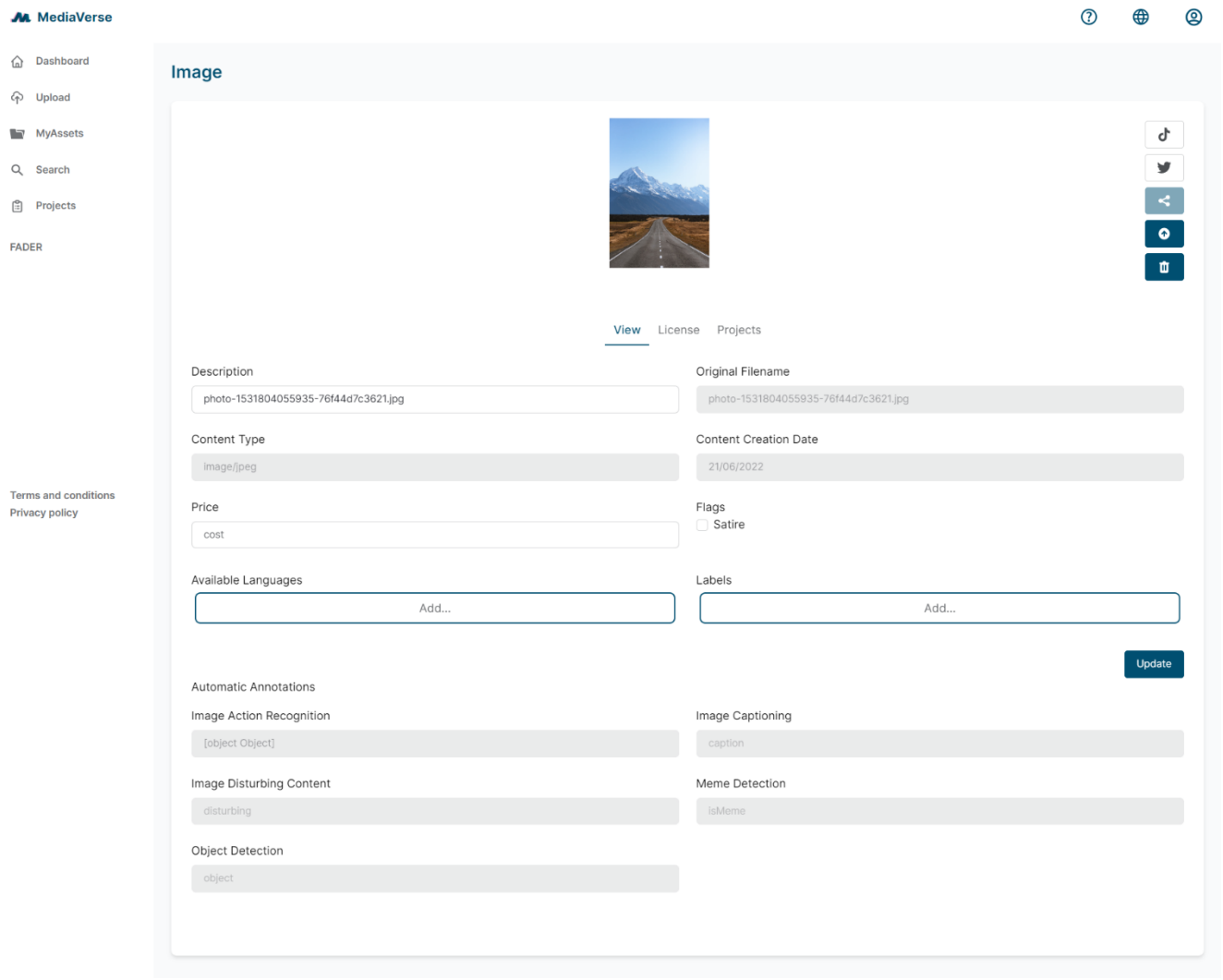

Figure 9. Assets details

Figure 10 provides an example of asset license registration. For a given asset uploaded by a user, a specific license from the supported ones (e.g. CC0[5] as depicted in the example) can be selected and the corresponding transaction is committed in the Ethereum network.

[5] https://creativecommons.org/share-your-work/public-domain/cc0/

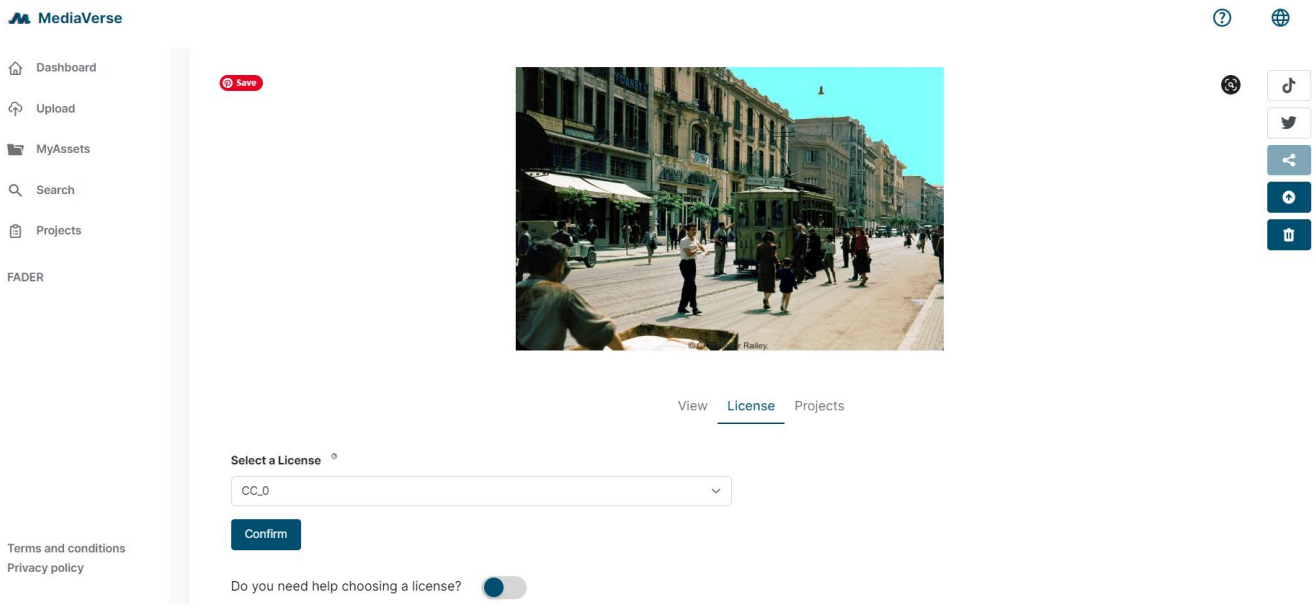

Figure 10 License registration for an asset

In the sequel, on the *Search* page (Figure 11) there are two search options. The local search will return any related result found in this specific node (e.g., cats results depicted in Figure 11), and the external search option which returns all the related results also from the other MediaVerse nodes. There are also some filter options that can be applied on the searches, such as to return only videos or only photos (this feature will be enriched in later releases).

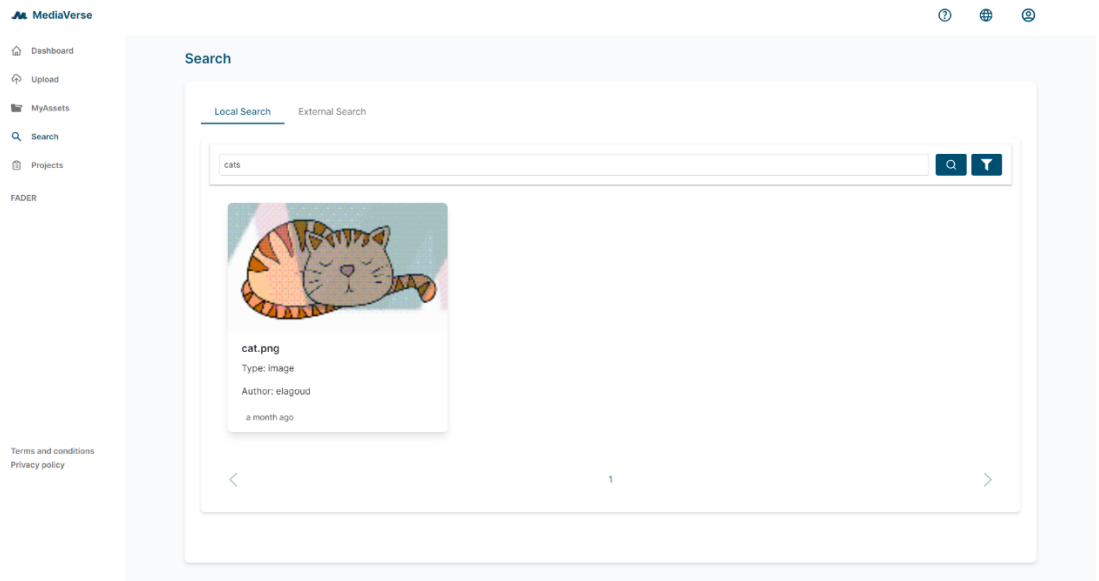

Figure 11. Search page

The final item in the main menu is the *Projects* tab (Figure 12), which contains all the projects that the specific user is involved in (either as an owner or as participant). These projects are collections are collections of assets which can be used for authoring procedure. With the integration of authoring tools in the next release of the MV platform (Fader and VRodos), projects created in MediaVerse and the associated assets, would be available within these authoring tools in order to create derivative works.

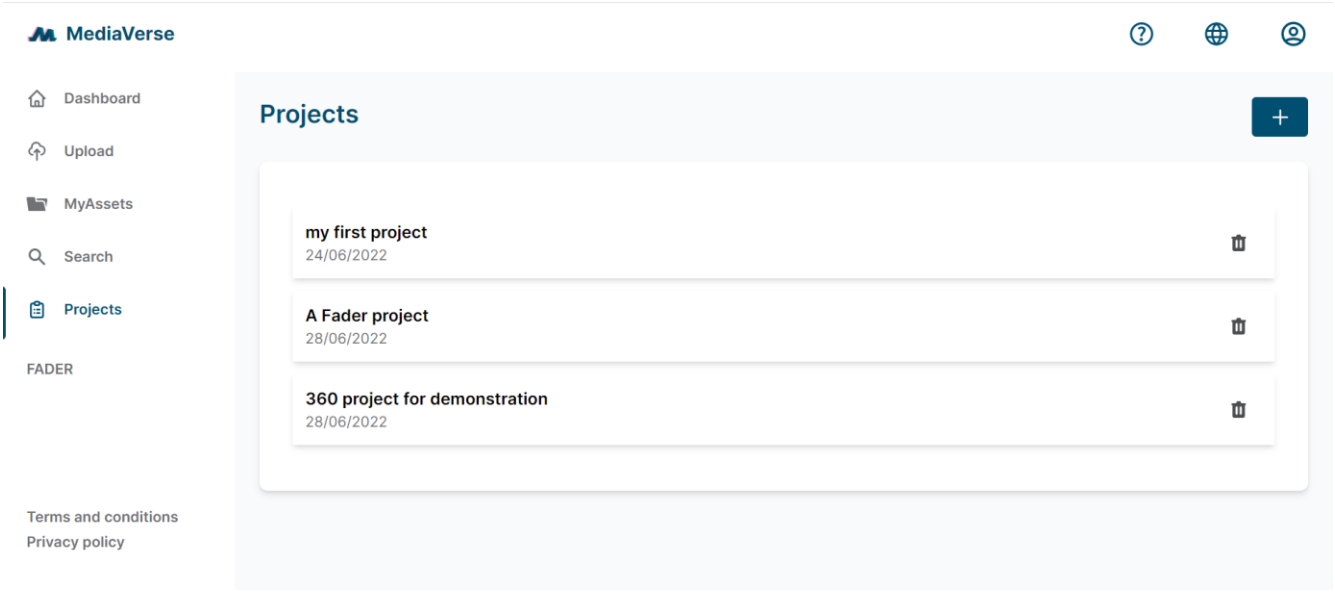

Figure 12. Projects page

Below the Projects tab there is the first external tool that is included in this release called Fader and is leading to Fader's webpage (Figure 13). In the next release this integration will be further improved with projects, as already explained.

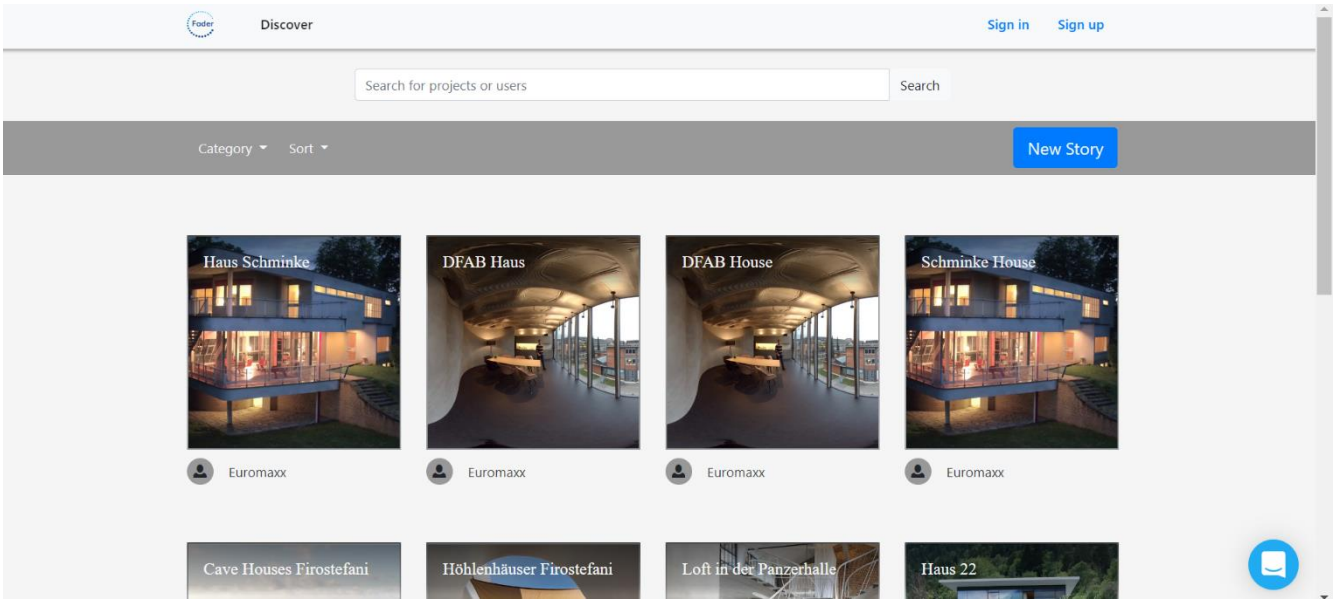

Figure 13. Fader webpage

In the secondary menu located in the upper right of the page (Figure 14), from left to right one can see the help functionality (pop up window that explains the main menu options), the language selection option (currently Catalan, English and Spanish) and the profile page option. On the profile page, the relevant information with respect to the profile of the user will be displayed (e.g., username, date of birth, billing information etc.). In addition this section can be used to update the user profile and set a registered office address which is mandatory from a legal perspective for the user to register assets.

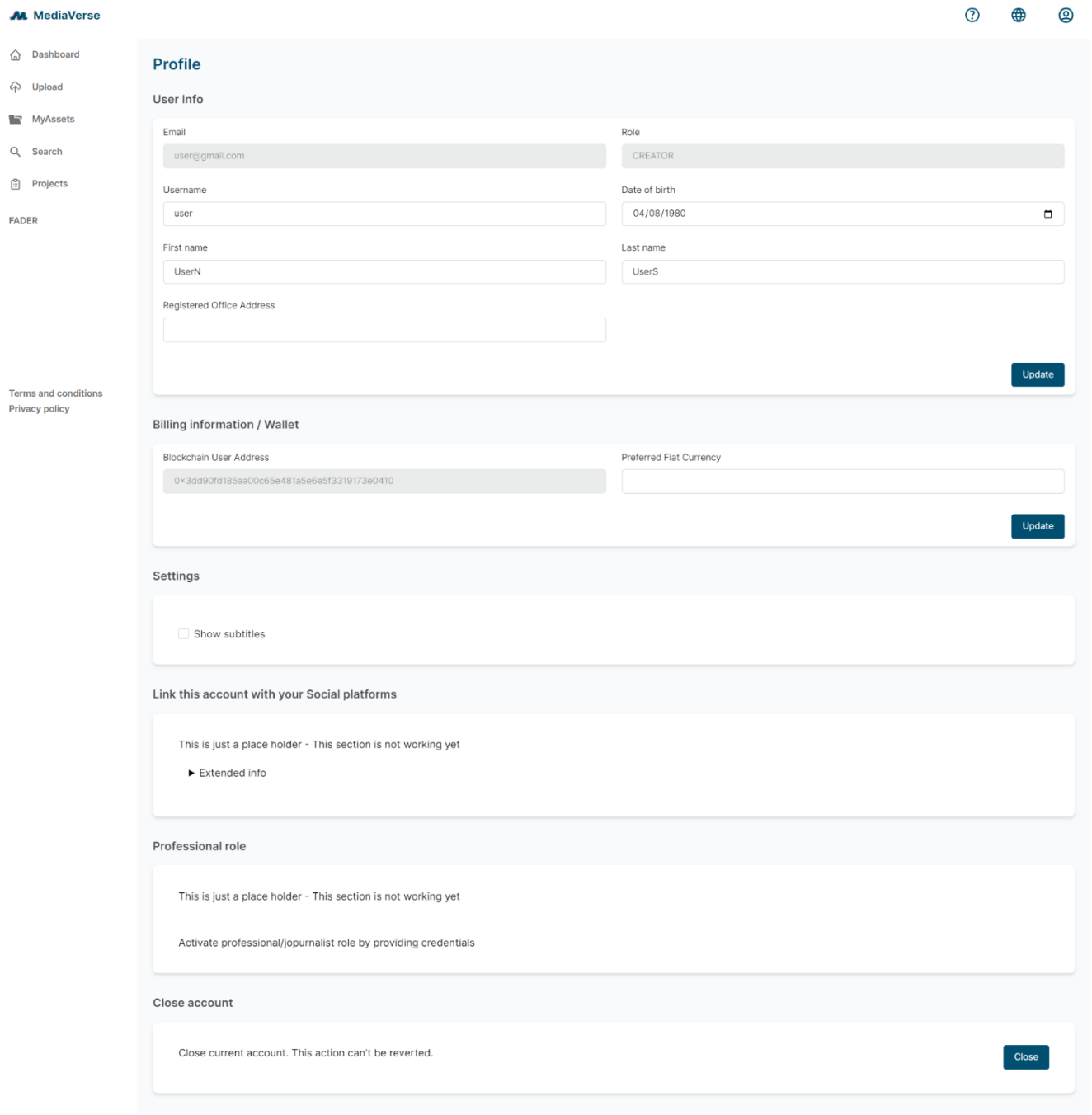

Figure 14. Profile details

4 Summary and Next Steps

This document accompanies the first MediaVerse platform release. This is the first stable version of the MV node and new components and functionalities will continuously be integrated and tested until the final release. In particular, there are several important features planned to be included to the node in the next release:

- Full integration of content authoring tools (Fader, VRodos): this will permit users to create projects in MediaVerse, add assets in them and then export them to the respective tool for further authoring.
- Integration of the content moderation toolset: models for misleading, inaccurate or tampered media content, content associated with hate speech and cyberbullying and not-safe for work (NSFW) content will be extended in the platform. In addition, a moderation UI is currently being designed and developed to be part of MediaVerse nodes, as a tool to set the appropriate moderation rules at node level.
- Integration of internode communication: this will be a functionality of great importance for content retrieval and purchasing across the MediaVerse network.
- Update of the Rights Management services to support Derivative Works such as those produced by authoring tools.
- Improve the integration of subtitle support provided by the RACU workflow engine: multilingual subtitles generated and stored automatically for each uploaded video will be used by the video player of the next release to facilitate accessibility for many of the user groups of the MediaVerse network. In addition, editing of the generated subtitles will be integrated to improve subtitle quality.
- Integration of social media analytics: this component, in conjunction with the already supported social login (in Twitter) will facilitate the sharing of content in social media platforms, the collection of analytics about the shared content and the discovery of trending topics/media assets to be imported in MediaVerse projects.
- Media annotations will further be leveraged to facilitate the discovery and accessibility of MediaVerse content, by allowing users to perceive, understand, navigate and interact with multilingual content.

With this first release of the platform, the use cases involved in WP7 will be able to validate and demonstrate in small-scale scenarios the MediaVerse platform. This is an essential step before being able to run a large-scale pilot, to minimise the risk of end-user rejection late in the project due to usability or technical reasons.

Additional MV node instances are also expected to be installed in the premises of the involved partners towards the final release of the node.