MediaVerse

A universe of media assets
and co-creation opportunities

D2.2

Conceptual Design of the MediaVerse Framework

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	Conceptual Design of the MediaVerse Framework
Deliverable number	D2.2
Deliverable version	V1.0
Previous version(s)	N/A
Contractual Date of delivery	30.09.2021
Actual Date of delivery	30.09.2021
Nature of deliverable	Report
Dissemination level	Public
Partner Responsible	ATC
Author(s)	Stratos Tzoannos, Spyros Papafragkos, Haris Bouchlis (ATC), Dimitrios Ververidis, Symeon Papadopoulos (CERTH), Rolf Nyffenegger (STXT), Stephan Gensch (VRAG), Enrique Quiros Fernandez (ATOS), Domenico Rotondi, Marco Giovanelli, Marco Saltarella (FIN)
Reviewer(s)	Manos Schinas (CERTH), Roberto Moncada, Giacomo Corrias (LINKS)
EC Project Officer	Alberto Rabbachin

Abstract	This deliverable presents the conceptual design of the MediaVerse solution. All the core components are presented and the interaction between them is devised. This document with its inputs is the basis for the implementation that will follow.

Keywords	Media Asset Management, Architecture, Media Rights, Services, Federated Services

Copyright

MediaVerse is an H2020 Innovation Project co-financed by the EC under Grant Agreement ID: 957252.
The content of this document is © the author(s). For further information, visit mediaverse-project.eu.

Revision History

Version	Date	Modified By	Comments
V0.1	01/10/2020	Stratos Tzoannos, Spyros Papafragkos	First Draft Table of Content
V0.2	21/07/2021	Stratos Tzoannos, Spyros Papafragkos	Final Table of Content
V0.3	27/08/2021	Stratos Tzoannos	First Draft
V0.4	02/09/2021	Stratos Tzoannos, Dimitrios Ververidis	Second Draft
V0.5	14/09/2021	Stratos Tzoannos , Spyros Papafragkos	Final Second Draft
V0.6	21/09/2021	Manos Schinas	Final Second Draft Review
V0.7	24/09/2021	Roberto Moncada, Giacomo Corrias, Manos Schinas	Review
V0.8	27/09/2021	Spyros Papafragkos	Refinements following review comments
V0.9	29/09/2021	Manos Schinas	Final QA
V1.0	30/09/2021	Stratos Tzoannos, Spyros Papafragkos	Final Document

Glossary

Abbreviation	Meaning
AD	Administrative Domain
API	Application Programming Interface
CID	Content Identifier
CI/CD	Continuous Integration and Continuous Delivery
CJ	Citizen Journalism
DAM	Digital Archive Management
DHT	Distributed Hash Table
GDPR	General Data Protection Regulation
GPU	Graphics Processing Unit
GUI	Graphical User Interface
HMD	Head-mounted Display
HW	Hardware
IPFS	InterPlanetary File System
IPR	Intellectual Property Rights
mTLS	mutual Transport Layer Security
MV	MediaVerse
NDD	Near Duplicate Detection
SC	Smart Contract
SLC	Smart Legal Contract

SPF	Single Point of Failure
SSO	Single Sign-On
SST	Speech to text
SW	Software
UI	User Interface
UUID	Universally Unique Identifier
VR	Virtual Reality
WCAG	Web Content Accessibility Guidelines
WP	Work Package

Table of Contents

Index of Figures

Executive Summary

This deliverable presents the conceptual design of the MediaVerse (MV) framework as a federation of MV nodes and lays the basis for the different strands of research and development in the project. This involves the description of the MV node, its modules as well as the basic interactions between them. The document aims to act as the reference guide throughout the development lifecycle of the project. Even though specific technologies are described, depending on the needs that will appear in the future, new technologies may be added on top or even replace the proposed ones in this document.

In particular, throughout the document, the MV ecosystem unfolds and all its components come into place. Building on the user requirements, the MV node and its key services and functionalities are defined. Then, the communication between the nodes is established and all the key technologies that will be used to achieve this. Furthermore, the external tools and services that will be connected to MediaVerse nodes and used by the MV users are taking their place in the MV ecosystem and are briefly described, to give the reader a first impression of the experience that will be available to the users.

More specifically, Section 1 presents the roadmap of this deliverable along with its purpose and relates it to other MediaVerse activities and work packages. Section 2 maps the requirements established in D2.1 to the conceptual architecture of MV and presents a high-level description of the MV. All the core components are introduced and their interaction with the other components. Furthermore, the main architecture diagram is given and depicts the global MV federated network. Section 3 presents the MV Node breakdown, i.e. its specifications and how it is decomposed in different tiers and the suggested technologies that form the architecture. Section 4 deals with the issues concerning the interactions between different nodes in MV. Section 5 aims at covering legal and technical aspects of IPR through a common digital rights management model. Section 6 describes the external services and tools before the conclusion in Section 7.

1 Introduction

This deliverable presents the Conceptual Architecture of the MediaVerse ecosystem. This will visualize the key elements of the system from a high-level and focus primarily on the relationships between key concepts, without defining implementation, process flow or other technical details.

Despite the generic nature of this document, we have tried to delve into concrete issues regarding workflows, technologies, deployment methods and others, moving a step beyond solely defining the architecture at an abstract level. This will help the MediaVerse implementation team to clarify any misconceptions and use this document as the set of common principles to follow throughout the project. For this purpose, we have reviewed several similar existing solutions like PeerTube[1], PixelFed[2], Plume[3], Nextcloud Social[4], Hubzilla[5] and Prismo[6] and we investigated several approaches and technologies like ActivityPub[7]. The output of the multiple brainstorming sessions and the fruitful exchange of knowledge among the members of the MediaVerse team resulted in a document, which brings together best practices and technical developments from a number of technical fields.

It is important to note that the content of this document is not binding but rather recommending a way forward for the upcoming research and development activities. Throughout the lifecycle of the project we may consider alternative implementation routes. This shall be clear to the development team who will use this document as the primary reference of the MediaVerse ecosystem.

Finally, this deliverable is also useful for the broader set of MediaVerse stakeholders. It shows a concise image of the system and defines the strategic objectives of MediaVerse. It also depicts some important decisions taken already, e.g. peer-to-peer communication, absence of central modules (registry), messaging through gateways, micro-services approach, choreography vs orchestration, the existence of remote services, and federated identity, among others.

[1] https://joinpeertube.org/
[2] https://pixelfed.org/
[3] https://joinplu.me/
[4] https://apps.nextcloud.com/apps/social
[5] https://fediverse.party/en/hubzilla
[6] https://gitlab.com/prismosuite/prismo
[7] https://activitypub.rocks/

2 Overall Description

2.1 Translating Requirements into a Conceptual Architecture

The first months of this Work Package had been dedicated to defining the MediaVerse Use Case scenarios and the corresponding technical requirements. Use Case partners stressed that all aspects of the system need to be very easy to use as it targets prosumers, ranging from freelance journalists to visitors at art events and members of vulnerable groups. The system aims to support a variety of people in creating, managing, and sharing content, including proper rights management. Even if some users may be skilled in some of the fields, no one will be experienced in all of them. It is therefore crucial that all technologies and interfaces will be usable without expert knowledge.

As part of the need for transparency, especially in rights and assets management, the vision of MediaVerse is to constitute a federated network of nodes, each node belonging to a different organization. In order to reach the network, users can run their own server, or just join one as a member.

An organization deploys an MV node and defines its own rules regarding the content. For example, DW or STXT could deploy MV nodes and make them accessible through the following domain names:

- https://mediaverse.dw.com
- https://mediaverse.swisstxt.ch

The users are then able to:

- Sign-in/Sign-up to an MV Node.
- Upload and organize content in their media library.
- Search and exchange content with any other user in the same or other node(s).
- Import/export content from/to external authoring tools through an API provided by each node.

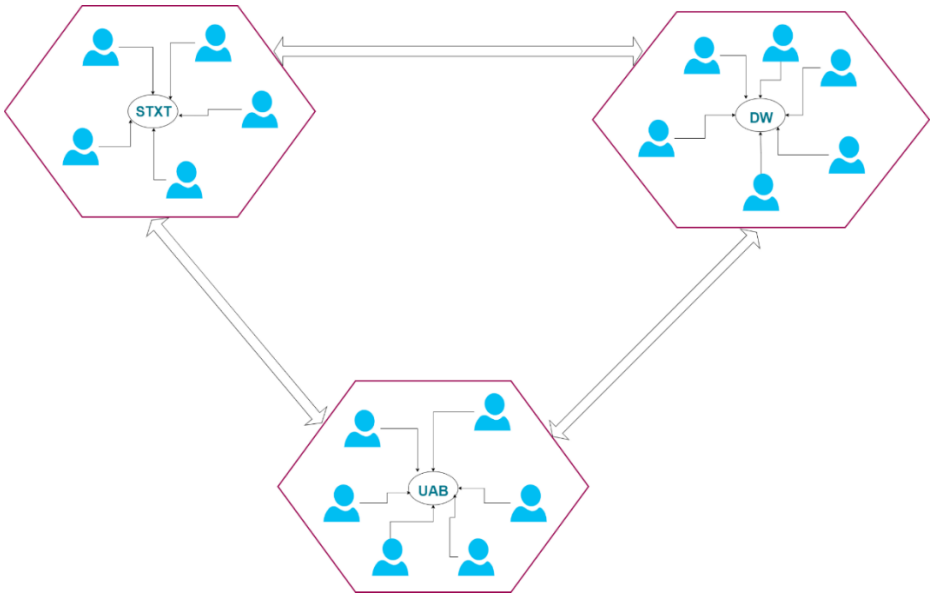

Figure 1: MediaVerse network consisting of MV nodes deployed by three different organizations. Users can register and log in to any of these instances.

This will support the requirement of larger content providers to enable full control of their own assets, yet enable single users to exchange content in both directions, i.e. by providing content to the network for others to consume and use for further media creation projects as well as finding and reusing content that others have shared in the overall network from their individual nodes. As previously described, the individual users have either the option of trusting one existing organization to handle their content, but they can set up their own node to have full control over their assets.

Any organisation can join the network, keeping control of what users of their nodes may be allowed to see and/or to consume or reuse when creating individual projects. Automated media rights management will be integrated in a way that any user will be able to manage their own content rights as well as access other people's content without explicitly asking or giving permission per media item.

The process of installing a new network node will be easy and the setup will be agnostic of the underlying infrastructure. Beyond the general components of each node, a variety of connected services and tools will be possible to connect to each node as needed to support individual requirements like automatic translation and subtitle generation, content moderation and verification and many others.

All these aspects, covering the range from content co-creation and authoring, content brokering, and media rights management, content annotation, search and retrieval to publishing follow the list of requirements collected in the first half year of the project as described in D2.1 "Use Cases and User Requirements". The system architecture of the MediaVerse Framework has been designed in a way that all node-based components and external tools will communicate in a federated, decentralised network.

2.2 MediaVerse Conceptual Architecture

The following diagram depicts the global architecture of the MediaVerse federated network:

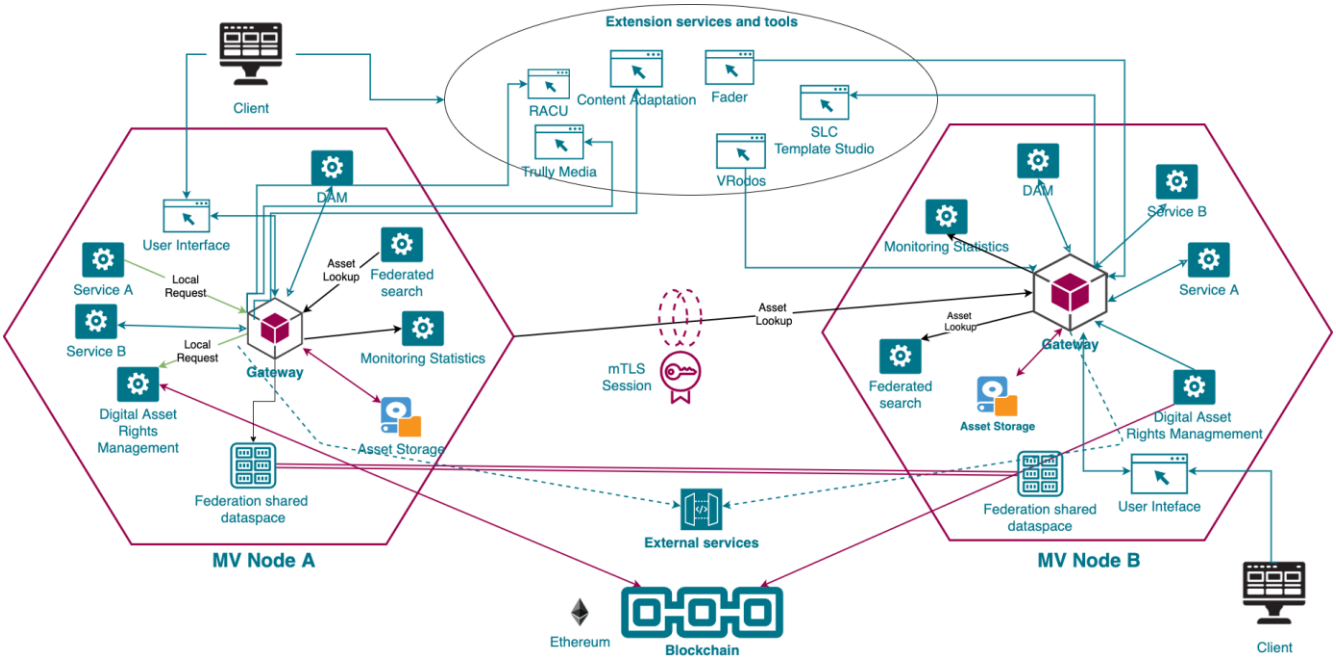

Figure 2: MediaVerse conceptual architecture

For the sake of simplicity, we have only included two MediaVerse nodes. In practice, there will be a network of nodes communicating with each other in a peer-to-peer manner. Therefore, the two nodes in the diagram can clearly depict the inter-node communication mechanism which will be applicable when adding more nodes in the network.

As already stated a MediaVerse ecosystem is a federation of **MediaVerse Nodes**. More formally, a MediaVerse ecosystem is made up of a set of **Administrative Domains** (**ADs**), where an **AD** has full responsibility of the administration of owned resources (e.g., digital assets, users, policies, HW/SW components). In the MediaVerse federation, each AD is autonomous in managing its resources even if the ADs in a MediaVerse federation cooperate according to a set of common rules governing the cooperation.

From an architectural point of view, an **AD** is represented by a **MediaVerse Node** that cooperates with the other MediaVerse nodes. Each MediaVerse node has a set of *MediaVerse core services*, which are present in every MediaVerse node, and could interact with additional services (*Extension Services* in Figure 2).

We can think of the MediaVerse node as the all-in-one installation needed by the individual organization to become a member of the MediaVerse federated network. A MediaVerse Node can be physically deployed to one or more physical or virtual machines.

With respect to architectural design, the MediaVerse node is not monolithic. On the contrary, it consists of several modules, each of these implementing a specific part of the functionality. The internal architecture of the MediaVerse node will follow a distributed microservices approach which will offer scalability and flexibility to the various components, minimising the risks and side-effects of parallel implementation by distributed development teams and facilitating the deployment and integration during the lifecycle of the project. The MV Node will be a distributed set of microservices, organized by a framework like Istio[8], described in detail in subsequent chapters.

The list of microservices is not fixed but it can be enriched with more services even after the project completion. Since the project will be offered under an open-source license to the community, external development teams will be able to extend the available features with additional ones. Since we are going to deliver most of the modules under open-source licenses, the description of the interfaces will be available to any developer who would like to replace, extend or add a microservice.

Some of these elements which will constitute the MediaVerse node are the following:

- The MediaVerse node will implement a set of features related to digital archive management. This is the **Digital Archive Management (DAM)** microservice, responsible for bringing together all necessary components for the basic functionality related to management of digital libraries available inside and outside the node. The DAM element will be also responsible for accessing and exposing the **Asset Storage**, which can be self-hosted by the node owner or could reside on the cloud (e.g. AWS, Azure) and will consist of the ***Metadata Storage, the Indexes (Metadata Index, NDD Index) and the Media (Object) Storage***.
- The access point between the front-end and the backend of the MediaVerse node will be an **API Gateway**, taking care of all the incoming and outgoing requests towards the user interface. The Gateway will also serve as a proxy for the outgoing requests and will offer services such as a firewall, load-balancer and aggregator for all the incoming requests.

[8] https://istio.io/latest/

- The **Federated Search** module will be responsible for the search and retrieval of content within the entire network. Although DAM is responsible for local search - i.e. given a query (regardless its source) searches on the local storage (metadata & indexed) for relevant content - the Federated Search module is responsible for the distribution of queries and search results across the network (taking into account permission rights, etc). Its purpose is to distribute the query among the entire federated network and then to collect back the results, merge them and return them to the initiator of the search. This is a demanding process which is described in the respective section of this module.
- The **Digital Asset Rights Management** service consists of a set of modules that will be implemented as part of the node, backed up by a decentralized Ethereum-based blockchain solution to be used for managing the copyrights and licensing. The Rights Management modules will communicate transparently with the blockchain and will expose an API facade to be consumed by the rest of the node elements and the User Interface. They will also implement blockchain-related functionalities, e.g. for micro-payments or digital wallets.
- The **Federation Shared Dataspace** will be responsible for a storage layer distributed among all the members of the federated network. Although there is no strict specification, we are investigating the use of IPFS (InterPlanetary File System)[9], a protocol and peer-to-peer network for storing and sharing data in a distributed file system. Also, by referring to the federated storage, we are also referring to a shared federated registry among the entire network, which will be stored and updated real-time by every node and will allow the discovery of all external nodes. It will serve as the catalogue of all the available nodes, acting as the facilitator of inter-node discovery. Moreover, it will be responsible for the circulation of the public keys of each node, which (in conjunction with the private keys of each node, stored locally) will allow the secure and trustworthy inter-node communication described in detail in subsequent sections of this deliverable. The role of this storage will not involve storing media assets, which will be hosted by the Asset Storage, as described previously.
- The **Front-End** of the MediaVerse node will be implemented as a Single-Page-Application (web app) which is served from a Web Server (e.g. Nginx) running inside the node. The code (HTML/CSS/JS) of the web application will run on each end-user's browser. It is important to highlight that each node will provide its own URL, in order to be accessible by end-users.
- Finally, there is going to be a set of **External Services and Tools** (e.g. Vrodos, Truly Media, Fader, RACU), which have been selected to be hosted and consumed as-a-service, centrally, due to restrictions in either proprietary licensing or high demands in hardware specifications, hard to be met by the nodes.

[9] https://ipfs.io/

3 MediaVerse Node

3.1 MV Node Specifications

The MediaVerse Node is the core element of the system. The Node will be responsible for the following tasks:

- Digital Asset Management
  - Media library: supports images, videos, 3D/VR models & scenes
  - User management: each user manages his/her own media library
  - Digital Rights Management: each user can apply licenses to his/her assets
  - Group Management: user and access rights management in the context of larger organizations

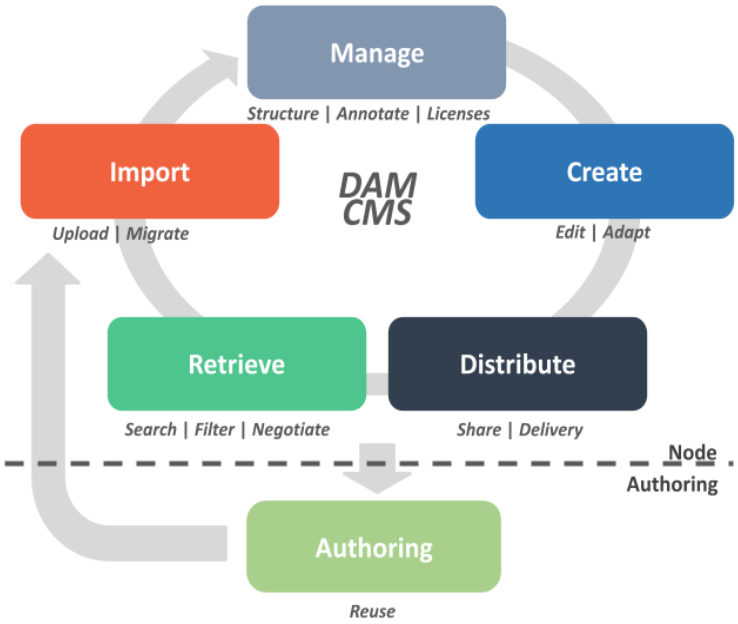

Figure 3: Digital assets lifecycle within MV Nodes

- Distribution
  - Adapt content and share it to other users
  - Share to social media or use links to deliver the content
- Content Retrieval
  - Search and filter content within your media library
  - Discover content from other users across the MV network
  - Automatic negotiations based on permissions
- Content Authoring
  - Re-use content to 3rd party authoring tools: Authoring tools are external to MV node, but a connection can be established for content re-use
  - Simple editing capabilities and filters can be part of MV nodes as extra functionalities.

3.2 MV Node Logical Decomposition

The MV Node consists of various modules that are described in Fig. 4.

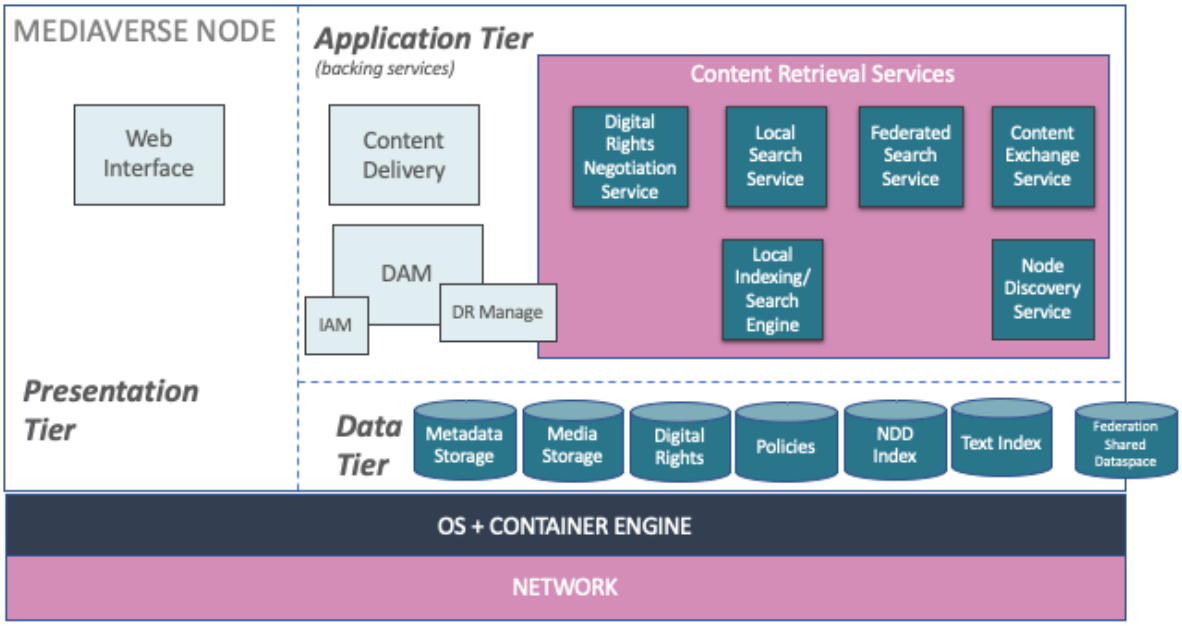

Figure 4: MediaVerse Node components

These components follow a logical grouping in tiers as described below.

3.2.1 Presentation Tier

This tier is referring to the web interfaces accessible by the users. In Figure 2 it refers mostly (but not exclusively) to the UI element, directly connected with the Gateway API, being part of MV Node. Apart from the UI of the MV Node, there can be more interfaces, exposed by relevant modules or external tools which address to the end users for accessing parts of their functionality.

- **MV Node User Interface**: The user interface has started to develop from scratch using React. It will allow the access to different services in just one place and also to give the users an easy way to use them. For now, this dashboard functionality is focused on user management and content management. The next step will be to allow the users to update the metadata of uploaded content.
- **Accessibility**: The interface will be WCAG2.2 compliant. WCAG stands for (Web Content Accessibility Guidelines), it's a set of guidelines created by the W3C to make web content more accessible to people with disabilities. 2.2 is the last version of this standard, recently updated, and will be followed in the interface to comply at least at the AA level which is the second more demanding one.
- **Sections of the dashboard**: In this phase of the development, the UI is focused on the user and content management, the next sections are the currently available pages, under development:
    - **Login / Create user:** Allow access to the app and create new users
    - **Profile:** Allows users to update their data and preferences
    - **Upload:** Users can upload here their content (pictures, audio, video, 3D models). In case the content is a video, it will be possible to choose the transcoding qualities

- **Search:** Allows to search content, filtering by name or others (category, date, language…)
  - **Detail:** Shows selected content. The user will be able to: a) Preview the content, b) Update content metadata, c) Publish / Unpublish the content, and d) Delete the content.

The content could be pictures, audio, video (regular or 360), or 3D models. In particular, Audio and Video section plays the content with/without subtitles, and 3D allows the user to preview the 3D model, rotate and zoom it.

3.2.2 Application Tier

This layer is implemented by a set of modules which collaborate in order to provide the business logic described in the specification document.

The access point for this layer is the **API Gateway**. The API Gateway is acting as a reverse proxy and is responsible for managing the calls to/from the various modules exposed as microservices. The main responsibilities are load balancing, security and service discovery. In addition to the API Gateway we may use an **Orchestrator** module, responsible for triggering the various modules of this layer and synchronizing their output before returning it back to the end-user.

Some modules of this layer refer to the following functionality:

- **Digital Assets Management (DAM):** Assets and user management. Core functionalities of MV node described in detail in previous and subsequent sections.
- **Processing:** Modules offering functionalities required for indexing, content understanding, simple editing, tagging, content adaptation, etc. These can also be external tools or libraries exposing an API or can be extensions/plugins of the current modules implementations.
- **Local and Federated Searching:** These are the modules to enable search and retrieval of content either locally or remotely within the entire network of nodes.
- **Identity and Access Management (IAM):** This module could be part of DAM and will be responsible for the authentication and Identity Management of users.
- **Digital Asset Rights Management (DRM):** The Digital Asset Rights Management combines Smart Legal Contacts (SLCs) and the blockchain Smart Contracts (SCs) to be able to fully cover operational and legal aspects related to rights (e.g., ownership right, asset reuse) on digital assets. The SLCs will give the possibility to fully cover the legal aspects, being SLCs able to structure and manage traditional, legal contracts. On the blockchain side, MediaVerse will use the SCs for managing the notarization of SLCs, for managing the representation of rights on digital assets to support their transferability, tracing and auditability, as well as rights related payments.
- **Near Duplicate Detection**: NDD will be part of MV nodes and will provide specialized calls for indexing and searching images and videos. Each MV node will have its own instance of the NDD module and will support the identification of duplicates each time a new asset is uploaded in any of the MV nodes. Cases of (near-)duplicates will be handled by the DRM module.
- **Monitoring and Tracking Statistics:** This will be responsible for tracking the usage of external tools as well as the usage (or search retrieval) of media assets. These statistics shall be stored inside the scope of the MV node and must be exposed via API in a format which can be visualised by a User Interface.
- **Storage Interface:** Interface to multiple storage solutions, from local storage to external 3rd party solutions (S3, Google Drive, etc).
- **Content Delivery Service** will be responsible for the adaptation of content and the delivery to end-users according to the specifications of each device. More details can be found in D3.1 “Next Generation Content Model and Algorithms for New Media Types”.

3.2.3 Data Tier

Data tier consists of the elements which are responsible for data persistence, fulfilling different role each one:

- **Asset Storage**: The Asset Storage is responsible for the persistence and retrieval of Media Assets stored in each node. It is combined of different types of storage media which are described below:
    - **Media (Object) Storage:** The storage layer of the MediaVerse node refers mostly to the owned digital archives of the MediaVerse organization members. This is where the media files reside. Usually, these archives pre-exist and need to be ingested to the system during the initialization phase of the node. As described in the overall architectural diagram, the MediaVerse node will support several alternative persistence layers either on the cloud (Azure, AWS) or locally. For this purpose, we are going to implement a common interface hiding the complexity of each vendor or local system. This interface will therefore be vendor and technology agnostic and will be consumed by the upper layers of the MediaVerse node components for accessing the storage layer. The configuration of the storage solution will be performed through the management dashboard of the MediaVerse node during initialization and bootstrap of the node.
    - **Metadata Storage**: The metadata storage contains all the structured information related to media files and user management. All the textual annotations which are linked to a file can be stored in this place. We are planning to use a document-oriented database for this purpose (e.g. MongoDB). Apart from the associated metadata to the media files, this storage can act as a general purpose database, storing information related to the server and the users. The Database exposes an API to be accessible internally to the elements of the MediaVerse node, meaning that each module can connect directly to it and submit queries to store and retrieve content. For facilitating the interaction with the Database, we are also going to implement an HTTP Interface exposed by the orchestrator and consumed by the modules, reducing complexity and offering an abstraction layer hiding all the necessary checks and interlinking occurring in the background.
    - **Text Index**: From the user requirements, it is obvious that there is demand for full-text indexing and retrieval, offering features like faceting, hit highlighting, lemmatization and stemming. For this reason, we are going to enrich the storage layer of each node with a full-text-search engine (e.g. Apache Solr) which will offer the necessary functionality and expose the referenced features through fully featured APIs. The full-text search storage will be used by the search and retrieval module of the nodes during the search phase of content inside the nodes.
    - **Near Duplicate Detection Index**: NDD component consists of an asset index and a corresponding search / retrieval module to support detection of duplicates. The index consists of two parts: a MongoDB instance that keeps metadata and global features of the assets (videos) and a special structure stored in the local filesystem that keeps fine-grained features extracted from the assets (region-level features of images and videos).
- **Federation Shared Dataspace:** The shared dataspace of the federated ecosystem is based on the InterPlanetary File System (IPFS). IPFS is a protocol and a peer-to-peer network to implement a distributed system for storing and sharing content. The traditional addressing of content in the Web is based on a location-addressing system with HTTP protocol and URL for accessing content in a centrally located server. IPFS uses **content-addressing** to access content in a global namespace inside a decentralized peer-to-peer network. This decentralized network provides better durability, availability, resiliency and avoids SPF hot spots (Single Point of Failure) to the sharing content and it can speed up the retrieval for far away content and improve bandwidth efficiency.

This content-addressing uses CID or Content Identifier to address content in a distributed network. This CID is a fingerprint of the content based on hash functions like SHA. This content-addressing method allows de-duplication, integrity, immutability, and universal ID for the content. IPFS uses Merkle DAGs to represent the different blocks that IPFS splits the content when it stores it. Splitting the content into blocks allows that different parts of the content can come from different nodes in the network. Every block has a unique CID, but the user only needs to know the root CID of the content to get the full content. Besides, every node in the network has a unique ID or Peer ID.

For more information about IFPS, please visit the official website (https://ipfs.io/). We plan to use IPFS as the base technology for the implementation of the Content Exchange Service and the Node Discovery Service of MediaVerse network.

3.3 MV Node Architecture and Deployment

3.3.1 Micro-services Architecture

The micro-services architectural pattern allows different development teams to use their own technology stack in order to build and package their modules separately and let them interoperate with the external system by means of APIs. These modules are usually packaged as containers (e.g. Docker) which also includes their own storage and web layer. The orchestration of such modules can be managed by either a central orchestrator or by applying the so-called choreography, meaning that each module is listening to a shared message queue for specific events and submitting its output again to a similar pipeline.

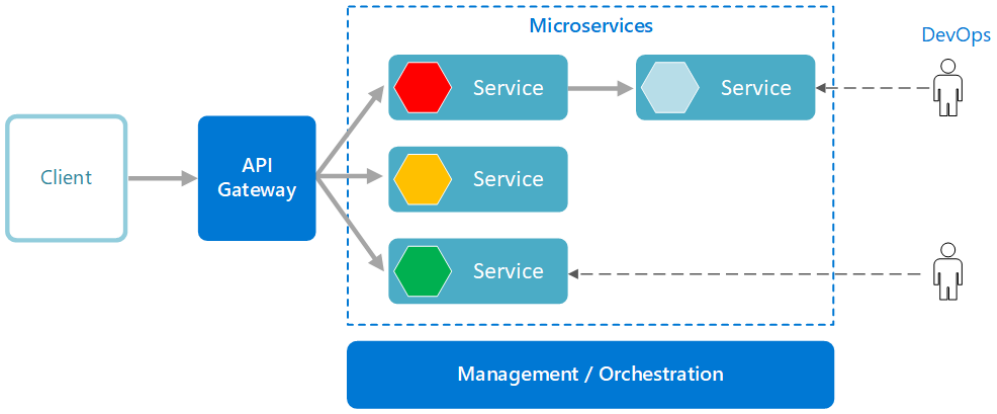

Figure 5: Microservices Paradigm, copyright: Microsoft Corporation

3.3.2 Containerized Packaging and Customization

The containerized packaging of each module will allow the extensibility of the MediaVerse node with alternative or additional modules in the future. For instance, a developer may implement and easily integrate a media-adaptation library to the node, by wrapping it up as a container. Each node owner will be able to customize the owned node by activating or deactivating the various features. From a marketing perspective, this “plugin” approach will also offer business opportunities for developers who want to monetize proprietary libraries and provide them under commercial license as part of the nodes. In general, the ability to easily customize a node in addition to the open source licensing of the project, will open MediaVerse node to the development community as a marketplace for their modules.

3.3.3 Distribution and Deployment

In order to attract more members to the Federated Ecosystem, we need to provide an easy way to deploy and set up a new node with a few actions. This involves not only the execution, but the distribution of the node elements.

We have examined various approaches for this purpose, such as the containerized approach, meaning that we will package the modules as software containers (e.g. Docker). These containers will also facilitate the DevOps process through applying easy-to-handle CI/CD pipelines to the various components of the system, automating the entire process of building, deploying and testing during the development phase. The management and orchestration of such containers can be managed by systems like Kubernetes or Istio.

Regarding the distribution of the MV node, the containerized packaging of various elements will be easily provided to the potential members of the network by being uploaded as images in public registries. The deployment and initialization of the node will be handled through a configuration file (e.g. Docker-compose, Kubernetes YAML, etc.) which will automatically download the images from the remote registry and run them on top of an environment-agnostic layer (Docker runtime). This facilitates the process of deployment and minimises the needs for specialized IT stuff from the side of the node owner. Any organization or even freelancer can set up a MediaVerse node with minimum software development knowledge by following the guidelines given in the public space of the MediaVerse project. Finally, this approach also offers the possibility that third parties will be able to offer easy setup and use of MV Nodes via a SaaS model (similar to Heroku and AWS).

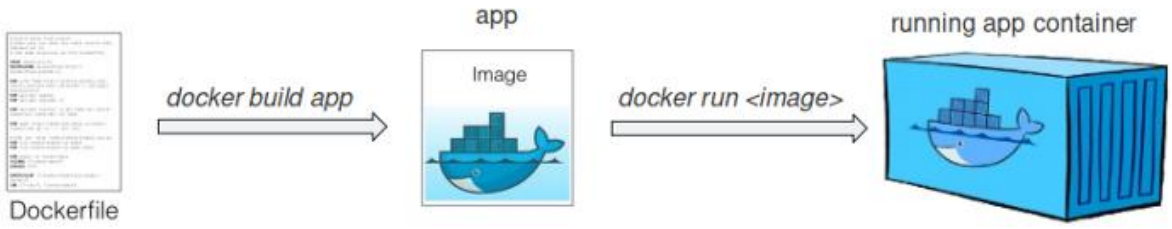

Figure 6: Development and deployment of MV services using a Docker-based workflow

3.4 Digital Asset Management

3.4.1 List of Features

The core element of MediaVerse node is the Digital Asset Management Module. The functionality of a DAM, split in steps can be described in figure 7, divided in ingesting-related (step 1 and step 2) and sharing-related (step 3 and step 4) features.

Each step reveals a set of different workflows that we are going to cover in the following parts of this document.

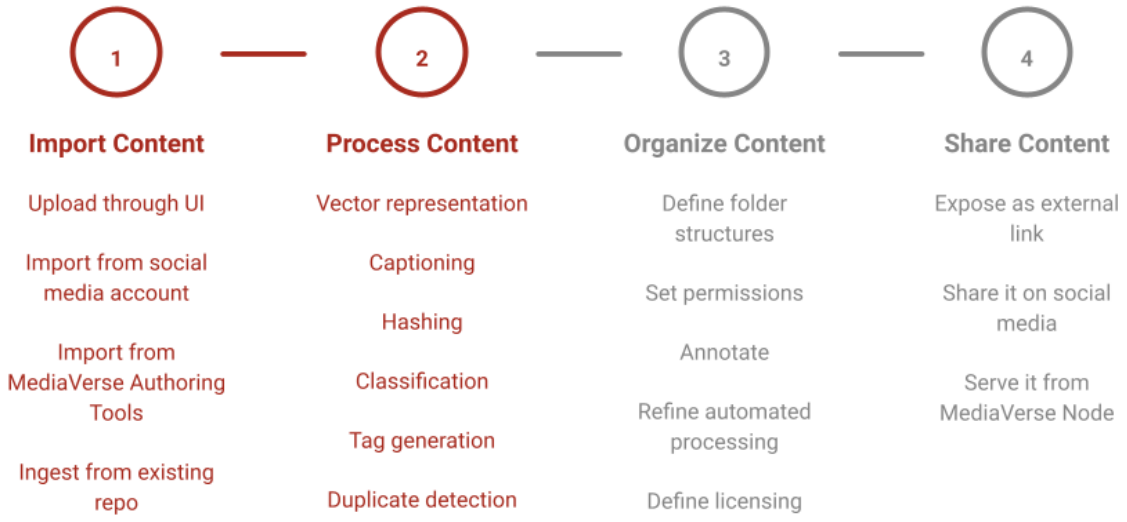

Figure 7: DAM basic functionalities

3.4.2 Import Content

A MediaVerse user can rely on multiple facilities to import media content to an MV Node:

- **Upload through UI**: The User Interface may use one of the various widgets and libraries to let the end-user upload a media file and send it to the backend through the MediaVerse Node Gateway. The MediaVerse Node on its turn, shall store the media file to the Media (Object) Storage, retrieve back its URL and start the processing pipeline, as described in following sections in order to populate the Metadata Storage and the various Indexes. This process of uploading is straightforward, so there is no further detail provided at this point.
- **Import from social media account**: For the interaction with social media, we are going to rely on the OAuth2.0 authorization mechanism provided by most of the major platforms. As described in the following diagram, the client is redirected through the web application to the relevant Authorization Server (e.g. Twitter, Facebook) in order to grant authorization access to the web application (MediaVerse node). Once approved, the Authorization Server will handle the necessary OAuth 2.0 tokens to the application. With these tokens, the MediaVerse backend will be able to interact on behalf of the user with the relevant social network and use its available API in order to retrieve content from it. Specifically for Twitter, there is a free-to-use public API which offers search and retrieval capabilities.
- **Import from MediaVerse Authoring Tools**: The MV Authoring tools will be mostly external applications which will interact with the MV Node through secured API calls. Each tool will provide its own mechanism to import and export content from/to MV Node, based on technology and capabilities. The integration approach for each one is described in detail in subsequent sections of this document.
- **Ingest from existing repository**: In some cases, the owner of a new MV node may store its own digital assets inside an existing object storage on the cloud, such as AWS S3 or Azure. In this case, we will extend the implemented APIs of the DAM in order to be able to manage the content from the existing cloud storage. This will be made possible by consuming the related vendor API, secured with the respective IAM API keys. These keys together with the storage address will be handled by the administrator of the MV Node and will be stored securely and encrypted inside the MV Node Metadata Storage. In general,

we are planning to follow software design patterns based on interfaces, thus to implement a common interface for all storage solutions (cloud vendors, local storage) encapsulating the logic and the APIs of each one. Various classes will implement the same interface, hiding the complexity and notations of each solution. This will facilitate development of the upper level code and will ease the switching between various solutions. This approach is illustrated in Figure 8.

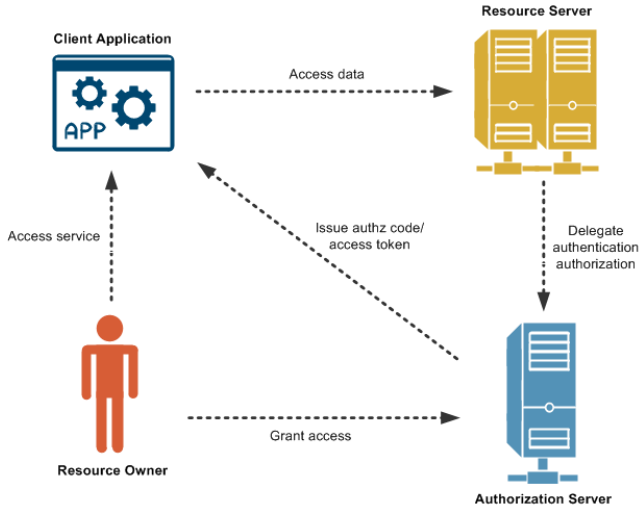

Figure 8: OAuth2.0 authorization mechanism. Copyright: oracle

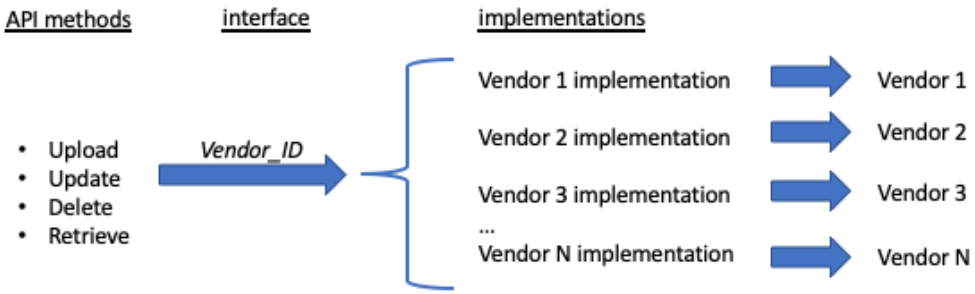

Figure 9: Interface design pattern

3.4.3 Process Content

The processing of the content will be performed through a pipeline managed by Orchestrator. Once a new file is ingested to the system, the DAM will be responsible for uploading the content to the linked storage (cloud, local). After the successful upload, the orchestrator will start triggering the various modules in synchronous or asynchronous way, based on their response time. It will also pass the URL of the file as a parameter (unless it is a simpler format like text) to let the modules download the file directly from the storage layer. It is important to note that the processing modules may reside inside or outside the node as external services.

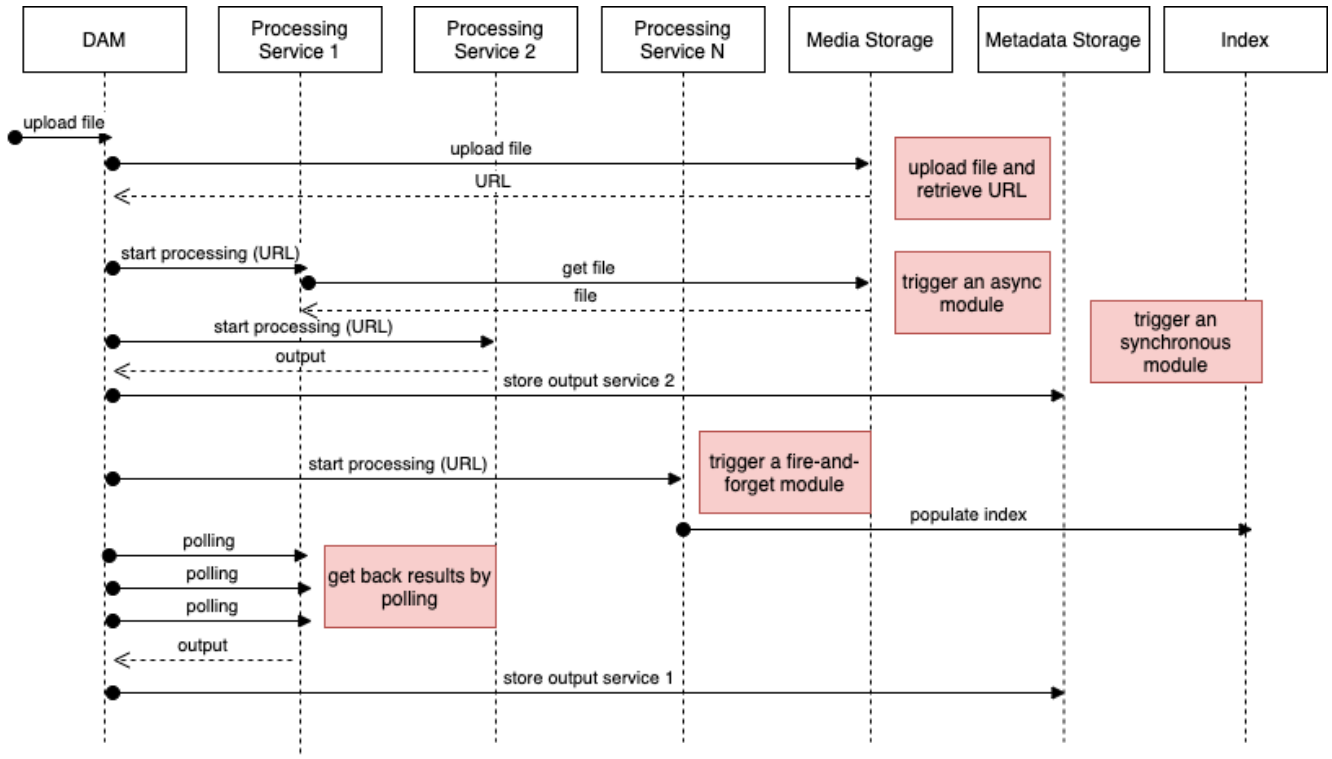

Figure 10: Media Processing Workflow UML Diagram

For the sake of example, in the sequence diagram above, we have added three processing services which behave in a different manner:

- **Asynchronously**: As seen in the diagram (service 1), the orchestrator performs polling (or uses alternative techniques like gRPC[10]) until the results are ready by the module. This fits to cases where the respective module needs longer time to process a media file. The long polling approach implies that the initiator (orchestrator) starts sending periodic HTTP GET requests to the processing module in order to fetch the results until they are ready. Once the results are fetched, the orchestrator stores them in the metadata storage.
- **Synchronously**: This way refers to modules with very short response time, which can return their output as a response to the HTTP call of the orchestrator. As in the previous case, once the orchestrator gets back the results, it is responsible for saving them to the metadata repository.
- **Fire-and-forget**: These kinds of modules do not necessarily return results to the sender, instead they deal with it independently. They may affect the indexes or metadata storage of the MediaVerse node, or they can keep internal indexes (e.g. media index for duplicate detection) inside their microservice container. With the fire-and-forget approach, the initiator gets immediately an acknowledgment of the initial call and never comes back to fetch the results.

[10] https://grpc.io/

3.4.4 Organize Content

The content-management features referring to grouping and annotating mostly involve interactions with the metadata storage. This is a functionality available to the end-users through the User Interfaces and will be reflected inside the MV Node Metadata Storage. All those actions will be handled directly by the API of the DAM, which will be responsible for executing the relevant business logic by orchestrating various calls to all involved modules. This task, in addition to grouping and annotating, may involve interactions with the Digital Asset Rights Management component, which will be covered by a specific set of modules described in following sections.

3.4.5 Share Content

With respect to social media, similarly to importing content, in order to share content, we are going to reuse the same tokens provided during the OAuth2.0 handshake[11]. This is a necessary step which has to take place either during the user login phase or when the user wants to interact with social media. Depending on the social media platform, these tokens may expire so they have to be renewed. It is essential to take into account that there are various levels of authorizations, so, in this case, when asking for authorization by the related social media, MediaVerse shall also ask for "publishing on behalf of the user".

For sharing content directly via the MediaVerse local storage we need to rely on a web server to serve the media files. There are various flavours of web servers available, nevertheless, the technical decisions will be taken during the implementation phase of the project. In order to hide content from public access and expose it only to privileged users, an idea could be to use long UUIDs inside the URL and of course remove list/browse capabilities of web server folders.

Finally, to serve content from cloud storage (e.g. AWS S3) there are various options, such as "presigned URL" provided by Amazon. The workflow is described in figure 11. Since it is a vendor-specific solution, we should first verify that a similar solution is available by all the vendors we are going to support for creating a ubiquitous interface to cover all cases.

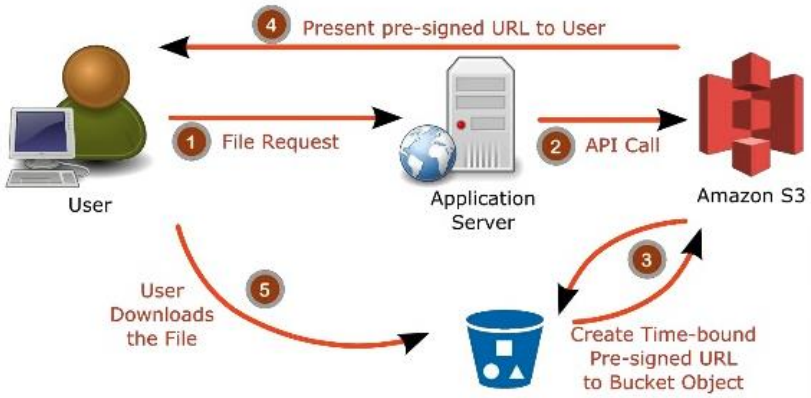

Figure 11: AWS Presigned URL - copyright: AWS

[11] https://oauth.net/2/

4 Inter-node Communication

The interaction between nodes involves tasks related to exchanging metadata information and multimedia files among nodes, searching content inside the entire network, authenticating users in a federated approach, sharing access rights, etc. This communication between nodes implies the implementation of certain components and mechanisms to ensure discovery, security and stability of the network. These mechanisms are described below.

4.1 Node Discovery

In order to maintain the distributed nature of the solution, we have decided to follow a decentralized approach for the node registry. This means that each node will have access (cached locally or inside Federation Shared Dataspace) to a synchronised record with all the existing nodes of the MediaVerse network, including their IP/domain and their public key. Each MV Node will expose a set of HTTP methods to read/update/delete records from the registry.

This Federation Shared Dataspace will also serve as **Node Registry and Public Keys Storage**: The nodes will be able to discover the location (IP, domain name) of all the active nodes of the network through a shared registry. Due to the federated nature of the solution, the plan is to keep a copy of the registry on each separate node. Every time a new node is added to or removed from the system, the registry has to be updated and made available to all nodes of the network. Moreover, the need for secure communication and trust between the nodes on a peer-to-peer basis, imposes the necessity of key pairs. The keys can be created on the initialization of any new node and the public keys will be propagated to the Node Registry files kept by each node. This will enable any node signing or validating the signature of any message exchanged with other nodes, thus securing their communication. After every change inside the registry of a node, an automated process must be able to synchronize the update with all external nodes. We are investigating the option to include this registry inside the Federation Shared Dataspace.

IPFS uses a distributed hash table (DHT) split across all the peers in the distributed network. This DHT is built using the Kademlia algorithm and it is a mix of catalog and navigation system. We can use this table to discover peer nodes and routing requests of content to any hosting peer using a defined lookup algorithm. Every peer node asks its near nodes and these in turn ask to their known peer nodes. There are three types of key-value pairings that are mapped in DHT and one of them is peer records (map a peerID to a set of multi addresses at which the peer may be reached).

4.2 Adding a New Node to the Network

From a business perspective, we want to eliminate the barrier to entry to the MediaVerse ecosystem and make MediaVerse attractive for any individual or organization. This will be possible by offering a simple process with minimum steps for downloading, configuring, launching the node and joining the network. For the registration of a new node to the network, the plan is to keep the process easy and transparent for everyone, maintaining only a basic mechanism to ensure that the network's terms and conditions are not violated.

Since we are using the Federation Shared Dataspace for the discovery of nodes, all we need is to create a mechanism that, upon first launch of a new MV Node, registers the necessary information for its discovery to the IPFS space. This information, together with the public keys of the node needed for secure authentication will be available to the rest of the network, so the new node will be immediately discoverable. In order to delete a node, a new entry must indicate that the node is no longer valid and has to be removed. It is not clear if the IPFS storage regarding the registry must be cached locally on each node and updated frequently. Also, the access to

the registry will be offered through an API, so the underlying mechanism will be transparent to the rest of the MV Node modules.

Finally, IPFS allows us to create public or private networks. In a private network one node can only connect with other nodes of the network if it has a shared secret key (swarm key). Nodes in a private network do not answer to nodes outside of the network.

4.3 Secure Communication and Authentication

As described previously, each node will create a set of keys (public/private) on startup. The private key will be stored inside the node and will be protected from external access. The public key will be propagated to all other nodes of the network through the Federation Shared Dataspace. With this pair, the nodes will be able to establish a secure channel, for encrypting and therefore protecting their communication from man-in-the-middle attacks.

The key pair, apart from encrypting the exchanged data, can also be used for ensuring authenticity of the counteracting part. Thanks to cryptographic algorithms, the sender node can sign the message with the private key and the receiver node can validate the signature by using the sender's public key, making sure that the messages are received from a trusted party.

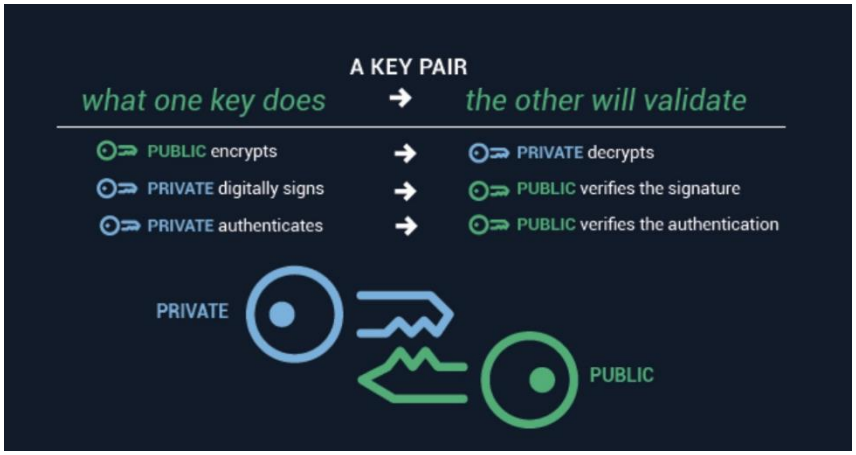

Figure 12: Asymmetric encryption algorithm key pair

Any incoming message towards a node has to be passed initially by the Gateway which will act as an interceptor in order to validate the signatures of the received message. Only the Gateway will be exposed through a public IP or domain name. The internal microservices will be able to send HTTP messages outside the node; however, in order to access any API of an external node they need to do it through the remote node's Gateway. Thus, they need to pre-sign the HTTP message with the necessary keys in order to avoid being rejected. For the signature management, there will be a specific module accessible through the Gateway.

The described mechanism for the communication between MV Nodes will be implemented with the use of mTLS (mutual Transport Layer Security). This is a protocol that establishes an encrypted TLS connection in which both parties use X. 509 digital certificates to authenticate each other. The implementation of the communication can be implemented by the solution of a service mesh such as Istio.

Istio and Kubernetes

A promising candidate which we will investigate regarding the implementation of the secure communication is Istio[12], an extension of Kubernetes offering a service networking layer that provides a transparent and language-independent way to flexibly and easily automate application network functions. X509 certificates for mTLS are automatically generated, renewed and distributed via specific facilities in the Istio Control Plane and side-proxies. As depicted in the figure 13, Istio has:

- the Control Plane through which the service mesh can be configured, controlled, monitored.
- the Data Plane where managed (micro-)services operate. When deploying a (micro-)service Istio automatically deploys a sidecar proxy (Envoy Proxy in the figure) that (a) mediates access to each service, (b) communicates with the Control Plane to apply access rules, traffic filtering rules, etc.

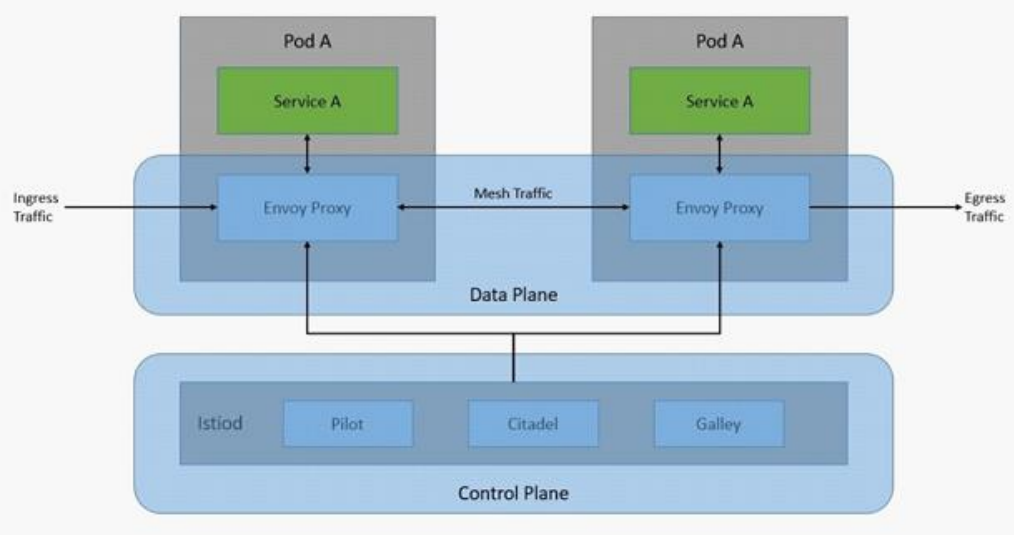

Figure 13: Communication through Istio service mesh

Istio also supports the automatic deployment of ingress/egress gateways[13].

These gateways are simply Istio proxies that don't control access to a specific (micro-)service:

- An Ingress gateway manages requests coming from the external environment (e.g., from user's browsers) directed to the managed (micro-)services.
- An Egress gateway manages requests from (micro-)services within the managed service mesh toward external services.

Both Ingress/Egress gateways support AuthN/AuThZ via the same mechanisms and similar rules as for the side-proxies inside the managed service mesh.

In the context of MediaVerse, to be aligned with the MediaVerse federation approach, each MV node will be deployed as an autonomous Istio service mesh. Therefore, each MV node will have its own Control Plane / Data Plane. Requests to MV node services coming from the MV node web UI (e.g., from a user using the MV node functionalities) or other external tools (e.g., Fader) will be managed via the MV node Istio Ingress gateway. On the other hand, communication between two MV nodes will be managed via the Egress gateway on the MV node requesting access to the other MV node and the Ingress gateway on the other MV node.

[12] https://istio.io/

[13] https://istio.io/latest/docs/concepts/traffic-management/#gateways

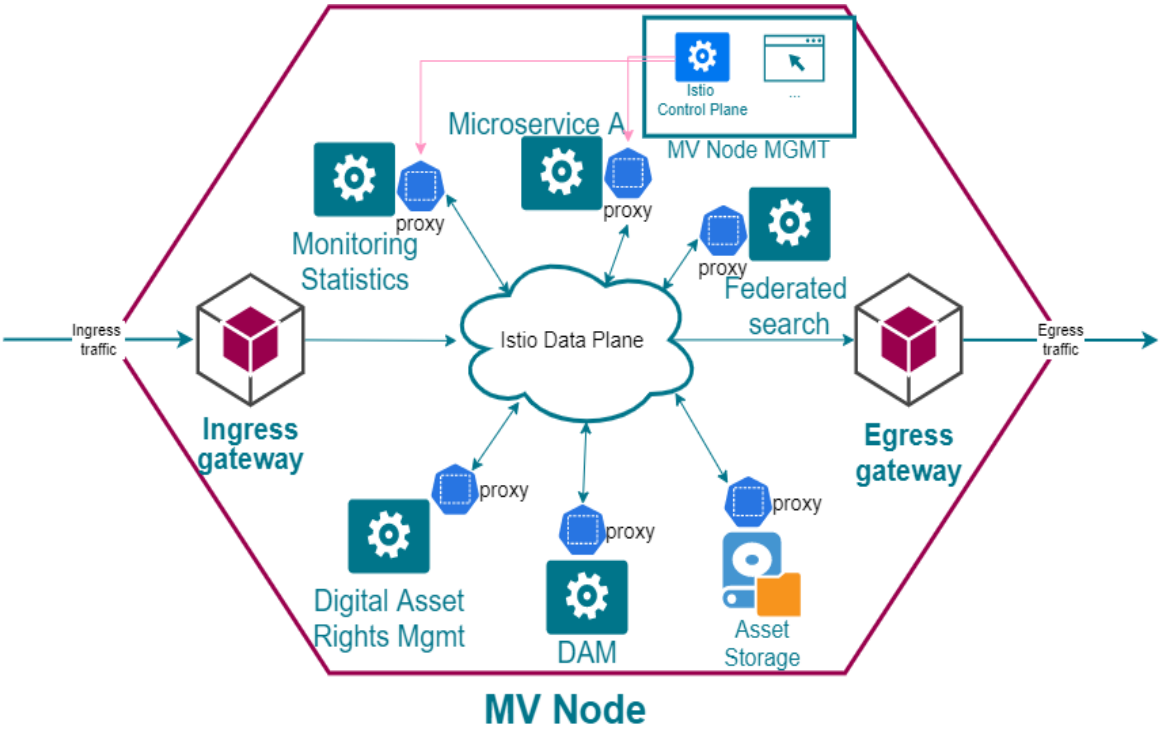

Figure 14: Istio-based MV Node Architecture

In this way there will be the possibility to both control that only specific services (e.g., the search service) on an MV node can access external services (e.g., another MV node) thanks to the Egress gateway AuthZ policies, and that only trusted MV node can submit request to another MV node thanks to the MV node Ingress gateway.

IPFS

When we talk of encryption of content in a distributed system we have two types: encryption in transit (transport) and encryption at rest (storage). IPFS uses encryption in transit by default. The content is secure when traveling from one IPFS node to another. IPFS intentionally does not implement any encryption at rest or force to use any particular encryption protocol. Any encryption at rest has to be done in the application layer before adding it to the IPFS network.

4.4 Federated Identity and Single-Sign-On between MV Nodes

Each node is responsible for storing the list of registered users, together with their role inside the node. These users will have a unique id inside the specific node. Therefore, the pair (node_id, user_id) is unique throughout the entire network and can be used globally for the reference to a specific user in a human-friendly way such as userA@nodeA. This pair can be used in the IAM module of each node in order to let users (or perhaps roles) of other nodes access their content. This mechanism will be used on retrieval time, when a user of a node attempts to browse and retrieve content from external nodes.

By using the global id of this user under the secure communication channel achieved between two nodes via their private/public key pair, there is no need for a user to hold various accounts in various nodes. By authenticating against one node, the user will be able to access content from another, external node if is given such permission by the administrator of the latter. This simplifies the architecture and bypasses the need for

Single-Sign-On mechanism or central federated-id provider which could affect the federated architecture of the system by adding centralized elements to it.

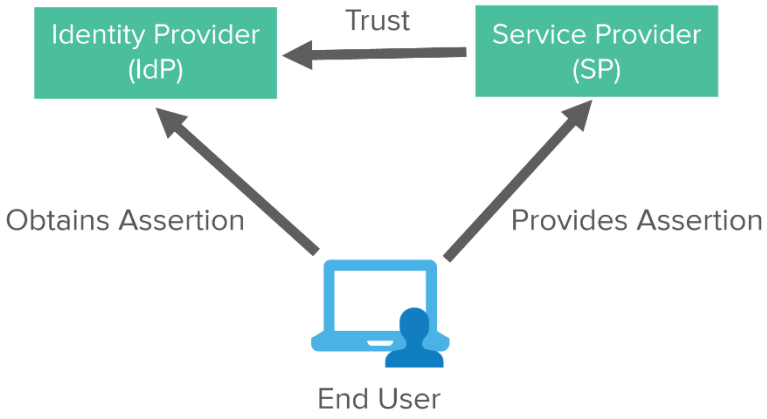

Figure 15: Federated Identity - copyright: okta.com

We need to highlight the fact that the user's credentials will never be transferred to another node but be securely maintained on the principal node. This means that the authentication handshake through the login workflow has to be performed always on the principal node, even in the case where the user has gained privileged access to a second node through his/her global id. This will be implemented through a similar to OAuth2.0 workflow where in order to login to a secondary node, the user will be redirected to the principal node and get back to the secondary node after successful authentication of the credentials. The details of this process are described in the diagram below. In this case, the user is originally registered to Node A and he has gained access to Node B (secondary). The user tries to login to Node B as illustrated in Figure 16.

Regarding the user credentials, we are going to follow all the recommended guidelines (hash, salt) to ensure that these are stored securely inside the database. We are also going to implement all the necessary features for updating or resetting passwords. Moreover, we are going to provide an option for authentication under the OAuth2.0 handshake with major OAuth providers (e.g. Google, Twitter) to avoid remembering passwords.

Finally, once the user is authenticated with the relevant MediaVerse node backend server, the server will sign and send to the client application a JWT token. This token can be saved on the session storage of the browser and be used to authenticate every subsequent HTTP request from the front-end to the back-end onwards, in order to avoid extra authentications. This workflow is described in the diagram of Figure 17.

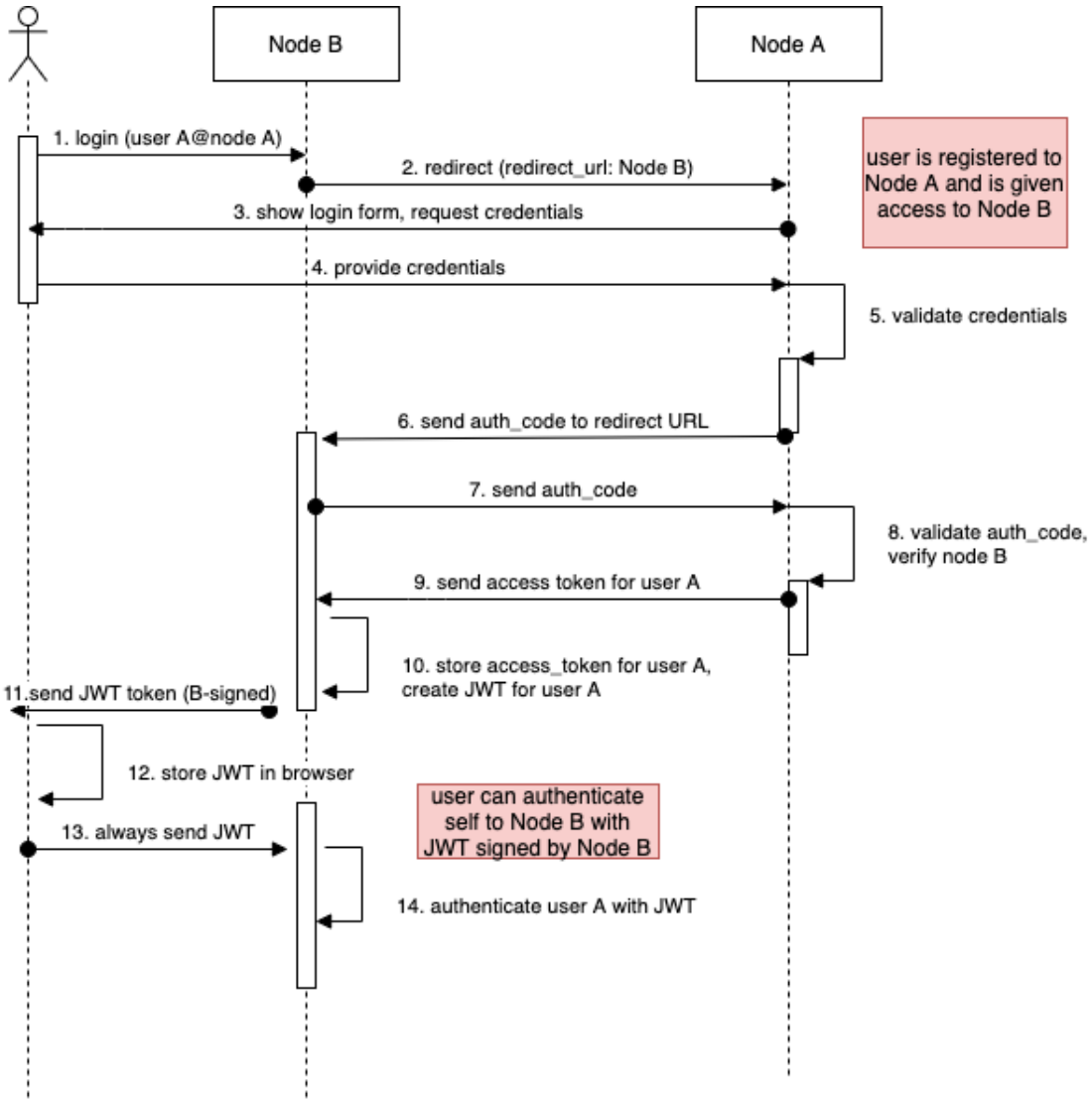

Figure 16: Federated Authentication workflow

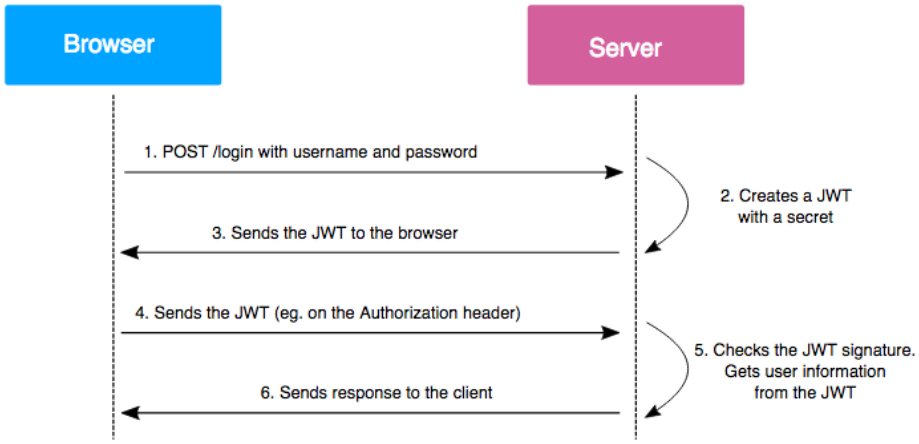

Figure 17: Authentication with JWT token

4.5 Federated Search

Every MediaVerse node will implement a "local" search service of content based on Solr Engine, NDD index and Metadata Storage. This service will allow users of this node to search for local content without searching outside the node content repository.

Users will also need to search in the MediaVerse network for content. For this reason, we will provide a new service to launch a search in the MediaVerse network. This federated service will act as a proxy to connect to the rest of the nodes of the network and send a search request to the local search service of each node.

This federated search has many challenges like node discovery, routing search request, request query format and protocol, response result format and protocol, aggregate search results, network communications, response time and timeouts, search strategies or approaches, and others. Aggregate search results will be an incremental and iterative process because the MV network will be composed of heterogeneous nodes with different infrastructures (computing capacity, network bandwidth and latency, repository size, number of users, etc.) and the responses will be asynchronous.

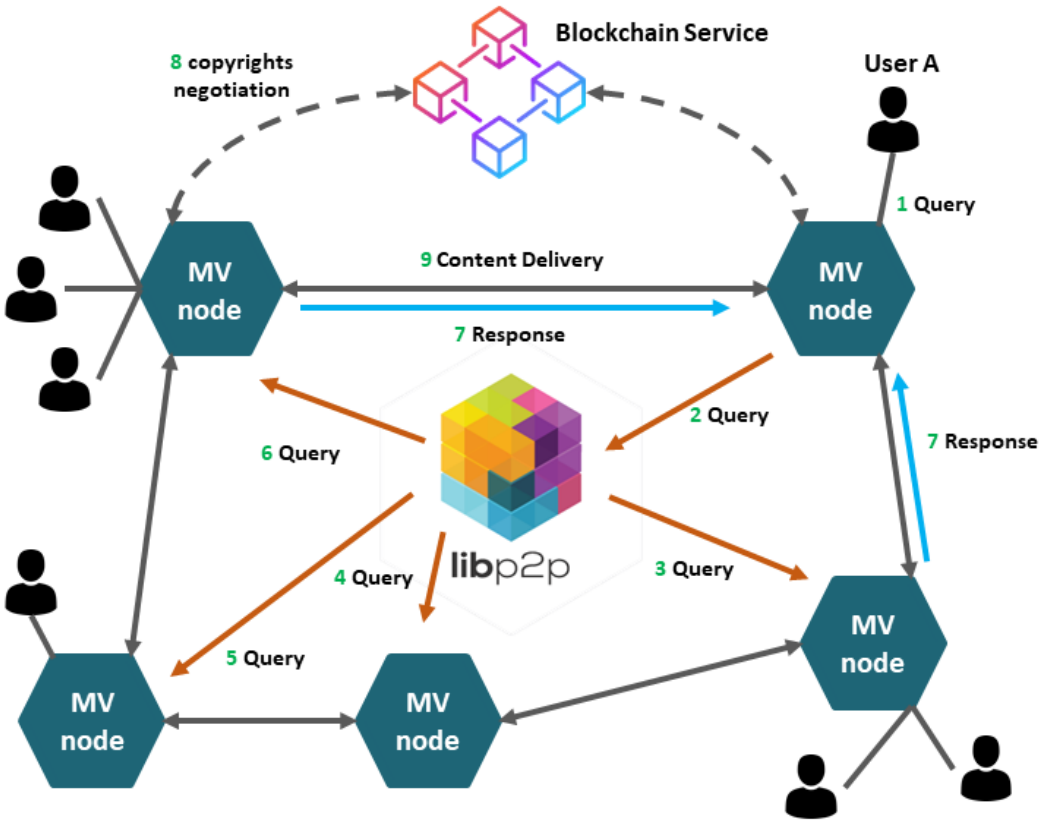

Figure 18: Federated Search Diagram

We plan to implement a federated search service with a public REST API to isolate the details of the "local" search service REST API (based on and visual index) and a pub/sub communication paradigm (one to many) to send the search request to the rest of MediaVerse nodes (inter-node communication). This paradigm has several advantages like loose coupling of producers (searcher) and consumers and better scalability.

For the pub/sub implementation we will try the pub/sub functionality of libp2p library used by IPFS and the DHT database for node discovery and routing. Figure 18 depicts a high-level overview of the distribution of queries across the MV network, with the routing being performed by the pub/sub mechanism of libp2p. More precisely, we can define a well-known topic that all the nodes can subscribe to receive remote search requests from the network. To figure out where to send the response we can use a stable topic unique per node (based on some public metadata like node ID) or to use an ephemeral topic sent in the request. Figure 19 provides an overview of the workflow of federated search using the IPFS network.

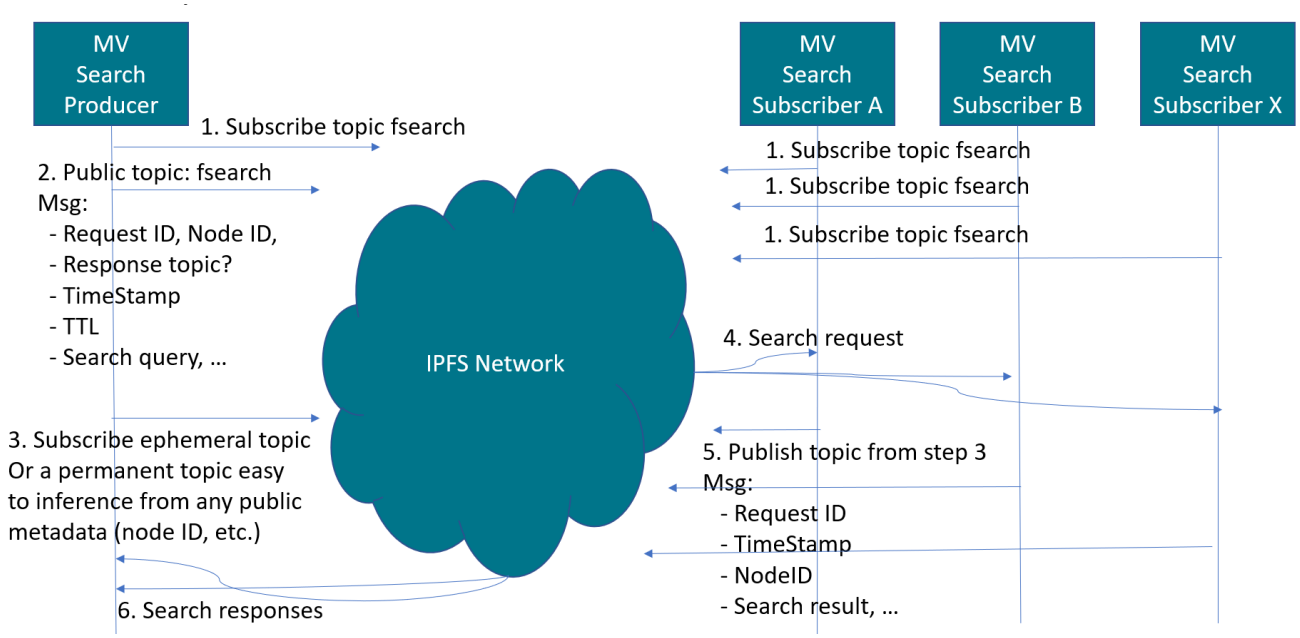

Figure 19: Federated Search pub/sub approach explained

5 Media Rights Management and Content Identification

5.1 Overview/Brief Description

Rights Management in MediaVerse aims at covering legal and technical aspects of IPR through a common digital rights management model, specifically by providing a machine-augmented legal framework to support the representation, registration, storage and negotiation of media content according to the corresponding IP definition and associated licenses.

Practically, The Rights Management component will be responsible for managing all the features related both to the Smart Legal Contracts (SLCs[14]) and the blockchain Smart Contracts (SCs[15]), ensuring the implementation of the required legal and technological aspects for the legally-enforceable management of digital assets rights within MediaVerse. Specifically, we refer to the Accord Project platform for the management of SLC templates and running the engine responsible for processing SLCs and related events or queries, and Ethereum as the blockchain platform.

The Rights Management component is composed of three sub-modules (see Fig. 20).

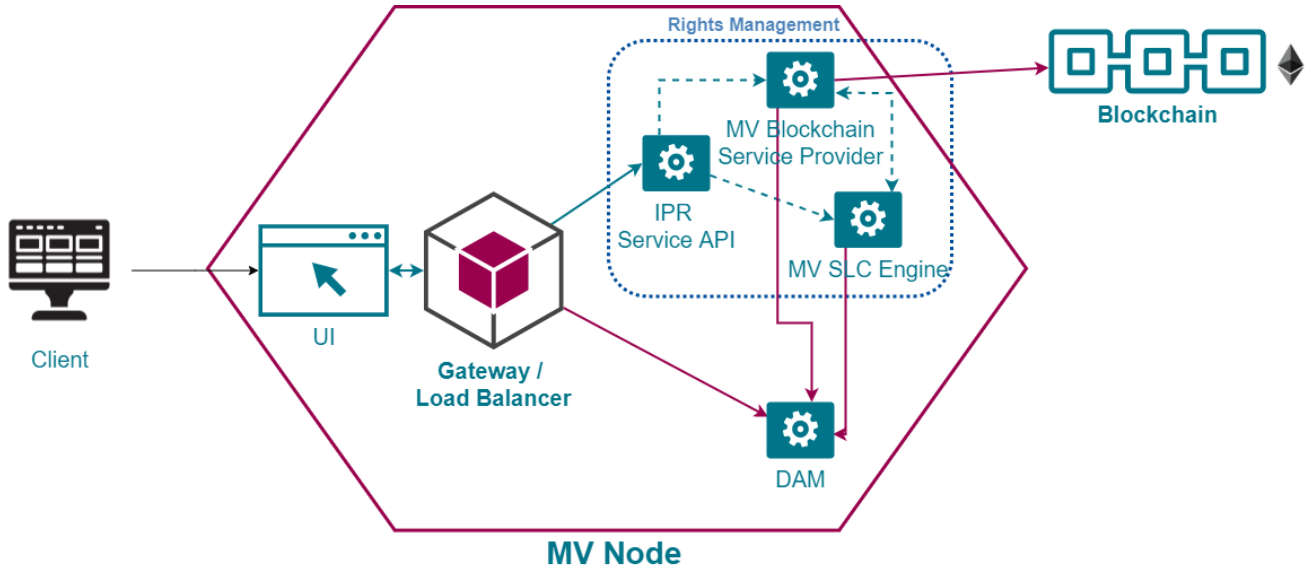

Figure 20: Rights Management Architecture Overview

- The IPR Service API, which will interface the component and will provide an API to interact with the Blockchain Service Provider and the SLC Engine.
- The MV SLC Engine, which will handle all the run-time features related to SLCs (e.g, the creation of SLC instances, check for the trigger of specific SLC-related events).
- The MV Blockchain Service Provider, which will handle the deployment and the management of the Smart Contracts 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 will be transparent with respect to the user.

[14] Human-readable and machine-readable agreements that are digital, consisting of natural language and computable components.

[15] A form of software that can be executed on the blockchain.

For further details, please refer to D4.1 Section 4 “Implementation of Copyright Management in MediaVerse”.

The Rights Management component will be an integral part of the MediaVerse Node. Users will interact with the Node through the UI that will interface with all the services provided by the node. However, the interaction with the blockchain will be completely transparent to the user. To this end, users will not have to manage a blockchain wallet; rather, they will operate directly on their MediaVerse account, which will abstract the blockchain layer by means of the MV Blockchain Service Provider that will perform blockchain transactions on their behalf, while being able, still, to keep track of their digital rights (e.g., Ownership deeds, right of use, etc.) and their balance in the fiat currency of their choice.

The Near Duplicate Detection component (NDD) identifies near and exact duplicates of assets uploaded in each MV node. Each node locally runs its own NDD service and leverages the same mechanism as federated search (section 5.5) to identify duplicates across the MV network. Local NDD consists of an asset index (e.g. a MongoDB instance) and a corresponding search/retrieval module to support detection of duplicates. Given a query multimedia asset (i.e. the newly uploaded asset), it returns the similarity of the near-duplicate items found in the index. Duplicates can be either exact copies or segments of an asset embedded in another asset (e.g. video segments). The retrieval procedure is performed in two steps:

- A set of candidate assets is retrieved from Metadata Storage (e.g. MongoDB using a special MongoDB query).
- The candidates are compared with the query asset using the corresponding visual signatures, i.e. representations based on state-of-the-art deep learning architectures that capture the content of each asset and allow their efficient similarity-based comparison.

Further details about the NDD component will be provided in the deliverable “D4.3 - Content Identification Services”, due by M24 (September 2022).

6 Extension Services and Tools

Despite the federated and decentralized nature of the project, there is a set of tools and extension services which are foreseen to be provided externally (even though there may be cases when they can be packaged/deployed together with the MV node). This occurs due to technical reasons (e.g. complexity in deployment, high demand in hardware resources) or business restrictions (security, commercial licensing). The integration of these tools will be performed either through exposed APIs or as-a-service, meaning that the MediaVerse users will be able to access them under the same authentication mechanism of their MediaVerse node. The lists and integration approach for each external tool are described below.

6.1 Content Moderation Toolset

This module consists of various content analysis libraries which are responsible for the automated moderation of the ingested content, such as Hate Speech Detection, NSFW content, disturbing content, and misleading and fake content.

The toolset will consist of a number of microservices. It will also expose its own User Interface for configuring and reviewing the annotated content, but also for defining moderation rules, and facilitate the training and fine-tuning of the underlying AI components. Since the moderation is always semi-automated and requires a final (manual) step to approve the content to be ingested, this module also requires an internal database to act as an intermediate buffer for the incoming assets, until they are finally selected to be part of the DAM repository. Figure 22 provides an overview of the moderation tools and its place within a MV node. The user interface can be used by the admin of a MV node to define and set up a set of moderation rules, applicable within the boundaries of the node. Users can provide feedback for the retraining and fine-tuning of the AI moderation models by inspecting the annotated content. The main functionality of the tools, i.e. the annotation of content with moderation-related tags (hate speech, profanity, etc) is exposed through an API, which is used by the DAM to analyze the newly uploaded content.

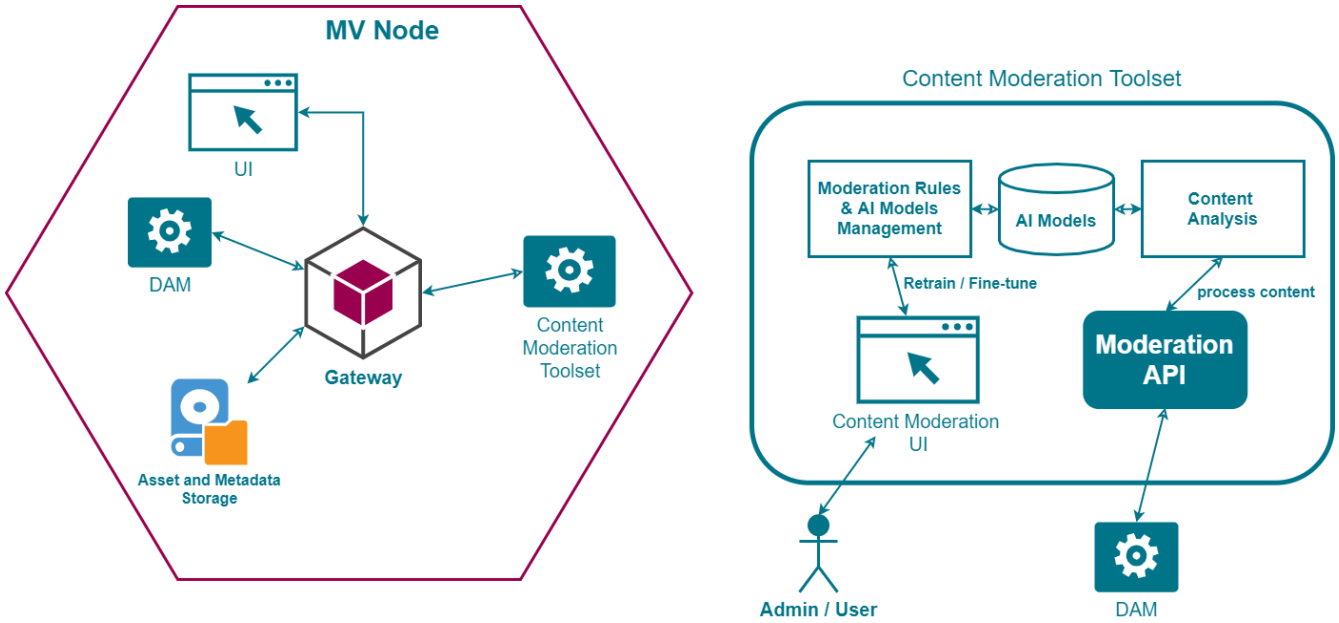

Figure 21: Content moderation toolset as part of the MV node

6.2 Truly Media

Truly Media is a collaborative environment for bringing together teams of users for fact-checking purposes. It also provides a set of fact-checking tools to the users. It is hosted on the cloud and exposed through a Software-as-a-Service (SaaS) model, providing an intuitive UI for the users and a management dashboard for the organisation administrators.

In the scope of MediaVerse the idea is to use it as-a-service, in two possible ways:

- Via consuming its API, in order to push content from a MediaVerse node inside Truly Media collections. The collections are common workspaces for fact-checking teams. By ingesting a media item to a Truly Media collection, a user will make this visible and accessible to a team of fact checkers for further investigation. Truly Media API offers a method for polling and retrieving the current verification status of a media item including the notes and detailed annotations provided by the fact checkers.
- By allowing privileged MediaVerse users to access the Truly Media fact-checking tools available. A MediaVerse user shall be redirected to the web interface of Truly Media in order to access it. The authentication shall be made transparent. However, since OAuth2.0 handshake is a prerequisite for Truly Media, we will need to design this process on a two-step verification process. An alternative way would be to create and launch a minimised version of Truly Media, keeping only the fact-checking features available to the users. The definition of the entire process will be part of the discussions during the integration phase of the project.

6.3 TruthNest

TruthNest is an online tool used to analyse Twitter accounts in order to retrieve metrics about their activity, network and influence, as well as account labels and a bot probability score for each account. There will be a soft integration with MediaVerse Node user interface by enriching any possible references to Twitter accounts with relevant URL links which open the corresponding analysis of TruthNest on a new browser tab. For instance, the button highlighted in the following screenshot will open a new tab with the URL washingtonpost, which corresponds to the analysis results of the specific account.

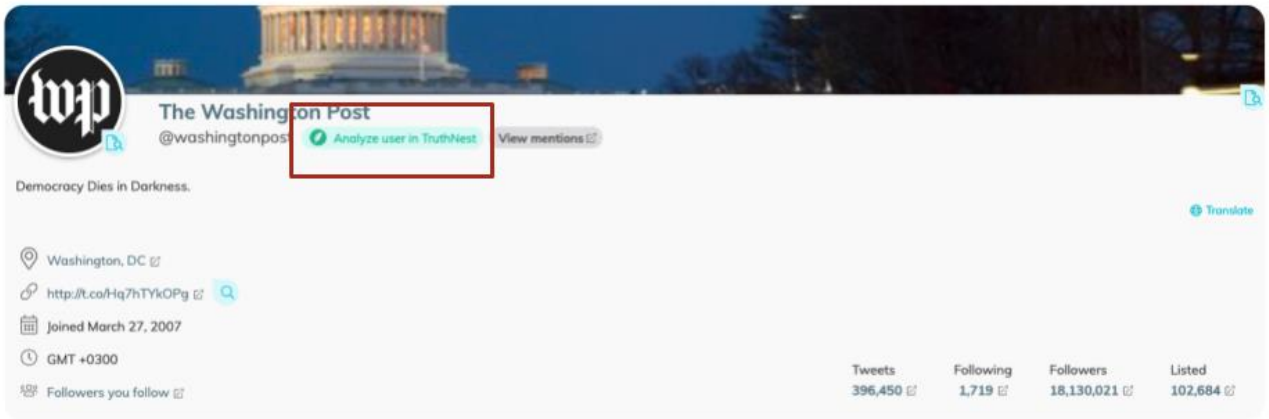

Figure 22: TruthNest User Interface

6.4 Media Annotation Service

This component will be responsible for the annotation of media assets (images, videos, 3D objects) to support search and retrieval. The annotation consists of automatic tag extraction and caption generation. In addition, vector representations of media items will be produced by this model to be used for cross modal retrieval by the models developed in T3.2.

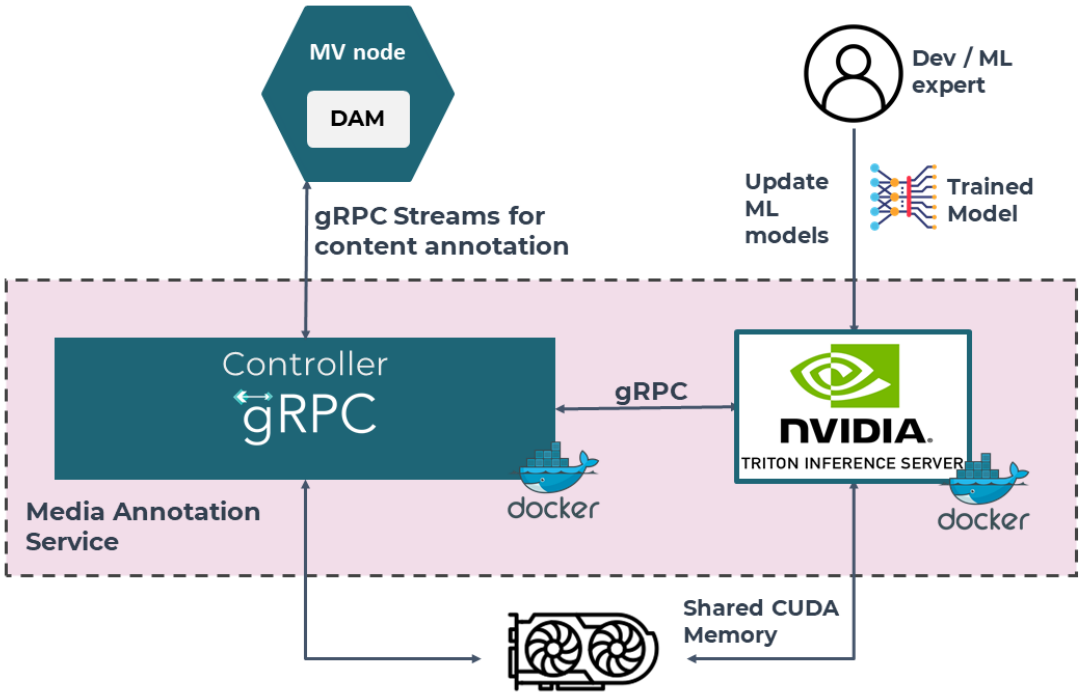

Figure 23: Architecture of the media annotation service and interaction with MV DAM

The media annotation service will offer a gRPC-based API[16] which can be used to upload media assets and get back annotations. Its internal architecture is depicted in Figure 23. The communication, e.g. with the MediaVerse DAM, is handled with an gRPC based controller, while the deep learning models that will be used to annotate media assets are managed by the Triton Inference Server[17] component[18]. We opted for Triton as an easy way to deploy deep learning models, considering its functionalities such as model versioning, concurrent model execution and the support of multiple backend frameworks. Given the number of models that need to be trained and deployed, that component is a key decision for the media annotation service. The gRPC Controller acts as a proxy between the service that requests asset annotation (e.g. DAM in our case) and the Triton server. An asset is uploaded through the controller, preprocessed and then is loaded in the shared GPU memory for processing in the Triton Inference Server. The results are directed back to the caller through the same gRPC channel. Note that media annotation service is stateless, assets are not stored and all the representations of assets needed by the ML models are deleted after processing. Both components, gRPC Controller and Triton Inference Server, are deployed as separate docker containers communicating with the gRPC protocol.

[16] https://grpc.io/

[17] https://developer.nvidia.com/nvidia-triton-inference-server

[18] For more details on the ML models that will be used, please refer to section 2 of D3.1.

6.5 Fader

Fader is a SaaS to create and publish interactive 360-degree stories with added 3D capabilities. Fader will be available for MediaVerse accounts from specified nodes to create and sign in to an account. This will also allow access to media from MV nodes to be used in Fader stories.

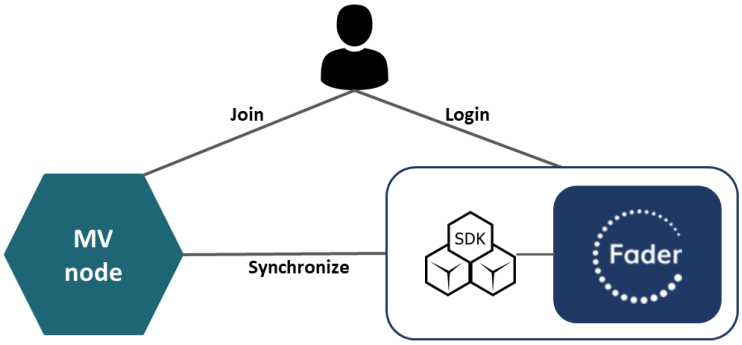

Figure 24: Fader Integration with MV Node

Integration: Fader will add MV authentication via OpenID as a means to create and sign into an account. This will allow access to MV nodes media, by using the API exposed by MV nodes. The current media library for Fader will be extended to allow selecting MV media as explained in the following. Currently, users upload media directly via the Fader Editor. It can be viewed and managed in the Media Library window. Media can be set private or public, thus allowing others to reuse media in their stories. With the extension or integration of MV media, Fader users can have access to MV DAM functionality and select media processing services that are aligned with the requirements, available via MV API and feasible within the context of the project.

The Fader Editor is a web front-end editor specifically designed for desktop use. It allows the composition of single media assets (video, images, audio, text) into an interactive 360-degree story that itself may consist of several scenes that can be linearly or interactively connected. As such, Fader allows the creation of various content types from tours to product presentation to quizzes.

The Fader Player is a web and mobile web JavaScript application to play Fader stories. It offers optional scene access, can be configured in colours, logos and fonts and is aimed to run on any web browser and various HMDs.

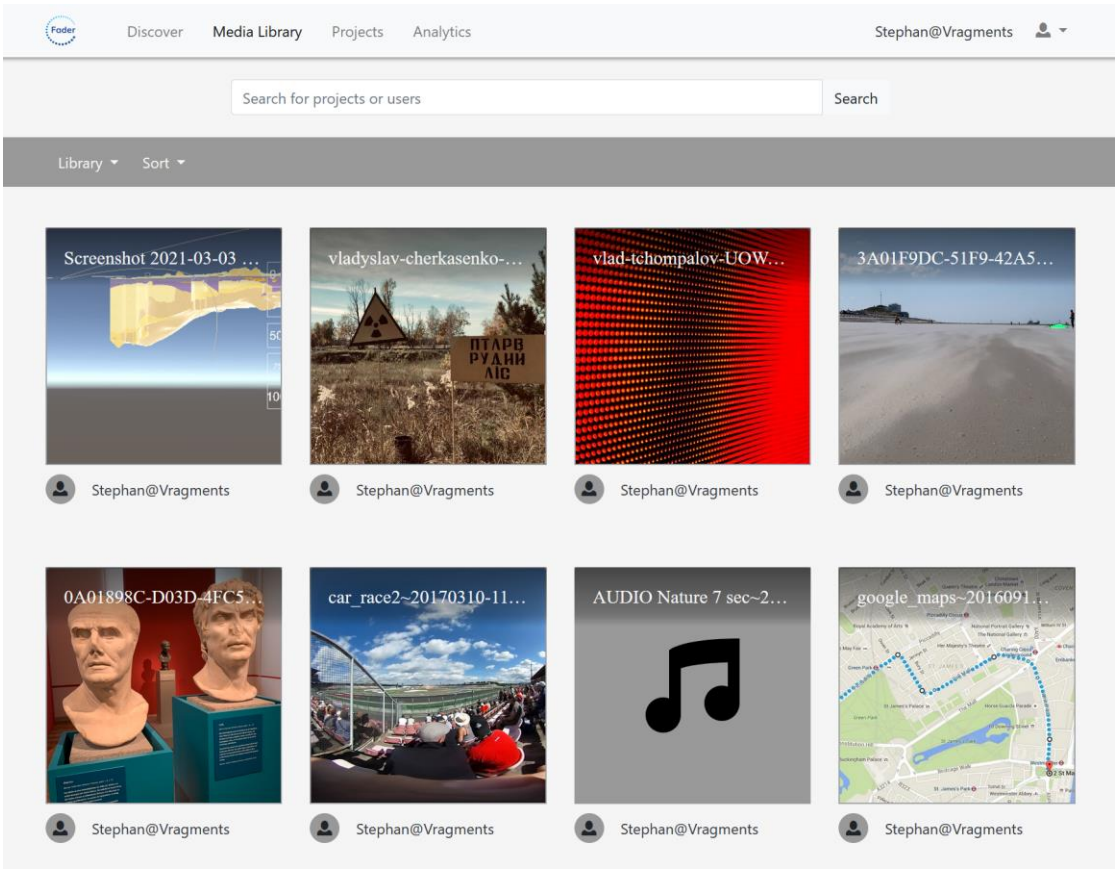

Figure 25: Media library access within Fader

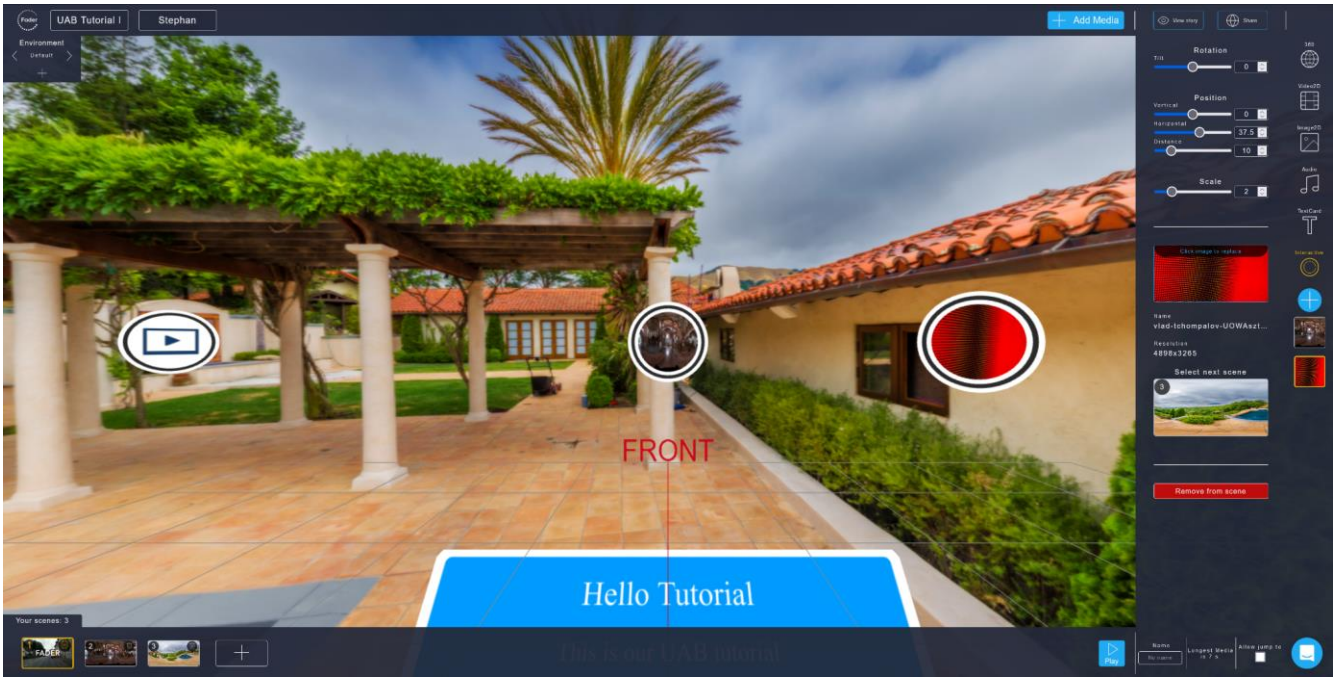

Figure 26: Fader Editor view

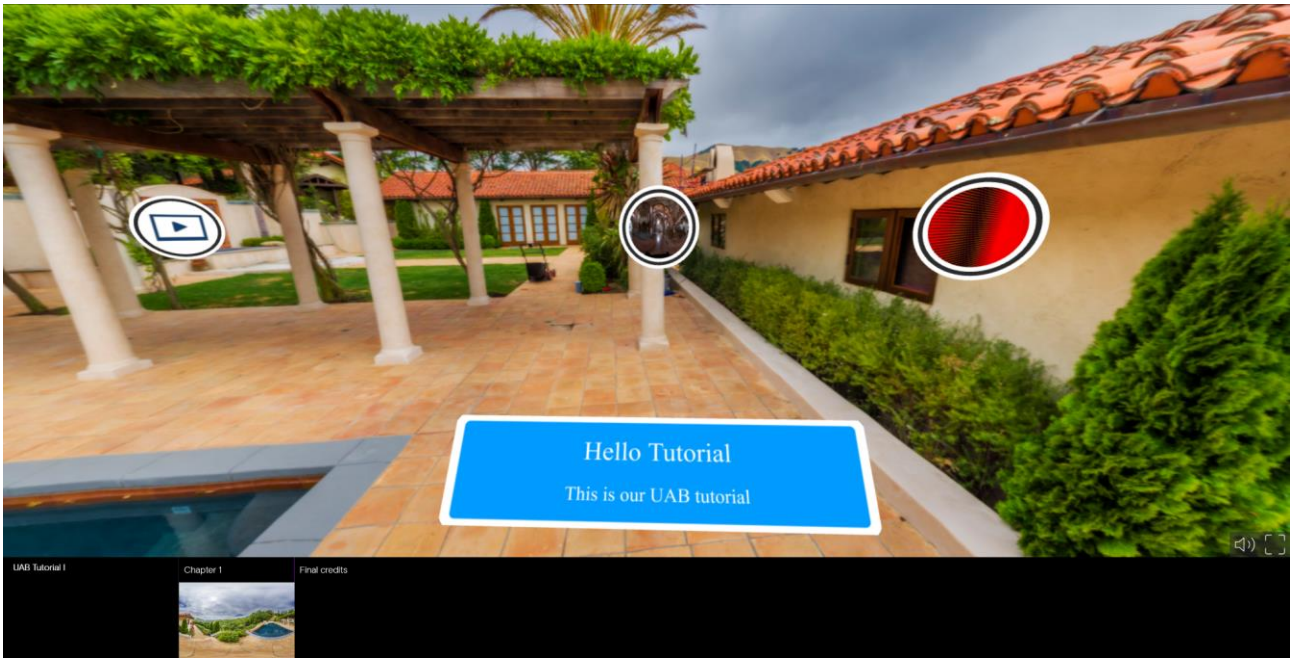

Figure 27: Fader Player view

6.6 VRodos

VRodos' ultimate target is to allow the democratization of VR experiences (based on 3D models, but not on 360 videos) through easily accessible interfaces that are suitable to non-programmers, i.e. to allow citizens as programmers. In the past, we have exploited templates and transpiling methodologies to transform the final experience into Unity projects, and thus compile the final experience as a Unity application[19]. However, as of 2021, the web technologies Aframe, Networked-Aframe, and MediaPipe AI, have become mature and therefore it is a good opportunity to shift the authoring tool to exploit the aforementioned libraries in order to produce the final application as a web technology multi playing VR experience.

Specifically, VRodos is a plugin for WordPress that transforms any website into a site that can deliver several side-products, namely:

a) it can generate the necessary structures in WordPress to be used as a 3D models repository as Figure 28, this side-product can be used by companies that do not want to upload their models in known repositories (sketchfab, turbosquid, CGTrader) for certain reasons (high cost, privacy, internal company use only);

b) it can transform a WordPress website into a 3D models service provider through [WordPress GraphQL API]. This is actually an API that allows the WordPress to serve 3D models content to XR applications, e.g. mobiles AR applications that need dynamic content;

c) a 3D models visualization site - service (Figure 29); This can have two uses. First, the 3D models that were uploaded in the repository can be visualized with a widget 3D viewer anywhere in the WordPress website; and secondly, through an iframe these widgets can be inserted into 3rd party websites. For example, the 3D models that were found in VRodos Assets List page were inserted (some of them) in this blog post in MediaVerse website[20].

[19] Ververidis, D., Migkotzidis, P., Nikolaidis, E., Anastasovitis, E., Chalikias, A. P., Nikolopoulos, S., & Kompatsiaris, I. (2021). An authoring tool for democratizing the creation of high-quality VR experiences. Virtual Reality, 1-20.
[20] https://mediaverse-project.eu/2021/05/20/democratizing-vr-content-creation-for-artists/

d) a 3D scenes editor (Figure 30). This is an interface to set the position and the properties of 3D models integrated into a single scene - experience (under-development).

e) a VR experiences creation mechanism which is under development in order to include more important features such as multiplaying with Networked-Aframe and user selfie segmentation with Mediapipe library .

An instance of VRodos plugin can be found in VRodos website[21].

VRodos in MediaVerse can benefit from the collaboration with MV nodes towards the following pillars:

- fetching content from MV nodes such as 3D models, Images, Videos, 360 videos;
- editorial license - license that allows reusing the assets in a different application - with blockchain;
- Using the speech to text translation service of MV (through STXT). Especially interesting in VRodos as it offers a multiplaying experience where cultures meet live.

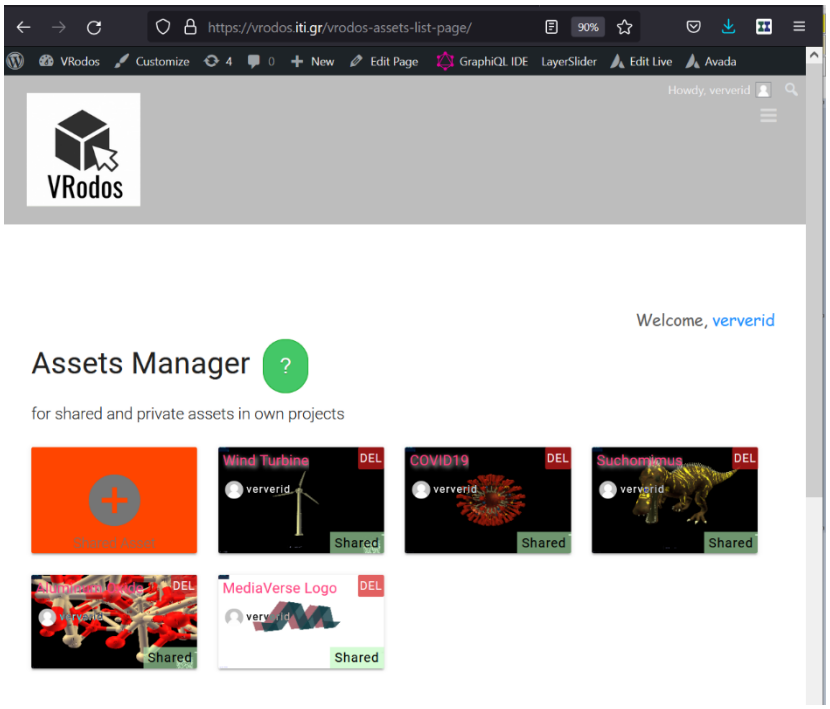

Figure 28: VRodos plugin allows to transform your wordpress into a 3D models repository (Obj, Fbx, Glb formats supported; animation and 3D sound supported for Fbx and Glb)

[21] https://vrodos.iti.gr/

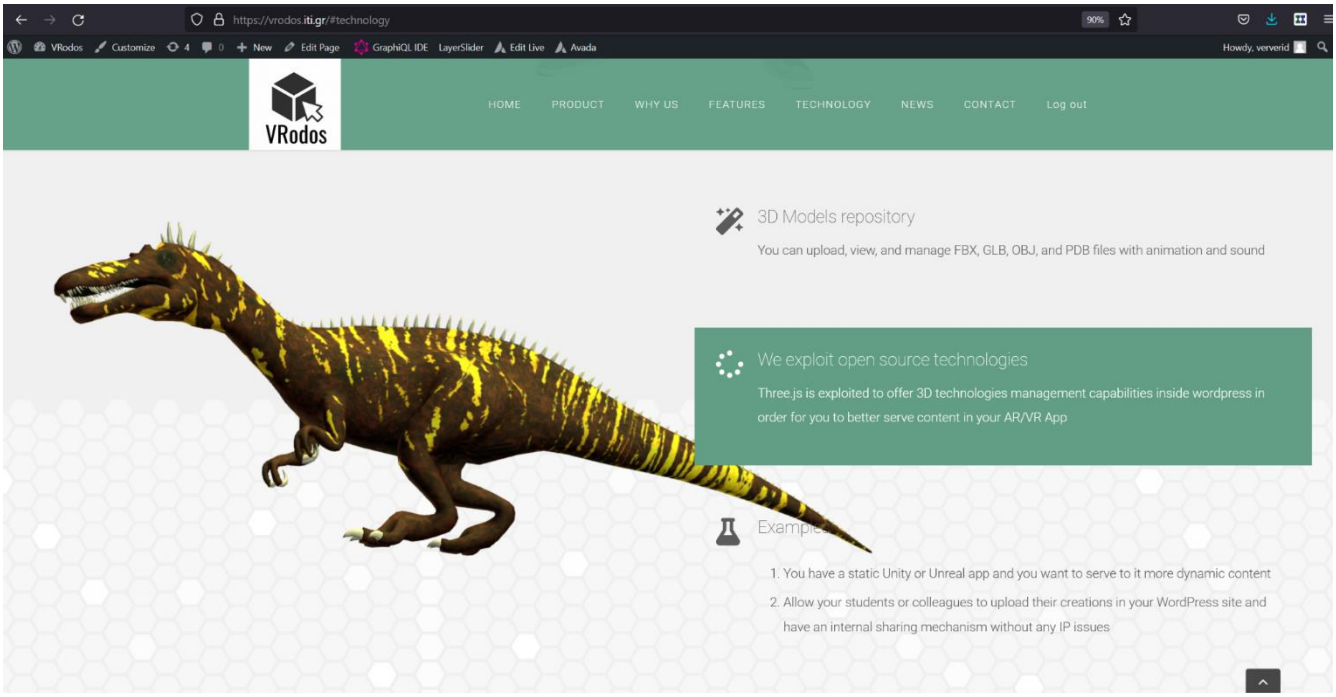

Figure 29: VRodos plugin allows to offer 3D models (animated with 3D sound) visualization service within the same website (https://vrodos.iti.gr/) as an iframe [MediaVerse News item]

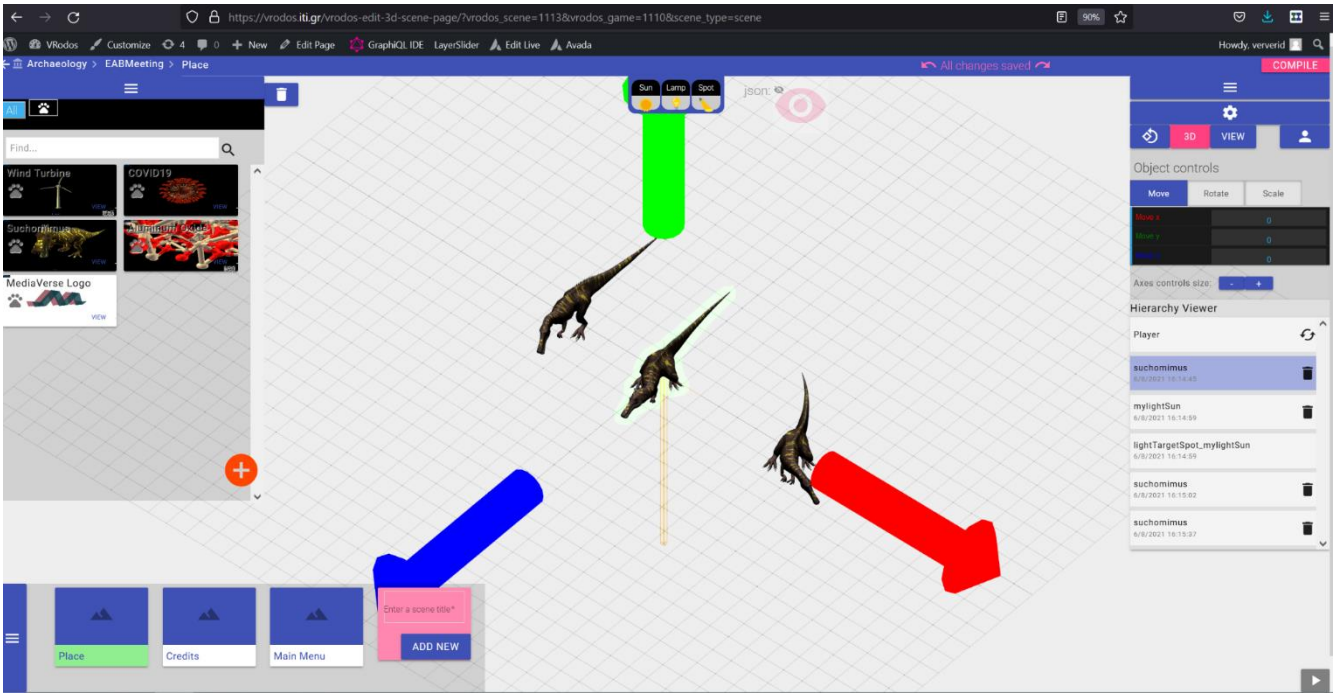

Figure 30: VRodos scene editor allows to make scenes out of the uploaded 3d models as a configurator for a VR experience.

The first progress towards the development of a template for a multi playing VR experience for pure Web technologies can be found in figures above. In the upper figure, we have used Networked-aframe for the provision of VR multi playing capability in web experiences [Networked-aframe]. It supports Firefox and Chrome for all devices (PCs, mobile devices, and VR glasses). The second example (in the middle), shows the video streaming capability of networked-aframe that allows the use of either a 3D model or video stream as a

representation for the avatar or both. The lower image shows the virtual production feature that is actually based on using user separated video streams from the background onto 3D model surfaces. It is based on the MediaPipe AI based Selfie-Segmentation capability to allow edge computing within browsers[22]. We hope that by combining the last two examples, we will make a cheap Virtual Production experience without the use of expensive equipment (Unity or Unreal engines, chroma key backgrounds, Professional LED lamps). Thus we will be able to perform the first ever virtual production concert in WebGL. This will benefit use case 3 for allowing artists to remotely collaborate to demonstrate 5G network or to remedy pandemics. More information can be found in Deliverable D5.2 "Immersive Storytelling Authoring Tools v1" of MediaVerse.

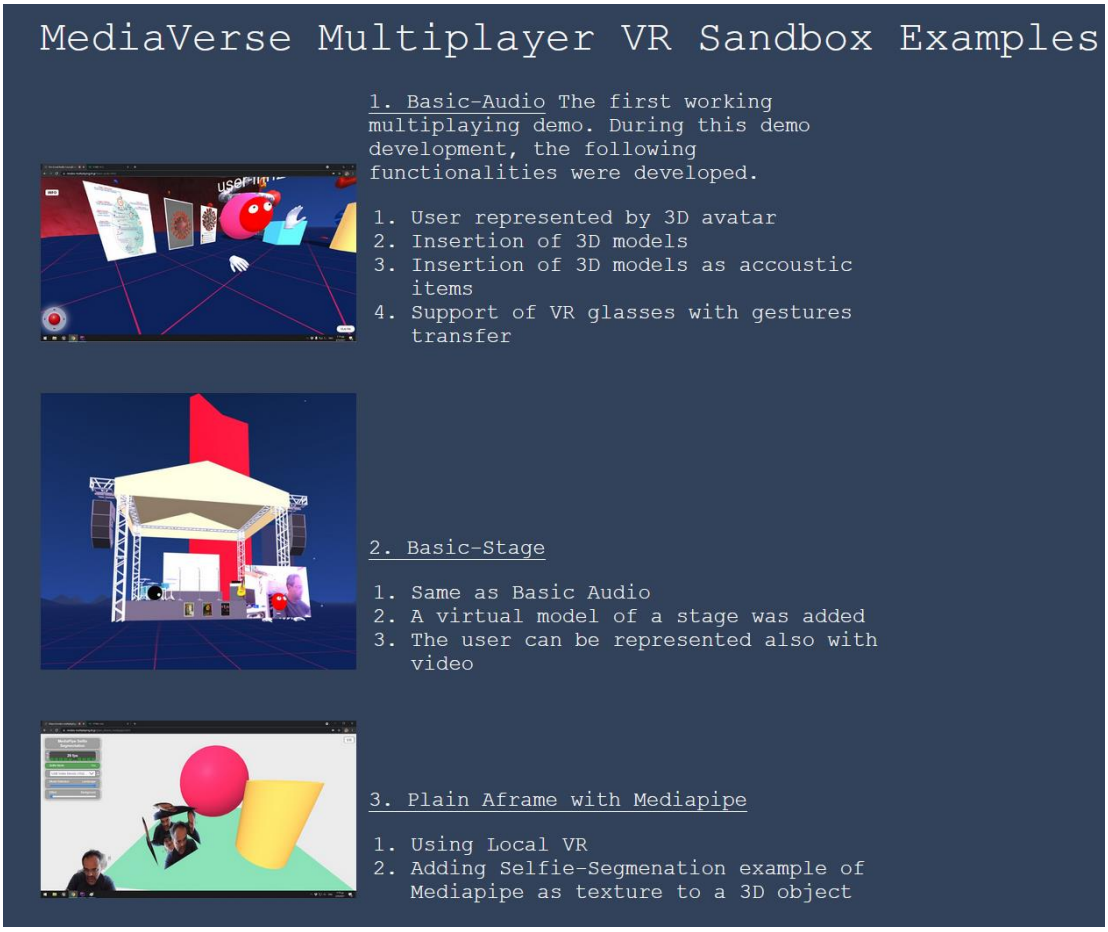

Figure 31: First unit test developments towards a multiuser VR experience with virtual production capabilities.

6.7 RACU

RACU stands for "reconnaissance automatique avec corrections ultérieures" (automatic recognition with retrospective corrections) and is used to describe the processing of accessibility files in an automatic manner.

STXT has developed a central workflow system called Mediahub. Within the media asset management system, there is the possibility in a proprietary RACU process of automatic speech recognition which is based on videos that are uploaded into the Mediahub and are automatically transcribed into text. The text can then be checked and corrected, it can be used to generate subtitles and it can be translated. Some of these steps can be done automatically, others manually.

[22] https://github.com/google/mediapipe

This media asset workflow platform-based system is used to automatically handle media assets among other services. It encodes, stores, transcodes, and delivers content automatically in a predefined sub-workflow, referred to as template. In the bigger picture, the system handles micro services, which are available via APIs.

Although some services are cloud-based, the platform itself is centralized and closed-source.

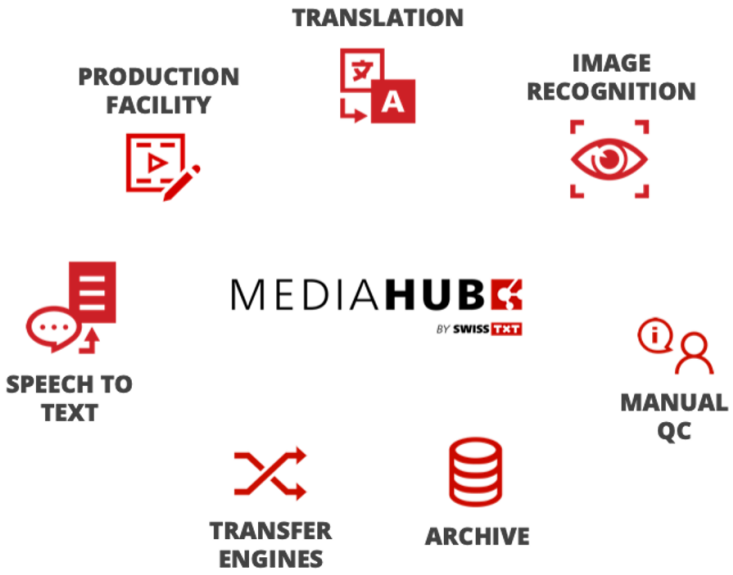

Figure 32: A conceptual illustration of Mediahub

STXT integrates the functionalities of this broadcast-grade workflow system engine that is capable of performing the orchestration of microservices. A specific workflow is designed via a visual interface using Business Process Model and Notation (BPMN). At any step in the workflow, the system can perform a task with any number of sub-tasks connected to it. The transactional information can be consolidated for billing. Also, as a result, the workflow system Mediahub will perform a Smart Contract for any task or sub-task.

This also supports the automatic generation of subtitles, which can be translated into other languages such as German, French, or Italian. In this way, videos with multilingual subtitles can be made available.

This allows running through various tasks that are created in a work-order and offers the user a high degree of flexibility in the creation of a task, such as automatic subtitling.

RACU also allows for manual intervention, especially after the first step of the speech-to-text (STT) automachine generation. This allows the first important linguistic corrections to be made manually in an SRT file and improves the result of the translation into other languages.

The media hub allows editing of subtitles directly in the corresponding video sequence in a certain length. Additional subtitle sequences can easily be added if the text for a sequence becomes too long. The text together with the time and position is created in a so-called SRT file. In MediaVerse we will be able to edit the SRT files directly in the node with a web-based subtitle editor.

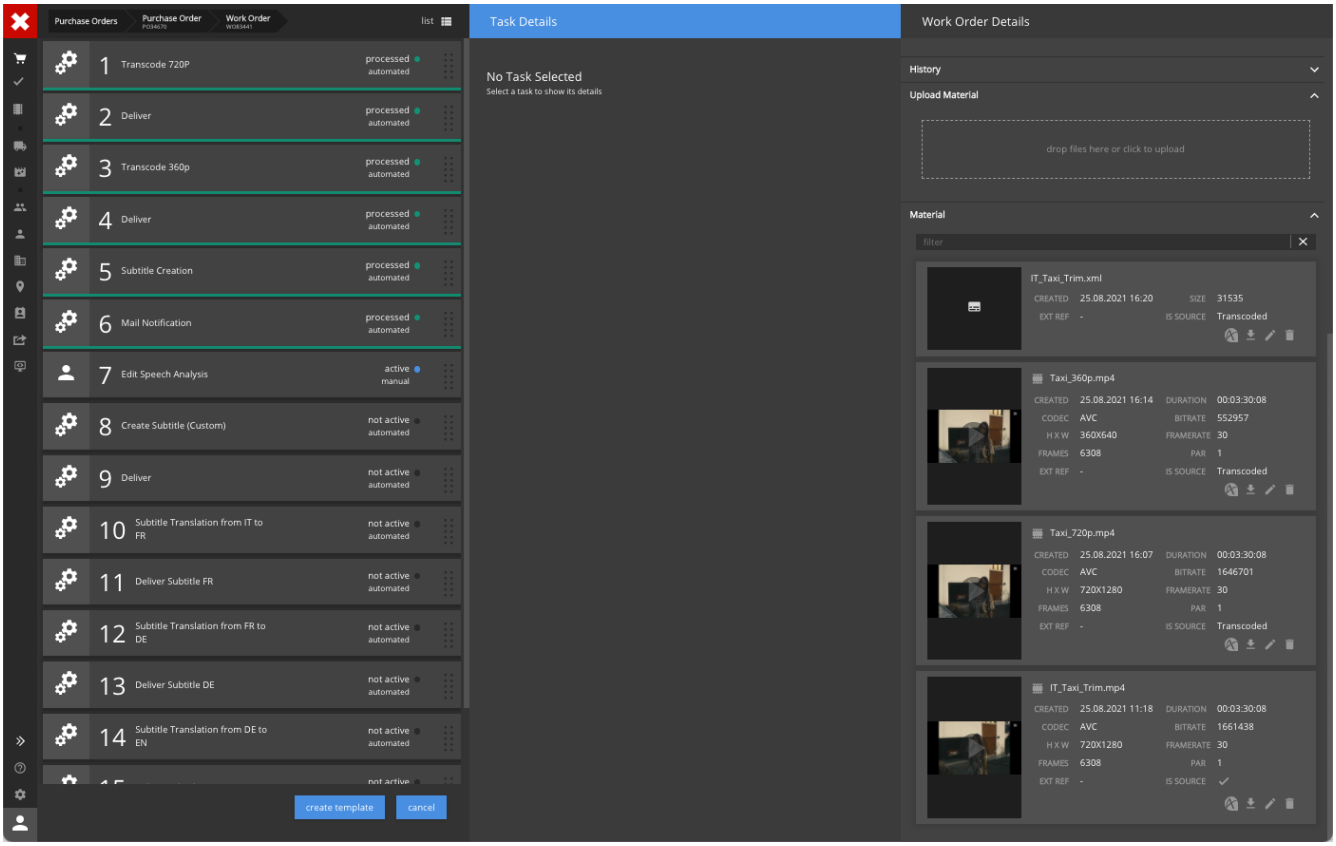

Figure 33: Work order in backend of the Mediahub

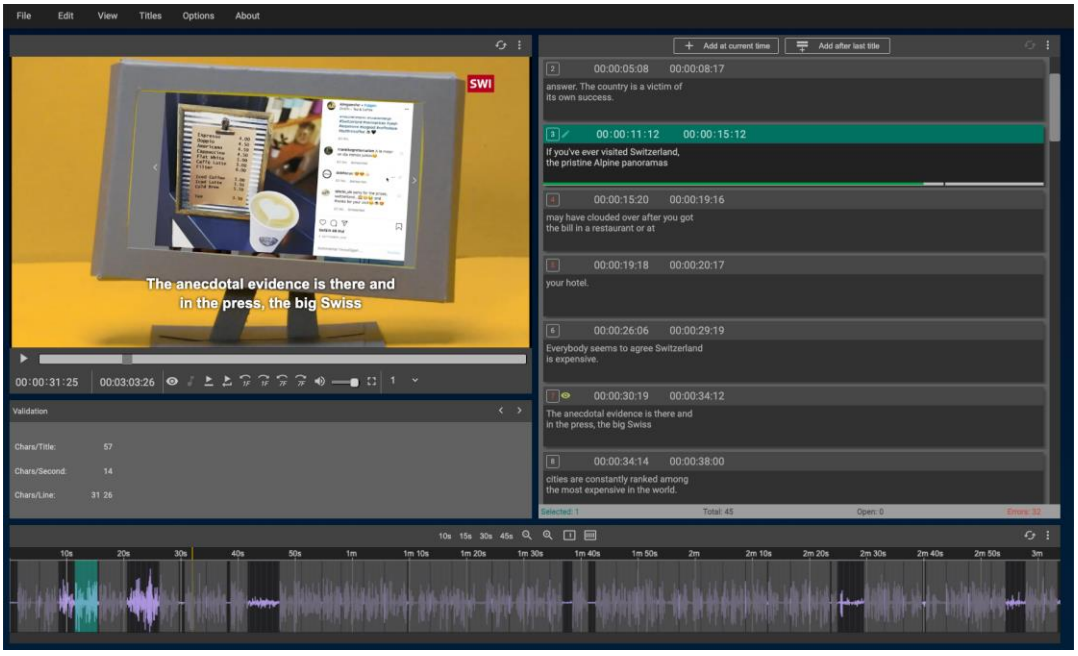

Figure 34: Manual editing of subtitles

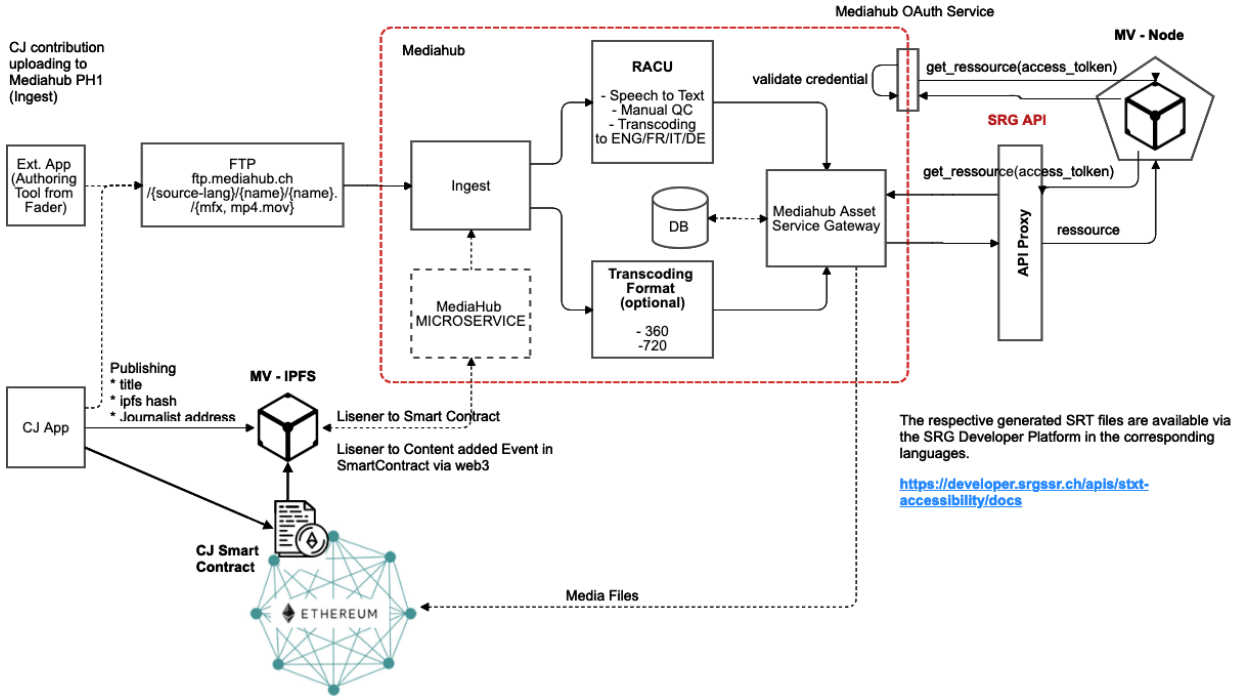

Figure 35: Asset ingest via FTP and IPFS

Integration: Authentication at the Mediahub is done via OAuth 2.0 through the MediaVerse Node. This will allow uploading media via the API exposed by MV nodes to the public API providet from the SRG Developer portal. Content will be uploaded into the Mediahub, which contains the basic functions of a media asset management system. As soon as all the individual RACU tasks for generating the required accessibility data like the multilingual subtitles have been completed, the files are available via the SRG Developer API[23] of the Swiss public service broadcaster, from where they can be transferred to the MediaVerse node and stored in the designated data storage.

In order to be able to carry out first tests, videos already can be uploaded into the Mediahub via FTP data transfer or manually via the Mediahub frontend GUI, from where they are automatically transferred to the RACU process for further processing. The generated SRT files are then available via the SRG Developer Platform in the corresponding languages.

As already considered in the project, media assets could also be stored in the MV Node Storage in the upcoming releases. The CJ Reporter app, which was developed outside of MediaVerse, is highly suitable for the products by citizen journalists and already supports the storage of video and images. The Mediahub could allow the storage of data with the integration of a corresponding micro-service, which listens to the entries in the Smart Contract and, if there are any changes, automatically triggers the RACU process. This would also have the advantage that the data would be unalterable, anonymised and a monetisation in the form of a reward for the citizen journalist would be possible.

[23] https://developer.srgssr.ch/apis/stxt-accessibility/docs

6.8 SLC Template Studio

The SLC Template Studio is a customized version of the Accord Project's Template Studio[24], that provides a Web UI for creating, editing and testing SLC Templates using a markdown editor, without the need of specific tools or previous developing experience. Moreover, it provides a template library from which contract templates can be selected and filled or be used as a starting point for making new templates.

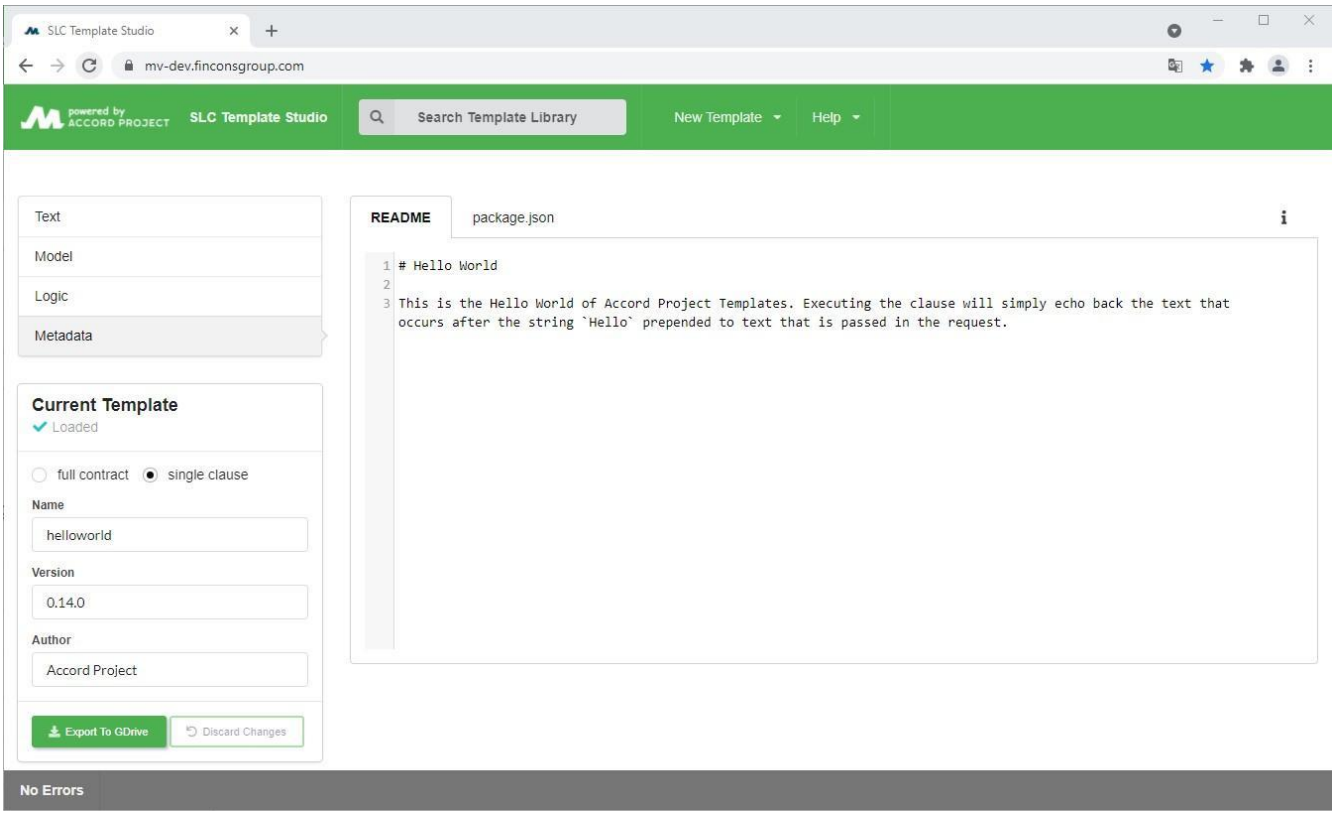

Figure 36: SLC Template Studio Web UI

The Template Studio is an open-source Web tool that allows editing the three parts of a SLC Template: the Text (the natural language), the Data Model and the executable business Logic associated with the template itself. It also provides features such as Importing and Exporting SLC Template files (.cta), checking for syntax errors, and simulating the execution and the usage of a SLC.

The SLC Template Studio is reported as part of Extension Services and Tools because it is required only for creating new SLC templates and, therefore, it is an optional feature that would be useful for some of the MediaVerse nodes. If a MediaVerse node provides it, it will be available as an additional feature of the MediaVerse UI. The SLC Template Studio will therefore be accessible to a MediaVerse user after completing its login to the MediaVerse node and will interact, via the MediaVerse node *Gateway* with the MediaVerse node DAM service.

[24] https://accordproject.org/projects/template-studio/

6.9 Content Adaptation

This module will take care of the automatic adaptation of media content (pictures, audio, flat video, and 360 videos). The content uploaded by the creators will be of top quality that will need to be adapted in order to be played on different kinds of device and over different network connections. For making this possible, MediaVerse will rely on technologies, including adaptive streaming, progressive downloads, and 360 video optimization.

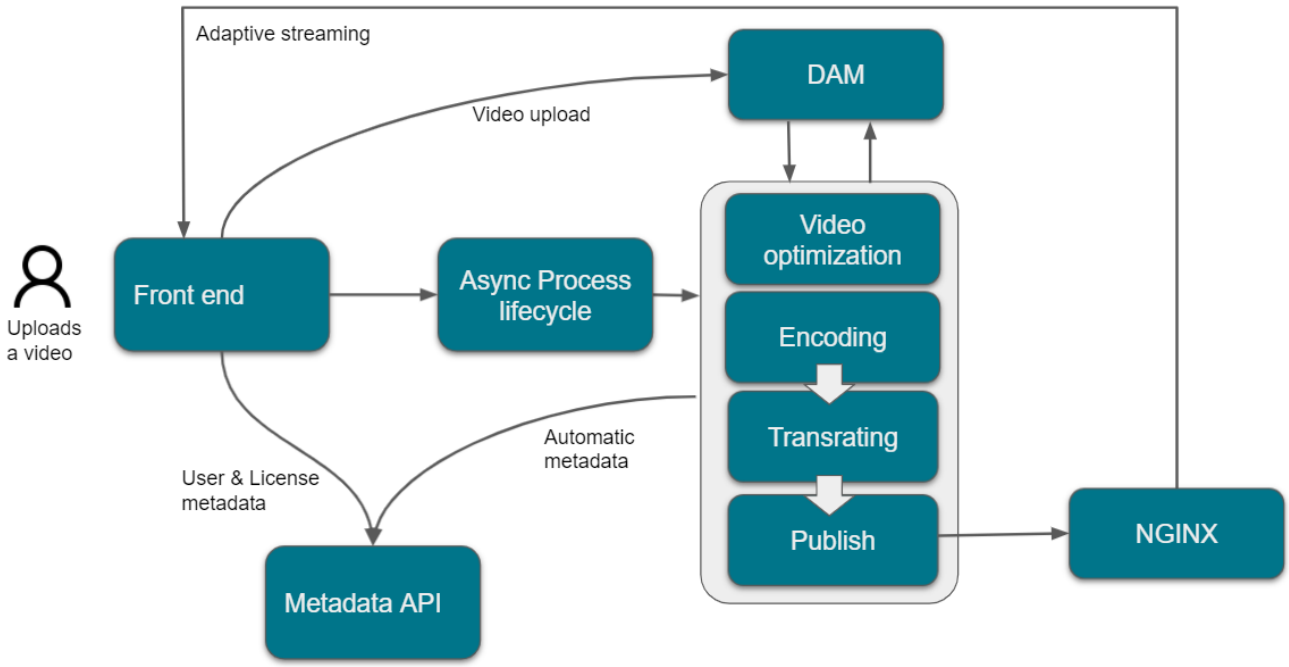

Figure 37: MediaVerse Content adaptation workflow

The content adaptation workflow will start at the moment that a new multimedia asset is uploaded by the user. The component is divided into several services that will take care of tasks like encoding or trans-rating the content when it is needed and that will be integrated with the DAM through an API. This communication with the DAM will be crucial as these different services will need to exchange the content with it initially to retrieve the original files from the author and later return the optimized versions back to the DAM.

For further detail please refer to D3.1 Section 5 “Content adaptation pipeline”.

7 Conclusion

This document has defined the conceptual design of the MV framework. All the MV components have been described and placed in the context of the MV framework and their main functionalities and interactions are presented. This document provides a high-level overview of MV as well as moderate detail in the MV components descriptions, and the full detail is referenced throughout the text of this document via links to specific sections in other MV deliverables or external sources. Hence, this document will be the central reference point for all the technical WPs and in particular WP6, which aims to develop all necessary applications and APIs towards integration, and to assess and validate its efficiency.

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