Platform Architectural Specification

January 16, 2017

Deliverable Code: D6.1

Version: 1.0

Dissemination level: Public

The aim of this document is to present a detailed description of the OpenMinTeD platform. The document focuses on the description and communication among the different services designed by the partners and aims at driving and coordinating system development.

H2020-EINFRA-2014-2015 / H2020-EINFRA-2014-2
Topic: EINFRA-1-2014
Managing, preserving and computing with big research data
Research & Innovation action
Grant Agreement 654021

Document Description

D6.1 – Platform Architectural Specification

WP6 – Platform Design and Implementation	
WP participating organizations: ARC, University of Manchester, UKP-TUDA, INRA, OU, CNIO, USFD, GRNET	
Contractual Delivery Date: 31/07/2016	**Actual Delivery Date:** 16/01/2017
Nature: Report	**Version:** 1.0
Public Deliverable	

Preparation slip

	Name	Organization	Date
From	Antonis Lempesis	ARC	08/01/2017
	Dimitris Galanis	ARC	08/01/2017
	Kostas Koumantaros	GRNET	01/12/2016
Edited by	Stefania Martziou	ARC	16/01/2017
	Antonis Lempesis	ARC	16/01/2017
Reviewed by	Byron Georgantopoulos	GRNET	12/01/2017
	Richard Eckart de Castillo	UKP-TUDA	14/01/2017
Approved by	Natalia Manola	ARC	16/01/2017
	Stelios Piperidis	ARC	16/01/2017
For delivery	Mike Hatzopoulos	ARC	16/01/2017

Document change record

Issue	Item	Reason for Change	Author	Organization
V0.1	Draft version	Initial version sent for comments	Antonis Lempesis	ARC
V1.0	First version	Incorporating reviewers' comments	Antonis Lempesis	ARC

Table of Contents

Table of Figures

Disclaimer

This document contains description of the OpenMinTeD project findings, work and products. Certain parts of it might be under partner Intellectual Property Right (IPR) rules so, prior to using its content please contact the consortium head for approval.

In case you believe that this document harms in any way IPR held by you as a person or as a representative of an entity, please do notify us immediately.

The authors of this document have taken any available measure in order for its content to be accurate, consistent and lawful. However, neither the project consortium as a whole nor the individual partners that implicitly or explicitly participated in the creation and publication of this document hold any sort of responsibility that might occur as a result of using its content.

This publication has been produced with the assistance of the European Union. The content of this publication is the sole responsibility of the OpenMinTeD consortium and can in no way be taken to reflect the views of the European Union.

The European Union is established in accordance with the Treaty on European Union (Maastricht). There are currently 28 Member States of the Union. It is based on the European Communities and the member states cooperation in the fields of Common Foreign and Security Policy and Justice and Home Affairs. The five main institutions of the European Union are the European Parliament, the Council of Ministers, the European Commission, the Court of Justice and the Court of Auditors. (http://europa.eu.int/)

OpenMinTeD is a project funded by the European Union (Grant Agreement No 654021).

Acronyms

OAI-PMH	Open Archives Initiative - Protocol for Metadata Harvesting
TDM	Text and Data Mining
AAI	Authentication & Authorization Infrastructure
REST	Representational State Transfer
JAR	Java Archive
URL	Uniform Resource Locator
UUID	Universally Unique Identifier
ID	Identifier
XML	Extensible Markup Language
JSON	JavaScript Object Notation
JMS	Java Messaging Service
XSD	XML Schema Description
CRUD	Create, Read, Update and Delete
DOI	Digital Object Identifier
API	Application Programming Interface

Publishable Summary

OpenMinTeD aspires to enable the creation of an infrastructure that fosters and facilitates the use of text and data mining technologies in the scientific publications world and beyond, by both application domain users and text-mining experts.

OpenMinTeD builds upon existing tools and text mining platforms, rendering them discoverable, through appropriate registries, and interoperable, through an existing standards-based interoperability layer.

This document presents a detailed description of the OpenMinTeD platform, the services that comprise it and the interactions between these services.

1. Introduction

OpenMinTeD aspires to enable the creation of an infrastructure that fosters and facilitates the use of text and data mining technologies in the scientific publications world and beyond, by both application domain users and text-mining experts.

OpenMinTeD builds upon existing tools and text mining platforms, rendering them discoverable, through appropriate registries, and interoperable, through an interoperability layer building largely on existing standards.

OpenMinTeD supports awareness of the benefits and training of text mining users and developers alike and demonstrates the merits of the approach through a number of use cases identified by scholars and practicioners from different scientific areas, ranging from life sciences (bioinformatics, biochemistry, etc.) to food and agriculture and social sciences and humanities related literature.

The goal of the project is to establish an open and sustainable TDM platform and infrastructure where researchers can collaboratively create, discover, share and re-use knowledge from a wide range of text-based scientific related sources in a seamless way to advance research, promote interdisciplinary open science, and ultimately support evidence based decision making.

1.1 Purpose of this document

The aim of this document is to present a detailed description of the OpenMinTeD platform. The document focuses on the description of the different services, that are designed by the partners, and the communication among them and aims at driving and coordinating system development.

1.2 Document layout

This deliverable is structured as follows: in the second section we present the overview of the platform, with a brief description of the main services the platform provides to its users. In the third section we present the services in the *enabling layer*, i.e. the services that provide support to every other service in the platform. In the fourth section we present the services in the *data layer*, which are responsible for storing, retrieving and managing the data that the platform is using and are also responsible for interacting with all the external to the project services and data sources. In the next, section we present the *service layer*, i.e. the layer containing the main services of OpenMinTeD that are providing the main functionality to the end users. Finally, in the fifth section we are describing in brief the user interfaces of the platform (for more details, see D6.4 - Platform UI specification).

2. Overview

OpenMinTeD will offer a **Registry** service for storing, browsing, downloading, searching and managing various resources such as publications, processing components (e.g. a named entity tagger or sentence splitter) and language resources (e.g. machine learning models, lexica, thesauri). These resources will be imported/registered in OMTD by using a set of specifications/protocols (e.g. OAI-PMH[1], Maven[2]) and they will be documented with high quality metadata. The **Workflow Editor** service of the platform will guide users (via an appropriate UI) in creating interoperable workflows of TDM components, which will be executed by the **Workflow** service in a cloud infrastructure (or on a local machine). OMTD users will be able to annotate the publications (texts) using the **Annotation Editor** service to create datasets which can be used in workflows; e.g. for evaluation purposes. Below, in Figure 1, an overview of the OMTD architecture is presented. In the next sections, the layers and components of the OMTD platform are presented in detail.

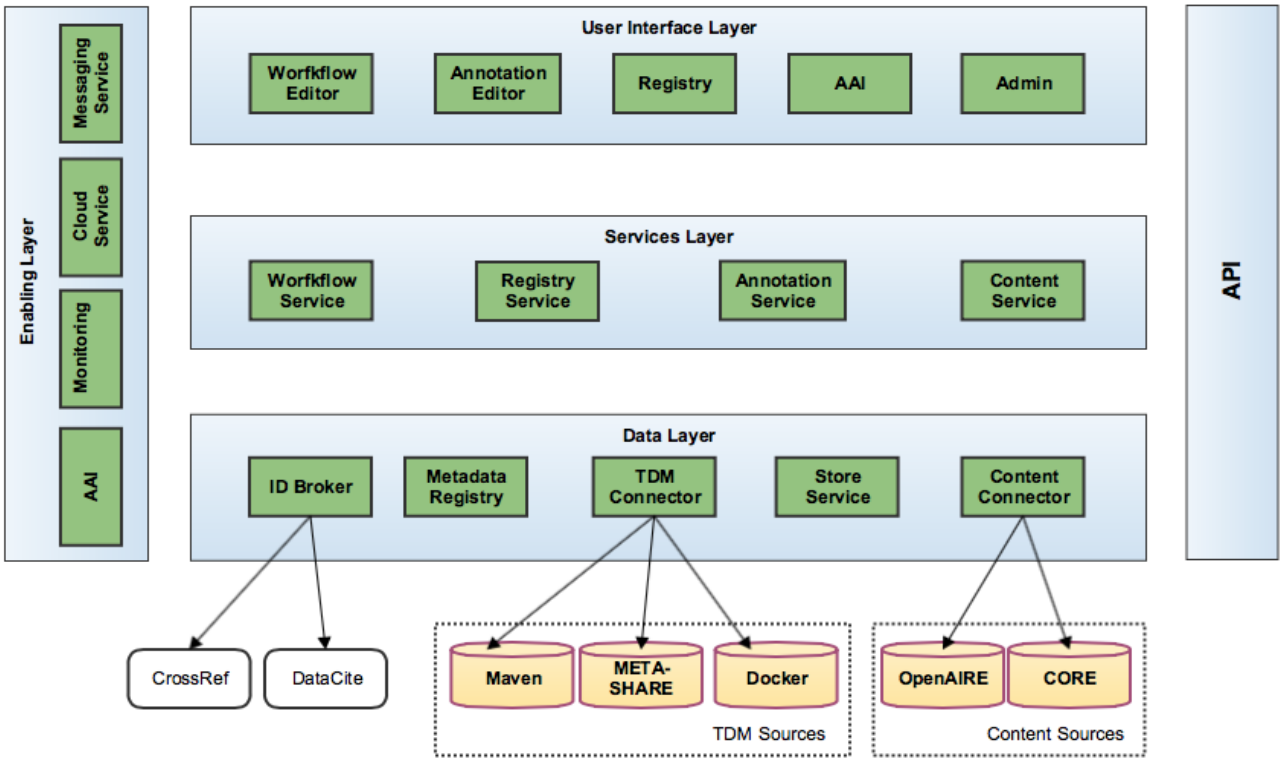

Figure 1 OpenMinTeD Architecture

2.1 Corpus

[1] https://www.openarchives.org/pmh/
[2] https://maven.apache.org/

In the context of OpenMinTeD, a corpus is either 1) a collection of publications along with their metadata or 2) the result of a workflow execution: the publications, the generated annotations (one file per publication), again along with relevant metadata. Each corpus is assigned a persistent identifier by the platform and is immutable; i.e. its contents cannot be modified allowing the reproducibility and validation of an experiment. In OpenMinTeD, a corpus is stored as an archive in the Store Service. Each archive contains:

- A file with the metadata of the corpus itself.
- The metadata of each (raw or annotated) publication.
- The full text of each publication or the annotated file.

Since the corpora are immutable and the OpenMinTeD users or services are not allowed to modify their contents, a corpus resulting from an OpenMinTeD workflow (i.e. an annotated corpus) must be a replica of the input corpus containing one or more extra archives with annotations. This policy does not lead to a waste of storage space, since the Store Service contains optimizations that allow it not to store duplicate files.

The Corpus is the unit of work for the OpenMinTeD workflows. Every workflow that is executed by the Workflow service accepts as input a corpus (that may already contain annotated publications) and the output is again an annotated corpus, i.e. a corpus containing annotations on the inut publications. Even if, for ease of use or for demonstration purposes, a user executes a workflow by uploading a few publications, in the background a new corpus is created and provided as input to the workflow. This allows for a simple and more uniform design of the workflow service.

3. *Enabling Layer*

The enabling layer of the OpenMinTeD platform contains all the services that are performing the management of the infrastructure or are providing functionalities used by every other service in the platform. These services are the following:

- **AAI service:** It provides authentication and authorization functionality
- **Monitoring service**: It is monitoring the use of the hardware resources and can be used to generate usage statistics and provide support for resource allocation decisions). It is also monitoring the state of the platform's services and notifies the system administrators when a service is no longer available.
- **Cloud service:** It is managing the hardware resources and is also responsible for executing the TDM components and tools of each workflow
- **Messaging service:** It provides asynchronous communication between the platform's services.

In the next sections, each service's functionality will be described in detail and its architecture and implementation details will be presented.

3.1 AAI

3.1.1 Functionality

The AAI activity in OpenMinTeD started in Month 12 of the project (May 2016). During the first months of the project, we worked together with the AARC[3] project in order to identify the requirements of the scientific communities. This work resulted in a set of guiding principles:

- Users should be able to access the OpenMinTeD Services using the credentials they have got from their Home Organizations using eduGAIN when possible, but alternative methods should be available.
- OpenMinTeD should expect to receive at least an identifier that uniquely identifies the user and that is stable across different sessions by the same user, coming from within the scope of the authentication source.
- Within the OpenMinTeD environment, a user should have one persistent non-reassignable non-targeted unique identifier.
- OpenMinTeD should define a set of minimum mandatory attributes, without which a user cannot access the OpenMinTeD platform.
- OpenMinTeD should attempt to retrieve these attributes from the user's Home Organization. If this is not possible, then an alternative process should exist in order to acquire and verify the missing user attributes.
- There should be a distinction (Level of Assurance/LoA) between self-asserted attributes and the attributes provided by the Home Organization/VO.

[3] https://aarc-project.eu

- Access to the various services should be granted based on the OpenMinTeD roles assigned to the user.
- OpenMinTeD Services should not have to deal with the complexity of multiple IdPs/Federations/Attribute Authorities/technologies. This complexity should be handled centrally and should be hidden from the OpenMinTeD Services.

Based on these principles and following the guidelines from the AARC project, we designed architecture for the OpenMinTeD AAI and a roadmap in order to incrementally introduce the new service elements on the OpenMinTeD platform.

The core of OpenMinTeD RCIAM[4] Service is the IdP/SP Proxy component, which acts as a bridge between the OpenMinTeD services and external authentication sources and identity providers. This separation between the internal services and the external authentication sources/identity providers allows the service developers to focus on the service features and abstract away the complexity of multiple IdPs, Federations, Attribute Authorities and different authentication and authorization technologies. This complexity is "outsourced" and handled centrally by the proxy. Services need to establish trust with just one entity, the IdP/SP proxy. Typically, services will have one static configuration for the IdP/SP proxy. Having one configured IdP, removes also the requirement from the services to operate their own IdP Discovery Service, which is a common requirement for services supporting federated access. Furthermore, all internal services will get consistent and harmonized user identifiers and attributes, regardless of the home organization or the research community that the user belongs to. Finally, this separation allows the change management of the internal services to be independent of the change management cycles at the home organizations IdPs. IdPs establish trust with one entity, the operator of the IdP/SP proxy and they are not impacted by the change operations of each individual service.

[4] https://github.com/rciam

3.1.2 Architecture

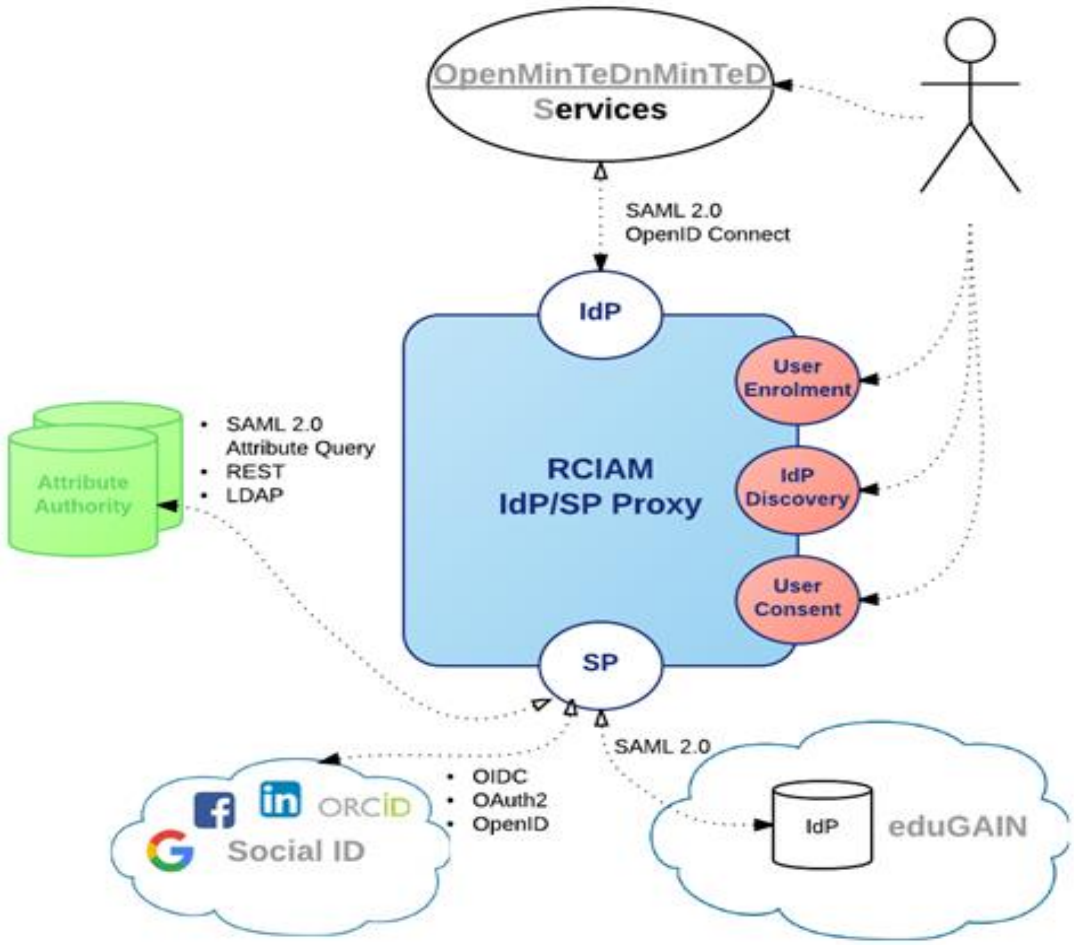

Figure 2 AAI Service

3.1.3 Interactions

The following list of external services consists of identity providers, each one implementing a different authentication protocol:

- eduGAIN
- ORCID
- Social ID
- Facebook
- Google
- LinkedIn

3.2 Cloud Service

3.2.1 Functionality

The Cloud service of OpenMinTeD is responsible for managing the hardware resources of the platform. It is responsible for allocating new virtual machines with configurable amounts of CPU power, RAM and disk space to the new workflows to be executed and decide which workflows will be executed at any given point and which will be queued for later execution.

Since the consortium decided that Docker[5] will be used extensively to distribute and execute the code of TDM tools and workflow components, the cloud service provides a Docker oriented API, in which the unit of work is a Docker image instead of an arbitrary executable file.

The service is currently at a very early stage of design, waiting for the full specifications and development of the Workflow service and the integration of the Galaxy[6] workflow engine (for more information, see the description of the Workflow Service in section 5.3), so a much more detailed and complete description will be provided in the next version of this deliverable.

3.3 Messaging Service

A messaging service is a service that allows other services to communicate indirectly by exchanging messages. In the messaging paradigm, a message is a piece of information (e.g. an event, a request for an action, a notification that an action was completed, etc.) that is generated by one service (the *producer*) and is eventually received by another service (the *consumer*). A message can either have only one recipient, in which case a *queue* is used where the producer is adding messages and the consumer is removing them, or multiple recipients, in which case the producer is sending a message under a topic and all interested consumers subscribe to this topic and receive any new messages. The main feature of a messaging service is that it allows decoupled communication between the consumers and producers; the producer is not aware and not interested in which services will eventually receive the messages. This allows for the creation of simpler services, with fewer software dependencies and thus a much simpler overall architecture.

Since all the core services of the OpenMinTeD platform are developed in Java, the natural decision was to use the Java Messaging Service specification and more precisely the Apache ActiveMQ[7] implementation.

[5] https://www.docker.com/
[6] https://galaxyproject.org/
[7] http://activemq.apache.org/

4. Data Layer

The Data Layer of the OpenMinTeD platform contains the services that manage the content (individual publications or corpora), TDM tools and components and TDM related resources (e.g. a lexicon, a machine learning model, etc.). These services are responsible for managing the metadata of all these entities (Metadata registry), storing the actual data of the entities that are stored locally in the platform (Store service), acquiring the content from external sources (Content service) and acquiring information about TDM tools and resources again from external sources.

Each one of these services has a limited scope and its functionality is not strictly limited to OpenMinTeD. As such, with minor modifications or extensions, these services could be used from other projects that require similar functionality.

In the next sections, each service's functionality will be described in detail and its architecture and implementation details will be presented.

4.1 Store Service

4.1.1 Archive

An *archive* in the OpenMinTeD platform is an organized collection of files and potentially other archives and is completely analogous to a directory of a typical filesystem. Each archive is described by a set of metadata, including an identifier, a name, access rights, etc. and is managed by the Store service.

4.1.2 Functionality

The Store service of OpenMinTeD is responsible for storing, retrieving and managing archives. An archive may contain a TDM resource (a linguistic resource, a lexicon, etc.), the contents of a corpus (e.g. publications in PDF format) and any other piece of data that must be stored in the OpenMinTeD platform. The service allows its users to either upload a single file (in which case an archive that contains this file is created automatically by the service) or incrementally create an archive and append files or other sub-archives to it.

The Store service supports two different deployment scenarios. The first one is used in the OpenMinTeD platform and involves using Pithos+, the cloud based storage service provided by GRNET[8]. The second deployment scenario is used when the OpenMinTeD services are deployed in local sites, without access to GRNET services. However, the service is designed in such a way that more deployment options could be easily implemented in the future (e.g. use Amazon S3 storage or any other cloud based solution).

[8] https://okeanos.grnet.gr/services/pithos/

4.1.3 Implementation

The Store service is implemented in Java 8 and consists of two Maven projects; the *omtd-store-api* and *omtd-store-rest*.

- ***omtd-store-api:*** It contains the *StoreService* interface that has two implementations, *StoreServiceLocal* and *StoreServicePITHOS;* each of them is based on a different filesystem connector. In particular, the former implementation offers access to the local filesystem (e.g. hard drive of a VM) based on the functionalities that are provided by the respective built-in JAVA APIs. The latter provides access to *Pithos+* cloud storage using a REST-based client implementation available at a GitHub[9] repository maintained by GRNET.
- ***omtd-store-rest:*** It provides a REST API for the *StoreService.* The implementation is based on Spring[10] framework and the respective Controller(s). The REST service is deployed to the Apache Tomcat Servlet container and it is offered as a standalone executable JAR file. The JAR is built using Spring-Boot[11] framework/library, which facilitates the creation of standalone applications; e.g. by offering the embedding of a Tomcat container. *omtd-store-rest* provides access to either *StoreServiceLocal* or *StoreServicePITHOS*; this is specified in a configuration file (storeServiceApp.properties) that is loaded on startup. The same configuration file also stores the parameters for connecting to Pithos+ service; e.g. URL endpoint, token, UUID. The Store Service was designed to run on a separate server (from Registry), which makes the OMTD platform more flexible and robust.

The Store Service provides operations such as:

- Upload an archive
- Delete an archive
- Store a file in an archive
- Create an archive
- Create an archive in another archive

Each archive is assigned a (unique) persistent identifier, which is automatically produced using the ID Broker.

4.1.4 Interactions

- Metadata Registry service, to store the metadata of OMTD Resources.
- ID Broker, to assign identifiers to newly created archives.
- Pithos+, the cloud based storage service provided by GRNET

4.2 Metadata Registry

[9] https://github.com/grnet/e-science/tree/master/pithosfs
[10] https://projects.spring.io/spring-framework/
[11] https://projects.spring.io/spring-boot/

4.2.1 Functionality

The OpenMinTeD platform manages a large and diverse set of resources: TDM tools and services, TDM linguistic resources, lexica, publication corpora, workflows, etc. These resources must be easily discoverable and in order to use them correctly a set of metadata must be used to describe them. The Metadata Registry is a general-purpose service whose responsibility is to store and manage the metadata of the various OpenMinTeD resources.

The diversity of the resources dictates that the metadata registry must be resource type agnostic: administrators of the metadata registry must be able to add/modify and delete resource types while the service is running without the need to modify the source code of the registry or the underlying database schema. Another reason for this versatility is that the OpenMinTeD platform is under continuous development, with new features, services and resource types being added regularly and the metadata registry should be able to easily cope with these changes.

Metadata are traditionally described using XML but in the recent years JSON is gaining popularity among developers. The metadata registry is able to store metadata in either XML or JSON format and provide the same functionality. Another feature of the service is that it supports notions from the relational databases: primary keys, unique constraints, and even referential integrity between the values of different resource types. Finally, the metadata registry is able to notify interested users about changes in the contents by creating topics in JMS and posting events for every insertion/modification or deletion of a resource.

4.2.2 Architecture

4.2.2.1 Resource Type

In order to start storing and managing resources of a certain type (e.g. TDM components, lexica or workflows), a structure called ResourceType must be created and submitted to the metadata registry. The ResourceType structure contains the following information:

- **Name.** The name of the resource type. Which must be unique among all the resource types.
- **Payload type.** The format of the resource (XML or JSON). While the metadata registry supports both formats, the resources of each resource must be expressed in one format, XML or JSON.
- **Schema** (or schema URL)**.** The schema of the resources (XSD for XML, JSON Schema[12] for JSON) or the URL of the schema, if it can be found online. The schema is used when a resource is added or modified for validation and to ensure that all stored resources are valid.
- **Indexed fields** (or IndexMapper)**.** While the underlying storage or the source code make no assumptions on the schema and contents of the stored resources, it is necessary for the service to extract some information from each resource type to allow the execution of queries. One such example is the email and name of a "Person" resource, which is described by an XSD

[12] http://json-schema.org/

schema. While a "Person" resource may contain a lot of information about the user, the users of the metadata registry need to search persons by their name or email. This is accomplished by specifying an IndexedField, which contains information about the searchable fields (the name of the field, its type (a string, a number, etc.) and the location of the value in the resource (XPath for XML resources or JSON Path[13] for JSON resources). When new resources are added or existing are modified the registry is using this list of IndexedFields to extract the values and store them in an index to allow for efficient queries.

4.2.2.2 Resource

The contents of the metadata registry are defined in a structure called Resource. The users of the service are storing and retrieving Resources. Each Resource contains the following information:

- **Id.** A unique identifier, which is assigned by the registry to each new resource.
- **Resource type.** The type of the resource. Before attempting to store a resource in the registry, its resource type must have already been registered.
- **Payload.** The actual contents of the resource. It is either an XML or a JSON file, depending on the payload type of the respective resource type. When storing a resource, instead of the actual payload, a URL of the payload can be supplied.
- **Creation** and **modification date.** The dates when this resource was inserted in the registry and when it was last modified.
- **Version.** Whenever a resource is modified its version number is incremented and the previous versions is retained. Only the latest version of a resource is available through the search APIs and the users can access the previous versions only by using the version API.

[13] https://github.com/jayway/JsonPath

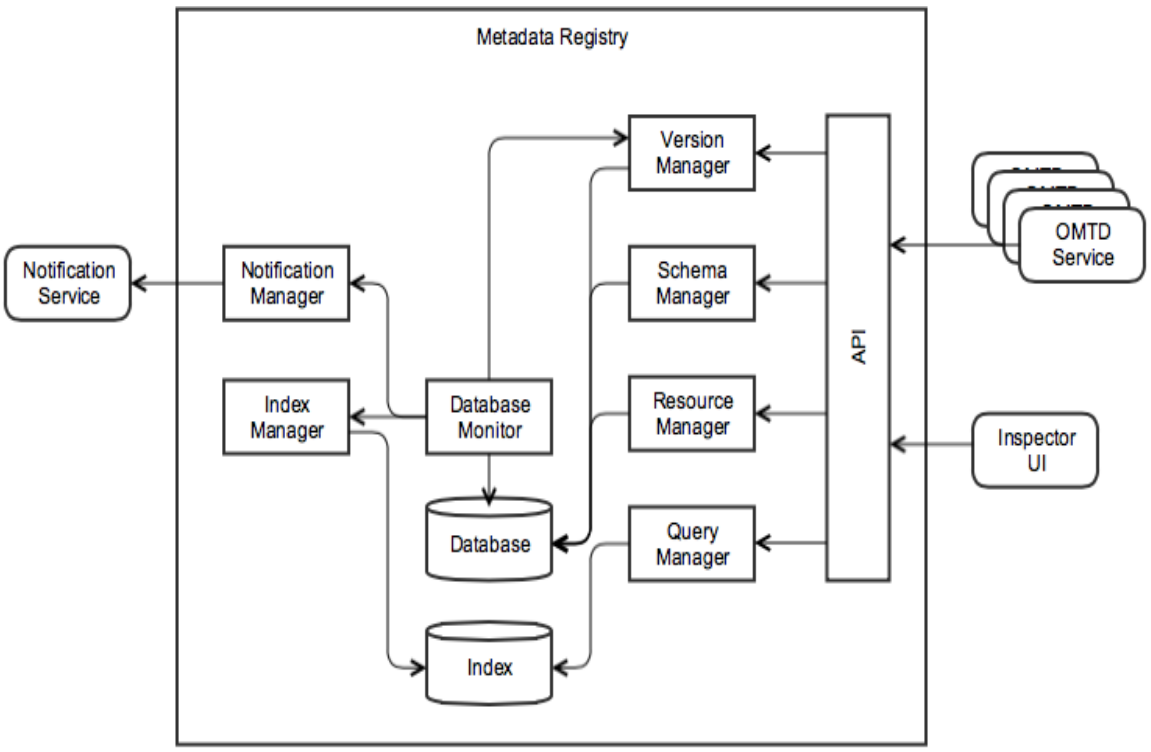

Figure 3 The Metadata Registry architecture

The Metadata Registry Service is composed from a number of different modules, each one responsible for a different task:

- **Schema Manager** is responsible for registering resource types, validating the schema provided, and preparing the database to accept new resources of the new types.
- **Resource Manager** manages resources: validating new resources against the schema and the constraints (referential, primary keys and unique values), and registering in the database. The resource manager is also used to update and delete existing resources.
- **Query Manager** performs queries to the index (based on the index fields defined in the resource type) and returns the results. While the resource manager is using the database to register the resources, the queries are performed at the index, which allows for far more efficient queries.
- **Database Monitor** monitors the database for any changes in registered resource types or resources and notifies other modules that are interested in these changes. For example, the Version Manager updates the version of resources when it receives a notification by the Database Monitor. These notifications from the database monitor to other modules allow for faster completion of the basic features of the registry (i.e. CRUD operations on resource types and resources) since the more computationally heavy operations (indexing, versioning, etc.) happen asynchronously.
- **Version Manager** manages the different versions of the resources. Whenever a resource is updated, the version manager stores the previous version and assigns a new version number to

the modified resource. Using the Version Manager, users can trace the evolution of a resource from the moment it was created up to its latest version.

- **Index Manager** is responsible for updating the full text index of resources. The full text index allows the execution of queries on the resources providing results much faster than the relational database that is used to store the resource types and resources. By responding to notifications from the Database Monitor, the Index Manager keeps the underlying index synchronized with the contents of the database.
- **Notification Manager** provides the connection between the Metadata Registry and the Messaging service of the OpenMinTeD platform. Whenever a new resource type is registered, the Notification manager creates a new JMS topic (e.g. "registry.person.create" or "registry.person.update" for insertions and updates of resources of type "person" respectively) and when resources are created, updated or deleted the Notification manager is sending new messages in the respective topic. The notification manager allows any other service in the platform to be asynchronously notified of the changes in the metadata registry without explicitly interacting with the metadata registry. Again, as is the case with the previous two modules, the Notification manager is notified for the changes in the resources by the database monitor.

The Metadata Registry is developed using Java 8, the underlying relational database is PostgreSQL and the full text index used is Elasticsearch**.**

4.2.3 Interactions

Since the Metadata Registry is one of the most basic and low-level services of the OpenMinTeD platform, it does not interact with almost any other service to implement its functionality. Instead, almost all services that handle metadata are using the Metadata Registry to manage them.

The only service that the Metadata Registry directly interacts with is the Notification Service, to post notifications about changes in the contents.

4.3 ID Broker/Resolver

4.3.1 Functionality

The ID Broker is a service with two responsibilities. The first is to provide new, persistent identifiers for new entities created in the OpenMinTeD platform. Those entities vary in kind and include

- **Corpora,** either created using the Content service, uploaded by users or the result of a workflow execution.
- **Workflows** created using the Workflow editor.
- **TDM related resources** like tools, components, lexica, machine learning models, etc.
- Any other **auxiliary resources** that are registered in OpenMinTeD.

It is worth noting that in cases where there exists both a metadata element and an actual resource (e.g. a corpus), a distinct identifier is assigned to each of the entities.

At this point in the project, it is not yet clear whether the ID Broker will create new, internal identifiers, use an external service like CrossRef or Datacite to acquire new identifiers or even adopt a hybrid approach where for some cases an external service will be used (e.g. Datacite for new corpora) while new OpenMinTeD identifiers will be generated for other kinds of resources (e.g. auxiliary resources like organizations, etc.).

The second functionality of the service will be to resolve existing identifiers against external services (for example, use CrossRef or DataCite to resolve a DOI) and return the metadata of the referenced entity.

4.3.2 Architecture

Since the implementation of the service is at early stages, its architecture has not been decided yet. It will be described in a next revision of this document (D6.2).

4.3.3 Interactions

The following list of external services that the ID Broker Service might interact with is indicative, since no concrete decisions have been made yet:

- Crossref[14], for publication or corpora identifiers (DOI).
- DataCite[15], for persistent identifiers for generated or uploaded datasets.
- ePIC[16] handle system.

4.4 TDM Connector

4.4.1 Functionality

The TDM connector is the service whose responsibility is to locate and harvest metadata of TDM related resources from various external sources and transform them to the OpenMinTeD schema. A second functionality of the TDM Connector is to fetch the actual TDM resources (e.g. a lexicon or a machine learning model) when they must be stored in the OpenMinTeD platform. Although there is a wealth of information about TDM resources from various repositories and registries, for the first version of the platform we are going to focus on three external sources:

- Maven repositories, for metadata of GATE[17] and UIMA[18] TDM tools or models stored in JAR files.
- Docker registries, for OpenMinTeD compatible standalone TDM tools and web services.
- META-SHARE[19], for any kind of TDM resources like tools, services, lexica, corpora, models, etc.

[14] http://www.crossref.org/
[15] https://www.datacite.org/
[16] http://www.pidconsortium.eu/
[17] https://gate.ac.uk/
[18] https://uima.apache.org/
[19] http://www.meta-net.eu/meta-share

In the next versions of the platform, we plan to include more sources of TDM resources, like the AgroPortal[20] or CIARDRING[21].

4.4.2 Architecture

The implementation of the service is at an early stage, so details about the architecture and implementation will be described in the next revision of the document (D6.2).

4.4.3 Interactions

- Maven repositories, for metadata of GATE and UIMA TDM tools or models stored in jar files.
- Docker registries, for OpenMinTeD compatible standalone TDM tools and web services.
- META-SHARE, for any kind of TDM resources like tools, services, lexica, corpora, models, etc.

4.5 Content Connector

4.5.1 Functionality

The Content connector is the service responsible for bridging the services in the OpenMinTeD with the external content providers. OpenAIRE[22], CORE[23], and Crossref have been identified as the most suitable providers for the first phase but more may follow later, depending on the needs of the project. There are multiple implementations and instances of the content connector, one for each external content provider. The content connector offers the following functionality:

- Performs mapping both from the OpenMinTeD metadata schema to the external provider's schema and the reverse, allowing the connector to return metadata in a common format.
- Provides search functionality by using the proprietary search API of the data provider and returning the results in a common format, usable by the Content Service.
- Provides access to the full text of the publications, allowing the Content service to build new corpora.

4.5.2 Architecture

Since there are multiple implementations of the Content connector, there is no single architecture that can describe all implementations. The only common thing between all content connectors is the API they implement and contains the following functionalities:

- **Search,** which provides faceted search functionality, used by the user interface when a user is locating the publications she needs to build a new corpus.
- **Fetching metadata**, which returns all the metadata of publications that belong to the new corpus
- **Fetching fulltext,** which returns the full text of a publication, again for inclusion in a new corpus

[20] http://agroportal.lirmm.fr/
[21] http://ring.ciard.net/
[22] https://www.openaire.eu/
[23] https://core.ac.uk/

4.5.3 Interactions

- OpenAIRE, CORE and CrossRef, to access their search APIs and the full text of their hosted publications.

5. Services Layer

The Services Layer of the OpenMinTeD platform contains the services that are providing the OpenMinTeD functionality to the end users. These services are the following:

- **Registry Service,** which stores and manages TDM-related resources.
- **Workflow Service,** which is managing and monitoring the execution of workflows.
- **Annotation Service**, which is responsible for storing, managing and exposing the annotations.
- **Content Service**, which is responsible for fetching publications and building corpora from external sources.

In the next subsections, we will describe the functionality and the architecture of each service in details and the interactions with the other services of the platform.

5.1 Registry Service

5.1.1 Functionality

The Registry Service stores and manages TDM-related resources: TDM tools (e.g. tokenizers, sentence splitters, POS Taggers), lexica, ontologies, annotation schemas, machine learning models etc. It allows users to register new resources, search for them and use them to build and execute workflows/applications (using the workflow editor/service) or manage the output of workflows (annotation editor). The service is also used to manage auxiliary resources, not strictly TDM related: information about persons, organizations, projects, etc.

While the Metadata Registry hands the bulk of the metadata management operations (storing, retrieving and searching of metadata), it does so in a resource type agnostic way and provides an API so generic that cannot be easily used by the OpenMinTeD Registry UI or the other services. The Registry Service is building on the functionality of the Metadata Registry and provides an API that is tailored to the needs of the UI or the services of the platform.

Another functionality of the Registry service is to harvest (on demand or at specified intervals) metadata about TDM related resources from external sources, like Maven repositories or Docker registries (for TDM components) and META-SHARE (for any kind of TDM related resources).

5.1.2 Architecture

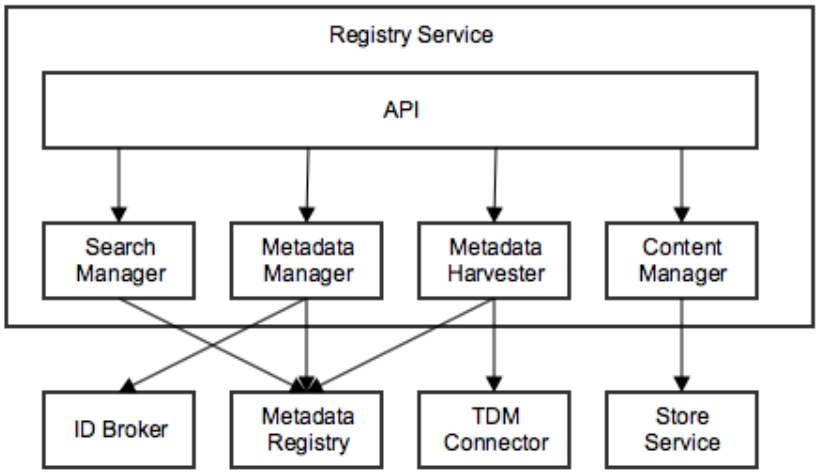

Figure 4 The Registry Service architecture

The Registry service comprises of a small number of modules, each one responsible for interacting with an OpenMinTeD service:

- **Search Manager,** which is responsible for performing queries on the metadata of the stored resources.
- **Metadata Manager,** which is responsible for retrieving, updating and storing the metadata of new and existing resources.
- **Content Manager,** which is responsible for storing and retrieving the actual TDM resources (e.g. a lexicon or a machine learning model) from the Store service.
- **Metadata Harvester,** which is responsible for harvesting metadata of TDM-related resources from external sources and storing them to the metadata registry. The harvest manager can be configured to either perform harvesting periodically or on demand.

5.1.3 Interactions

- Metadata Registry, to store the metadata of the resources.
- Object store, to store the actual contents of the resources.
- TDM Connector, to acquire metadata about TDM resources.
- ID Broker, to acquire persistent identifiers for newly created resources.

5.2 Content Service

5.2.1 Functionality

The Content Service is responsible for providing access to content from external sources. It allows OpenMinTeD users to perform queries for content metadata (both basic keyword search and more advanced filtering) using a common query language. The service redirects the query to multiple sources of content (OpenAIRE, CORE, and CrossRef are examples of sources of publications), aggregates, homogenizes and transforms the results to the OpenMinTeD format and returns them to

the user. The service is configurable and extensible and thus able to use any external source that provides a search API.

Another responsibility of the Content Service is to provide access to the actual data (e.g. full text in case of publications). It allows users to either directly upload and describe with appropriate metadata a corpus or, by using the Content Connectors, create a new corpus containing publications from external sources (e.g. OpenAIRE or CORE) and edit the automatically generated metadata for it. The service is using the Object Store to store the contents of the corpora and the Metadata Registry to store and manage the metadata of each corpus.

When creating a new corpus, the input of the service is a user-supplied query (e.g. "the English publications published in PLoS ONE in 2016"). The service then uses the Content connectors to execute this query in all supported content sources and fetch both the full text files and the metadata of the publications that match the criteria. Using the Store Service, a new archive is created that contains the contents of the new corpus and a metadata record for the corpus is generated with information extracted from the metadata of the publications.

5.2.2 Architecture

The Content Service comprises of a number of modules, each one responsible for a specific functionality:

- **Search Manager,** which is responsible for performing searching for publication metadata in the external sources (OpenAIRE and CORE). For this functionality, it uses the Content connectors to perform the queries and retrieve the results.
- **Corpus Manager,** which is performing the main functionality of the service by coordinating the rest of the modules: it is downloading the contents of the new corpora from the external sources, creates new persistent identifiers for the newly created corpora, creates metadata for new corpora and stores them in the registry, and also stores the content of the corpora to the Store Service. The corpus manager is using an internal database to store the status of each corpus-building request.
- **Metadata Manager,** which is responsible for creating and storing metadata for new corpora to the Registry and fetching metadata of existing corpora from the Registry.
- **Content Manager,** which is responsible for storing and retrieving the content of corpora from the Store Service.
- **Content Connector,** which (as described in the Content Service section) is responsible for performing queries to the external content sources (OpenAIRE or CORE) and retrieving the metadata and publications of the results.

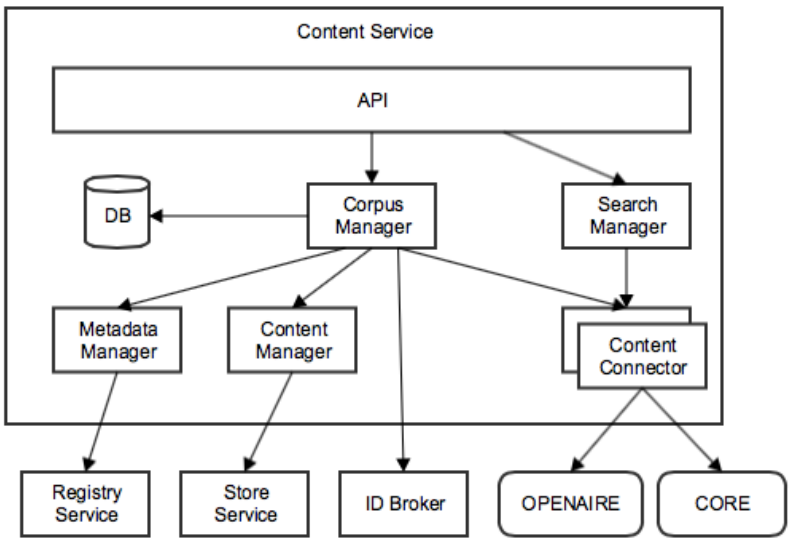

Figure 5 The Content Service architecture

5.2.3 Interactions

- ID Broker, to acquire persistent identifiers for new corpora.
- Registry Service, to store the metadata of the corpora.
- Store Service, to store the contents of the corpora.
- Messaging Service, to provide notifications about the status of the submitted workloads.

5.3 Workflow Service

5.3.1 Functionality

The OpenMinTeD consortium decided not to implement a new workflow execution service but instead opted to use the Galaxy Workflow Engine. Galaxy is used by the LAPPS[24] Grid project and a number of other TDM projects and has proven to be a reliable solution. However, Galaxy (as any other off-the-shelf software) cannot be used in the OpenMinTeD platform without an integration layer. The Workflow Service plays the role of the integration layer between the OpenMinTeD services and Galaxy. More specifically, the workflow service is responsible for:

- Managing and monitoring the execution of workflows, allowing the users of the registry UI to start, stop and cancel jobs without necessarily using the Galaxy UI.
- Moving the content to be processed from the Store service to the Workflow engine before the execution of a workflow and storing the results back to the Store service.
- Notifying users about the status of the execution by sending messages to the Message Service.

[24] https://www.lappsgrid.org/124-2/

5.3.2 Architecture

The implementation of the Service is at an early stage, so details about the architecture and implementation will be described in the next revision of the document.

5.3.3 Interactions

- Registry Service, to retrieve information about TDM tools and resources and to store the information about the workflow execution.
- Content Service, to retrieve information about corpora and the corpora themselves and to store the annotated corpora once a workflow execution has finished.
- Annotation Service, to store the output of the workflows
- Messaging Service, to notify interested services about the progress of workflows.

5.4 Annotation Service

5.4.1 Functionality

The Annotation service is responsible for managing annotations. Its main responsibilities include the following:

- Store and retrieve annotations from the Store service.
- Expose annotations in different formats.

5.4.2 Architecture

The implementation of the Service is at an early stage, so details about the architecture and implementation will be described in the next revision of the document.

5.4.3 Interactions

- Metadata Registry to acquire information about executed workflows and corpora.
- Store Service to store and retrieve annotation.

6. *User Interface Layer*

This layer includes the user interfaces (UI) for interacting with the OMTD platform. Below we briefly we present them; a detailed description is given in "D6.4 - Platform UI Specification".

6.1 Registry

The Registry UI is the main portal of the platform and the starting point for every user that wants to visit the OpenMinTeD platform. It offers the following functionality:

- Search and browse for registered TDM tools, components, corpora and TDM resources
- Register new TDM related resources
- Different views of data depending on the user type (simple user or TDM expert)
- Discovery of publications from the external sources (using the content service)
- Creation or uploading of new corpora
- Discovery and registration of TDM related resources and tools from external source
- User registration and account management
- Personal space for users containing "their own" data and resources that have not been made public (e.g. a closed access corpus, TDM resource, etc.)
- Ability to start the execution of workflows and monitor their progress.
- Links to the other portals (Workflow and Annotation editors) to edit or create new workflows or annotations.

6.2 Workflow Editor

The Workflow Editor UI of the OpenMinTeD platform is the editor provided by Galaxy; (for more details on this choice, see the Workflow Service section). The Galaxy UI offers the following functionality:

- Find the components that the user is interested in (e.g. a named entity recognizer).
- Create workflows of interoperable tools by connecting them in a pipeline (or acyclic graph).
- Set the parameters of a processing component (e.g. a lexicon or a machine learning model).
- Create a new workflow by modifying an existing workflow one.
- Start and monitor the execution of a workflow.

Since the integration of Galaxy in the project was decided only a short while before the writing of this deliverable, a more extensive description of the Workflow Editor will happen in the next revision of this deliverable.

6.3 Annotation Editor

The OpenMinTeD consortium decided to use WebAnno[25] as the annotation editor of the platform instead of creating a new editor. WebAnno is *"a general purpose web-based annotation tool for a wide range of linguistic annotations including various layers of morphological, syntactical, and semantic*

[25] https://webanno.github.io/webanno/

annotations". It has been developed by UKP-TUDA and provides a large number of features, the most important of which are:

- Support for different user roles (annotators, project managers, administrators, etc.).
- Creation and management of annotation projects.
- Support for new annotations, curation and correction of existing annotations.
- Support for a large number of annotation formats.

A more detailed description of the features of WebAnno will be provided in the next revision of this deliverable.

7. References

[1] D4.3 – OpenMinTeD Functional Specifications
[2] D6.4 - Platform UI Specification
[3] D6.7 - OpenMinTeD Platform services distribution specification