X Modal
X Cultural
X Lingual
X Domain
X Site
Global OER Network

Grant Agreement Number: 761758
Project Acronym: X5GON
Project title: X5gon: Cross Modal, Cross Cultural, Cross Lingual, Cross Domain, and Cross Site Global OER Network
Project Date: 2017-09-01 to 2020-08-31
Project Duration: 36 months
Document Title: D2.2 - Final server side platform
Author(s): Erik Novak
Contributing partners: JSI, Nantes, UCL
Date:
Approved by:
Type: P
Status: Final
Contact: Erik Novak (erik.novak@ijs.si)

Dissemination Level		
PU	Public	x
PP	Restricted to other programme participants (including the Commission Services)	
RE	Restricted to a group specified by the consortium (including the Commission Services)	
CO	Confidential, only for members of the consortium (including the Commission Services)	

Revision

Date	Lead Author(s)	Comments
01/08/2019	Erik Novak	Initial Draft
08/08/2019	Walid Ben Romdhane	Added contributions from Univerité de Nantes
10/08/2019	Sahan Bulathwela	Added contributions from University College London
20/08/2019	John Shawe-Taylor	Draft Review
30/08/2019	Jasna Urbančič	Final Version

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABBREVIATIONS

Acronyms	Definitions
OER	Open Educational Resource
XML	eXtensible Markup Language
API	Application Programming Interface
JSON	JavaScript Object Notation
URL	Uniform Resource Location
MIME	Multipurpose Internet Mail Extensions

ABSTRACT

In this report we present the final server-side platform architecture. We describe the different components which are able to acquire, process and enrich the material and user data. In addition, we present different services and the platform API used to access the processed material metadata and the services.

1. INTRODUCTION

The purpose of this document is to report on the final server-side platform. During the process of platform development, we have evaluated and considered the different component requirements identified in deliverable 2.1 – Requirements & Architecture Report.

The platform consists out of four major components – the database, ingesting and processing pipeline, services and platform API – each employed to perform a separate task. The platform is able to process three types of OER materials: text, video and audio. In addition, it enriches them through a process called Wikification. The processed materials are stored into the database. The database also contains data about user activities on the OER repositories that integrated the X5GON Connect Service, a library developed for acquiring behaviour data. The database is accessed by the different services developed within the project – the recommender engine, the quality assurance tool and the learning analytics tool – which can be accessed through the platform API.

The document is structured as follows. Section 2 contains a high-level description of the platform architecture, its main components and how it was developed. Section 3 describes the components of the ingesting and processing pipeline – the components tasked to acquire and enrich the material metadata. Next, section 4 explains the reasoning behind choosing PostgresQL as the main database system used in the platform. In addition, it describes the database schema of the most relevant data tables. A description of the available services is available in section 5. Afterwards, section 6 describes the platform API and provides examples of the API endpoints. Section 7 is dedicated to the X5GON Connect Service – describing its functions and the data it acquires. Afterwards, we present future work in section 9 and conclude the document in section 10.

2. PLATFORM ARCHITECTURE

In this section we present the X5GON platform architecture. The initial version has been presented in deliverable D2.1 Requirements & Architecture Report, from which we have extended its design.

The platform is developed to connect four different yet important components:

- **The database.** The service which contains all of the processed OER material metadata, and user activity data.
- **The ingesting and processing pipeline.** The component which extracts the appropriate OER material metadata, enriches it, process it and sends it to the database.
- **The services.** The components which consumes the data stored in the database and provides recommendations, quality and insight into the OER materials, as well as into how users consume the materials.
- **The API server.** The component connecting all previously mentioned components. In addition, it provides an access point for the users to access the developed services and the processed material metadata.

Figure 1 show the high-level overview of the platform architecture. It shows how the server sends the user requests and data to the appropriate endpoints – either to the services or to the ingesting and processing pipeline.

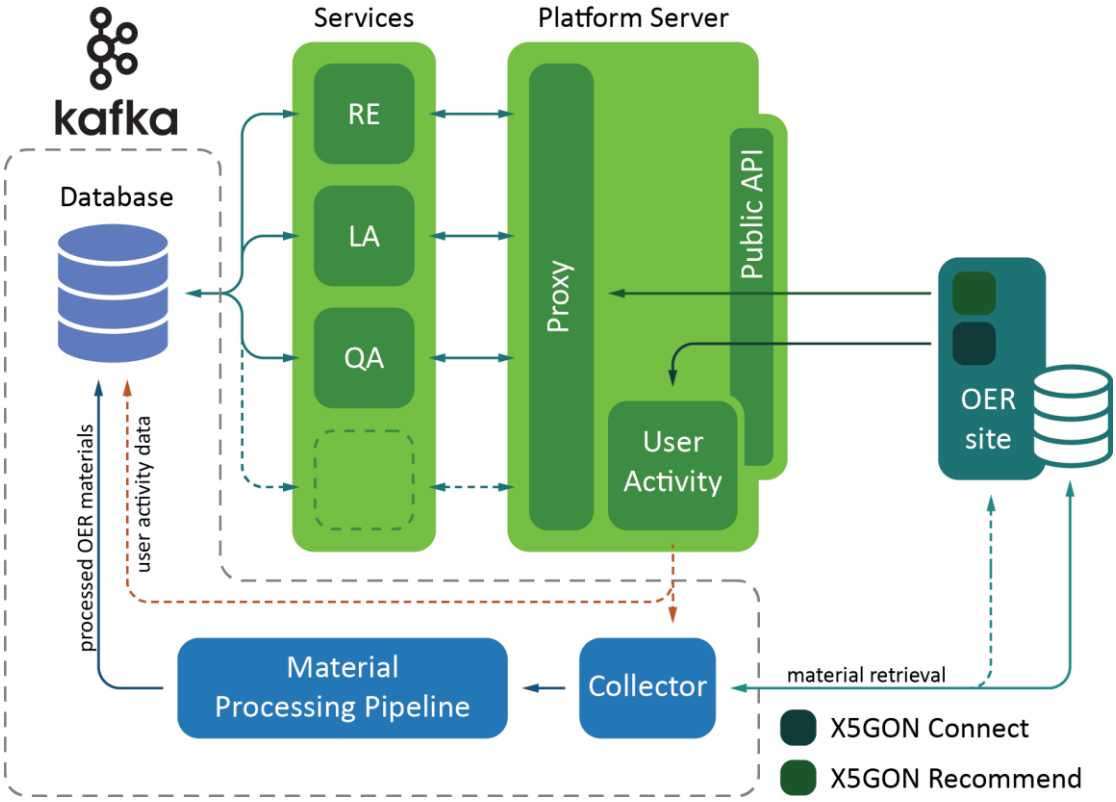

Figure 1: *The X5GON platform architecture. It connects the database, ingesting and processing pipeline, its services and the API server.*

The majority of the platform was developed with the javascript (nodejs), while some services are developed with the Python programming language. Parts of the architecture are connected with the use of the Apache Kafka service [1]. The database system in use is PostgresQL. The code of the main platform components is available on GitHub [2] while the platform itself is served on its own website [3]. The platform runs on a machine with 150GB of space, 32GB of RAM and 8 CPUs. The operating system installed on the machine is Linux Debian 8.6 (jessie).

In the following sections 3, 4, 5, and 6, we describe each corresponding platform component in detail. In addition, section 6 describes the API access point through which users can access the services and the processed OER material metadata.

3. THE INGESTING AND PROCESSING PIPELINE

In this section we describe the ingesting and processing pipeline. This component is developed to retrieve the OER material metadata from the providers' site – either through their API or through retrieving the metadata from their website – and send it through the pipeline which processes the material metadata, enriches it and stores it in the database. Figure 2 shows the components that this section describes. Most of the pipelines components are connected with Apache Kafka – a data messaging system – which enables distribution of data across multiple machines (see section 3.4). The code for running the ingesting and processing pipeline is available on GitHub [4]. What follows is the description of the components in the pipeline.

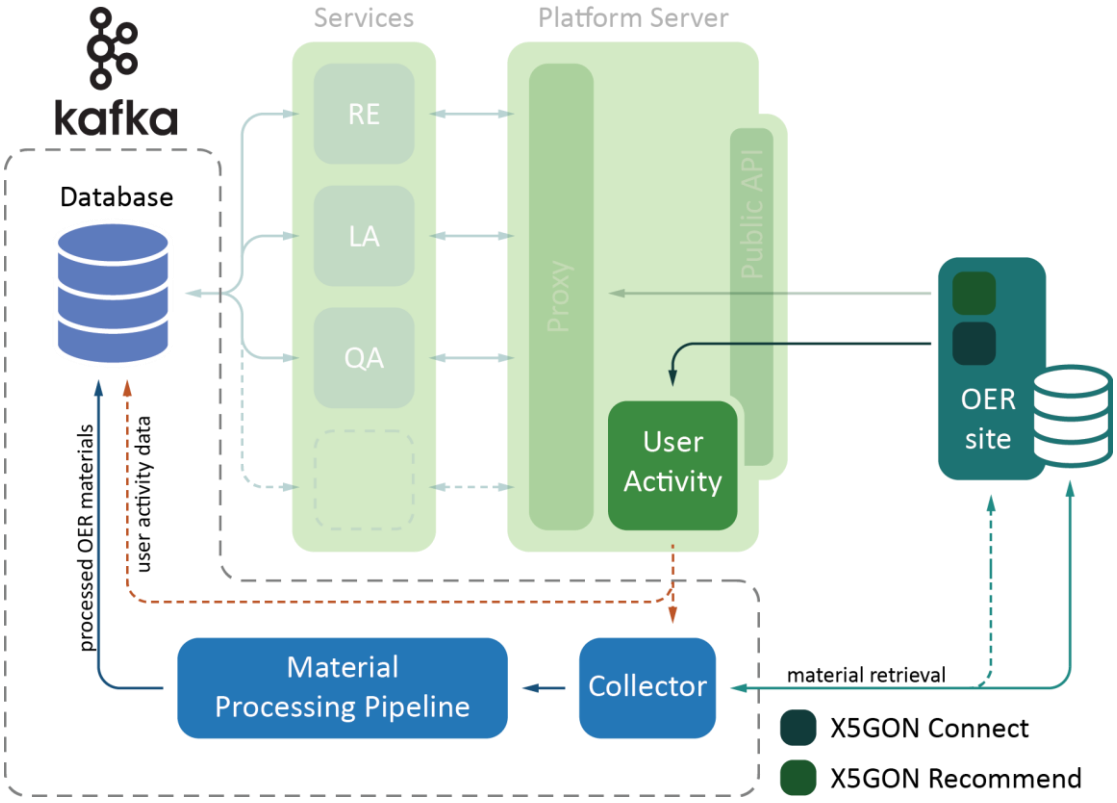

Figure 2: *The X5GON platform architecture with the focus on the ingesting and processing pipeline. The pipeline retrieves the appropriate material, process, enriches and stores it in the database.*

3.1. MATERIAL COLLECTOR

The first component in the ingestion and processing pipeline is the collector – whose task is to retrieve the material metadata from the OER provides and send it to the processing pipeline. It contains a list of different scripts – each corresponding its corresponding OER provider – which are triggered when a user viewed a material. This is done in the following steps:

1. The collector retrieves the user activity data which is sent by the **X5GON connect service** (for more detail see section 7).
2. The collector checks if the OER material was already acquired by the platform. If the material was acquired, it continues to retrieve and process the next user activity data. Otherwise, it continues to the next step.
3. Using the user activity data, the collector identifies which retrieval script it needs and retrieves the OER material metadata. Afterwards, based on the material

type (video, audio or text) it sends to the appropriate **material processing pipeline** (see section 3.2).

This approach allows us to automatically process OER materials on request without the periodic checking for new materials. This however does mean the platform will not contain the newest materials until at least one user views it. To solve this, we have also provided a REST API endpoint to which an OER provider can send their material metadata to be processed. More on the upload REST API is described in section 6.1.

Currently, the platform contains scripts for acquiring material metadata from the following OER providers:

- Videolectures.NET – an award-winning free and open access educational video lectures repository. Link: http://videolectures.net/
- eUčbeniki – a Slovenian repository containing textbooks for primary and secondary school. Link: https://eucbeniki.sio.si/
- engageNY – a repository provided by the New York State Education Department. Link: https://www.engageny.org/

The remaining materials were acquired either by importing data dumps provided by the corresponding OER providers or by uploading the material metadata through the upload REST API.

3.2. Material Processing Pipeline

The next component in the pipeline is a collective of steps where each process or enriches the material metadata. The steps include formatting the material metadata into a common format, extracting the content of the material, annotate it with Wikipedia concepts through a process called Wikification, validate if all of the required attributes were filled and store it in the database. In addition, if any of the steps detect an error, it stores the partially processed material into its own location in the database for future evaluation on what went wrong.

Figure 3 shows an overview of the steps presented in this section. The material processing pipeline was developed using qtopology [5], a distributed stream processing layer. Each step of the pipeline was developed in a single script, which qtopology joins them together to form a pipeline.

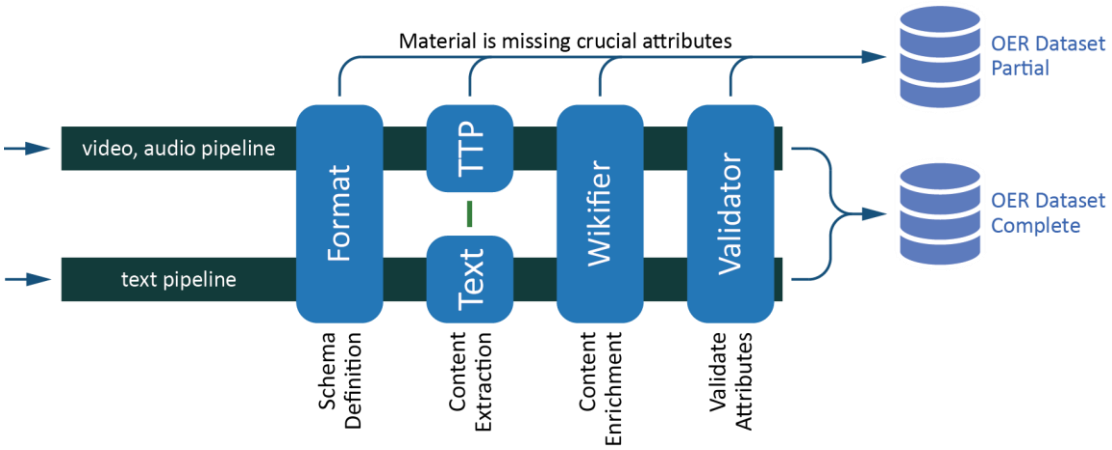

Figure 3: *The material processing pipeline. Depending on the materials type, it extracts its content, enriches and validates it before storing it in the database.*

3.2.1. Schema Definition

Since OER materials come from different providers and contain different values, we first format the provided metadata into a common format. In this step, we take the available material metadata and transform it into a JSON object containing the following attributes (some of which are optional).

- **Title:** The material title
- **Description (optional):** The description of the material
- **Provider URI:** The website containing the link to the material
- **Material URL:** The direct location of the material
- **Author (optional):** The authors of the material
- **Language:** The ISO 639-1 language code of the material, e.g. *sl*, *en*, *es*, etc.
- **Type:** The object containing the extension and mimetype of the material
- **Date Created (optional):** The creation date of the material in *YYYY-MM-DD* format
- **Date Retrieved:** The material retrieval date in *YYYY-MM-DD* format
- **Provider Token:** The token associated by the provider of the material
- **License:** The license of the material, which should be some Creative Commons version
- **Material Metadata (optional):** The object containing the metadata acquired through the acquisition or enrichment process

If any of the required attributes is missing, the material will be detected as partial and will be stored in a separate data table. Once the material has been formatted it is sent to the text step, which is content extraction.

3.2.2. Content Extraction

In this step we extract the content of the material and represent it in a text format. Currently, the platform is able to extract the content from three different types of material – text, video and audio.

Text. Extracting the content from text documents is performed using the textract [6] library – a library for extracting raw text from documents. The library is able to extract content from a range of text document formats (e.g. word documents, power point presentations, pdfs, etc.) and returns the content of the document in text form. The output text preserves the structure of the document, which is can be afterwards used in the content enrichment process.

Video and Audio. To retrieve the content of video and audio materials the platform sends a request to the Transcription and Translation Platform (TTP) [7], which is maintained by Universitat Politècnica de València. The platform provides an API endpoint through which we are able to request transcriptions and translations in a variety of languages - *Spanish, English, Slovene, German, French, Italian, Portuguese, and Catalan* - and formats. The platform requests for results in the mentioned languages and in three formats - *dfxp, webvtt*, and *plain* - which are the formats most used by the video OER providers. Doing this it will enable OER providers to integrate the transcriptions and translations into their videos.

The TTP platform also provides translation of textual documents, but does not preserve the structure of the provided document. Because of this, we have decided to omit the translation step from text documents in the processing pipeline.

More on the Transcription and Translation Platform capabilities can be found in Deliverable 3.4 – Early support for cross-lingual OER.

3.2.3. CONTENT ENRICHMENT

Once the materials content was extracted, we enrich the material metadata. This is done through the processed called Wikification. It identifies and links the textual components to the corresponding Wikipedia pages. To retrieve the annotations, we used Wikifier [8], a web service which returns a list of Wikipedia concepts that are most likely related to the textual input. The web service also provides cross- and multi-linguality which enables extracting and annotating materials in different languages. The output annotations contain the following attributes.

- **Concept URI:** The Wikipedia concept identifier in the material language
- **Concept Name:** The annotation name in the material language
- **Concept Secondary URI:** The annotation identifier in the English Wikipedia
- **Concept Secondary Name:** The annotation name in the English Wikipedia
- **Language:** The language code of the Wikipedia database from which the annotation was taken
- **Wiki Data Classes:** A list of classes to which the annotations belongs to according to WikiData [9]
- **Cosine:** The cosine similarity between the Wikipedia page corresponding to the annotation and the material content
- **PageRank:** The pagerank [10] of the annotation
- **DBPedia IRI:** The DBPedia [11] identifier
- **Support Length:** The number of keywords that support the relevance of the Wikipedia concept to the material

Wikifiers input text is limited to 20k characters, because of which processing longer text cannot be done. To this end, we have split longer documents into chunks of at most 10k characters and pass them to Wikifier. Here, we are careful not to split the document in the middle of a sentence, if possible.

Afterwards, we send each chunk of text to Wikifier and retrieve the Wikipedia concepts that are related to it. To calculate the similarity between the concept and the whole material we aggregated the concepts by calculating the weighed sum – which takes the relevant coverage of the chunk to the whole text and multiply it with the cosine similarity of the Wikipedia concept found in the corresponding chunk. We perform this for all concepts and assign the result to the material metadata.

3.2.4. ATTRIBUTE VALIDATION

The last step before storing the material into the database is to validate if all of the required attributes are present in the material metadata. The validation is performed with the use of the jsonschema [12], a library for validating the schema of JSON objects. For each material metadata object, the systems check if the required attributes are present, as well as the extracted material content in text and Wikipedia concepts. If all required attributes are present, the material metadata is stored in the database. Otherwise, we store the metadata in a separate data table for future evaluation of the missing data.

3.3. PIPELINE PROCESS ANALYSIS

The ingesting and processing pipeline have processed more than 91k material metadata – out of which 1.7k materials were partially processed. The reason of the partial material processing is due to the structure of the pdfs which instead of text can be a set of images – from which we are unable to extract the content.

While text materials are usually processed in under a minute, video materials are processed longer due to the intense process of generating transcriptions and translations, e.g. to generate transcriptions it takes approximately the duration of the video or audio material. Because of this, we have initialized multiple pipelines to enable parallel processing of materials: five pipelines for processing text materials and three pipelines for processing video and audio materials.

3.4. Real-Time Data Distribution

The ingesting and processing pipeline was designed to allow data distribution across multiple machines – enabled with the use of Apache Kafka. An instance of the Apache Kafka service as a Docker [13] container that was initialized using an existing Dockerfile for Apache Kafka [14].

In addition to the mentioned production machine, we initialized an additional (development) machine on the Pošta Cloud Infrastructure – used for testing and development of services, implementing methods and storing processed data. The development machine also retrieves the processed materials and enables an almost-to-real-world development experience. Figure 4 shows the data distribution flow between the machines.

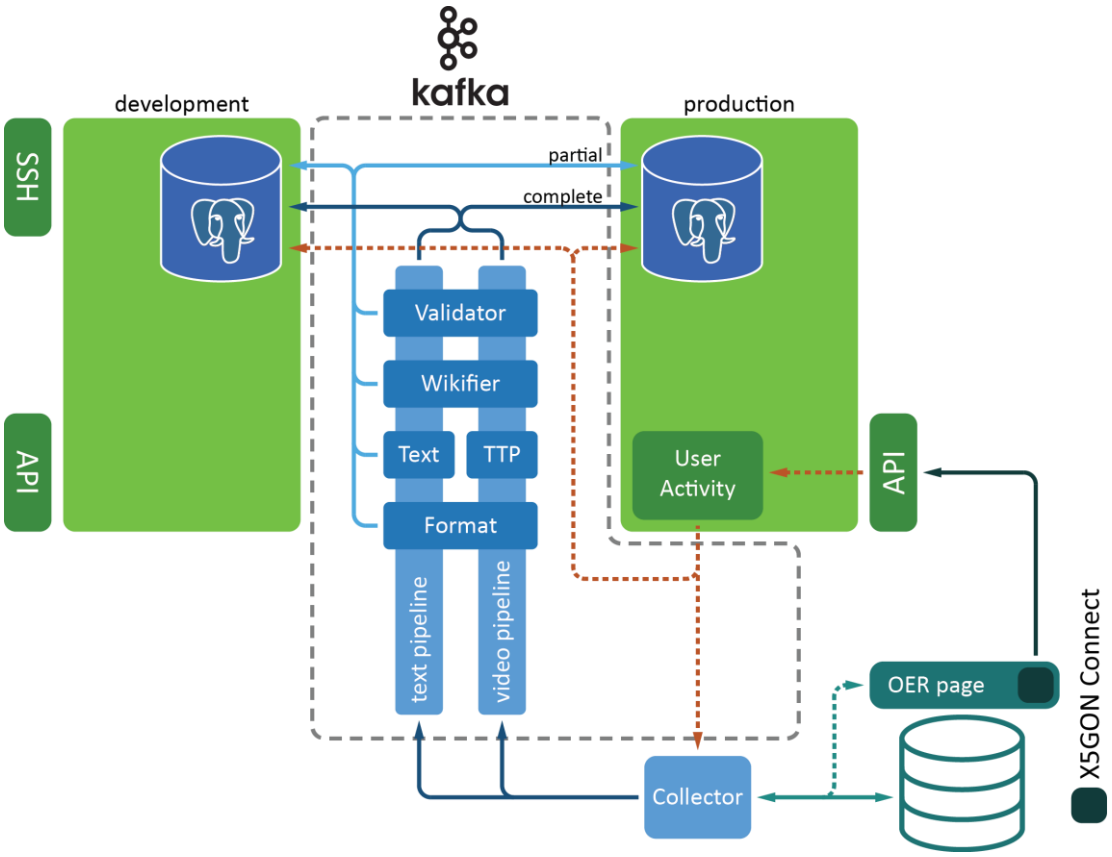

Figure 4: *The dynamics of data distribution with Apache Kafka. It shows the data distribution flow between the production and development machines.*

4. DATABASE

There is a wide range of database systems that available by different providers. To decide which one to integrate in the platform we set a list of requirements:

- **Non-dynamic schema.** We believe that to have a robust database where the platform would exactly know which fields are available in the data tables, the database must not allow dynamic schemas (e.g. where each record in the database contains different values). With this, we have a concrete schema of the data available in the database, making it more predictable what to expect.

- **Error instead of warning.** When a record contains a value that does not match the restrictions set in the schema, a database system can decide what to do with such value. Some systems modify the value to match the schema restriction, which we do not want – thus we want a system that throws an error and notifies the system when such an event happen.

These restrictions narrowed the list of database systems – finally deciding with PostgresQL [15]. The database system was integrated into the X5GON platform and contains the acquired data.

4.1. DATABASE STRUCTURE

The database was structured to contain both OER material metadata and user activity data – acquired through the X5GON Connect Service or the embedded recommendation list. In addition, the database contains user and material models used in the recommender engine. The script for creating the database structure is available on GitHub [16] – it is based on the proposed database structure described in Learning Analytic Engine 2.0 (D3.2) deliverable. Figure 5 shows the database schema of the most relevant data tables. What follows is a description of each table in the figure.

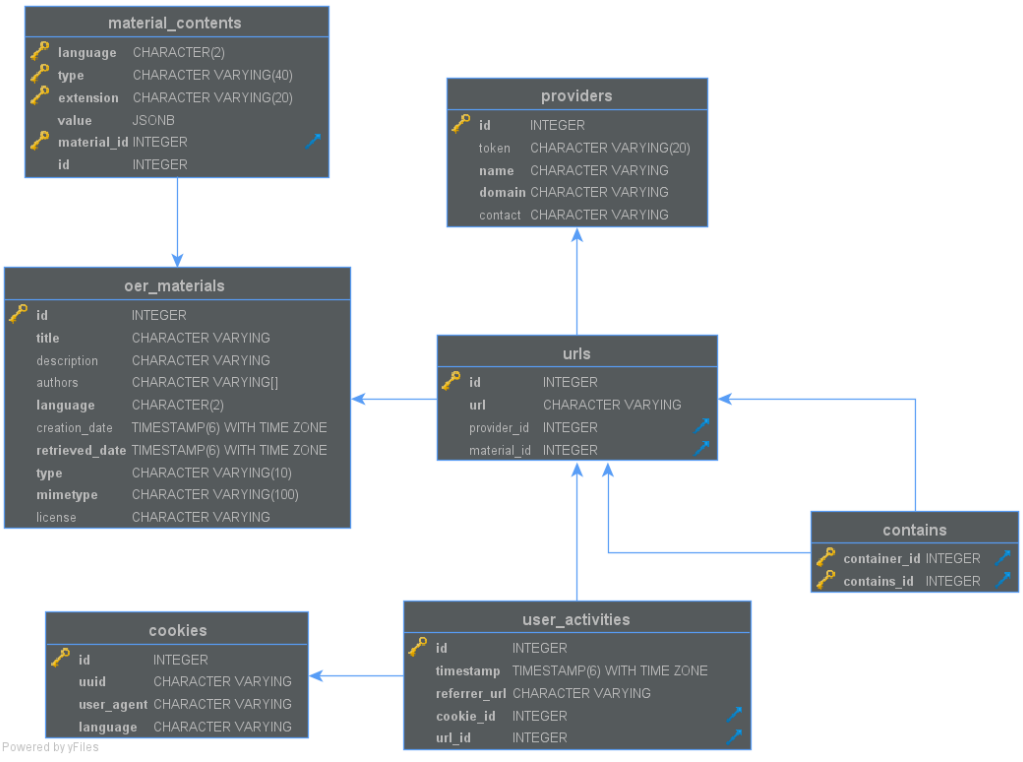

Figure 5: *The database schema of the most relevant data tables. It consists of data from user activities and OER material metadata.*

OER Materials. This table contains all of the open educational resources metadata that were successfully processed by the ingesting and processing pipeline. It contains the following columns:

- **ID:** The material ID
- **Title:** The material title
- Description: A short description about the material
- **Authors:** The authors associated with the material
- **Language:** The language in which the material is presented
- **Creation Date:** The date when the material was created
- **Retrieved Date:** The date when was the material retrieved
- **Type:** The material type (short)
- **Mimetype:** The full material mimetype
- **License:** The license of the material

OER Materials Partial. This table contains all of the open educational resources metadata which were not fully processed by the ingesting and processing pipeline. The columns in the table is similar to the ones in the OER Materials table with an additional column named *"message"* which contains the error message triggered when processing the material.

Material Contents. The table contains the extracted text, transcriptions and translations of the successfully processed materials. It contains the following columns:

- **ID:** The content ID
- **Language:** The language in which the content is present
- **Type:** The content type (e.g. extracted text, transcriptions and translations, as described in section 3.2.2)
- **Extension:** The content extension (e.g. the formats described in section 3.2.2)
- **Value:** The body of the content
- **Material ID:** The ID of the material from which the content was extracted in the *OER Materials* table

Providers. This table contains the provider information. It contains the following columns:

- **ID:** The provider ID
- **Token:** The token associated with the provider
- **Name:** The providers name
- **Domain:** The provider domain – where their repository is found
- **Contact:** The contact to the lead maintainer

URLs. The table containing all of the URLs that were collected through the materials or user activity data. The table contains the following columns:

- **ID:** The ID of the URL
- **URL:** The actual URL address
- **Provider ID:** The ID of the associated provider in the *Providers* table
- **Material ID:** The ID of the associated material in the *OER Materials* table

Contains. This table contains information about which URLs are found on the website corresponding to a different URL. It serves as a relation table for the records in the *URLs* table. It contains only two columns:

- **Container ID:** The ID of the record in the *URLs* table, which contains one of more other URLs
- **Contains ID:** The ID of the record in the URLs table, which is found in the website associated with the container URL

Cookies. This table contains the information about the user associated cookies. The cookies serve as user identifiers – without having to store any user data, such as its name, location or IP. The table contains the following columns:

- **ID:** The ID of the cookie
- **UUID:** The random string serving as the user identifier
- **User Agent:** The user agent (technology specification) associated with the cookie
- **Language:** The language configuration of the browser associated with the cookie

User Activities. The table contains information about the users' activities on the OER repositories with integrated X5GON Connect Service (see section 7). The table connects the *cookies* and *URLs* tables and contains the following columns:

- **ID:** The ID of the user activity record
- **Timestamp:** The time of when the user activity happened
- **Referrer URL:** The URL from where the user came to visit the current website
- **Cookie ID:** The ID of the cookie corresponding to the acting user
- **URL ID:** The ID of the website URL where the user triggered the activity

4.2. DATABASE STATISTICS

This section shows the database statistics – mainly the number of records in each database table. Table 1 shows the number of records in different database tables in the time of this writing.

Database Table	Number of records
OER Materials	89,923
OER Materials Partial	1,741
Material Contents	218,656
Providers	14
URLs	189,322
Contains	89,901
Cookies	438,648
User Activities	1,547,458

Table 1: *Number of records in the database statistics.*

In the duration of the project the platform as acquired a large amount of OER materials. In addition, we see that the number of unique users which accessed the OER repositories with the integrated X5GON Connect Service is large. The number user activities is also large – providing a rich information which can be used in the Learning Analytics and Recommender Engine development. Additional database statistics can be found in deliverables Final Prototype of User Modelling Architecture (4.2), Final Prototype of Recommendation Engine (4.4), and Prototype of Cross-Site Recommendation Engine (4.5).

5. PLATFORM SERVICES

This section is dedicated to the existing and planned services in the platform. There are three main types of services – the recommender engine, the learning analytics tool and the quality assurance tool. The services have access to the platform database from which they take the appropriate data for building the models. In addition, some of the service functionalities can be accessed via the platform API (see section 6). Figure 6 shows the highlighted architecture components that this section focuses on.

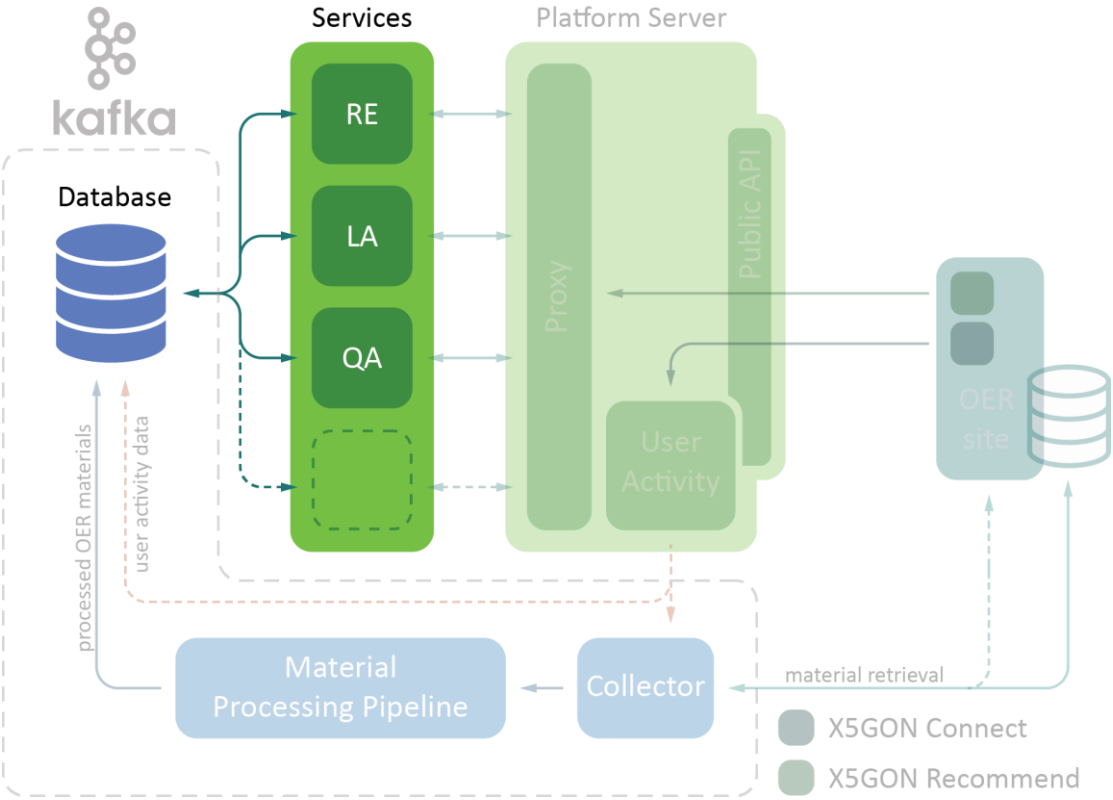

Figure 6: *The X5GON platform architecture with the focus on the X5GON services. The services retrieve and stores the resulting data to the database.*

What follows is a short description of each service with the references to the document which contains a longer description of the functionality and methods used in the service.

5.1. RECOMMENDER ENGINE

The recommender engine produces recommendations of materials and bundles. It analyses the data produced by the material processing pipeline and user activity data, and creates material and user models. These models are then employed to provide a list of recommendations based on the user query.

More can be found in deliverables Final Prototype of User Modelling Architecture (4.2), Final Prototype of Recommendation Engine (4.4), and Prototype of Cross-Site Recommendation Engine (4.5).

5.2. QUALITY ASSURANCE TOOL

The quality assurance models that are developed by X5GON project mainly enable two services: 1) Extraction of quality features from educational resources, and 2) Providing quality score prediction for Open Educational Resources.

The first service envisages a feature that will present 13 quality features that are extracted from the text representation of the educational resources. The features categorise into multiple quality verticals such as Understandability, Authority, Presentation, Freshness and Topic Coverage. These features can be extracted from a variety of educational resources that span across different modalities (e.g. video, audio, text etc.). These features will provide informative summary of how different quality features are present in a document enabling the learner/ stakeholder to take a more informed decision taking these features into account. A detailed description of how these features were identified and extracted can be found at Deliverable D1.1 - Quality Assurance Models.

The second service envisages creating scalable, automatic quality evaluation models using the above features. Currently, videolectures data has been used to train a supervised learning quality model that can perform with 71% pairwise accuracy when comparing pairs of lectures using model predictions. Any relevant open educational dataset can be used to train the quality model using the above features and the model performance can be evaluated.

A detailed discussion about how the models are developed is outlined in Deliverable D1.1 - Quality Assurance Models. Furthermore, Deliverable D1.2 - Evaluation of Quality Assurance Models discusses how these models are evaluated.

5.3. LEARNING ANALYTICS TOOL

The learning analytics tool currently provides two public models – the difficulty and order models.

Difficulty Model. This model is able to measure the resource hardness based on lexicon and grammar properties of the materials. An implementation for this approach is available in the WP 3 API through the service *wikification2conpersec*.

Order Model. One of the learning approaches is to have a logical order during the learning process when consuming the educative resources. So, this second model tries to evaluate a pair of resources and give a relative order to consume these resources based on the “continuous wikifier model”. The main idea of this method, is to catch the common concepts between the resources, and to use their distribution over the resources to infer the order. More precisely, we assume the keys concepts to predict the order are those, which are present in the end a resource and in the beginning of the other. From our observation these patterns correspond to a concept introduction, and our goal is to choose the order which maximizes this kind of transition in order to have a fluid transition between the resources and to introduce as many prerequisites as possible. Implementation for this approach is available in the WP3 API through the service *continuouswikification2order*.

Further details on these metrics can be found in the Learning Analytic Engine 2.0 (D3.2) deliverable.

6. PLATFORM API

This section describes some of the platform API. The API enables accessing different services, as well as access to the processed OER material metadata. The platform API is publicly accessible and it currently allows to perform the following requests:

- Upload OER material metadata;
- Retrieve the processed OER materials;
- Query for OER material metadata;
- Request for different types of recommendations.

Figure 7 shows which components of the X5GON platform architecture are the focus of this section.

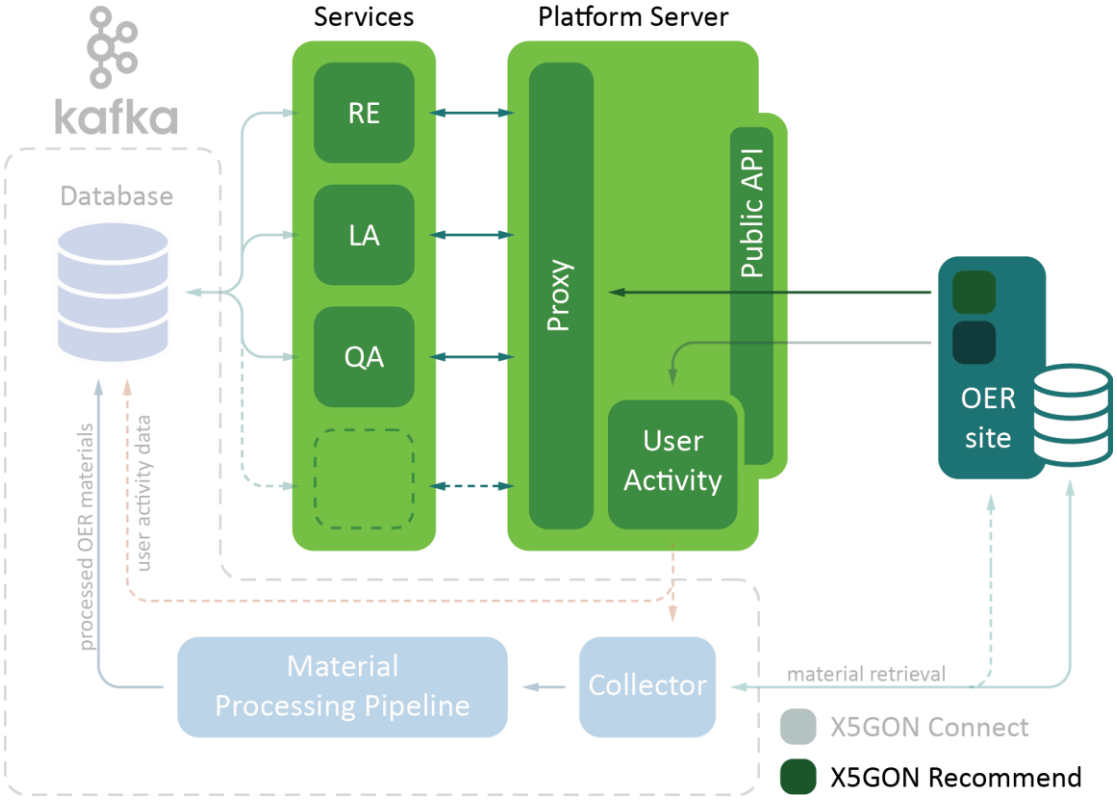

Figure 7: *The X5GON platform architecture with the focus on the components associated with the platform API. The API retrieves the user request and proxies it to the appropriate service.*

What follows is a description of some of the supported REST API. A full list of available API endpoints is available on the platform website – under the documentation [17].

6.1. MATERIAL UPLOAD API

The material upload REST API endpoint enables dynamic uploading of material metadata to the X5GON pipeline. The endpoint enables OER providers to upload their material metadata to be processed and be stored into the database. Afterwards, the provider can access the extracted content and annotations, as well as enable their materials be present in the recommendations provided by the X5GON platform.

The upload endpoint requires an API key which can be provided by one of the platform maintainers.

In addition, the endpoint validates if the sent material metadata is in the correct format and if all required attributes are present. If not, the endpoint returns the information about the number of successfully uploaded materials, as well as the submission errors, describing the number of materials that are in an incorrect format, and the description of the error for each invalid attribute. Once the material was successfully submitted and processed, it can be retrieved through the **material retrieval API**.

6.2. MATERIAL RETRIEVAL API

The material retrieval API endpoint allows to retrieve material metadata information from the database. The retrieval is performed by providing a set of (optional) query parameters.

- **Languages:** a comma separated list of ISO 639-1 language codes
- **Limit:** The number of records to be returned
- **Offset:** The number of records from a collection to skip
- **Page:** The page number of the record collection

Afterwards, the endpoint returns a list of material metadata that correspond to the provided query parameters. The material metadata returned by the endpoint contains the following attributes:

- **Material ID:** The unique ID of the OER material
- **Title:** The title of the OER material
- **Description:** The description of the OER material
- **URL:** The URL of the OER material
- **Language:** The language of the OER material in ISO 639-1 code
- **Type:** The type of the OER material (e.g. *text, video* or *audio*)
- **Mimetype:** The mimetype of the OER material
- **Contents IDs:** The IDs of the material contents (e.g. extracted content text, transcriptions and translations)
- **Provider:** The JSON object containing the provider's name and domain

The content IDs can be used to retrieve the specific transcription and/or translation of the material – and including them as captions in the material video player – through a designated API endpoint. That endpoint returns the following content attributes:

- **Content ID:** The unique ID of the content
- **Type:** The type of content (e.g. *text_extraction, transcription*, or *translation*)
- **Extension:** The extension of content (e.g. *plain, dfxp,* or *webvtt*)
- **Value:** The JSON object containing the value of the content
- **Language:** The language of the content in ISO 639-1 code

6.3. MATERIAL SEARCH API

The material search API endpoint enables users to search through the indexed OER materials. The material search functionality is part of the Recommender Engine presented in deliverable 4.1 The search is initialized by providing a set of (optional) query parameters.

- **Text:** The seed text from which the system finds the relevant OER materials
- **Type:** The type of OER materials to retrieve (e.g. *all, video, audio,* and *text*). The default value is **all**
- **Page:** The page number of the provided list of relevant OER materials. The default value is **1**

For a given user query text, it provides a list material metadata with the following attributes:

- **Material ID:** The unique ID of the OER material
- **Weight:** The number representing the relevance of the OER material. Materials with a greater weight means bigger relevance to the user query
- **URL:** The URL of the OER material
- **Title:** The title of the OER material
- **Description:** The description of the OER material
- **Provider:** The name of the OER material provider
- **Language:** The language of the OER material in ISO 639-1 code
- **Wikipedia:** A list of extracted Wikipedia concept objects, where each object contains the concepts URI and its support length (as described in section 3.2.3)
- **Type:** The type of the OER material (e.g. *text, video* or *audio*)

In addition, the endpoint provides additional metadata such as the total number of found OER materials and the maximum number of pages.

6.4. MATERIAL RECOMMEND API

The material recommender API endpoint provides recommendations based on the user preferences. The API proxies the request to the Recommender Engine which it then processes and returns the recommendations. Currently, the platform API provides four types of recommendations:

OER Materials. This type of recommendations is similar to the material search API endpoint. For a given user query text, it provides a list of most relevant recommendations.

OER Bundles. This type provides a list of recommended educational bundles – where a bundle is defined as the website containing one or more OER materials. Contrary to the OER Materials type, this endpoint does not accept user query text as an input. Instead, it accepts the URL of the educational bundles for which the user wishes to find similar bundles.

Personalized. This type provides a list of recommended educational bundles that are personalized to the particular user. This endpoint does not require any query parameters since the only value it requires is the user identifier (e.g. cookie ID).

Collaborative Filtering. This type provides a list of recommended educational bundles provided by the collaborative filtering algorithm. Similar to the personalized type, it does not require any query parameters – only the user identifier.

8. X5GON CONNECT SERVICE

Part of the projects goals is also to understand how the users are consuming the educational material and provide insight for the Learning Analytics and Recommender Engines tasks. In the deliverable 2.1 – requirements & architecture report, we have foreseen the use of a library for acquiring information about the user activity on the OER repositories. We have developed the X5GON user activity tracker snippet library and rebranded it to the **X5GON Connect Service.** This section describes the functionalities of the Connect Service and the components associated with it.

8.1. APPLICATION FORM

The Connect Service allows the OER repository to get an insight into which materials are the users consuming and how they are transitioning from one material to another. But before the repository integrates the service into their system, they need to register to the X5GON platform, where they get a unique token. The token is a short string which enables the platform to identify which repository is being accessed. In addition, it also serves as the ID for accessing the connect services information on their repository – how many unique users have visited their repository and how many times. Figure 8 shows a snapshot of the application form.

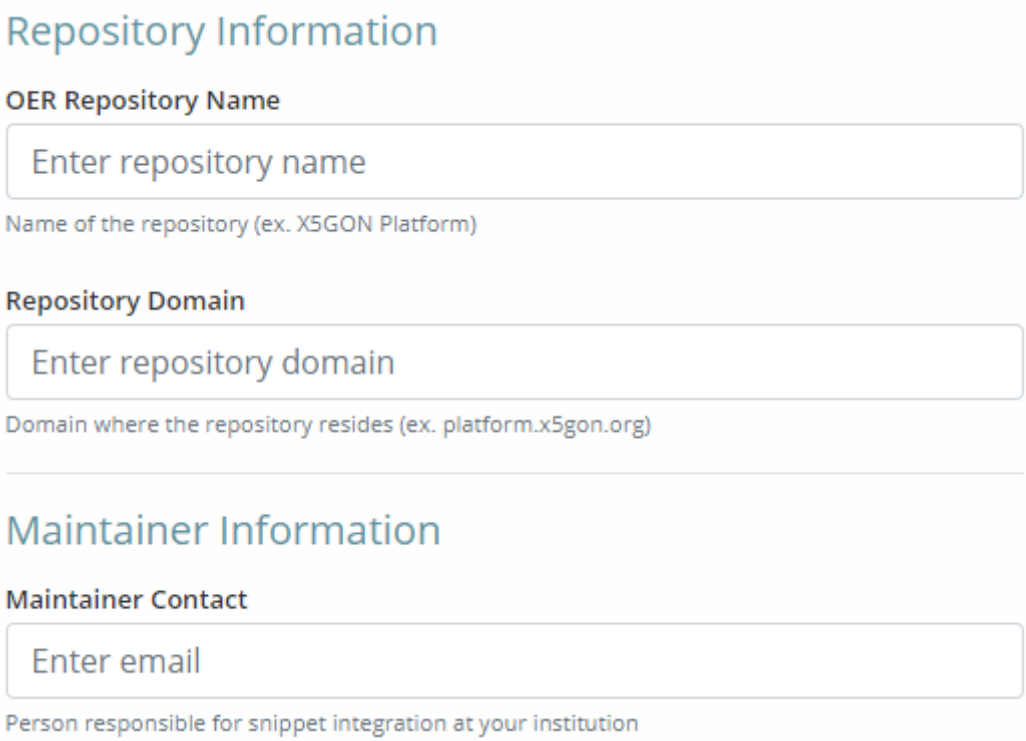

Figure 8: *A snapshot of the application form.*

The form does not request a lot of information – only the OER repository name, its domain and the contact information of the person responsible for the service integration into the repository.

8.2. SETTING UP THE CONNECT SERVICE

The connect service is design for easy integration – at the same time when the user retrieves the unique token it is also informed about the privacy policy [18] including what kind of user information the service collects, how to integrate the library into their system and the security measurements (e.g. subresource integrity [19]) are provided at service integration. In addition, we provide a solution for managing cookie policies

enabled by the Cookie Consent [20] API – providing a thorough description of how to use the API in combination with the Connect Service. This solution enables the visiting user to decide if it wants the library send its activity information to the platform – disabling the functionality if it does not provide an **active** consent.

Note that we ask for the connect service to be integrated only on the repository pages which contain OER materials. This is because the service is intended to retrieve information on the materials that were visited by a user. Currently, the service does not have any capabilities of detecting if the page contains materials or not.

8.3. CONNECT SERVICE FUNCTIONALITY

Once the connect service is integrated in the repositories and the user gives an active consent, the service performs the following actions.

1. **Assigns a cookie ID to the user.** When the user first visits an OER repository with an integrated Connect Service, the service assigns a cookie ID which is then used to identity the user in the future. This ID is randomly generated, does not take any user data into account and cannot be used to get back to the user.

2. **Sends user activity data to the platform.** When a user visits a page with an integrated connect service, it sends the following user activity data to the platform:

    - **User ID:** the cookie ID used to identify the user accessed the learning material
    - **Material URL:** The material identifier and the link that the user visited
    - **Referrer URL:** The URL from which the user arrived to the material
    - **Access Date:** The date of the visit
    - **User Agents:** The information about the technology used to access the materials
    - **Language:** the language configuration of the user's technology

The sent data is then stored in the database and used in the ingesting and processing pipeline (see section 3) for material retrieval. In addition, the user activity data is then used in both the Learning Analytics and Recommender Engine research for understanding the education process of the users and for creating personalized recommendations – more is described in deliverables Learning Analytics Engine 2.0 (3.2), Final Prototype of User Modelling Architecture (4.2), Final Prototype of Recommendation Engine (4.4), and Prototype of Cross-Site Recommendation Engine (4.5).

The Connect Service has also been modified and extended as a Moodle plugin – enabling integration of the connect service into the Moodle LMS. More is described in the deliverable Learning Analytics Engine 2.0 (3.2).

9. FUTURE PLANS

Even though this document describes the final server-side platform, we have identified parts of the platform which could be improved. This section describes the future work for the platform.

9.1. CONTENT EXTRACTION

The ingesting and processing pipeline is able to process large volumes of text, video and audio materials – but due to the variety of formats and content there is still some room for improvement.

Text Materials. The current library for extracting content is unable to represent equations in a meaningful way. The equations are transformed into a set of characters which are not informative. In addition, the extracted content does not preserve the format and structure. To this end, we will consider using and developing tools which will solve these problems.

Additional Enrichment Components. To have a better insight of the extracted content we are considering to develop additional components that will a) extract named entities, and b) provide word-level information of the contents works. In addition, we will explore the option of mapping the learning materials to scientific fields.

9.2. SERVICE INTEGRATION

In the current state, we have three services that are in the platform integration process. While the recommender engines and learning analytics tools are already integrated and are ready for use, the quality assurance is in late-development phase and are planned to be integrated before the end of the project. In addition, we will decide which functions of the services we will make public – the decision will be based on the results of the services, its implementation and use-cases.

In addition, we are considering to develop a quick-to-integrate-service component which will enable easy integration of services that will be developed in the third year of the project, as well as after it ends.

9.3. PROCESS ANALYSIS COMPONENTS

To analyse the different processes of the platform and to have an overview of the data in the database we intend to develop components that will analyse these aspects of the platform. In addition, we will develop a GUI through which we will be able to monitor and control the processes, as well as see the dynamics of the data.

9.4. CONTINUOUS INDEXING OF OER MATERIALS

To have a wide range of learning materials we will continue acquiring and indexing learning material. We will continue to write scripts for acquiring the material metadata. In addition, we will find repositories which will be prepared to share their OER material metadata directly by uploading the material metadata through the material upload API (see section 6.1).

10. CONCLUSION

In this document we present the final server-side platform architecture. We present the four main components, e.g. the database, ingesting and processing pipeline, services and platform API, and how they are connected. The platform is running on two machines on the Pošta's Cloud Infrastructure – one machine containing the production instance, and the other containing the development instance (used for development and testing purposes).

While the server-side platform is mostly developed we still identified part of the platform that could be improved, more precisely extraction and enrichment components of the ingesting and processing pipeline, finalize the integration of existing services, developing analysis components for monitoring and controlling the platform processes, as well as continue indexing OER materials.

REFERENCES

[1] The Apache Software Foundation, “Apache Kafka,” 5 August 2019. [Online]. Available: https://kafka.apache.org/.

[2] Jožef Stefan Institute, “JozefStefanInstitute/x5gon: Connecting and processing content from OER repositories,” [Online]. Available: https://github.com/JozefStefanInstitute/x5gon. [Accessed 8 August 2019].

[3] Jožef Stefan Institute, “Home | X5GON Platform,” [Online]. Available: https://platform.x5gon.org. [Accessed 8 August 2019].

[4] Jožef Stefan Institute, “x5gon/src/server/preproc at master - JozefStefanInstitute/x5gon,” [Online]. Available: https://github.com/JozefStefanInstitute/x5gon/tree/master/src/server/preproc. [Accessed 8 August 2019].

[5] V. Jovanoski, “qtopology | Distributed stream processing layer,” [Online]. Available: https://qminer.github.io/qtopology/. [Accessed 8 August 2019].

[6] D. Bashford, “textract - npm,” [Online]. Available: https://www.npmjs.com/package/textract. [Accessed 8 August 2019].

[7] J. a. G. D.-M. G. V. a. C. J. a. J. A. Iranzo-Sánchez, “The MLLP-UPV Supervised Machine Translation Systems for WMT19 News Translation Task,” in *Proc. of Fourth Conference on Machine Translation (WMT19)*, Florence (Italy), 2019.

[8] G. L. M. G. Janez Brank, “Annotating Documents with Relevant Wikipedia Concepts,” in *Proceedings of the Slovenian Conference on Data Mining and Data Warehouses*, Ljubljana, Slovenia, 2017.

[9] Wikimedia Foundation, “Wikidata,” [Online]. Available: https://www.wikidata.org/. [Accessed 8 August 2019].

[10] Wikipedia, the free encyclopedia, “PageRank - Wikipedia,” [Online]. Available: https://en.wikipedia.org/wiki/PageRank. [Accessed 8 August 2019].

[11] DBpedia, “DBpedia,” [Online]. Available: http://dbpedia.org/. [Accessed 8 August 2019].

[12] T. d. Grunt, “tdegrunt/jsonschema: JSON Schema validation,” [Online]. Available: https://www.npmjs.com/package/jsonschema. [Accessed 8 August 2019].

[13] Docker, Inc., “Enterprise Container Platform | Docker,” [Online]. Available: https://www.docker.com/. [Accessed 8 August 2019].

[14] wurstmeister, “wurstmeister/kafka - Docker Hub,” [Online]. Available: https://hub.docker.com/r/wurstmeister/kafka/. [Accessed 8 August 2019].

[15] “PostgreSQL: The world's most advanced open source database,” The PostgreSQL Global Development Group, [Online]. Available: https://www.postgresql.org/. [Accessed 27 03 2018].

[16] Jožef Stefan Institute, “x5gon/create-postgres-database.js at master - JozefStefanInstitute/x5gon,” [Online]. Available: https://github.com/JozefStefanInstitute/x5gon/blob/master/src/load/create-postgres-database.js. [Accessed 8 August 2019].

[17] Jožef Stefan Institute, “API Documentation | X5GON Platform,” [Online]. Available: https://platform.x5gon.org/documentation. [Accessed 8 August 2019].

[18] Jožef Stefan Institute, “Privacy Policy | X5GON Platform,” [Online]. Available: https://platform.x5gon.org/privacy_policy. [Accessed 8 August 2019].

[19] moz://a, “Subresource Integrity - Web security | MDN,” [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/Security/Subresource_Integrity. [Accessed 8 August 2019].

[20] Osano, Inc., “Cookie Consent by Osano | The most popular solution to the EU cookie law,” Osano, Inc., [Online]. Available: https://cookieconsent.osano.com/. [Accessed 8 August 2019].