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: D2.1 – Requirements & architecture report
Author(s): Erik Novak
Contributing partners: JSI
Date:
Approved by:
Type: R
Status: Draft/Final
Contact:

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
March 2nd	Marko Grobelnik (MG), Mitja Jermol (MJ), Davor Orlič (DO)	MG: The deliverable has a good structure, platform architecture has a good high-level description DO: images describing the platform architecture components should show how the component fits in the overall architecture
April 12th		

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Acronyms	Definitions
OER	Open Educational Resources
XML	eXtensible Markup Language
RDF	Resource Description Framework
SVM	Support Vector Machine
NLP	Natural Language Processing
API	Application Programming Interface
JSON	JavaScript Object Notation
URL	Uniform Resource Locator
MIME	Multipurpose Internet Mail Extensions

ABSTRACT

In this report we present the technology and data sources available within the project consortium as well as the X5GON platform architecture which will connect all of the services developed within the project. Additionally, we describe the core requirements needed for the architecture to work as well as how the requirements will be achieved.

1. INTRODUCTION

The purpose of this document is to gather, from all partners, a comprehensive list of existing technology and data sources that are available within the consortium and to recognize the core requirements needed for the development of the X5GON platform with respect to the social network and the learning analytics & recommendation engine plugin components.

The X5GON project starts with a significant amount of pre-existing operational technology providing a good baseline for extending them during the project evolution. This document focuses on retrieving existing elements with regard to the next two issues:

- Existing data sources and data requirements
- Existing technology and technical requirements

Based on these two points, an initial set of requirements and functionalities were created to initialize the development of the overall architecture and functionality for the X5GON project.

The overall template is structured into three main parts: existing technology, data sources and core requirements as described in the following sections.

This document will be accessible and used by any partner for the purposes inside the X5GON project.

2. EXISTING TECHNOLOGY

This section describes the different technology which are available for the X5GON project and can be provided by any of the partners of the consortium. The technology will be used in the platform architecture development as well as work done within other work packages. This technology must also include the services which are going to be served by the X5GON platform or services the platform will be based on.

Section 2.1 present a quick list of all of the available technology as shown in Table 1. The table was collaboratively filled by the consortium. Meaning of individual columns in the technology table is provided:

- **Technology:** name or identification of the technology i.e. QMiner
- **Responsible:** name of the institution or company responsible of the technology described, i.e. JSI
- **Sources availability:** type of availability of the source code yes/no/partly
- **How it is provided:** the type of accessibility i.e. library, web service, etc.
- **Programming language:** type of language used by the technology, i.e. C++, JavaScript, etc.
- **License:** type of license, i.e. GNU/GPL, proprietary, etc.
- **Web site URL:** address of the web site which includes the documentation and information regarding the technology

A more detailed description of each technology and how it's going to be used is presented in Section 2.2. For each technology we provide the following attributes:

- **Description:** the description of the technology
- **Usage within the project:** how will the technology be used within the project
- **Related technology:** what technologies are related or associated to it
- **Constraints:** the technical constraints when using the technology, i.e. hardware, operating system, internet connection etc.
- **Dependencies:** external dependencies when using the technology, i.e. libraries or programs that need to be linked or included
- **Bottlenecks:** does the technology slow down the process pipeline

2.1 TECHNOLOGY QUICKLIST

Technology	Responsible	Sources Availability	How is it Provided	Programming Language	License	Web site URL
Qminer	JSI	Yes	Library	C++, JavaScript	BSD 2-Clause “Simplified” License	https://qminer.github.io/
Enrycher	JSI	Partly	Web Service	C++, Java	Proprietary	http://enrycher.ijs.si/
Wikifier	JSI	Partly	Web Service	C++	Proprietary	http://wikifier.org/
Xling	JSI	Partly	Web Service, Library	C++	Proprietary	http://xling.ijs.si/
EventRegistry	JSI	Partly	Web Service	C++	Proprietary	http://eventregistry.org/
TTT	UPV	Yes	Web Service & API	C++, C, Python	Apache License 2.0	http://www.mllp.upv.es/tlk/
TLP	UPV	Yes	Web Service & API	Python, PHP, JavaScript	Apache License 2.0	http://www.mllp.upv.es/tlp/

Table 1: *List of existing technology*

2.2 Technology Descriptions

This section describes the different technology in the consortium's arsenal. For each technology we provide the following attributes:

- **Description:** the description of the technology
- **Usage within the project:** how will the technology be used within the project
- **Related technology:** what technologies are related or associated to it
- **Constraints:** the technical constraints when using the technology, i.e. hardware, operating system, internet connection etc.
- **Dependencies:** external dependencies when using the technology, i.e. libraries or programs that need to be linked or included
- **Bottlenecks:** does the technology slow down the process pipeline

2.2.1 QMiner

Description	QMiner is an analytics platform for large-scale real-time streams containing structured and unstructured data. It is designed for scaling to millions of data points on high-end commodity hardware, providing efficient storage, retrieval and analytics mechanisms with real-time response.
Usage within the project	The platform includes machine learning methods, such as k-Nearest Neighbours, kmeans clustering, SVM, as well as a storage mechanism which will be used for developing recommendation methods as well as pre-processing OER materials and user activity data. It also contains feature extraction methods for information extraction from raw textual data.
Related technology	Enrycher, Wikifier, Xling, EventRegistry
Constraints	No constraints
Dependencies	Node.js
Bottlenecks	The system imposes only a small overhead on the processing pipeline. The performance mostly depends on the processing technology.

2.2.2 Enrycher

Description	Enrycher is a service-oriented system, providing shallow as well as deep text processing functionality at the text document level. The system consists of two major components. First is the architecture, which is design to easily scale with respect to number of requests that can be processed in parallel. Second is a set of components, which perform particular tasks in the processing pipeline (e.g. part-of-speech tagging, named entity extraction). The output can be either in Enrycher-defined XML schema or in RDF.
Usage within the project	The service will be used to extract part-of-speech tags and named entities as part of the OER material pre-processing pipeline
Related technology	Any NLP technology can be used to provide support for additional languages.
Constraints	Internet connection
Dependencies	No external dependencies

Bottlenecks	The system imposes only a small overhead on the processing pipeline. The performance mostly depends on the processing technology.

2.2.3 Wikifier

Description	The JSI Wikifier web service takes a text document as an input and annotates it with links to relevant Wikipedia concepts. It supports any language for which a localized Wikipedia with at least 10k pages is available – currently accumulating to 135 languages. The web service can auto-detect the language in which the text was written and can process millions of requests per day.
Usage within the project	The service will be used to extract Wikipedia concepts found within the OER material. These are going to be used to map OER material written in different languages into a common semantic space allowing us to do cross-lingual recommendations
Related technology	QMiner, Xling, EventRegistry
Constraints	Internet connection
Dependencies	No external dependencies
Bottlenecks	Each request sent to the web service takes up to 3 seconds to respond which is still efficient within the pre-processing pipeline.

2.2.4 Xling

Description	Xling is a web service which handles cross-lingual similarity computation and cross-lingual categorization of two text documents written in different languages. It uses Wikipedia concepts returned by the Wikifier web service to map the input documents into a common semantic space in which the service then calculates similarity and dmoz classification.
Usage within the project	The service will be used to calculate similarity between different OER materials and to extract dmoz classifications.
Related technology	Wikifier, QMiner
Constraints	The system imposes only a small overhead on the processing pipeline. The performance mostly depends on the processing components.
Dependencies	No external dependencies
Bottlenecks	Request processing duration depends on the amount of content sent to the service. The service will be used within pre-processing pipeline.

2.2.5 EventRegistry

Description	The Event Registry service-oriented system monitors world events by collection and processing news articles from more than 30k news sources globally in 15 languages. It identifies named entities using natural language processing and use machine learning methods to identify world events and extracts information about the event. Additionally, the events are connected and related to each other – showing the previous and following related events.

Usage within the project	The methods developed within the service will be used and modified for the X5GON platform.
Related technology	Wikifier, QMiner
Constraints	Internet connection
Dependencies	No external dependencies
Bottlenecks	No bottlenecks. The service will be used as inspiration and starting point for platform development.

2.2.6 TTT

Description	UPV result R-16731- 2013 - TTT: "Transcription and Translation Tools for Video Lectures"
Usage within the project	Transcription and translation of video and audio OER material
Related technology	No related technology
Constraints	Internet connection
Dependencies	No external dependencies
Bottlenecks	No bottlenecks

2.2.7 TLP

Description	UPV result R-17943- 2016 - TLP: "The transLectures-UPV Platform. Multilingual subtitling and text translation for MOOCs and media repositories"
Usage within the project	Assisted multilingual media subtitling
Related technology	No related technology
Constraints	Internet connection
Dependencies	No external dependencies
Bottlenecks	No bottlenecks

3. DATA SOURCES

This section lists data sources identified at the start of the X5GON project, which are available within the consortium. The data sources will be used for testing platform functionality as well as developing recommendation and learning analytics models within WP3 and WP4. The data sources are listed in Table 2.

Meaning of individual columns in Table 2:

- **Data Entity:** name or identification of the data source, i.e. Videolectures.NET
- **Data Responsible:** name of the institution or company responsible of the data source described, i.e. JSI
- **Data sources:** the type of data which is gathered, i.e. main stream news/blogs/videos/presentations/books/articles/…
- **How can be accessed:** the method to get access to the data, i.e. API/WS/files/databases/…
- **Type of data:** the type of data which is stored, i.e. raw_text/categories/ontology/json/…
- **Amount of data and covered languages:** the size of data which needs to be stored for being processed in the pipeline. This information can be expressed in M/T/P/Bytes or as a data flow per time, i.e. 1TB or 150.000x15kB streams news per day. Languages which are represented in the data (when applicable). If possible, list languages.
- **License:** identifies if the data is only available for the project purposes (PR) or if it is also public for any other purposes (PU). The identification of the type of license would be desirable.
- **Web site URL:** address of the web site which includes the documentation and information regarding the data source.

Data Entity	Responsible	Data Sources	How can be accessed?	Type of data	Amount of data and covered languages	License	Web site
Videolectures.NET	JSI	Videos	API	JSON	25k videos Languages: English, Slovene, Spanish, German, Chinese	PR	http://videolectures.net
UPV Media Video Lecture Repository	UPV	Videos	Web Service	MP4	1.8k videos, 800 related slides Languages: Spanish	PR	https://media.upv.es

Table 2: *List of data sources*

4. ARCHITECTURE & CORE REQUIREMENTS

This section describes the initial architecture of the X5GON platform and lists the core requirements of each architecture component. The overall platform architecture is presented in section 4.1 while the platform components and their respective requirements are described in section 4.2Platform Components and Their Requirements.

4.1 PLATFORM ARCHITECTURE

The X5GON platform architecture is designed to efficiently handle different types of OER material as well as mapping material written/spoken in different languages into a common semantic space, i.e. same vocabulary. This handling and mapping is done in the pre-processing pipeline described in section 4.2.1. Once the OER material are processed they will be stored in a PostresQL database [1] making it available to all consortium members. The data will be used for development of recommendation and learning analytics engines as part of WP3 and WP4 as well as quality assurance done in WP1 and other unforeseen services developed within the X5GON project.

The platform will need to handle millions of requests per day that will come from users accessing material from different OER repositories. To this end, the platform will scale and take advantage of the cloud infrastructure provided by Pošta Slovenije where we will build machine learning models used in the platform, create database and model backups, and run the platform in production. Figure 1 shows the architecture of the X5GON platform.

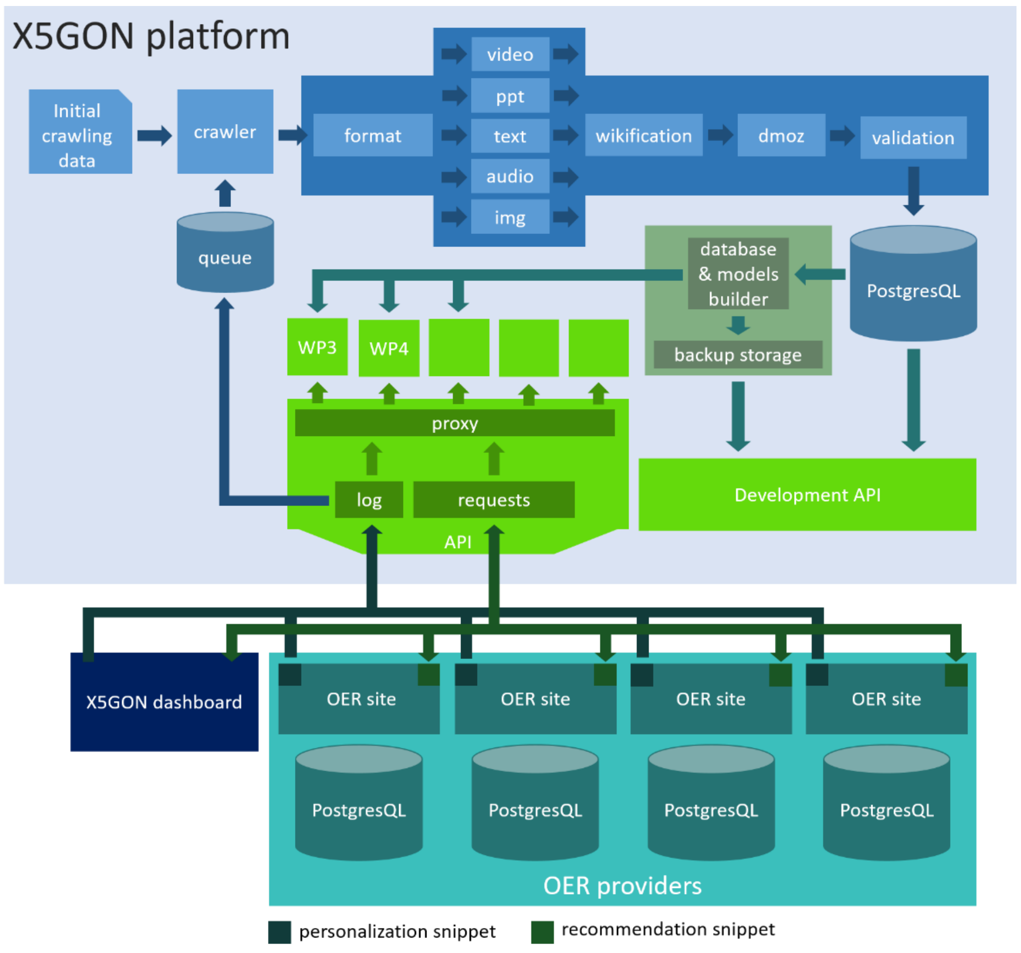


Figure 1: *The X5GON platform*

In the following section 4.2 we present the components that build the X5GON platform, the technology used in the component and its core technical requirements.

4.2 Platform Components and Their Requirements

In this section we present the X5GON platform components. There are three components: crawling and pre-processing pipeline, user and development API, and backup mechanism. The role of each component and its requirements are presented in the following subsections. Additionally, the user activity tracker library is required for tracking user activity across different OER repositories. How it is used and what is the data it will track is presented at the end of this section.

4.2.1 Crawling and pre-processing pipeline

The crawling and pre-processing pipeline's role is to acquire OER materials and prepare them for other platform components' consumption. It is responsible for retrieving OER material, extracting information from the material and storing the data in the database. It also maps the material into a common semantic space, i.e. common vocabulary, which is essential for comparing materials written in different languages. Figure 2 shows the pipeline and its location within the architecture. Pipeline elements are labelled with numbers for easier identification. What follows is description of elements in the pipeline.

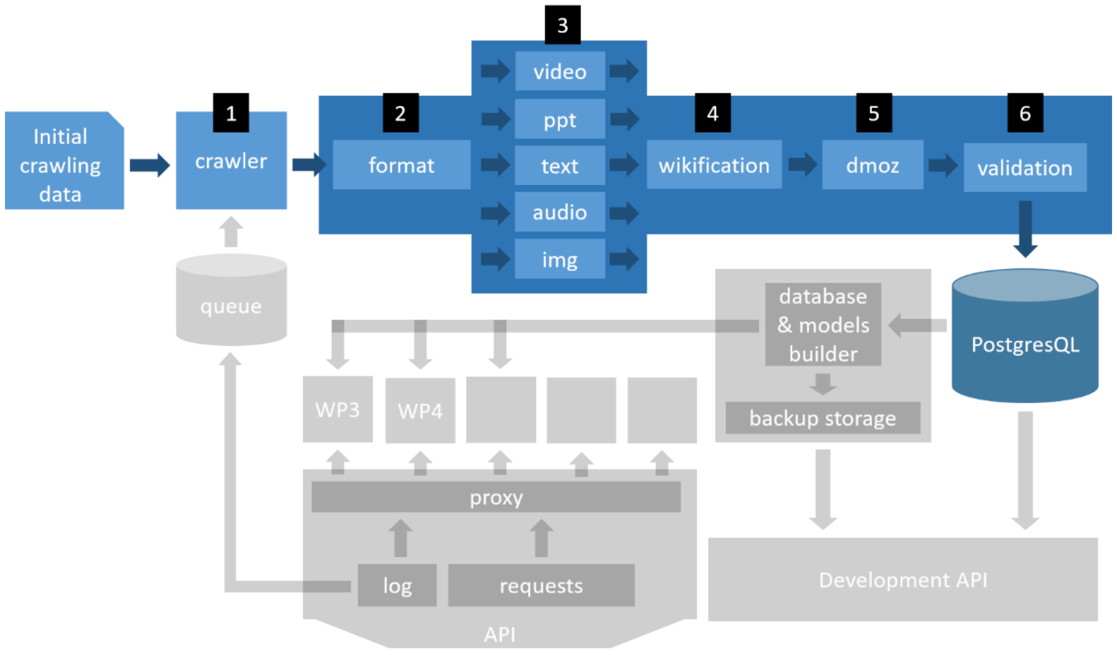
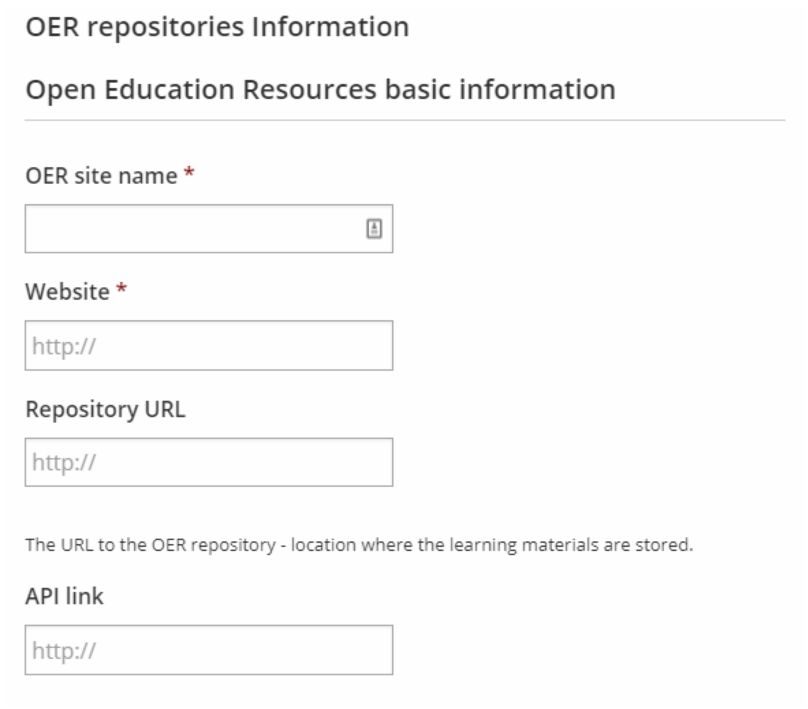


Figure 2: *Crawling and pre-processing pipeline*

Crawler (1). The first element in the pipeline is the crawler whose task is to retrieve material from OER providers and send it to the pre-processing pipeline. When initialized it will crawl OER repositories which are provided in the initial crawling data. The initial data consists of a) OER repository URL, b) material MIME types and c) material extensions. A MIME type is a standardized way to indicate the nature and format of a document [2]. For our project MIME types are used to identify OER materials in the crawling process.

The initial crawling data will be created by OER experts within the project consortium as well as external OER providers who expressed interest in the X5GON project. To acquire this data we prepared a form which is available on the X5GON website [3].

The form consists of OER provider basic information, the technology it uses, and material information such as material formats, MIME types and file extensions. A screenshot of part of the form is shown in Figure 3.


OER repositories Information

Open Education Resources basic information

OER site name *

Website *

http://

Repository URL

http://

The URL to the OER repository - location where the learning materials are stored.

API link

http://

Figure 3: *OER repositories information form*

Format (2). After the crawler retrieves OER materials it sends to the format element which transforms the data into a common format. This element transforms the material data to a JSON object containing the following attributes.

- **title:** the title of the material
- **provider:** the OER provider of the material
- **materialURL:** the URL of the material
- **author:** who created the material *(optional)*
- **created:** when was the material created or published *(optional)*
- **type:** what is material type, i.e. video, audio, presentation, image, text, etc.
- **language:** the language in which it is written, i.e. en, sl, es, etc.
- **metadata:** additional metadata acquired from the material, i.e. Wikipedia concepts, extracted features, etc.

The attributes labelled as optional are not necessarily required. The metadata attribute is a placeholder and will contain material information that will be extracted in the pre-processing pipeline, i.e. type based features extracted by Feature Extractors (3), Wikipedia concepts received by Wikification (4) and data retrieved by Dmoz classification (5).

Feature Extractors (3). After material formatting the next step is extracting features and content from the material. Because OER material are found in different formats and types we need to handle each type separately. To this end, we will develop feature and content extractors for each material type where the output will be raw text. The types we will initially support are video, audio, presentations and text. The approaches used to extract raw text from the material type is shown in Table 3.

OER Material Type	Feature Extraction and Enrichment Approach
video	Translation, Transcription
audio	Speech-to-text
presentation	Text extraction
text	Conversion to raw text

Table 3: *OER material types and target feature extractions & enrichments*

Video and audio materials will be sent through the Translectures service provided by Universitat Politecnica de Valencia which will generate transcriptions for videos and text from audio using speech-to-text technology. To extract features from text and presentation materials we will search for existing libraries and develop new software.

Wikification (4). After feature extraction we will send the materials through the wikification process. This will identify and link textual components to the corresponding Wikipedia pages. We will use Wikifier [4] developed by JSI which also supports cross and multi-linguality enabling extraction and annotation of relevant information from OER material in different languages. This feature allows us to create a common semantic space which will be used for comparing materials, retrieving materials based on user's query and other analysis done within the X5GON project.

DMOZ classification (5). The next step in the pipeline is DMOZ classification. DMOZ now renamed to Curlie is a multilingual open-content directory of Word Wide Web links [4]. It is used to identify in which category and subcategory a given material falls. The number of categories is large, ranging from high-level categories such as Arts, Science, Computers, Games and Sports, to low-level categories like Contests, Home Videos, People and Evolution. The categories are described in [5].

Within the pipeline DMOZ classification will be used to extract in which DMOZ category the material falls. This will give us a hierarchical structure of the OER material which will be used in data presentation as well as analysis done in WP4.

Validation (6). The last step before storing the material data is material validation. In this step we will validate if the JSON representing the OER material contains all of the required data. After the JSON object is validated it is sent to PostgresQL database and made available for analysis and user API described in Section 4.2.2.

Pipeline requirements

- **OER repository URLs, MIME types and file extensions.** To extract materials from OER repositories we need a list of MIME types and file extensions that are associated with OER materials. We will acquire this information using the OER Repositories Information Form found on the X5GON website.
- **Feature and information extraction methods**. To effectively extract material information we need to prepare feature and information extractors that can handle the material types provided in Table 3. Video and audio materials will be sent through Translectures service while for text and presentations we will search for existing libraries and develop new software that solves our problem.

4.2.2 User and Development API

The user and development API are in charge of handling requests and returning the appropriate data. While the user API focuses on storing user activity, providing recommendations and material information to the user, the development API serves as an access point to enriched data stored in the PostgresQL database for the project partners. User API will also serve as a proxy, i.e. redirection, to other services such as the Recommendation Engine developed in WP4 and Learning Analytics Engine developed in WP3. Figure 4 shows the user and development API architecture as well as labels of elements described in the following paragraphs.

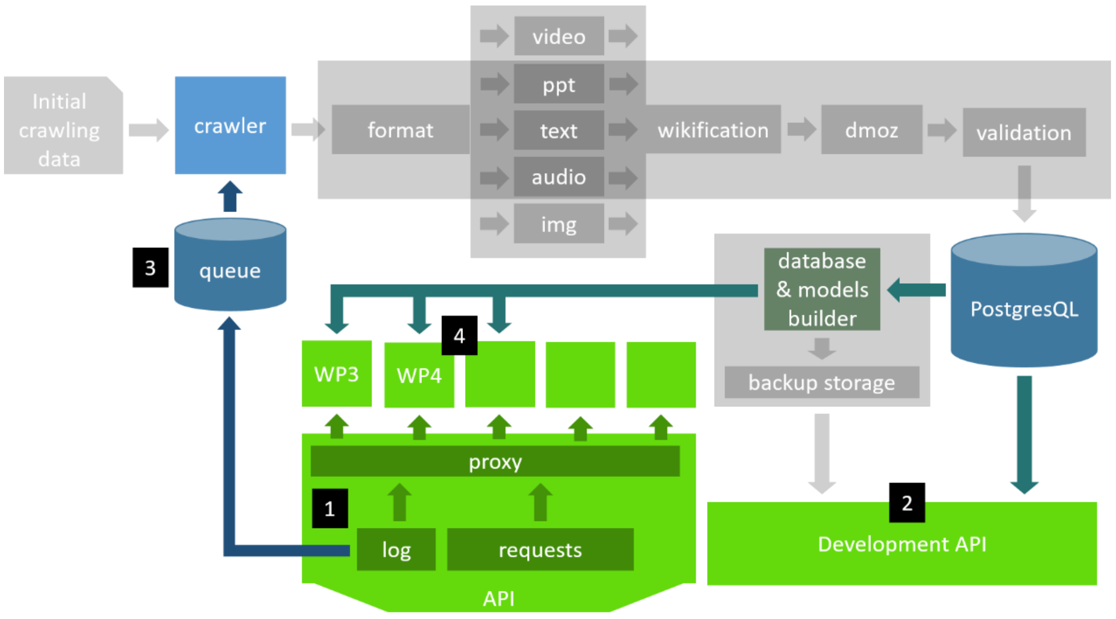


Figure 4: *User and development API*

User API (1). This element will handle requests mainly given by OER providers and their users. It will retrieve material data from the database as well as proxy requests to external services (4). Additionally, the user API will retrieve user activity data generated by the X5GON user activity tracker library (see section 4.2.4). The data will contain the URL of the visited OER material which is used to provide OER material recommendations. If the visited material is not found in the database the API sends the material URL to OER material crawling queue (3) where it waits to be used in the crawling and pre-processing pipeline described in section 4.2.1.

Development API (2). This element is designed to be used by consortium members for retrieving data stored in the database. Each developer will need to provide an identification token, which is used to identify the user and avoid accessing the database by people outside the consortium. The token will also contain information about the access level the user has, i.e. read-only access, editing access, etc.

OER Material Crawling Queue (3). This element's role is to store OER material URLs that were not yet acquired. The URLs are provided by the user API using the user activity data provided by the X5GON user activity tracker library. The crawler will periodically check the crawling queue for new material URLs and take the oldest insert as the material crawling target. Once the material is acquired it is sent through the pre-processing pipeline described in section

External Services (4). External services contain services developed within the X5GON project. Examples of these services are the Learning Analytics and Recommendation Engines developed in WP3 and WP4, respectively. Additionally, unforeseen services developed within the project also fall here. The services will have access to the material and user activity data stored in database.

User and development API requirements

- **Automatic token generation for development API.** For smooth development workflow we need to develop an automatic token generation. This will exclude the human middleman who can slow down the work process.
- **User activity data storing.** User activity data is crucial for the development of Recommendation and Learning Analytics Engines as well as other analysis. To this end, we will implement a mechanism that stores user activity data to the database making it available to the consortium.
- **Crawler queue pipeline implementation.** A complete collection of OER material the users have viewed is required for the development of Recommender and Learning Analytics engines. To this end we need a crawler queue pipeline which stores URLs of OER material that is not in the database. The crawler will then check the queue and acquire the missing materials.
- **Scalable user API.** We expect millions of request per day will be sent to user API once it's in production. To effectively handle this number of requests we will need to implement a scalable API server. This will be done using Node.js [4], a JavaScript runtime built on Chrome's V8 JavaScript engine, which contain features and methods for scaling servers, and the cloud infrastructure provided by Pošta Slovenije.

4.2.3 Backup Mechanism

To have a stable and scalable platform we will need to develop a backup mechanism which periodically creates data dumps, i.e. snapshot of the data available in the database, and save machine learning models used in the platform. The mechanism will contain a storage of data dumps and machine learning models which will distribute the data to multiple instances of the server allowing it to scale and take advantage of the servers processing power.

The storage will be available to the consortium members for data analysis and checking if the machine learning models are working correctly. In the case of data and model corruption we will use backups to revert the platform to the last stable version until we repair the corrupted data and models. This will ensure the platform is still runing while the problems are being solved.

Figure 5 shows the architecture of the backup mechanism as well as labels of elements described in the following paragraphs.

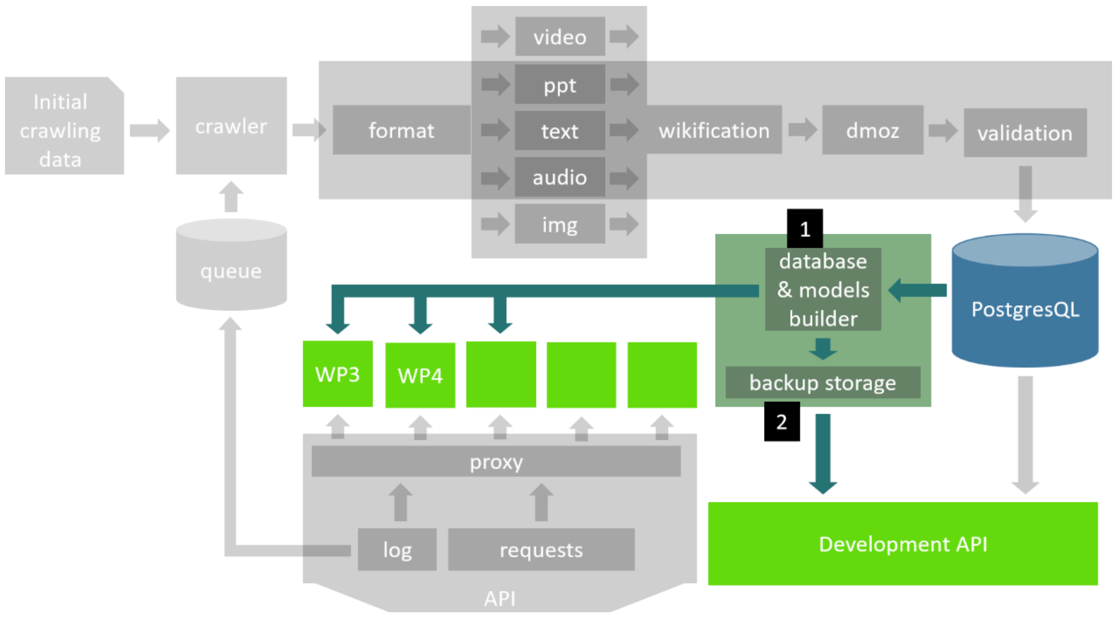


Figure 5: *Backup mechanism*

Database and Models Builder (1). This element will create data dumps and machine learning models using the data present in the database, i.e. OER material and user activity data, and store them in the backup storage (2). The models and data dumps are also sent to services that are developed within the X5GON project.

Backup Storage (2). The backup storage will contain all machine learning models and data dumps generated by the Database and Models Builder (1) element. The data stored will be versioned to describe the state of the platform at different platforms. This will give better understanding of the code evolution. For the purpose of service development and code revisions the storage will be accessible through the development API.

Backup mechanism requirements

- **Cloud based infrastructure.** Within the consortium we will develop different database and model building methods which can quickly become scattered across different machines. To have the whole architecture localized we will need a cloud based infrastructure which will be accessible for every developer. To this end, Pošta Slovenije will provide their cloud infrastructure and allow access to it. There we will store and run the X5GON platform as well as some services needed for the platform to function.

4.2.4 X5GON user activity tracker snippet

For the development of Learning Analytics and Recommendation Engines as well as other unforseen services developed within the project we will develop a user activity tracker library for acquiring information about user activity on OER repositories. The library will be included on the OER repositories and where it will create and add a cookie containing the information associated with the user's visits. The cookie will also be persistant over all OER repositories that included the library allowing us to track the users learning pathway. The cookie will be linked to the domain of the user API to which it will also send the data. The user activity data includes the following attributes:

- **userId:** the identification of the user accessing the OER repository

- **materialURL:** the link to the material the user visited
- **referrerURL:** from which website did the user accessed the material
- **accessDate:** when was the material accessed
- **userAgents:** what technology did the user used to get to the material

Additionally, we will prepare a code snippet, i.e. a script the OER providers would add to their website, which will send the data to the user API. An example of the snippet the OER provider will need to include to their website is shown in Figure 6.

```html
<!-- prerequisite for using the snippet -->
<script type="text/javascript" src="jquery-3.3.1.min.js"></script>
<!-- library that contains the x5gonActivityTracker function -->
<script type="text/javascript" src="x5gon-log.min.js"></script>

<script>
    // snippet code for sending user activity data
    $(document).ready(function () {
        // send user activity to x5gon server
        // the 'x5gonPartnerToken' will be changed for each OER partner
        x5gonActivityTracker('x5gonPartnerToken');
    });
</script>
```

Figure 6: *X5GON user activity tracker library and snippet*

X5GON user activity tracker snippet requirements

- **Integrated X5GON user activity tracker and snippet.** To acquire user activity data we need OER repositories to integrate the X5GON user activity tracker library and snippets into their websites. To this end, we will invite OER providers outside the consortium to be part of the OER Network we are creating within the X5GON project. We will describe the benefits of joining the network and the efforts the providers will need to provide to integrate the snippet.
- **Generate token for each OER repository.** As seen in Figure 6 each OER provider will have its own token used to identify the location of the user activity. To distinguish different repositories we will need to generate a token for each OER repository that included the snippet. This technology will be similar to token generation for development API mentioned in requirements within Section 4.2.2.

5. FUTURE PLANS

In the future we will develop the X5GON platform following the architecture described in section 4. During development process we will address every requirement and use technology and data sources available within the consortium. The source code will be versioned and stored on a Github repository. Parts of the code will be open-sourced, motivating external developers to collaborate and contribute to the project.

The platform will be developed and sent to production on the cloud infrastructure provided by Pošta Slovenije. The infrastructure allows to modify the processing power the platform has available making the platform scalable and sustainable. It will also be maintained and regularly updated with new features and mechanisms described in this report.

6. CONCLUSION

In this report we present the technology and data sources that are available to the consortium at the start of the X5GON project. The technology includes analytics platform, i.e. QMiner, as well as services for processing text, i.e. Enrycher, Wikifier, Xling. The technology will be used for feeding the Recommendation and Learning Analytics Engines developed in WP3 and WP4.

The data source available to the consortium is Videolectures.NET which contain more than 25k videos of lectures, interviews, conferences and events. This data source will be used for creating the initial recommendation and learning analytics models. In the future we plan to add other data sources which in theory could improve the results of the engines.

Additionally, we present the architecture of the X5GON platform in Section 4. The architecture includes:

- A crawler for acquiring new OER materials
- A pre-processing pipeline which for each OER material type extracts textual data which are then projected in a common semantic space using the Wikifier and DMOZ classification services
- An API server which handles user requests and redirects them to the recommendation and learning analytics engines as well as other services developed within the X5GON projects
- A models building and cloud storage component allowing having multiple API servers running and redistributing the user requests shortening the response time

We will also develop and provide a user activity tracker library which will be used to acquire user activity data within the OER repositories. This data will be then fed to the recommendation and learning analytics engines and potentially improve the engines results.

REFERENCES

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

[2] “MIME types - HTTP | MDN,” MDN web docs, [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/HTTP/Basics_of_HTTP/MIME_types. [Accessed 26 03 2018].

[3] “Invitation to connect your OER repository - X5GON,” X5GON, [Online]. Available: https://www.x5gon.org/about/connect/. [Accessed 26 03 2018].

[4] J. Brank, G. Leban and M. Grobelnik, “Annotating Documents with Relevant Wikipedia Concepts,” in *Proceedings of the Slovenian Conference on Data Mining and Data Warehouses (SiKDD 2017)*, Ljubljana, Slovenia, 9 October 2017.

[5] “DMOZ - Wikipedia,” Wikipedia, [Online]. Available: https://en.wikipedia.org/wiki/DMOZ. [Accessed 26 03 2018].

[6] “Curlie - The Collector of URLs,” Curlie.org, [Online]. Available: https://curlie.org/. [Accessed 26 03 2018].

[7] “Node.js,” Node.js Foundation, [Online]. Available: https://nodejs.org/en/. [Accessed 27 03 2018].