Europeana – Core Service Platform

DELIVERABLE

D7.1: Strategic Development Plan, Architectural Planning

Revision	Final
Date of submission	30 October 2015
Author(s)	Marcin Werla, PSNC Pavel Kats, Europeana Foundation
Dissemination Level	Public

Co-financed by the European Union
Connecting Europe Facility

REVISION HISTORY AND STATEMENT OF ORIGINALITY

Revision History

Revision No.	Date	Author	Organisation	Description
1	1 August 2015	Marcin Werla	PSNC	Initial version
2	14 September 2015	Pavel Kats	Europeana Foundation	Revision of initial version
3	30 September 2015	Marcin Werla	PSNC	Revision
4	30 October 2015	Pavel Kats, Marcin Werla	PSNC, Europeana Foundation	Final version

Statement of originality:
This deliverable contains original unpublished work except where clearly indicated otherwise. Acknowledgement of previously published material and of the work of others has been made through appropriate citation, quotation or both.

The sole responsibility of this publication lies with the author. The European Union is not responsible for any use that may be made of the information contained therein.

Europeana DSI is co-financed by the European Union's Connecting Europe Facility

1. Introduction

Europeana is the central access point to tens of millions of records from cultural heritage institutions from all over Europe. The amount of data which is processed, and the amount of sources of this data and finally the heterogeneity of data introduces a number of conceptual and technological challenges which have to be faced using most modern IT solutions. In its operational history technical environment of Europeana had several ups and downs, starting with the overloading of the server which took place just after the official opening of the portal, when the news started to be transmitted be the media. The infrastructure of Europeana is now of course much more advanced and almost incomparable to its initial state from 2008, but the continuous growth of content and interest, new features like faceted search, enrichments, Europeana API and Content Reuse Framework, all the time create new challenges which have to be overcome by technical part of the Europeana team. In such expected and unexpected situations, the key to proper and fast solution is very often dependent on the architecture. Good architecture can save the operational capacity of the portal or ingestion backend, but even small architecture flaws can lead to bottlenecks and small failures which escalate to big disasters.

In order to make sure that the future of technical operations of Europeana is safe and stable, as a part of technical development work in WP7 of Europeana DSI project, architectural planning was made and documented in this deliverable. It starts with the overview of current Europeana architecture in section 2. Next section starts with a brief description of key elements of Europeana Cloud services infrastructure, as it is key environment for the future technical operations of Europeana. It is followed by detailed description of four directions of architectural development which are at the moment seen as the most important. The deliverable ends with a summary which explains how these four directions can be joined in the future to make Europeana systems ready for development of new features and processing of much bigger amounts of data.

The development of Europeana systems is continuous and complex activity and takes into account many factors, including the general development of IT tools and services as well as results of many Europeana-related projects. Therefore, architectural planning presented in this document may become outdated even before all plans described in this document are implemented. To reflect the current state of architectural planning this deliverable may be update throughout the project lifetime and released as subsequent versions.

2. Current architecture overview

Good overview of the current architecture of Europeana systems can be found in "D5.2. Review of Europeana's logical and technical architectures"[1] released in May 2015 under Europeana v.3 project. The Figure 1 below, coming from this deliverable, shows schematic overview of Europeana architecture.

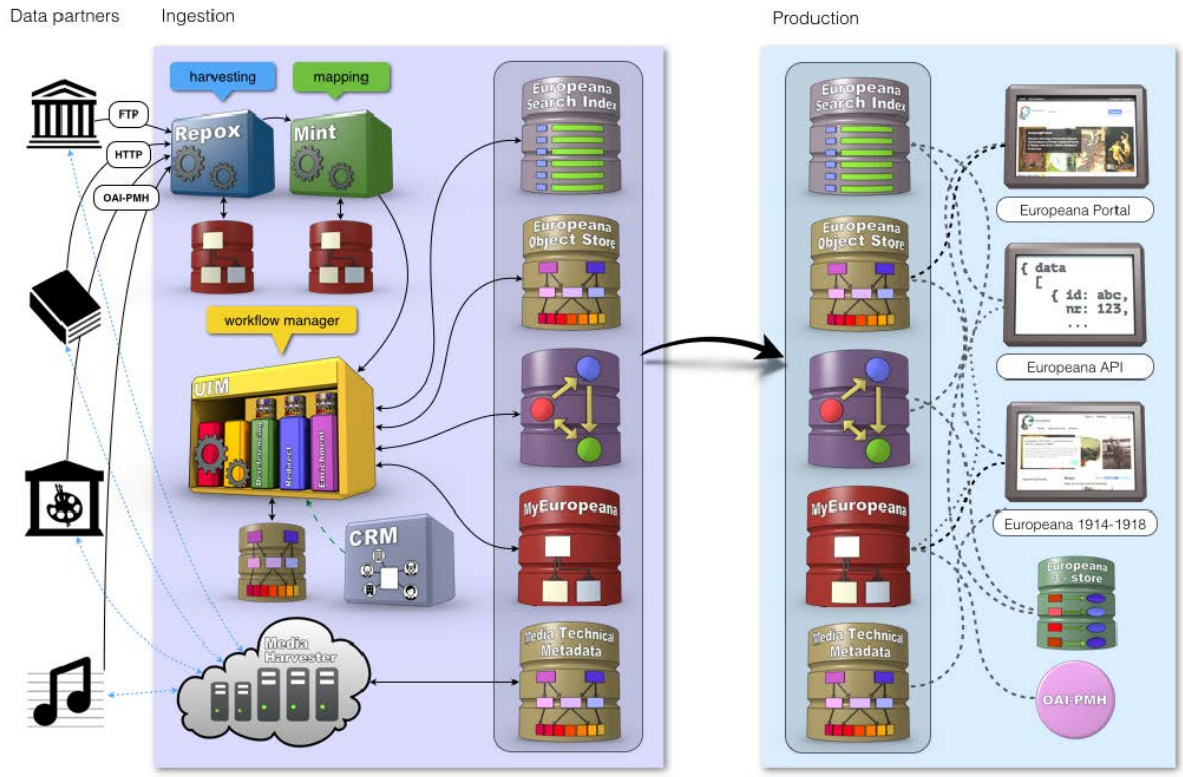

Figure 1. Overview of Europeana architecture [source: Europeana v.3 - D2.5].

The current environment is divided into ingestion backend and production frontend. Key components of ingestion backend include:

- Harvesting subsystem based in REPOX
- Mapping subsystem based in MINT
- Workflow management subsystem based in UIM, connected with SugarCRM data providers' information management
- Media harvesting and analysis subsystem

The above subsystems are responsible for providing Europeana staff with functionality needed to aggregate data from Data Partners, map the data to format proper for further processing, run processing and enrichment workflows and finally store the data in a number of output components:

- Search storage (for indexing and searching data) – based on Solr
- Object storage (for storing and exposing data) – based on MongoDB
- Graph storage (for storing and exploring relations) – based on Neo4j
- Technical metadata storage (for storing media harvester data) – based on MongoDB

1 http://pro.europeana.eu/files/Europeana_Professional/Projects/Project_list/Europeana_Version3/Deliverables/Ev3%20D5_2%20Review%20Logical%20and%20Technical%20Architectures.pdf

Above three storage components are used to validate the aggregated and processed data before it can go live in the production frontend. An internal replica of Europeana portal and API endpoint are used for that.

After the validation is done and data is accepted for publishing, the synchronization process is started. This process, as described in D5.2, was done manually by file-level synchronization of storage components from ingestion to production environments, which required temporary switching off of the synchronized components. To minimize downtime, all storage components in the production environment were replicated so that at least one of them is running when others are synchronized offline.

In 2015 Europeana team took steps to change the synchronization approach from off-line to on-line synchronization. The work is currently (October 2015) being finished and the architecture looks as depicted on Figure 2 below. Main changes are the following:

- UIM workflow management subsystem, extended with dedicated applications took over the task of synchronization of data between backend and frontend systems and does this in an automated way.
- All three storage components (search, object, graph) were reconfigured from “independent replicas + load balancer” approach to proper clustered solutions (Solr Cloud, MongoDB master/slave, Neo4j HA). This allows the modification of production data without the necessity of shutting down of components that are being synchronized.

The new architecture decreases the amount of work that has to be done in order to publish new/updated data. It also increases the availability and robustness of the entire production environment.

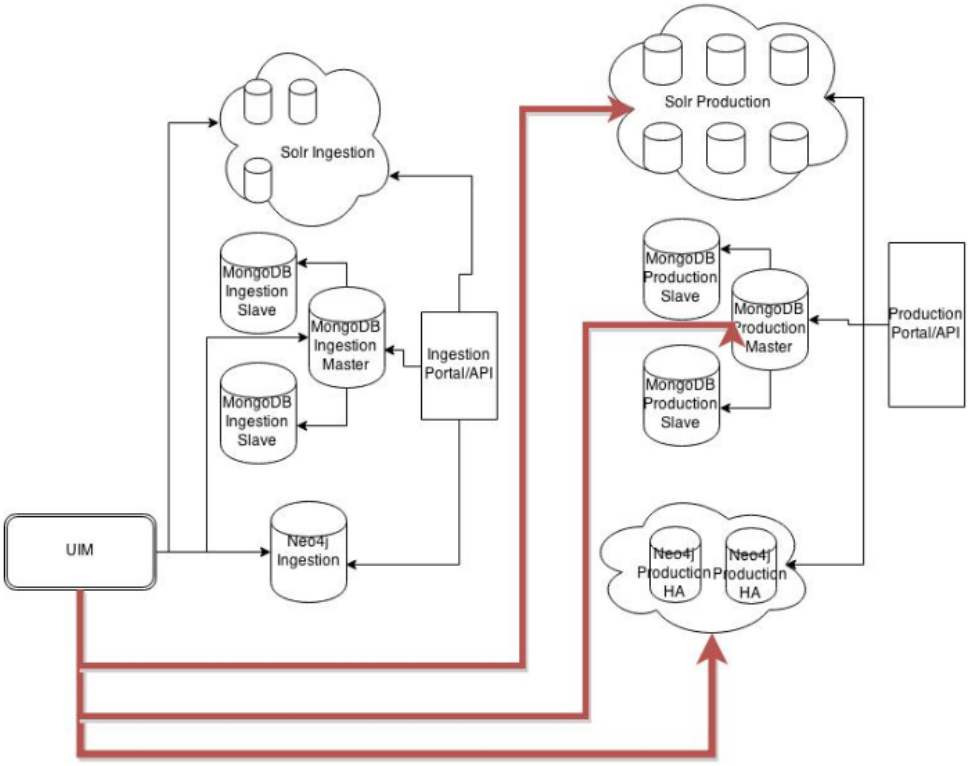

Figure 2. Europeana architecture with the new synchronization approach.

Regarding the further development of Europeana architecture, it should be focused on addressing the following issues:

- Lack of integration with EuropeanaCloud.
- Complex ingestion system which consists of four different applications (Repox, MINT, UIM, SugarCRM) with independent storages.
- Media harvesting subsystem very loosely connected with the rest of system.
- Lack of shared messaging queue/bus for future integrations.
- Lack of centralized logging and auditing facilities.

Architectural planning presented in the following section takes the newest Europeana architecture and provides directions for its development, in the context of issues described above.

3. Architectural planning

The main direction of architectural changes in Europeana infrastructure should be gradual migration towards the usage of Europeana Cloud infrastructure (see deliverables of Europeana Cloud project: D2.2. Europeana Cloud Architectural Design[2] and D2.4. Prototype of Content Cloud[3]), which is designed and developed to be a shared backend infrastructure of European aggregators, including Europeana itself. As depicted on Figure 3 below, Europeana Cloud system provides a number of services which offer following capabilities:

- Registration and resolution of unique identifiers
- Metadata and content storage and access
- Flexible data processing with good scaling capabilities
- Notification about data changes (currently under implementation)
- Data annotations (currently under implementation)

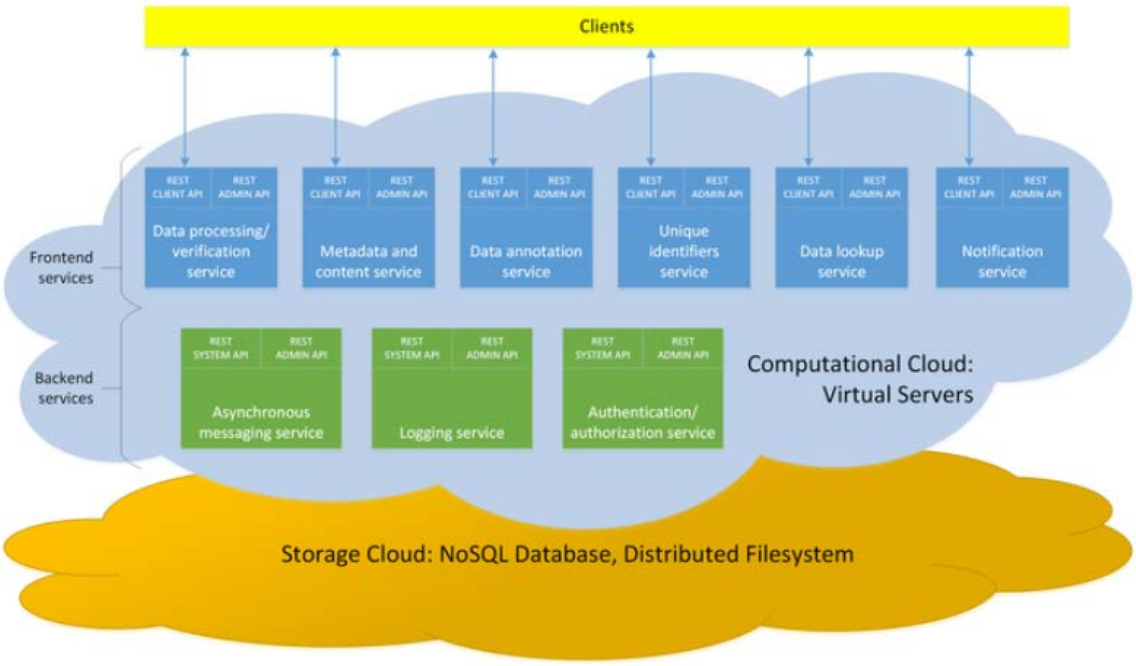

Figure 3. Europeana Cloud architecture (source: Europeana Cloud D2.4).

From the point of view of present Europeana architecture, Europeana Cloud can provide first of all shared data storage and data processing components. This functionality, adopted gradually, can be very useful to address other issues listed at the end of the previous section. In order to do that, the several directions of architectural developments are proposed below.

2 http://pro.europeana.eu/files/Europeana_Professional/Projects/Project_list/Europeana_Cloud/Deliverables/D2.2%20Europeana%20Cloud%20Architectural%20Design.pdf

3 http://pro.europeana.eu/files/Europeana_Professional/Projects/Project_list/Europeana_Cloud/Deliverables/D2.4D2.5PrototypeofMetadataandContentCloud.pdf

Direction 1: Reconstruction of complex ingestion system

Current ingestion system of Europeana is composed of several tools which were developed and somehow integrated on the previous stages of development of Europeana system. These tools have sometimes partially overlapping functionality (e.g. both MINT and REPOX are able to do mapping and harvesting), are based on different technology stacks (Java, PHP) and have separate relational databases. Proper maintenance and development of such complex combination is challenging and hard to optimize.

The UIM component, responsible for Europeana ingestion workflow management, is natural candidate for taking the lead role and becoming the ultimate tool for Europeana team. It is already visible in the recent architectural changes, which are putting UIM as the component responsible for managing of synchronization of data between ingestion and publication environments.

The reconstruction of ingestion environment should utilize the architectural concept of microservices[4] and related architecture design patterns[5]. This concept was introduced to allow splitting monolithic and inflexible systems to more modular and lightweight ones. It also significantly supports deduplication of services functionality, as each core function can be run as separate microservice, which can be reused by many tools. So decomposition of best features of currently used tools into microservices can lead to creation of much better ingestion platform than usage of few loosely connected bigger tools. Another benefit is that each service can then be adapted and redeployed separately, and can even have different technology stack (if justified), as long as communication interfaces are agreed. Major challenges which have to be taken into account in this context are: right level of services granularity, proper distribution and monitoring of services, consistent error handling (when referenced microservice/s will be down), consistent transaction handling in the cross-service context, shared storage or other way of data synchronization and consistency between services.

The microservices approach should result in the future UIM being a central point responsible for orchestration of all ingestion related operations. The Metadata and Content Service provided by Europeana Cloud should be used as the shared storage allowing UIM and all microservices to easily exchange and process data without the necessity of copying it between different storage systems. Microservices responsible for data processing (mapping, enrichments) should be implemented as plugins to Europeana Cloud Data Processing Service, allowing to scale the processing operations easily.

Current object storage of Europeana, based on Mongo DB should be as a result replaced with Europeana Cloud Metadata and Content Service. Depending on the directions of development of Data Annotation Service, the graph storage of Europeana could be also possibly replaced with Europeana Cloud functionality.

Direction 2: Integration of media harvesting system

The media harvesting subsystem which is now very loosely connected with the rest of system should be as well integrated into the UIM+microservices environment. Separate MongoDB which is now used as a storage of technical metadata should be replaced with proper usage of EuropeanaC loud Metadata and Content Service and the media harvesting and analysis process should be deployed within Data Processing Service to have homogeneous processing environment.

[4] http://martinfowler.com/articles/microservices.html
[5] http://microservices.io/patterns/index.html

The two steps described above are in fact very complex operation which will also result in changes in the Europeana portal and API, at least in the part which use object storage, as it will be using Metadata and Content Service instead.

Two more architectural planning directions presented below have more backend nature and are not specifically connected to the nature of Europeana functionality. They are rather good practices which appear often in complex distributed system and could be also introduced in Europeana system to increase its flexibility, extensibility and security.

Direction 3: Introduction of shared messaging queue

Europeana system is a complex and distributed environment and in such systems very common component is shared messaging queue. Such queue is used to send messages about activities undertaken by specific component. The component which is doing some activity is also sending a message to the queue and other components may (if they want) receive this message and react to it. The communication is asynchronous – the sender is not waiting for the receiver to confirm that the message was read – and receivers can be added and removed in runtime. So in fact the sender does not have to be aware who will get the message. Good scenario describing usefulness of the shared queue in the context of Europeana is the following:

1. Harvesting microservice delivers new metadata record to shared storage (Metadata and Content Service) and sends a “New record with id X” message about that fact to the queue.
2. Enrichment microservice gets the message and reacts to it by doing some enrichments on the new data record.
3. Later on, new microservice for media harvesting and analysis is added. This microservice also begins to listen and react to “New record…” messages. This microservice can be plugged in into the system without any additional modifications of existing services (if they are already generating proper messages).

To implement this approach, not only a setup of shared messaging queue is required but also each component of Europeana systems should be reviewed in the context of:

- the list of messages that this component should generate – messages which will be useful already and may be useful in the future;
- the communication with other already existing components – if it can be transformed from synchronous, based on direct calls, to asynchronous, based on messages.

Using queues for message passing and asynchronous communication in general is also very good way to ensure scalability of the system and avoid bottlenecks. For example, in the scenario above, if the enrichment microservice will be overloaded it will not block the harvesting process. Instead, messages will be buffered in the queue until the time when they can be processed. If more processing power will be needed, then more enrichment microservices can be instantiated and connected to the same queue to start parallel processing. With this approach and proper amount of resources even near real-time processing performance can be achieved. Of course the key is the stability and performance of the queueing system, but there are several systems like this (e.g. Apache Kafka) that proved their reliability in several big data processing environments.

Direction 4: Introduction of centralized logging and auditing

Shared messaging system described above can be also utilized to introduce centralized logging, which is very good way to achieve good overview of all components of complex and distributed systems. Open source tools like Logstash, Ganglia and Kibana can be used to

aggregate, process and visualize log data. There are also possibilities to utilize machine learning approaches to provide smart automated monitoring of such systems and create error detection mechanisms which are even able to generate warnings before regular monitoring alerts will be triggered.

Europeana portal and API can be also connected to the centralized logging. This should allow to gather all usage data in one place and process it in order to provide analytical view on how resources available in Europeana are accessed and what are the paths of users which are exploring the rich content of Europeana.

4. Summary

Four directions of architectural planning described in this deliverable are indication of the most important aspects of technological developments of Europeana system in the context of coming months. The major challenge will be for sure the migration to Europeana Cloud which should be connected with reconstruction and deep optimization of the ingestion environment. This process should be divided into several steps:

- Identify all functionality that is needed in the ingestion process, on the path from aggregation to production (including the media harvesting part).
- Decide which functionality can be extracted as microservices, which functionality can be completely automated and which requires human interaction or supervision (which means that user interface is necessary).
- Design workflows, data flows and interactions between microservices and the coordination subsystem ("new UIM").
- Plan which microservices can be extracted from existing tools and services and which have to be rewritten from scratch, also in the context of functionality provided by Europeana Cloud.
- Design interfaces (APIs) of microservices, ideally in a unified way[6].
- Implement the coordination subsystem and microservices (on the basis of Europeana Cloud where possible).

As the re-implementation of ingestion system will most probably take several months, it has to be done in parallel with the operations of currently existing ingestion backend. The planning above covers two first directions described in section 3. Direction 3, introduction of shared message queue, should be done in the beginning of these works, so that all new components and microservices can utilize it. Direction 4, centralized logging and auditing, can be also started "by the way", as it will be relatively easy to add logging via messaging during the major changes described above.

[6] E.g. http://more.locloud.eu/hackathon/index.php?op=vre_api