Control Interface for the Digital Memory Platform

June-August 2021

AUTHOR:
Marco Donadoni

SUPERVISORS:
Jean-Yves Le Meur
Antonio Vivace

ABSTRACT

CERN produces a huge amount of data, such as experimental results, digital documents and multimedia content, which needs to be stored, archived and preserved for many years to come. For this reason, the Digital Memory project was started, with the goal of guaranteeing the long-term storage and preservation of content from various existing information systems, but also to be able to collect missing content that is currently not archived anywhere.

To reach this goal, the Digital Memory Platform is currently under development. It is an OAIS-compliant digital archive that will manage all the processes needed to harvest, ingest and preserve all the content that needs to be archived, in a single standardized way. This platform will be connected to the existing information systems, so that researchers at CERN can easily use it to archive all their resources.

This report describes the development of the Control Interface, a dashboard used to manage the Digital Memory Platform. It can be used to request the archival of a resource coming from one of the already existing information systems, to approve or reject such requests and to control the progress and the outcome of the various archival processes. To handle these features, we also developed part of the platform itself.

TABLE OF CONTENTS

1. INTRODUCTION

As part of its research efforts, CERN generates a very large amount of data that needs to be stored, archived and preserved so that it can be accessed in the future. That includes raw data measured by the different experiments, digital documents of various types such as published papers and reports, but also multimedia content like recorded lectures and meetings.

At the moment, this content is maintained and made accessible by different information systems, for example CDS (CERN Document Server), Zenodo and CERN Open Data, but none of them are OAIS-compliant. In addition, there already have been instances of lost data at CERN, due to hardware failure, unreadable data formats or because the data was simply never archived in any information system [1].

In 2016, the Digital Memory project was started, with the goal of preventing the loss of valuable data [2]. One of the main activities of the project is to develop a single OAIS-compliant digital archive, the *Digital Memory Platform*, in order to make sure that all the data that needs to be preserved is kept in a single, standardized, long-term archival service. This platform needs to be connected to the existing information systems, in order to be able to harvest and ingest the data already stored in them [3] .

a. OPEN ARCHIVAL INFORMATION SYSTEM

The *Open Archival Information System* (OAIS) reference model is a standard framework that describes the processes needed by an organization to preserve digital information and make it available to the community.

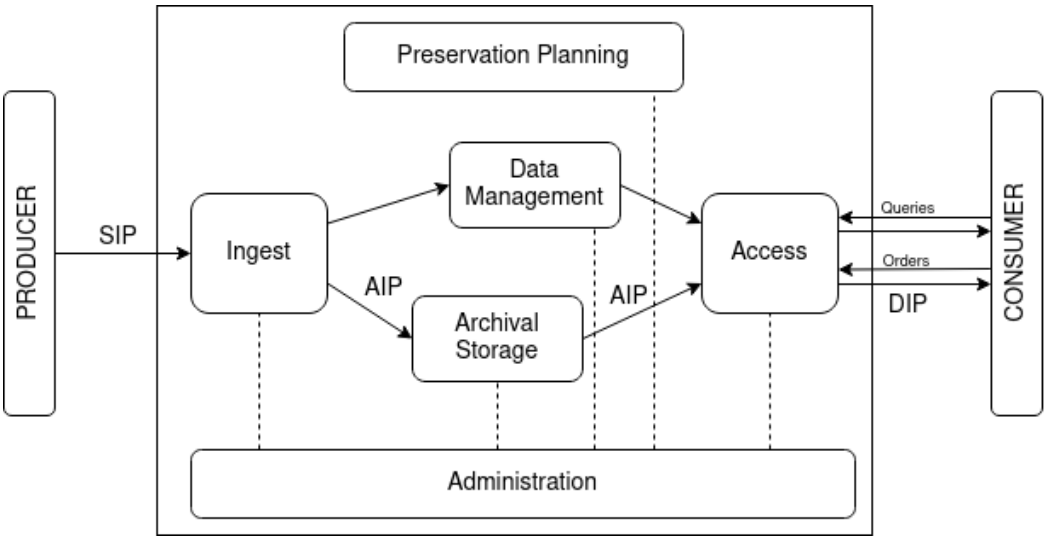

Figure 1. Open Archival Information System reference model

There are four cooperating actors in the OAIS model: producers of information, consumers, management and the archive itself. These actors use *Information Packages* to interact with each other. In particular, there are three types of information packages:

- **SIP (Submission Information Package)** is used by a producer to submit information to the archive.

- **AIP (Archival Information Package)** is used by the archive to store the data submitted by the user, along with its metadata.
- **DIP (Dissemination Information Package)** is derived from the AIP and is given to a consumer after an access request.

b. DIGITAL MEMORY PLATFORM

The Digital Memory Platform needs to be able to handle all of the following operations:

Harvesting	The system needs to be able to harvest the data and metadata of a specified resource from any of the existing information systems. This creates an archive composed of a hierarchy of directories, following the BagIt specification, which contains all the requested data and metadata in a standardized structure. This is handled by the bagit-create tool, which creates a SIP (Submission Information Package).
Ingestion	The system needs to be able to ingest an archive in the BagIt format. This archive can either be generated by the system itself by harvesting an external source, or it can be directly provided by the external information system. During the ingestion, the data will be normalized in order to be easily accessible and readable in the future. This creates an AIP (Archival Information Package).
Long term storage	After an archive is ingested, it needs to be stored in a long-term storage. This can be, for example, the CERN Tape Archive (CTA) or an external service [4].
Access to the archived content	Once archives are pushed to the long-term storage facilities, they must be made accessible to the users. This can be done by developing an ad-hoc solution or by exploiting existing systems like InvenioRDM.

Many different components will compose the Digital Memory Platform, but the ones we are interested in for the summer programme are:

Platform API	The core component of the system that controls and orchestrates all the other parts. It manages all the different operations supported by the digital archive. It exposes an API that can be used to control the platform and that will be also used by the Control Interface.
Control Interface	Web application used by the users to trigger archival requests, to approve them and to control the jobs running and the status of the platform.

During the summer programme we worked on both the Control Interface and the Platform API, focusing only on the harvesting of the various resources.

2. PLATFORM API

The Platform API [5] is the central component of the Digital Memory Platform. This component is very important because it will manage and orchestrate all the other components of the system. It exposes a REST API that is used to interact with the system.

It has been developed in Python using these libraries and frameworks:

Django	Web framework used to build web applications. It provides many features ready to be used out-of-the-box, including authentication, authorization, persistent storage and routing.
Django REST Framework	Toolkit that expands the set of functionalities provided by Django when dealing with Web APIs. It makes the development of the REST API much easier and faster.
mozilla-django-oidc	Library used to manage the login using CERN credentials, exploiting the OpenID Connect protocol.
bagit-create	Library developed at CERN, used to harvest the records coming from the various external sources.
Celery	Task queue used to distribute tasks to be run by different workers. Whenever a user requests the archival of a specified record, a new task that handles the harvesting of the record is created.
Requests	Library used to interact with the APIs of the various information systems, in order to search records and to gather information about them.

3. CONTROL INTERFACE

The control interface [6] is a web interface accessible through a browser. It is implemented as a Single Page Application using the following tools:

React	Framework used to build the user interface. The other teams in the Digital Repository section also use this framework.
React Router	Library to be used together with React to manage the routing to the different pages of the application.
Axios	Library to make HTTP requests, used to interact with the REST API.
Bootstrap	Framework used to style the user interface.

4. FEATURES

In this section, we describe the main features developed during the summer programme.

a. USER AUTHENTICATION

Before accessing and using the system, the user must be authenticated. We support two kinds of authentication, by using a username and password and by logging in with CERN credentials. The first one is supposed to be used only for special local accounts or for testing, otherwise everybody must use the CERN credentials.

We use the OpenID Connect protocol to login using CERN credentials. All the authentication flow is handled by the mozilla-django-oidc library, which automatically manages the users in the database, creating and updating them as necessary.

b. RECORDS SEARCH

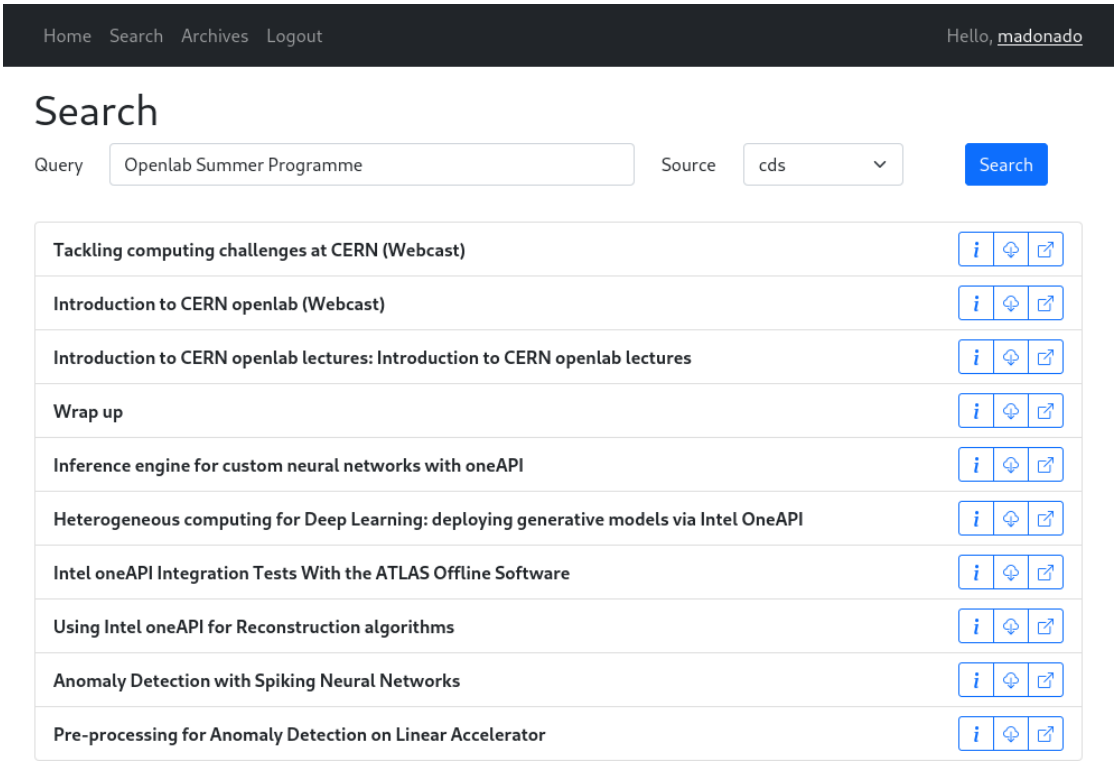

Figure 2. Records Search

From this page, the user can search for specific records, for example documents or video recordings, on some information systems. Right now, we only support CDS, but other information systems will be added in the future.

After searching and finding a specific record, the user can see its details and he can access the resource directly from the external information system. The user can also request the harvesting of a specific record: this creates a new archival request, which needs to be approved before actually being worked on.

c. ARCHIVAL REQUESTS

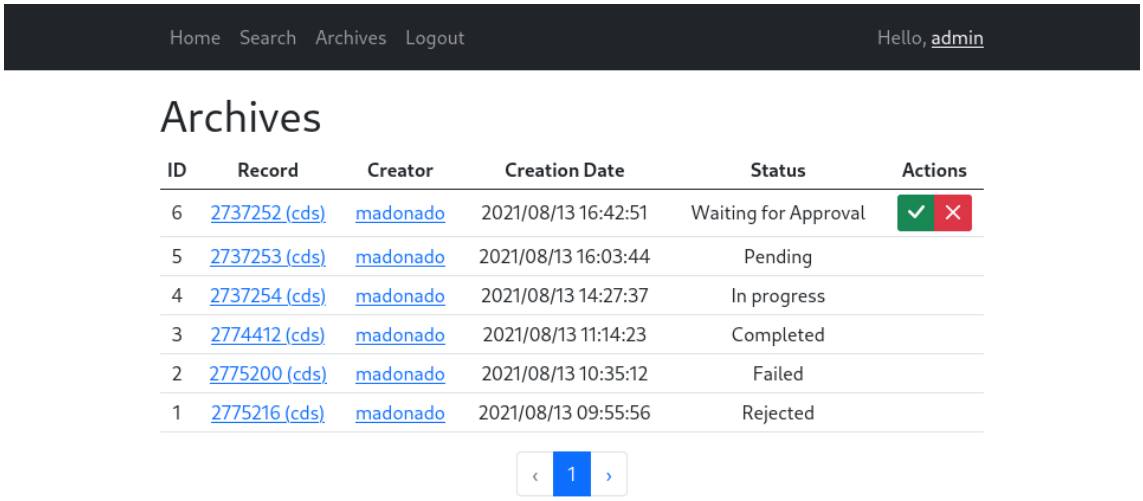

Figure 3. Archival Requests

From this page, the user can see all the archival requests made to the system.

After a user requests the creation of a new archive, a privileged user can either approve or reject the request. If the request is accepted, a new celery task is spawned. The job of this task is to run the bagit-create tool in order to harvest all the data and metadata associated to the record. In the future, this task will also have to start the ingestion process to store the data in the digital archive.

Each archival request can be in one of these states:

Waiting for Approval	The archival request needs to be approved before the system can start harvesting the specified record.
Rejected	The archival request has been rejected, so the record will not be harvested.
Pending	The archival request has been approved. A new Celery task to harvest the specified record has been created and it is now waiting to be scheduled on a free worker.
In Progress	The harvesting task is currently running on a worker of the system.
Completed	The harvesting task has successfully harvested the specified record.
Failed	Some error occurred during the execution of the harvesting task.

d. PERMISSIONS

Django has a built-in permission system, which we use to limit the actions a user can make. In particular, there are three custom permissions currently in use:

`can_access_all_archives`	The user can access all the archives, even those created by other users.
`can_reject_archive`	The user can reject archival requests he can access.
`can_approve_archive`	The user can approve archival requests he can access.

These permissions can be independently granted to each user. In particular, we can leverage the role based permissions system provided by the CERN authorization service. From its web interface, we can define roles that are linked to some of the existing e-groups used at CERN. Whenever a user logs in using the CERN credentials, the Platform API receives some information about the user (e.g. name, surname, email), including the user's roles. Based on these roles, the application can decide which permissions to grant to the user.

5. DEPLOYMENT

The Digital Memory Platform is hosted on OpenShift, a "Platform-as-a-Service" cloud computing service that provides an easy way to deploy multiple Docker containers, which are orchestrated by Kubernetes. The deployment configuration is described entirely by YAML files that define the various OpenShift or Kubernetes objects needed to run the application. They describe, for example, the storage volumes needed, which containers need to be executed and how they communicate with each other. These configuration files are stored in a Git repository [7], so that they can be easily shared, modified, but also rolled back to a previous version when some changes cause problems to the deployed application.

The Digital Memory Platform is deployed continuously and automatically each time the source code or the deployment configuration is changed. Thanks to the CI/CD pipelines of GitLab, after each push on one of the git repositories, the Docker image of the application is automatically rebuilt, it is published on the registry and the OpenShift configuration is updated.

6. CONCLUSIONS AND FUTURE WORK

During the nine-weeks of the summer programme we successfully developed a prototype of the Platform API and its Control Interface with many features, including user authentication and authorization, the interaction with existing information systems and the handling and approval of archival requests. I want to thank my supervisors Jean-Yves Le Meur and Antonio Vivace for giving me the possibility to work on this project and for their support and feedback during my summer programme.

This is just a starting point for the Digital Memory Platform and there are still several features that need to be added. First of all, the platform only supports the harvesting and not the ingestion and the long-term storage of the archives. In the future, users will also need to be able to access the archived content. Considering the features we implemented, many tweaks would make the user experience better, for example adding the possibility to filter the list of archival requests or the search results.

7. REFERENCES

[1] Jorik van Kemenade. *The CERN Digital Memory Platform: Building a CERN scale OAIS compliant Archival Service*. https://cds.cern.ch/record/2728246

[2] Jean-Yves Le Meur and Nicola Tarocco. *The obsolescence of information and information systems: CERN Digital Memory project*. https://cds.cern.ch/record/2649765

[3] CERN Digital Memory. *Digital preservation at CERN*. https://digital-memory-project.web.cern.ch/preserving

[4] CERN Digital Memory. *Archiver Project, Technical Summary*. https://www.archiver-project.eu/deployment-scenarios-technical-summaries/cern-digital-memory

[5] Repository of the Platform API. https://gitlab.cern.ch/digitalmemory/oais-platform

[6] Repository of the Control Interface. https://gitlab.cern.ch/digitalmemory/oais-web

[7] Repository of the OpenShift configuration files. https://gitlab.cern.ch/digitalmemory/openshift-deploy

8. APPENDIX

a. REST API ENDPOINTS

POST	`/login/`	Login using local accounts
GET	`/oidc/authenticate/`	Login using CERN credentials
POST	`/logout/`	Logout
GET	`/users/`	List of users
GET	`/users/<id>/`	Details of the user with given id
GET	`/users/<id>/archives/`	List of archives requested by the user with given id
GET	`/me/`	Details of the currently logged in user
GET	`/records/`	List of records
GET	`/records/<id>/`	Details of the record with given id
GET	`/records/<id>/archives/`	List of archives requested for the record with given id
GET	`/archives/`	List of archives requested
GET	`/archives/<id>/`	Details about the archive with given id
POST	`/archives/<id>/actions/approve/`	Approve the archival request with given id
POST	`/archives/<id>/actions/reject/`	Reject the archival request with given id
POST	`/harvest/<recid>/<source>/`	Request the archival of the specified record
GET	`/search/<source>/?q=<query>`	Search among the records of a given source