Cultural Heritage Preservation and Analysis

COURSE OF BIG DATA TECHNOLOGIES – 2024/2025

GROUP 8

Silvia Bortoluzzi email: silvia.bortoluzzi@studenti.unitn.it
Diego Conti email: diego.conti@studenti.unitn.it
Sara Lammouchi email: sara.lammouchi@studenti.unitn.it

Date of Submission: 17.07.2025

ABSTRACT

AIM:

Our project delivers a scalable big data platform to help museums, libraries, and heritage sites digitize, store, and analyze large collections of artifacts, images, and documents.
The system supports automated ingestion, performs advanced analytics (including deduplication and recommendations), and enables easy exploration and retrieval of cultural content.

KEY OBJECTIVES:

- Scalable ingestion and storage of multimodal cultural data (images, metadata, user comments)
- Automated data cleaning and deduplication
- Semantic enrichment and recommendation using vector search (Qdrant)
- Development of an interactive dashboard for search, recommendation, and data exploration

MAIN RESULTS ACHIEVED:

- Implemented a complete data pipeline from ingestion (Kafka) to serving (PostgreSQL, Streamlit dashboard)
- Ingested and processed thousands of Europeana records and user-generated annotations, and successfully joined them
- Achieved automated deduplication and semantic enrichment of collections using CLIP embeddings and Qdrant vector search
- Delivered an interactive dashboard for search, recommendation, and data exploration through Streamlit
- Demonstrated scalable, modular architecture deployable with Docker

PROBLEM STATEMENT AND MOTIVATION

THE PROBLEM

- **Manual digitization is slow and error-prone:** Limits how much can be digitized and introduces mistakes.
- **Scalability challenges:** hard to handle massive volume of high-resolution images, documents, and metadata being generated.
- **Physical artifacts are deteriorating:** There is a risk of permanent loss unless efficient digitization occurs.
- **Limited collection visibility:** Heritage sites struggle to present and connect their collections in ways that captivate, educate, and engage the public

WHY IS THIS IMPORTANT?

- Implements large-scale digitization for heritage institutions
- Enables research through accessible, high-quality digital collections
- Preserves artifacts before they are lost to time or damage
- Engages the public with better discovery and interactive exploration

DATA EXPLORATION INSIGHTS

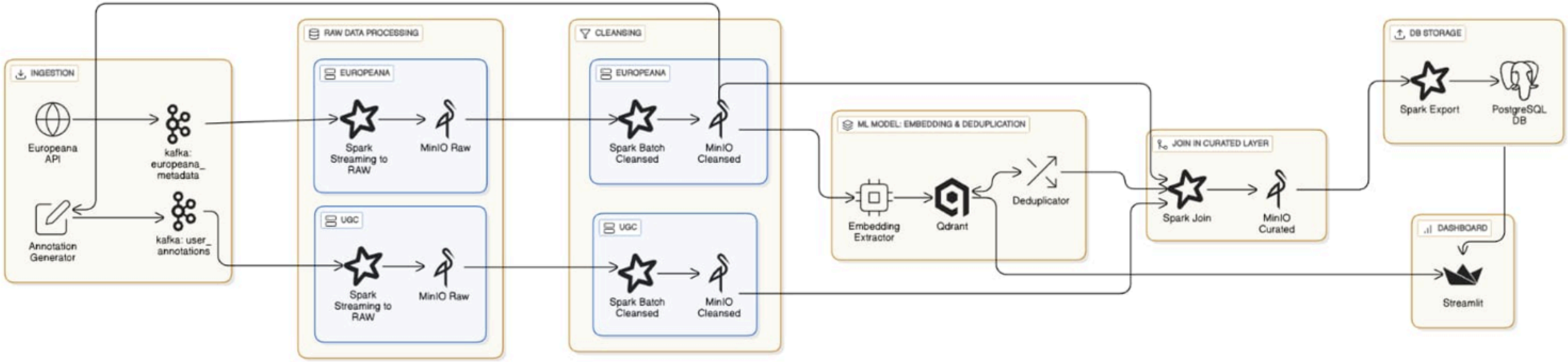
OVERVIEW TABLE

Layer	Data	Variables (Fields)	Description
Raw	**Europeana**	title,guid,image_url,timestamp_created,provider,description, creator, subject,language,type,format,rights, dataProvider,isShownAt,edm_rights	Original metadata as received from Europeana; may include incomplete or inconsistent records.
Raw	**Annotation Producer**	guid, user_id, tags, comment, timestamp, location, ingestion_time, source	Simulated user annotations; linked to Europeana objects via guid.
Cleansed	**Europeana**	title,guid,image_url,timestamp_created,provider,description, creator, subject,language,type,format,rights, dataProvider,isShownAt,edm_rights	Validated and cleaned metadata; normalized fields; only reliable records are kept.
Cleansed	**Annotation Producer**	guid, user_id, tags, comment, timestamp, location, ingestion_time, source, timestamp_cleansed	Cleaned and deduplicated user annotations
ML	**Qdrant Points (Embeddings)**	vectors: vector.image (512D), vector.combined (1024D), payload: guid, status, processed_at, canonical_id	Images and texts embeddings for deduplication and semantic recommendations
Curated	**PostgreSQL tables (Joined Metadata)**	title, guid, image_url, timestamp_created, provider,description, creator, subject, language, type, format, rights, dataProvider, isShownAt, edm_rights, guid, user_id, tags, comment, timestamp, location, ingestion_time, source, timestamp_cleansed	Final flat table joining user annotations and Europeana metadata (deduplicated), ready for SQL queries and dashboard.

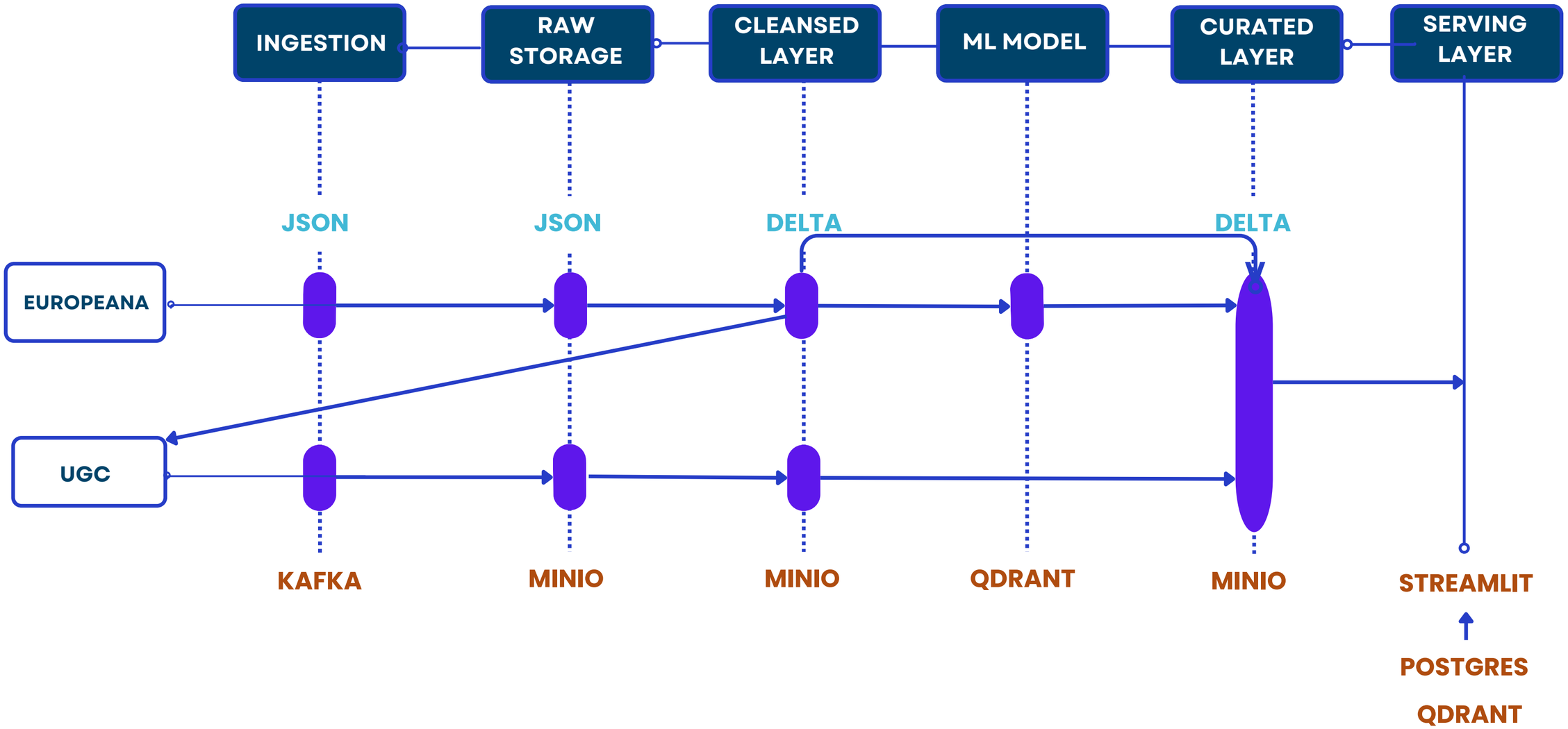
DATA HIGHLIGHTS OF A SAMPLE RUN

- **TOTAL OBJECTS COLLECTED:** around 5000 images from 17 providers
- **PROVIDER DISTRIBUTION:** Maximum 400 images per provider
- **DEDUPLICATION**:
  - threshold 95%
  - 84.12 % unique images
  - 15.88% duplicates detected
- **METADATA**:
  - title, guid, image URL, provider, type, rights: 0% missing
  - Missing metadata: Description(34.4%), Creator(43.7%), Subject(100%), Language(62.8%), Format(100%), isShownAt (reference URL) (22.2%), edm_rights (100%)

SYSTEM ARCHITECTURE



SYSTEM ARCHITECTURE: HOW DATA FLOWS



TECHNOLOGIES AND JUSTIFICATION

Kafka

Aim: Scalable data ingestion

Role: Ingests Europeana metadata and user annotations; buffers, partitions, and distributes data streams to processing components

MinIO

Aim: Distributed raw data storage

Role: Stores as a data lake Lake all ingested data (JSON, metadata, ugc) in a 3-layer Delta Lake Architecture (raw, cleansed, curated); provides S3 interface for Spark and other tools

Spark

Aim: Parallel data processing & cleansing

Role: Cleanses, deduplicates, and transforms ingested data in batch and streaming; maintains reliable ACID tables for downstream ML and serving

CLIP -OpenAi

Aim: Semantic enrichment & deduplication

Role: Generates image embeddings (for deduplication & recommendations) and text embeddings (for recommendations)

Qdrant

Aim: Fast vector search
and recommendations

Role: Indexes embeddings from CLIP; used for image deduplication and for powering semantic search and recommendations in Streamlit

PostgreSQL

Aim: Curated data serving & querying

Role: Hosts the final joined and deduplicated collections, supporting flexible queries and integration with the dashboard

Streamlit

Aim: User-facing analytics & exploration

Role: Provides an interactive web dashboard for searching, visualizing, and recommendations

Docker Compose

Aim: Modular, reproducible deployment

Role: Orchestrates and isolates all services (Spark, MinIO, Qdrant, dashboard, etc.) for easy local or cloud deployment

IMPLEMENTATION AND CODE REPOSITORY

Code Structure

```
Cultural-heritage-bigdata-project/
│
├── config/ #Configuration files and setup scripts for services
│   ├── kafka/
│   ├── minio/
│   ├── postgres/
│   └── README.md
│
├── kafka-producers/ #Ingestion scripts for Europeana data and simulated user annotations
│   ├── annotation-producer/
│   ├── europeana-ingestion/
│   └── README.md
│
├── ML-model/ #Embedding extraction (image/text) and semantic deduplication with Qdrant
│   ├── embeddings-extractor/
│   ├── qdrant-deduplicator/
│   └── README.md
│
├── spark-apps/  #Spark jobs for ingestion, data cleansing, joining, and serving
│   ├── curated-to-postgres/
│   ├── eu-to-cleansed/
│   ├── eu-to-raw/
│   ├── join-eu-ugc-qdrant-to-curated/
│   ├── ugc-to-cleansed/
│   ├── ugc-to-raw/
│   └── README.md
│
├── streamlit/ #Streamlit dashboard for exploring, filtering, and visualizing data and recommendations
│   ├── app/
│   └── README.md
│
├── .gitignore
├── docker-compose.yml
└── README.md
```

GitHub Link: https://github.com/diecccoo/Cultural-heritage-bigdata-project

How to run the project

PREREQUISITES:

- Docker (and Docker Compose) installed on your machine
- **.env** file placed in the kafka-producers/europeana-ingestion/ folder
- 12 GB RAM
- ≥10 CPU Cores

LAUNCHING:

- Clone the repo: git clone **https://github.com/diecccoo/Cultural-heritage-bigdata-project**

- Build & launch: Once inside the Cultural-heritage-bigdata-project directory run: **docker compose up --build –d**

  This will start up all containers
- Wait for services to initialize.

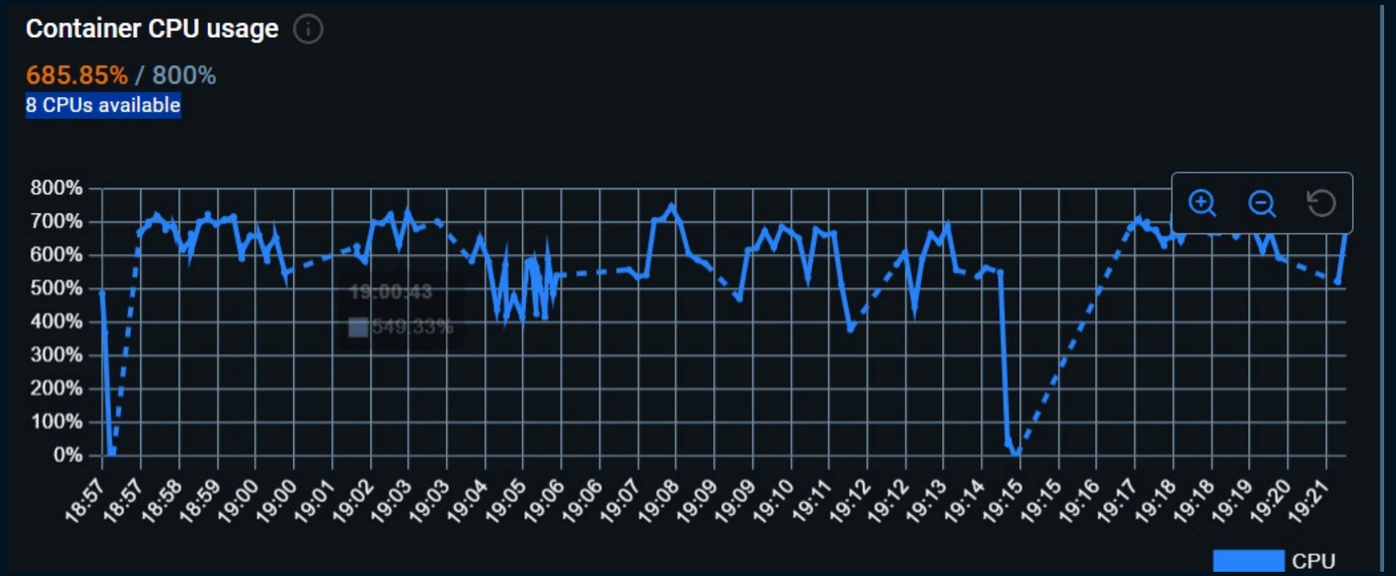
- Ingested data will be processed and available in the dashboard.
- Access the dashboard: Go to **http://localhost:8501** in your browser

  - Notes: See full setup and usage in README.md on GitHub!

RESULTS AND PERFORMANCES

CPU USAGE

- Initial spike in CPU utilization occurs during the system's startup phase, when multiple containers and services are launched in parallel.
- During most of the period, CPU usage fluctuates between ~400% and ~700%.
- In earlier tests, too much CPU usage sometimes caused crashes, so we added an environment variable called SLEEP to slow down data generation and keep things more stable.



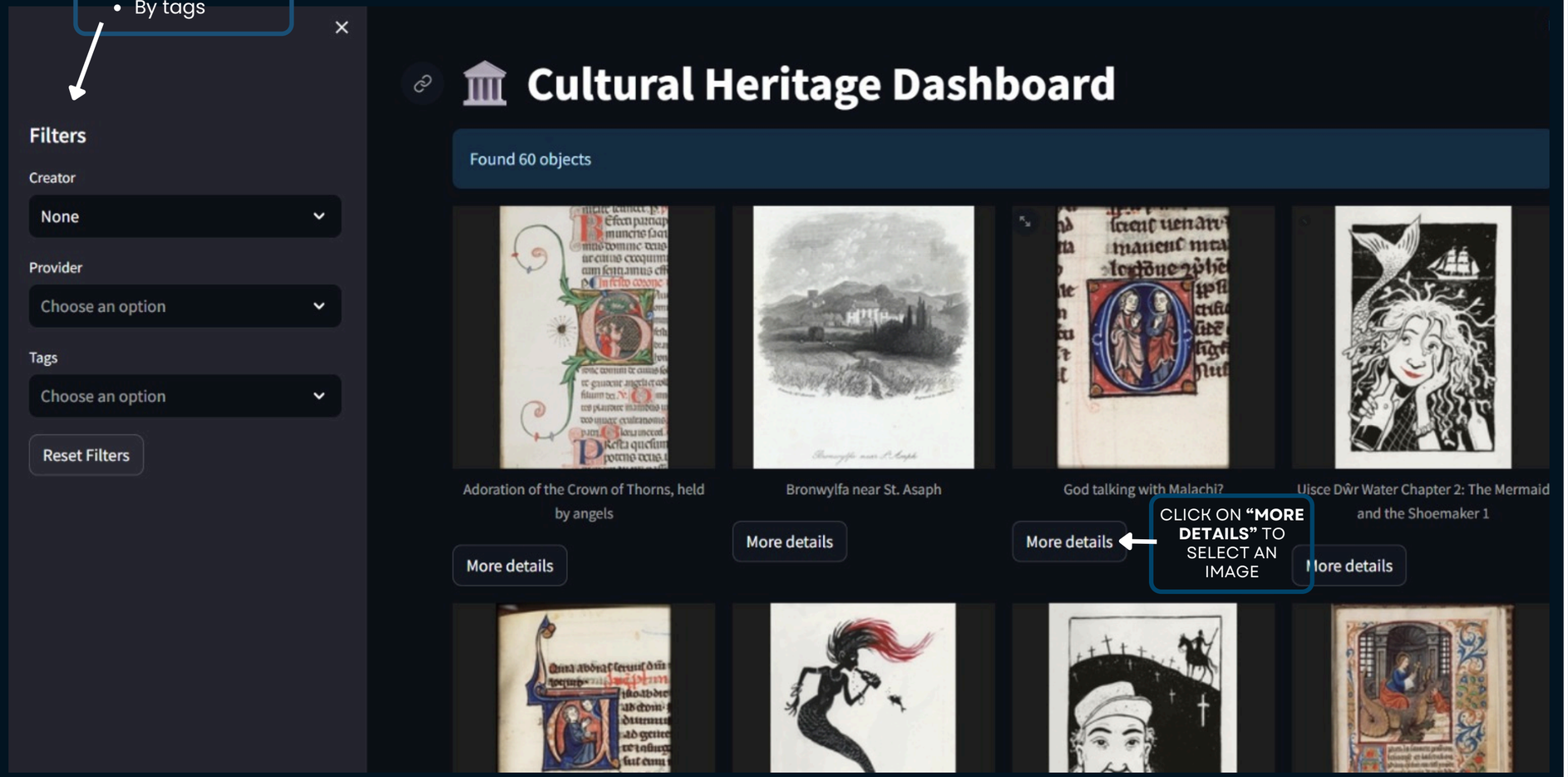
MEMORY USAGE

- Memory usage ramps up quickly at the start as containers are initialized.
- After startup, memory remains consistently high, fluctuating slightly but generally staying between 8 and 9 GB out of the available 15 GB.

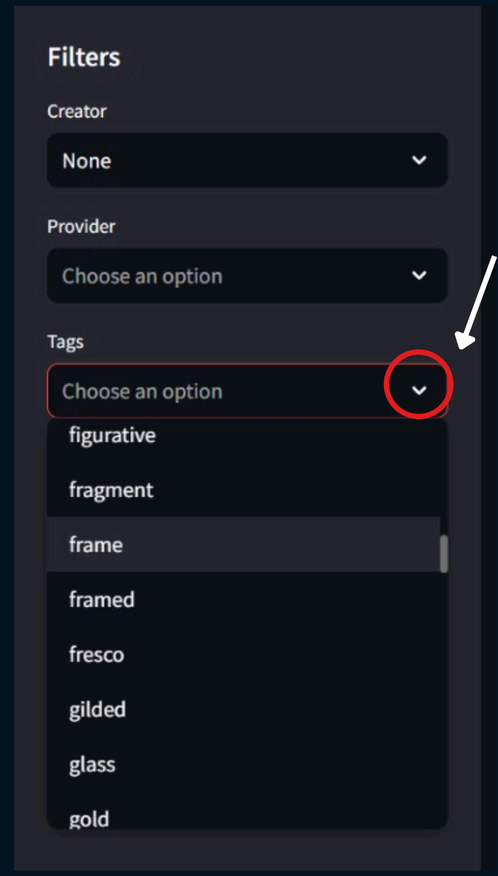
DASHBOARD

The dashboard allows users to explore, filter, and view detailed information about cultural heritage objects, including user annotations, and recommendations of similar items.

Filter results:
- By creator
- By provider
- By tags



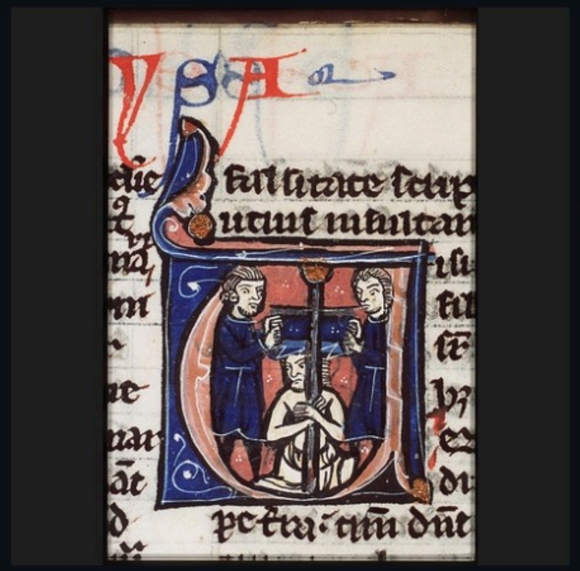
- Here is an example of **filtering by tags.** When selecting a tag, several options can be found
- Each filter, except Creator, is made with inclusive OR



Once an image is selected (after clicking “More Details”), a new page will appear. Here is an example:

Option to **go back to gallery**

Back to Gallery



The martyrdom of Isaiah: he is sawn in two

Deplov

You can view **the details** of each object:

- Title,
- Creator
- Description
- Type
- Rights
- Data Provider

Note: If information is not present in the metadata, you will see N/A

Description

Title: The martyrdom of Isaiah: he is sawn in two

Creator: N/A

Description: N/A

Type: IMAGE

Rights: N/A

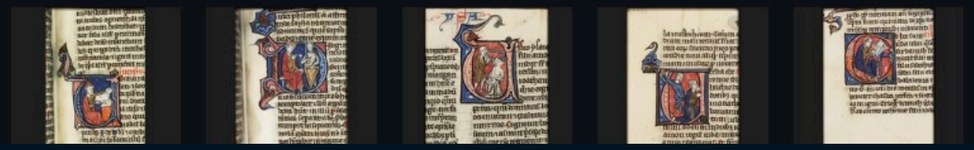
Data Provider: The European Library

By clicking on “Show Comments”, you can see all related user annotations

Show Comments

Dashboard shows up to 10 **similar objects** for each selected item

Similar objects



Example of User annotations (Comment, User ID, Timestamp, Tags)

Comment #7

User ID: user0342

Timestamp: 2025-07-17 16:20:25.982878

Comment: Could this be from the Venetian school?

Tags: `pastel` `textile` `figurative` `illumination`

Comment #8

User ID: user0114

Timestamp: 2025-07-17 16:20:08.729244

Comment: Hard to photograph due to reflections.

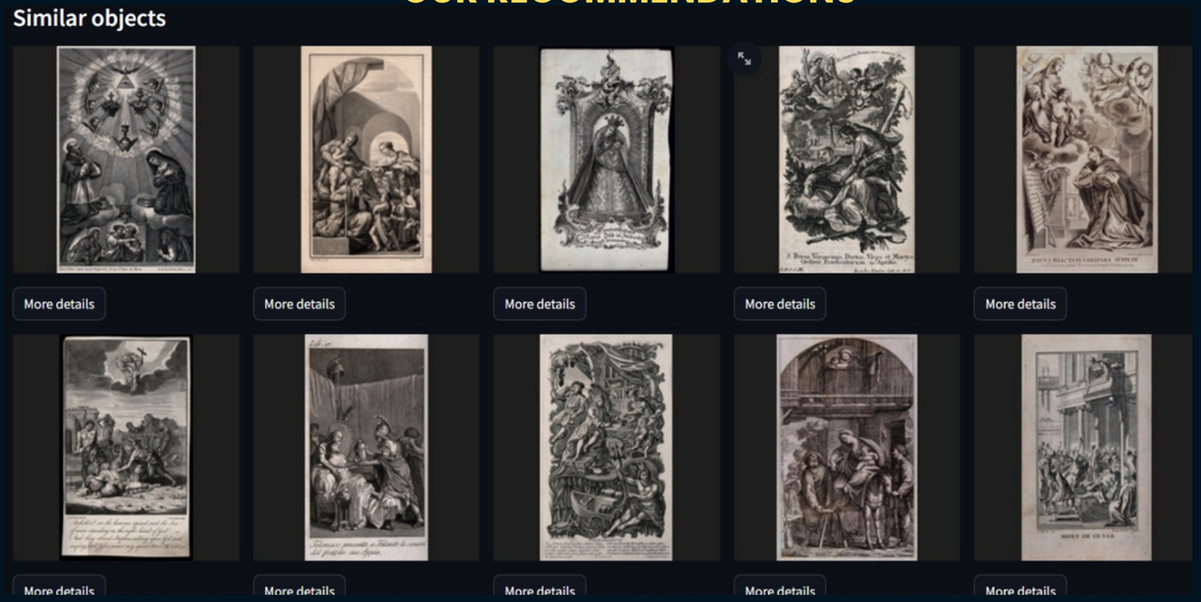
Tags: `mounted` `relief` `chalk` `armor`

COMPARING RECOMMENDATIONS APPROACHES (WITH EUROPEANA)

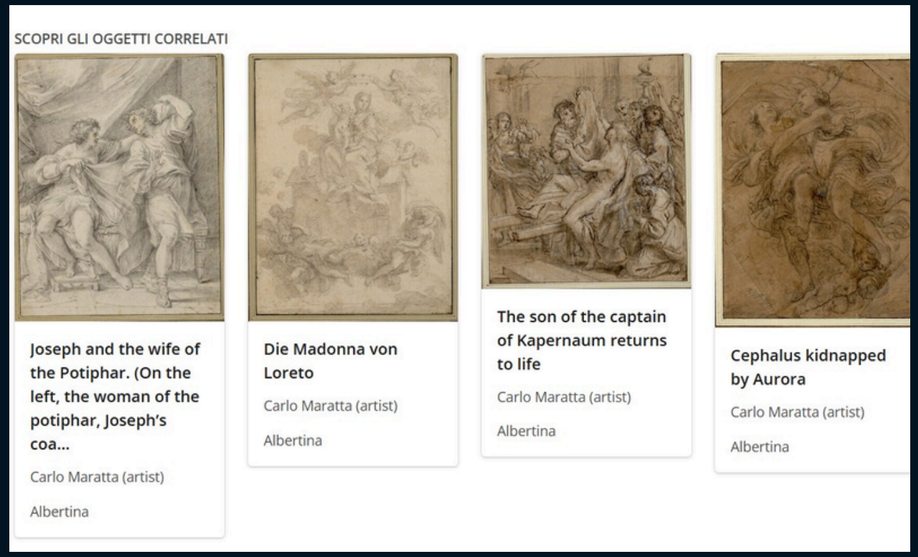
IMAGE SELECTED

EXAMPLE 1:

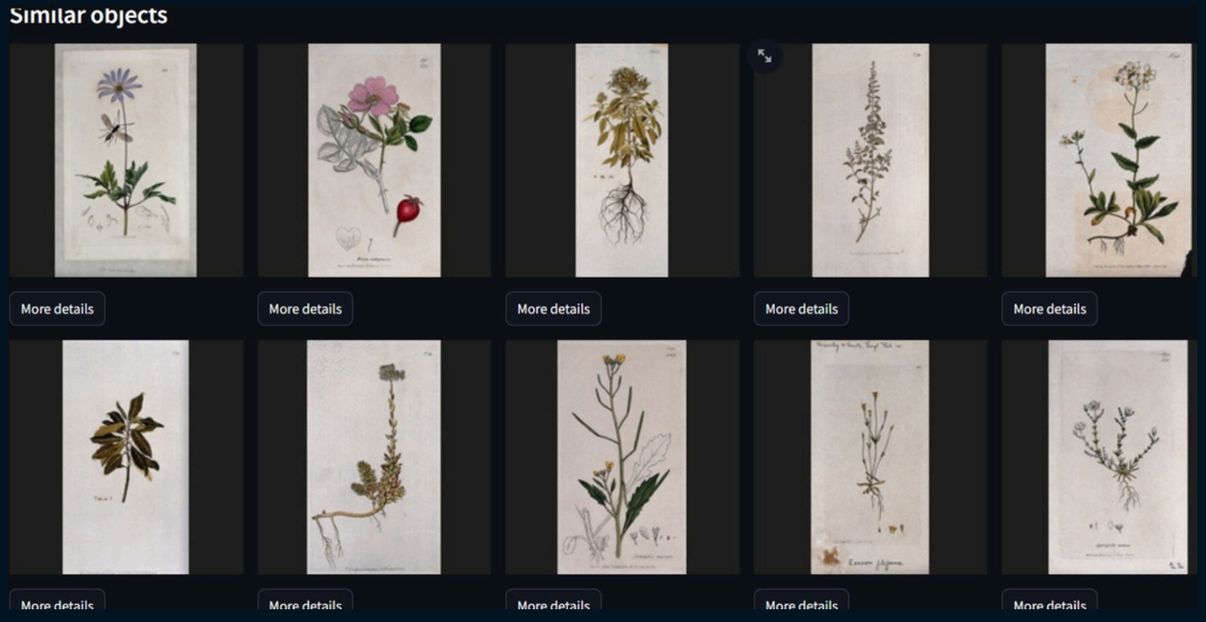
OUR RECOMMENDATIONS

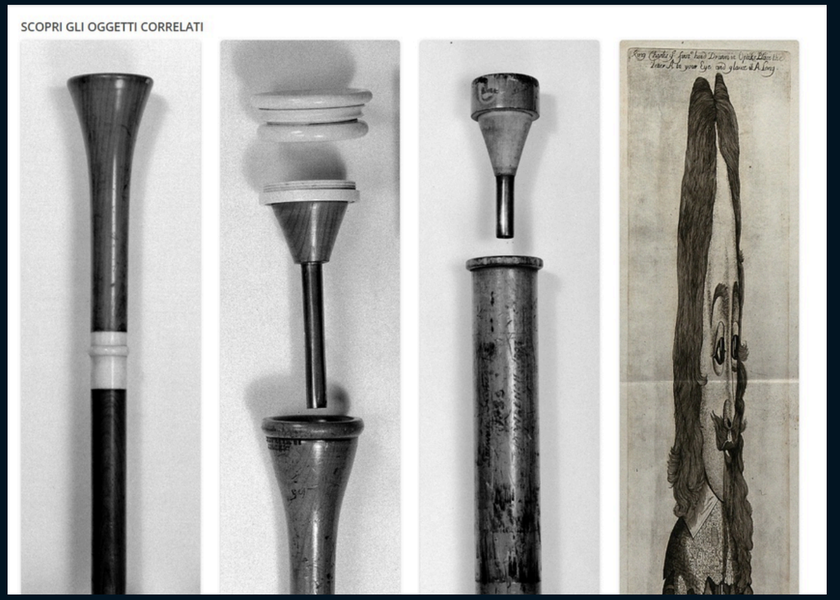


EUROPEANA'S RECOMMENDATION



EXAMPLE 2:





LESSONS LEARNED

MAIN CHALLENGES

- Finding a good image provider on Europeana
- Obtaining real and rich metadata
- Handling image embeddings
- Connecting Kakfa - Spark -MinIO
- The computational demands of the project often exceeded the capacity of our local machines, causing regular slowdowns and also often sudden outages.

KEY INSIGHTS

- Containerization is essential: Modular containers enable easier scaling, development, and maintenance
- Finalizing one part before moving on is important, otherwise, changes become much harder later as more dependencies build up.

WHAT WORKED WELL

- Team-work
- Api retrieval and setup
- Setup of docker compose containers
- Dashboard configuration

WHAT DID NOT WORK WELL

- Initially used overwrite mode in Delta Lake, which proved inefficient for scaling updates.
- Difficulties while trying to implement the join on the curated layer, especially due to changing schema and data dependencies.
- Deploying the project with low computational resources and a broken computer.

LIMITATIONS AND FUTURE WORK

LIMITATIONS

- Single Kafka broker limits fault tolerance
- Comments are synthetic, not real user data
- Ingestion from only one data source (Europeana)
- Metadata may contain missing or incomplete fields
- Europeana API does not support true scrolling, so limited batch download
- If Qdrant fails, pipeline breaks before PostgreSQL and dashboard

POTENTIAL IMPROVEMENTS

- Replace the .txt file (state) with a more scalable solution to track downloaded GUIDs
- Add real user annotations to analyze it over time to detect peaks in engagement
- Add orchestration and monitoring (e.g., Airflow, health checks, centralized logging)

WHAT WOULD BREAK FIRST WHEN THE SYSTEM SCALES, AND WHY?

- Spark Streaming bottleneck: micro-batches from Kafka (UGC) use .coalesce(1), which serializes output and limits parallelism
- Kafka bottleneck: only 1 partition, 1 replica, and 1 broker, so no scalability or fault tolerance
- Missing orchestration: no retry, scheduling, or alerting mechanisms if a job fails or hangs

FUTURE WORK

- Add Redis as a vector cache layer to speed up recommendations and reduce pressure on Qdrant/PostgreSQL
- Integrate additional open data sources for richer metadata and more advanced queries
- Use real annotations from external platforms (e.g., Wikipedia) to enrich user-generated content
- Implement Delta Lake OPTIMIZE to compact small files and improve performance
- Enable temporal analysis of artifact creation (e.g., group by year/month to identify historical trends)

Thank You.