MS15 End-user services development and implementation of the basic end-user services and their interface, defined based on the use cases

Version 0.3

Contract Number	2018-EU-IA-0104
Project Website	http://www.saintgeorgeonabike.eu
Contractual Deadline	31/5/2021
Milestone	MS15
Document type	Report
Author	Sergio Mendoza (BSC)
Contributors	Maria Cristina Marinescu (BSC), Cedric Bhihe (BSC)
Reviewer	Mónica Marrero(EF), Antoine Isaac (EF)
Keywords	end-user services, user interface, development

Co-financed by the Connecting Europe Facility of the European Union

Change Log

Version	Author	Description Change
V0.1	Sergio Mendoza (BSC)	First iteration
V0.2	Mónica Marrero, Antoine Isaac (EF), Maria-Cristina Marinescu (BSC)	Review and amendments
V0.3	Sergio Mendoza (BSC)	Final version

Table of Contents

1. Introduction

Saint George on a Bike (SGoaB) aims to improve the quality and quantity of metadata associated with imagery from European cultural heritage, using Artificial Intelligence technology, with an emphasis on the collections held by Europeana.

MS15 implements use case “3.3 Search based on enrichment” defined in MS3[1], following the Architecture defined in MS6[2].

SGoaB's Activity 8 (End-user general service definition) defines:

1) the functionalities that the SGoaB project will offer as services
2) a text-based image search tool

This document describes the following points to accomplish Activity 8:

- User Interface
- Services API and endpoints
- Search engine

The software implementing the end-user services has been deployed following the architecture defined in figure 7 of MS6 [3]:

- We build our search API and portal
- We store enrichments locally

In next iterations, we intend to store enrichments as Annotations in Europeana. Europeana Annotations API may have some limitations, which may restrict the shape or quality of the stored enrichments or the contents that may be accessed through the API. In such a case, the project-specific Search API would rely on its own enrichment storage and access layer, as shown in Figure 1 (Figure 7 IN MS6).

2. Definition of text-based image search

The objective of the text-based search engine is querying for images through the text in their enrichments and other kinds of (descriptive) metadata (e.g. author, year, etc.). This largely corresponds to use cases 3.3 (search based on enrichment) in MS3; this deliverable expects to satisfy this use case following the description given below:

1. The end user enters a search query that corresponds to her information need

2. The SGoaB API returns a list of Europeana objects

Note that the term 'text-based' term used here is a legacy from the original project proposal: in reality we are aiming to make this search a more "semantic" one, and where we would benefit from the contextual information that are attached to the entities that are connected to images, such as semantic relations between concepts, alternative labels (as described e.g. in SKOS[4]), potentially translations.

[1] See MS3 - Use Cases
[2] See MS6 - System and module-level architecture development .
[3] See MS6 - System and module-level architecture development .
[4] See SKOS alternative Lexical Labels

The *text-based image search* service follows the software architecture defined in the Figure 1, (see Figure 7 in MS6[5]). The yellow components have been implemented for the text-based image search service and are explained in their sections in this document:

- Section *Search Engine UI (SGoaB UI)*
- Section *Search Engine (SGoaB Enrichments)*
- Section *SGoaB Search API (SGoaB Search API)*

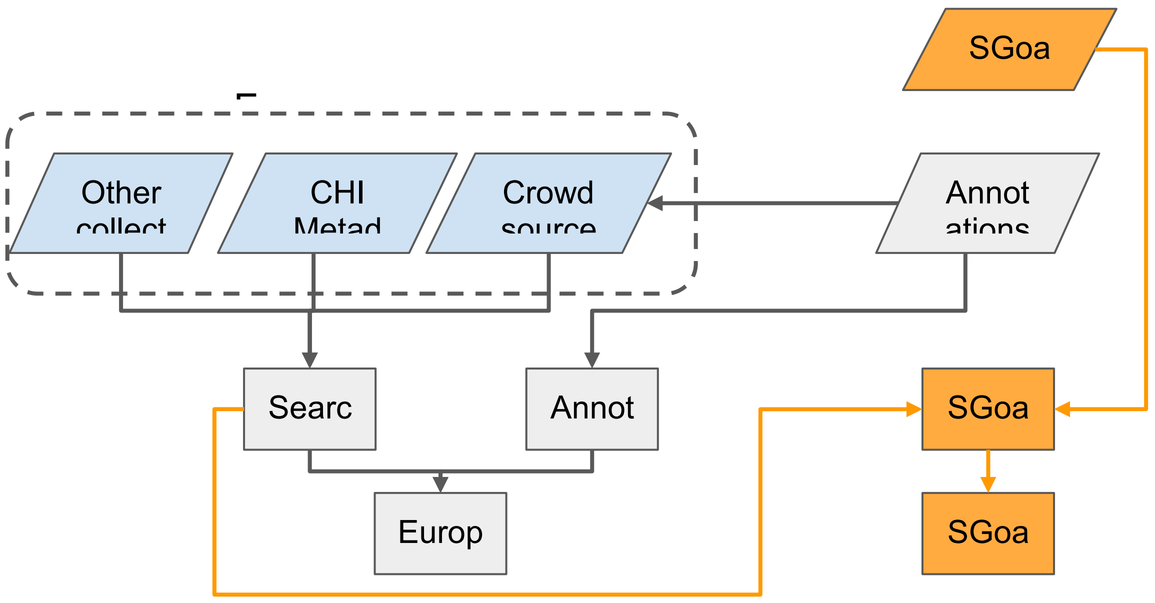


Figure 1. Building our search API and portal; local storage for enrichments

3. Search Engine UI

3.1 Production environment

The Search Engine is already accessible at the URL:

- http://vm1.nlpcase.bsc.es:5000/api/search/query

The production environment runs a user-friendly web application for accessing the SGoaB contents, including enrichments, querying the SGoaB Search Engine.

3.2 Querying the Search Engine

Although the SGoaB API has an endpoint for querying the search-engine (see "SGoaB API" section), a web-app has been developed to make it accessible for general users.

- The mandatory queries will be raw text.

[5] See MS6 - System and module-level architecture development .

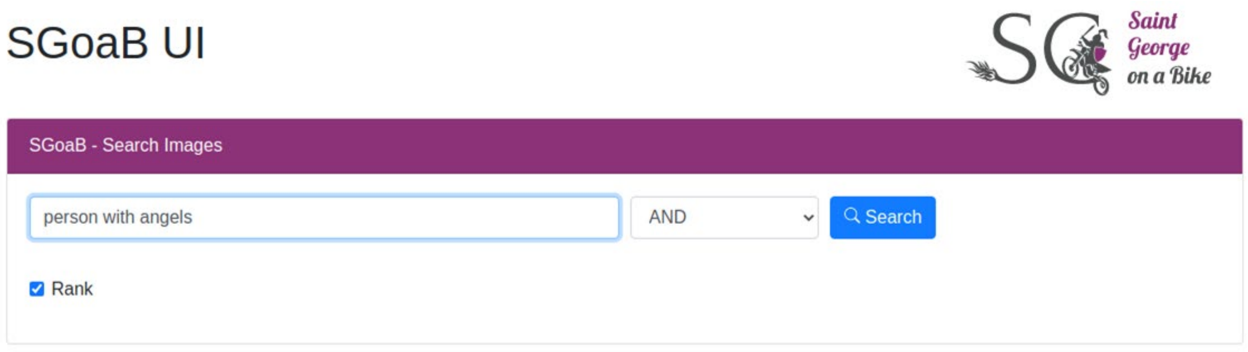
- The user may choose to search for either terms (OR) or all terms (AND) in the query text. AND will return images with all the search terms in the image metadata. OR will return images with at least one of the search terms.
- When a rank checkbox is selected, the results are ordered following the TF-IDF ranking algorithm. The TF-IDF algorithm scores higher the documents whose metadata contain a higher frequency of the queried terms.[5]

The example in Figure 2, with the query "person with angels" and the operator "AND" the results are:

- images with the class "crucifixion" and the class "angel". Both in the same image.

When querying "person with angels" with "OR" option, the results are:

- images with the class "person"
- Images with the class "angel"
- Images with the class "angel" and the class "person"


SGoaB UI

SGoaB - Search Images

person with angels

AND

Search

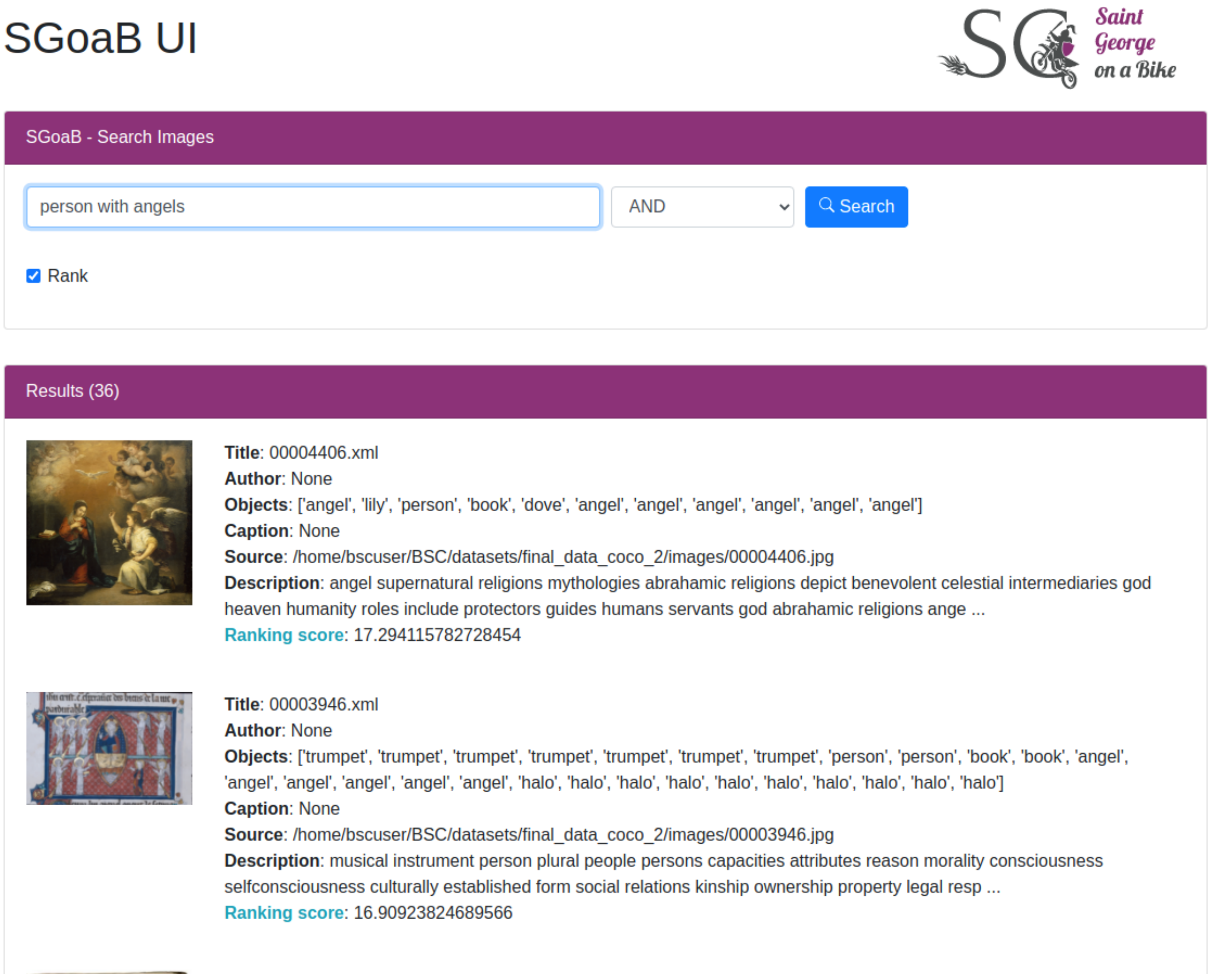
Rank: ✓

Figure 2. SGoaB UI - Search Engine Query

The results show the images, and their metadata enrichments, for those matching the query. The images are ordered by their descending rank score.

Each image in the results shows:

- **Objects:** list of the concepts identified in the image
- **Description**: relevant terms in the descriptions of the classes in the image. Limited to 200 characters.
- **Source**: is the Path of the image
- **Caption** (in this iteration is empty)
- **Title** (in this iteration contains the metadata filename)
- **Author** (in this iteration is empty)
- **Ranking score** (See Search Engine > Ranking)



SGoaB UI

SGoaB - Search Images

person with angels

AND

Search

☑ Rank

Results (36)

Title: 00004406.xml
Author: None
Objects: ['angel', 'lily', 'person', 'book', 'dove', 'angel', 'angel', 'angel', 'angel', 'angel', 'angel']
Caption: None
Source: /home/bscuser/BSC/datasets/final_data_coco_2/images/00004406.jpg
Description: angel supernatural religions mythologies abrahamic religions depict benevolent celestial intermediaries god heaven humanity roles include protectors guides humans servants god abrahamic religions ange ...
Ranking score: 17.294115782728454

Title: 00003946.xml
Author: None
Objects: ['trumpet', 'trumpet', 'trumpet', 'trumpet', 'trumpet', 'trumpet', 'trumpet', 'person', 'person', 'book', 'book', 'angel', 'angel', 'angel', 'angel', 'angel', 'angel', 'halo', 'halo', 'halo', 'halo', 'halo', 'halo', 'halo', 'halo', 'halo', 'halo']
Caption: None
Source: /home/bscuser/BSC/datasets/final_data_coco_2/images/00003946.jpg
Description: musical instrument person plural people persons capacities attributes reason morality consciousness selfconsciousness culturally established form social relations kinship ownership property legal resp ...
Ranking score: 16.90923824689566

Figure 3. SGoaB UI Search engine: response with the results (query: "person with angels").

3.3 Server: flask + jinja

The UI is deployed using flask and jinja templates rendering [1][2].

Flask is a light and fast server, which, if necessary, would allow moving to other production servers such as Apache httpd+mod_wsgi [3].

Jinja is a templating engine, which allows writing the code of the web-app with its own syntax, next to Python syntax. When the web page is requested, the template is rendered returning the final document with HTML syntax.

4. Search Engine

4.1 Introduction

The search engine is used for the implementation of the *text-based image search*. Its objectives are:

- Storing the image’s metadata
- Allow querying the image’s metadata and getting a list of Europeana objects as results

A Search Engine is a tool for storing documents and querying the Search Engine for retrieving the documents that best correspond with the user information requirements.

The SGoaB Search Engine contains metadata documents. The image metadata documents contain information such as[6]:

- **Classes in the image**
- Caption (in this iteration is empty)
- Path / URL
- Filename
- Title (in this iteration contains the metadata filename)
- Author (in this iteration is empty)

When loading the documents in the search-engine, the following information will be used for improving the results. They’re detailed at Indexation method:

- **Class alternative labels**
- **Class description**

Any of the attributes in the document could be used for indexing the documents. The indexation is an algorithm to organize the information of the documents in the Search Engine before querying it. Having an index allows fast response for queries.

A Search Engine makes it easier finding the documents. It is enough to query a text with what the user is looking for. For an end-user, it would be enough querying text with her requirements.

An example would be querying the text ”angels and dragons” which would return all the metadata documents with the classes knights and/or dragons.

4.2 The Search Engine

The search engine used by the SGoaB UI is based on the project *python-searchengine*[6]. The Search Engine has the following components:

- Data preprocessing
- Indexing with inverted index
- Searching
    - Rank algorithm

The simplicity and lightness of the source code allows an easy customization of its components for the *text-based image search*. The next sections explain how they have been customized.

[6] We put in bold the metadata that is exploited in the current stage of deployment.

4.3 Search Engine Storage

The Search Engine is using its own storage for enrichments. The SGoaB has its own local storage for the SGoaB generated enrichments, stored as image metadata. The enrichments are stored now as PASCAL-VOC. For the next iterations, other formats will be considered like json-ld, which may be useful if introducing knowledge graphs.

Currently, the search engine is built on a small subset of the descriptive metadata as detailed in the introduction. In the future, we will merge the metadata from SGoaB with Europeana's metadata (e.g. title, creator, etc.). It has 379 images, with its metadata documents that were generated by the object detection service. The dataset of 379 images and image metadata documents are 10% of the image dataset used for training the object-detection service model, which was trained with a dataset of 3790 painting images.

4.4 Preprocessing before indexation

Each image metadata document is extended with fields that will be used for the indexation and consequently will improve the quality of the search results. For each class in an image metadata document, the fields "class alternative labels" and "class description" are imported from wikidata and dbpedia (See section "Enrichment with Wikidata and DBpedia").

Before indexing the images metadata files, we prepare its contents following the steps below:

1. Tokenize the text
2. Lowercase the tokens
3. Remove punctuation symbols
4. Remove stopwords
5. Stem tokens

4.5 Indexation method: inverted index

The indexation method used is the inverted index algorithm. When loading the documents, it creates an index which maps each term to the documents (file) corresponding to the image metadata containing it.

The Search Engine has one index. The indexation of the image metadata documents was done using the image attribute:

- Classes label in the image

In addition, the indexation was improved using for each class of the image:

- Class alternative labels
- Class description

An example for the class "prayer":

- Class alternative labels: prayer, adorant, orans, orant, supplicant, suppliant
- Class description: person posture prayer

As a consequence, when querying "orant" or "supplicant" or "person", the documents containing the class "prayer" will be returned.

4.6 Ranking

The ranking algorithm used is tf-idf[7]. tf–idf is the product of two statistics: term frequency and inverse document frequency. For the ranking, it takes into account these metadata:

- **Classes labels in the image**

And also for each class, it's used:

- **Class alternative labels:** "Also known as", (aka), other names for the same object
- **Class description**

The fields "class alternative labels" and "class description" were obtained from wikidata and dbpedia (See the section "Enrichment with Wikidata and DBpedia").

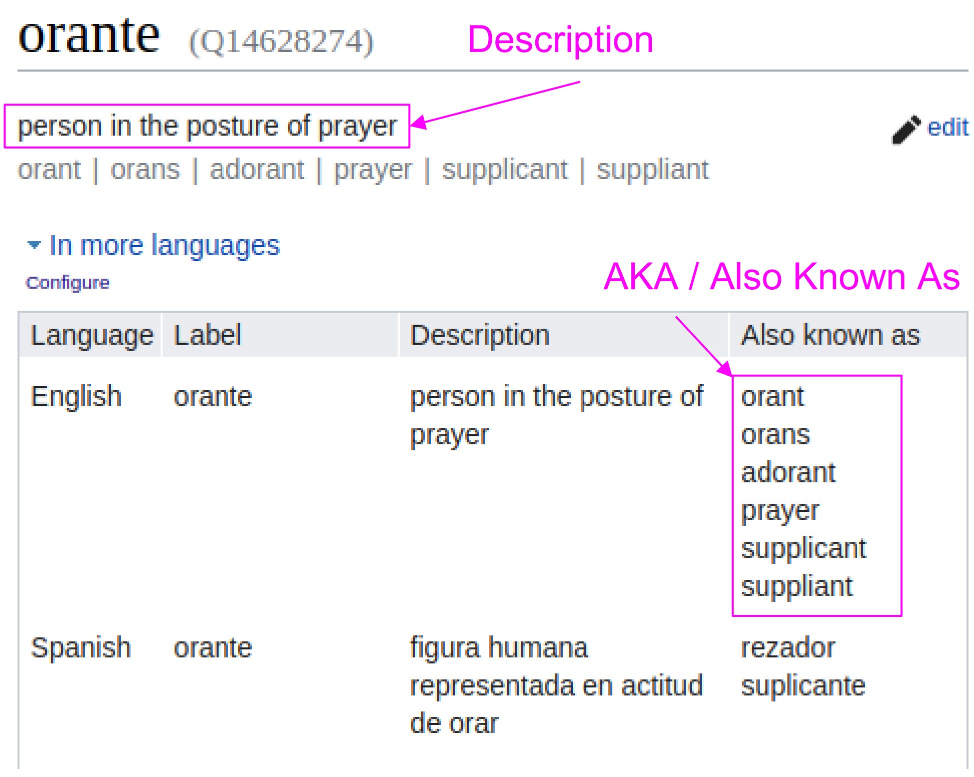
4.7 Enrichment with Wikidata and DBpedia

Enrichments in the form of "also known as" metadata for the classes that result from object detection can be included to the Search Engine enrichments as "alternative labels" for improving the search. For example, the class prayer is also known as:

- "adorant"
- "orans"
- "orant"
- "supplicant"
- "Suppliant"

When the input to the search engine contains the word "adorant", the tool will also return the images containing prayer. This information is typically available in cultural heritage vocabularies like Iconclass and Getty AAT, as well as in third party sources like Wikidata, see for example Figure 4 below:

[7] See "Using TF-IDF to Determine Word Relevance in Document Queries"



orante (Q14628274)

person in the posture of prayer

orant | orans | adorant | prayer | supplicant | suppliant

In more languages

Language	Label	Description	Also known as
English	orante	person in the posture of prayer	orant orans adorant prayer supplicant suppliant
Spanish	orante	figura humana representada en actitud de orar	rezador suplicante

Figure 4. Wikidata: "Also known as" and "description" of entity Q14628274

To implement such additional enrichment, we have provided each class selected for object detection[8] with a mapping with an URL, which corresponds to the same entity in wikidata or dbpedia. Synonyms and descriptions were extracted from Wikidata and DBpedia sparql endpoints:

- https://dbpedia.org/sparql
- https://query.wikidata.org/sparql

A python script was created for querying these endpoints and processing the results. The HTTP requests are done using the python Requests HTTP Library[9].

Query	Results

PREFIX wd: <http://www.wikidata.org/entity/>
PREFIX schema: <http://schema.org/>
SELECT ?aka WHERE {
wd:Q3341893 skos:altLabel ?aka .
FILTER (lang(?aka) = "en")
}
	aka
	aureole
	glories
	gloriole
	glory
	halos
	nimbus

Figure X. Sparql query for **wikidata** "also known as" extraction of entity Q3341893

[8] See Classes in SGoaB v1.1
[9] See Python Requests HTTP Library

Query	Results

PREFIX wd: <http://www.wikidata.org/entity/>
PREFIX schema: <http://schema.org/>
SELECT ?description WHERE {
wd:Q3341893 schema:description ?description .
FILTER (lang(?description) = "en")
}
	description
	religious symbol representing a ring of light

Table 1. Sparql query for **wikidata** description extraction of entity Q3341893

Query	Results

PREFIX dbo: <http://dbpedia.org/ontology/>
PREFIX dbr: <http://dbpedia.org/resource/>
SELECT distinct ?aka WHERE {
?a dbo:wikiPageRedirects dbr:Crucifixion .
?a rdfs:label ?aka
FILTER (lang(?aka) = "en")
}
	aka
	"Death on a cross"
	"Crucifixation"
	"Crucifixtion"
	"Crucifiction"
	"Cruxifixion"
	"Cross (execution)"
	"Crucification"
	"Crucified"
	"Crucifracture"
	"Crucify"
	"Crusification"
	"Roman crucifixion"
	"Stipes"
	"Christ-like pose"
	"Patibulum"
	"Self-crucifixion"
	"Devotional crucifixion"
	"Torture stake"
	"Haritsuke"

Table 2. Sparql query for **DBpedia** description extraction of resource https://dbpedia.org/resource/Crucifixion

Query	Results
PREFIX dbo: <http://dbpedia.org/ontology/> PREFIX dbr: <http://dbpedia.org/resource/>	**description** "Crucifixion is a method of punishment or

SELECT distinct ?aka WHERE {
 dbr:Crucifixion dbo:abstract ?aka .
 FILTER (lang(?aka) = "en")
}
	capital punishment in which the victim is tied or nailed to a large wooden beam and left to hang, perhaps for several days, until eventual death from exhaustion and asphyxiation. It was used as a punishment by the Romans. Crucifixion has been used in parts of the world as recently as the twentieth century. The crucifixion of Jesus is central to Christianity, and the cross (sometimes depicting Jesus nailed to it) is the main religious symbol for many Christian churches.”

Table 3. Sparql query for **DBpedia** description extraction of resource https://dbpedia.org/resource/Crucifixion

The aka and descriptions are stored in the classes in SGoaB Classes document[10] at the tab ”classes-aka-descriptions” of the spreadsheet . The tab contains the column with the descriptions and the column aka, which contains synonyms.

4.8 Data sources used in search engine - basic and "semantic" search

SGoaB foresees to deploy search based on the following metadata enrichment components, described in other milestones:

- Object detection
- Visual relations
- Caption Generation

For the current version, as progress on all three components is still ongoing, only the classes generated from object detection are included in search. The default Search Engine indexation algorithm (tf-idf) therefore only uses the name of the detected classes in an image (e.g. Prayer, The Holy Shroud, Centaur, etc.).

To provide more information for the search process, however, we have begun to source more "contextual" data from third party sources, in a way similar to what is done by Europeana and other similar CH services that implement basic forms of "semantic" search:

- Class alternative labels
- Class descriptions

The enrichments are pulled from Wikidata and DBpedia, as described in section “Enrichment with Wikidata and DBpedia”. If just using class labels for indexation, when a user searches for documents containing the class “halo”, the user just would find the document if the term queried is exactly “halo”. After the improvement with wikidata and DBpedia data, when the user introduces any synonym or term in the description of the class, the search engine will return images containing that class.

- Halo Synonyms: aureole, glories, gloriole, glory, halos, nimbusa

[10] See Classes in SGoaB / aka-descriptions

- Halo description: “religious symbol representing ring light”

5. SGoaB Search API

This section defines the HTTP API which:

- Offers the services through HTTP
- Is used by the SGoaB UI

The API endpoint offering the services are described by:

- HTTP-METHOD: {GET / POST / PUT / DELETE}
- INPUT: input parameters
- OUTPUT: response content

The API uses JSON as its communication format and the standard HTTP.

5.1 Endpoints

Below we define the endpoint implemented in the SGoaB Search API.

`/sgoab/search/query`
● **HTTP-Method**: POST ● **INPUT**: ○ Raw text: string ○ AND / OR: string ○ Rank: Boolean ○ Token: string (optional) ● **OUTPUT** ○ Search results in JSON
Constraints 6 Queries may use metadata filters: 6.1 author’s name 6.2 class / detected objects’ classes depicted in the painting 6.3 time range when the image was painted

6. Discussion: future work

6.1 Improvements for Search Engine:

6.1.1 Integration with Europeana APIs

For the next iteration we expect that the SGoaB Search Engine could:

- Store and access the full-fledged representation of the enrichments through the Europeana Annotation API
- Use the Europeana Search API to tap into the 'regular' (i.e. pre-SGoAB) descriptive metadata for objects that Europeana holds
- Query the Europeana Search Engine looking for images
- Query images containing classes through Europeana Search API (e.g. images with Crucifixion)
- Query images by searching user given natural language text in the SGoaB generated captions

6.1.2 Ranking algorithm

Introduce transformer language models for question-answer tasks for improving search engine results. The images could be ranked using the query text and image metadata like:

- Image metadata:
  - Basic Objects
  - Complex objects
  - Inferred objects
  - Visual relations
  - Caption

Also, Cosine Similarity could be an interesting option for ranking image metadata documents [4].

6.1.3 Auto-complete

Implement an auto-complete service for:

- query suggestion for the UI user
- entity suggestion would be based on the classes defined used in SGoaB now

About Use Case 3.6 Browsing based on enrichment:

- will suggest related links (Europeana's entities? external entities? websites?)
- would be more defined in future iterations

6.2 Enrichments Storage

Now the data used is stored as CSV, JSON and PASCAL-VOC.

In the next iteration, when having a stable search engine and services, the objective is storing the enrichments obtained from the SGoaB enrichment components (e.g. object-detection, visual-relations) in Europeana's resources.

The obtained enrichments will be stored as Europeana Annotations through the Annotations-API[11] following the software architecture defined in Figure 6 in the MS6[12]. During the development, the Europeana's acceptance resources will be used for testing, not the production environment.

[11] See Europeana Annotations API
[12] See MS6 - System and module-level architecture development .

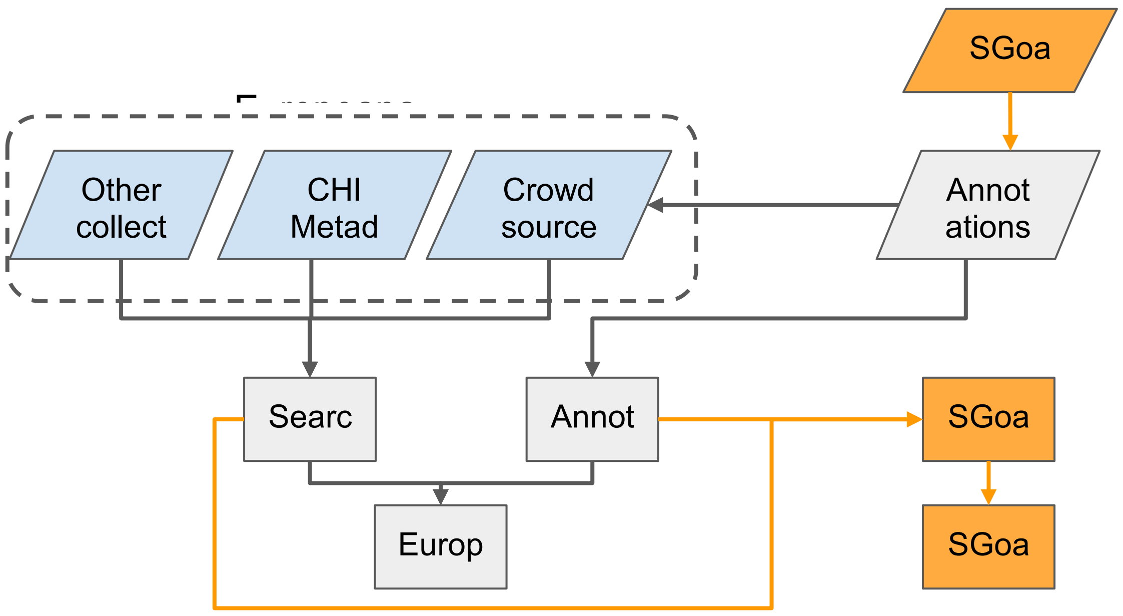


Figure 5. (Figure 6 in MS6) Building our own search API and portal; Europeana-stored enrichments

The code is stored in a Gitlab repository at BSC resources. [13]

- TBD:

- When storing enrichments as Europeana-Annotations:
  - Would it be enough searching through the Europeana's Annotations API?
  - Would it be necessary to have a SGoaB Search Engine?
- Evaluate if storing services results as JSON-LD is a real option.
- Evaluate if a unique storage as Annotations in Europeana is enough for satisfying the use cases.
- Which parts of the services outputs could be stored and shared using JSON-LD.

6.3 UI uploader

TBD - For the next iteration the following answer should be answered for continuing with the implementation.

- Will the results be always stored?
- How long will the enrichments be stored?
- Would a DataBase be necessary? (With documents storage: CSV, JSON-LD, etc.)
- Make a UI uploader with a selection of the service.
- Would uploading using URLs instead of files be an option?

[13] See SGoaB-Gitlab .

- Would uploading just metadata be an option? (It should contain URLs of the image to be analyzed.

6.4 Authentication

Introduce token authentication for using the services.

7. Acronyms and Abbreviations

- AKA - Also Known As
- API - Application Programming Interface
- BBx - Bounding Boxes
- BERT - Bidirectional Encoder Representations from Transformers
- EDM - Europeana Data Model
- HPC – High Performance Computing
- MS – Milestones
- ROI - Region of Interest
- TBD - To Be Decided
- UI - User Interface

8. References

[1] “Flask Documentation (2.0.x),” *flask.palletsprojects.com*. https://flask.palletsprojects.com/en/2.0.x/.

[2] “Jinja Documentation (3.0.x),” *jinja.palletsprojects.com*. https://jinja.palletsprojects.com/en/3.0.x/ (accessed May 26, 2021).

[3] “mod_wsgi (Apache) — Flask Documentation (2.0.x),” *flask.palletsprojects.com*. https://flask.palletsprojects.com/en/2.0.x/deploying/mod_wsgi/ (accessed May 26, 2021).

[4] B. Li and L. Han, “Distance Weighted Cosine Similarity Measure for Text Classification,” *Intelligent Data Engineering and Automated Learning – IDEAL 2013*, vol. 8206, pp. 611–618, 2013, doi: 10.1007/978-3-642-41278-3_74.

[5] A. Mishra and S. Vishwakarma, “Analysis of TF-IDF Model and its Variant for Document Retrieval,” *2015 International Conference on Computational Intelligence and Communication Networks (CICN)*, Dec. 2015, doi: 10.1109/cicn.2015.157.

[6] B. de Goede, “bartdegoede/python-searchengine,” *GitHub*, May 29, 2021. https://github.com/bartdegoede/python-searchengine (accessed May 31, 2021).