
Open WebSearch
Build Your Own (Conversational) Search Engine
Hackathon Guide
Open Web Search
Funded by the European Union
SUPPORTED BY
NGI

Overview

- ➜ OWS Partner TU Graz (CoDiS Lab)
- ➜ Concept of Search Applications
- ➜ Technical Details of the Prototype Search Application
- ➜ Create Your Own (Conversational) Search Application

OWS Partner: TU Graz, CoDiS Lab

- Graz University of Technology, Austria
- **Co**gnitive and **Di**gital **S**cience **Lab**
- Interdisciplinary research by connecting computer science with cognitive psychology

Contributions to ows.eu

- Search applications
- Ethical, legal, and social aspects

OWS Workflow

Index Generation

Web resources are selected and retrieved, their content and metadata are analysed, and all data stored in the index database.

(1) Selecting web resources
Web pages are navigated, prioritized and collected

(2) Storing web documents
Multiple gatherers collect web documents and store them in web archives on a European server

(3) Content extraction
The content of web documents is extracted (e.g. words, images)

(4) Metadata extraction
Metadata (e.g. publisher, author, date) are extracted

(5) Content analysis
Features of web documents are extracted (e.g. topic, language, quality, genre, legal constraints, ethical aspects, etc.)

(6) Index building
All extracted data from web documents are stored in a specialised database, the so-called webindex

(7) Index deployment
The index is deployed in its full version at European data centres or sliced into smaller portions for specific purposes and made available for download

Search Applications

A user search request will be answered by a search application that makes use of the open web index.

(a) Selecting web documents
Web documents are selected that fit to the user search request

(b) Ranking web documents
The selected Web documents are sorted (ranked) according to their assumed relevance for the user

(c) Purpose-specific search
An application with user interface enables the search for general or specific purposes

(d) User searches and receives result
The user is supported to better understand the search process

Data Products

Knowledge representation models will be created using the open web index, in order to be used by any agent and for many applications

Building knowledge graphs
Using the extracted information from web documents, a knowledge graph is created that supports specific search requests

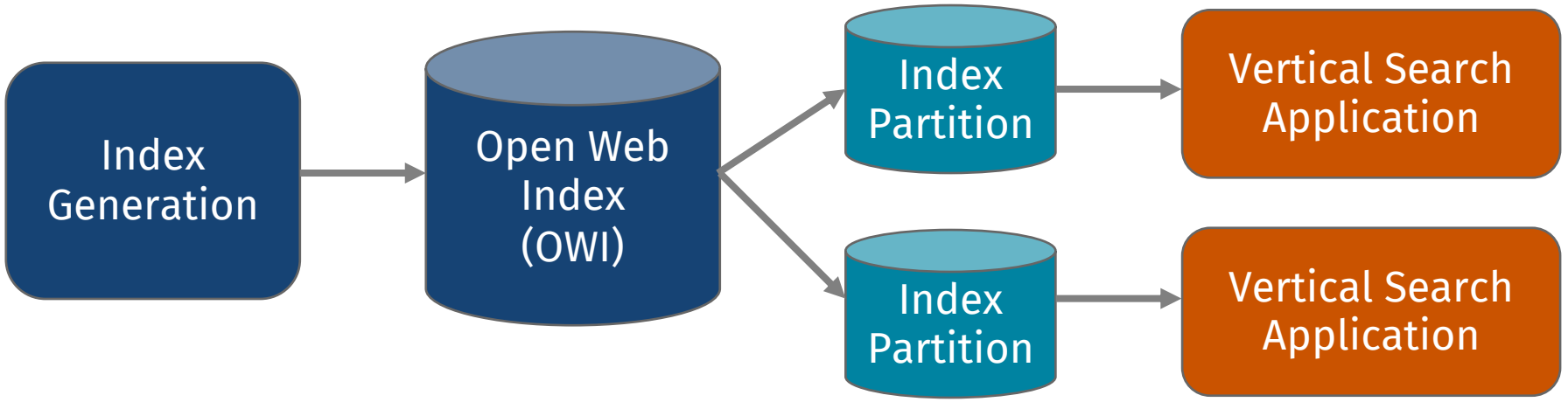
Building AI Language Models
Creation of different types of language models by using Web documents

Any agent, multiple applications
Language models and knowledge graphs can be used by any agent (or application)

...

Vertical Search Engines in OWS

- Vertical Search Engines serve special purposes, topics, or domains, e.g. product search
- Search applications are a key concept in OpenWebSearch.eu
- Based on Open Web Index and index partitions

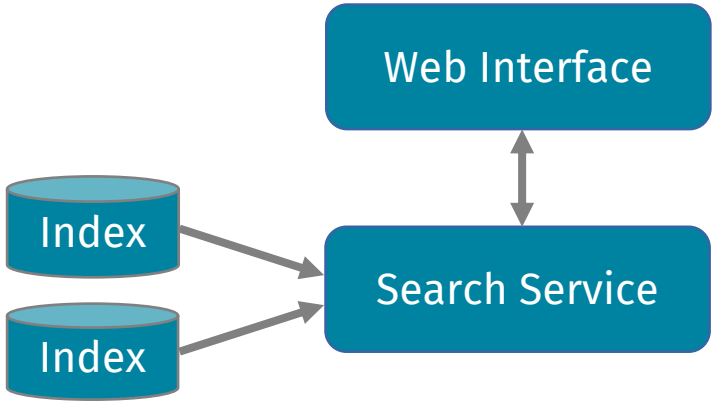


Prototype Search Application

Minimal implementation of an OWS vertical search engine

➜ Index partitions in reasonable size on a local server

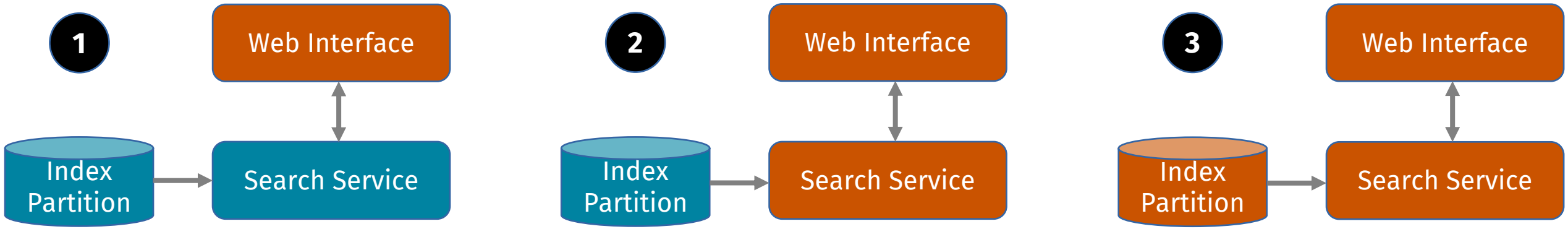
➜ Search service that searches the index and provides REST API



Prototype Search Application

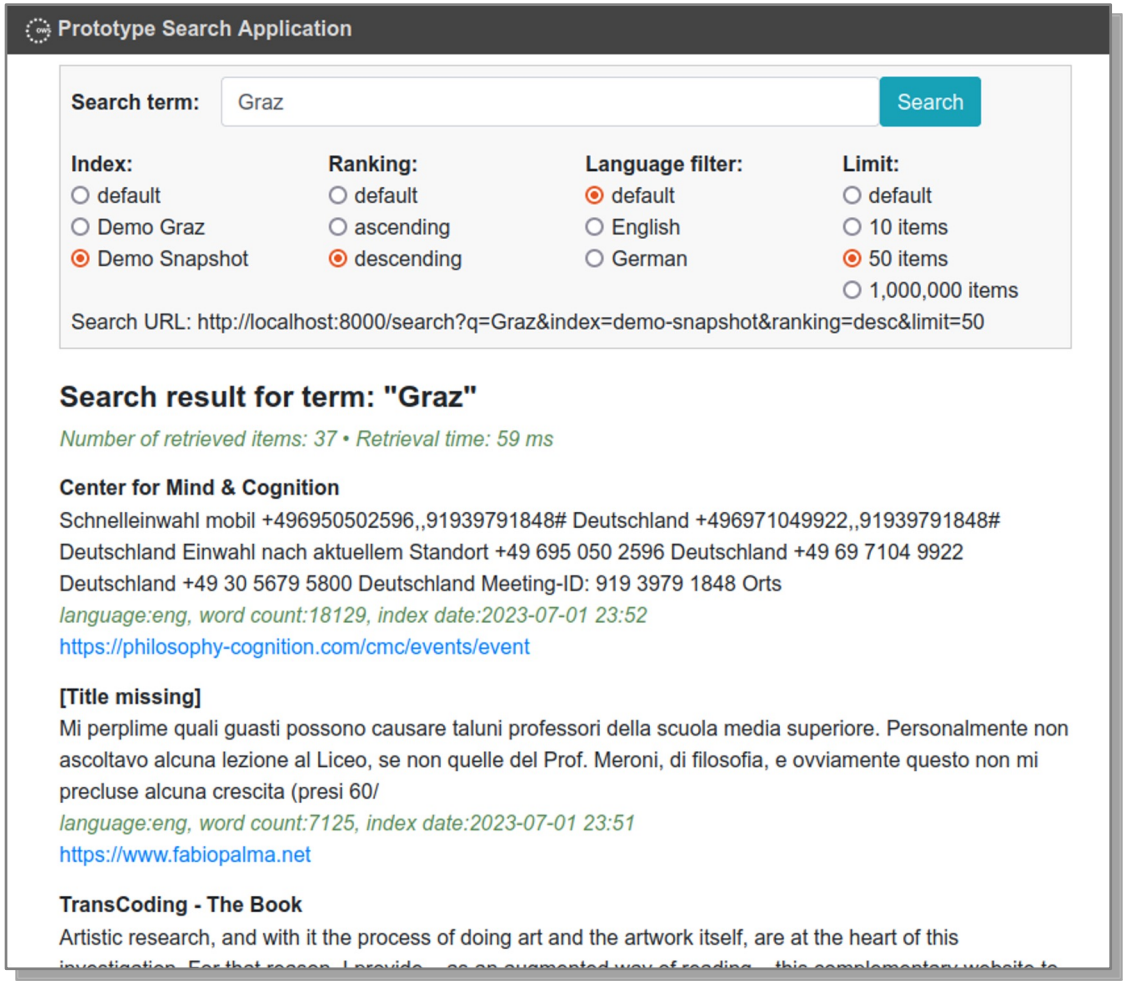
Guides the development of an OWS search application

- ➜ Using the search service and development of a web interface (1)
- ➜ Using the concept and development of the whole application (2)
- ➜ Using the concept to explain the architecture of search applications (3)



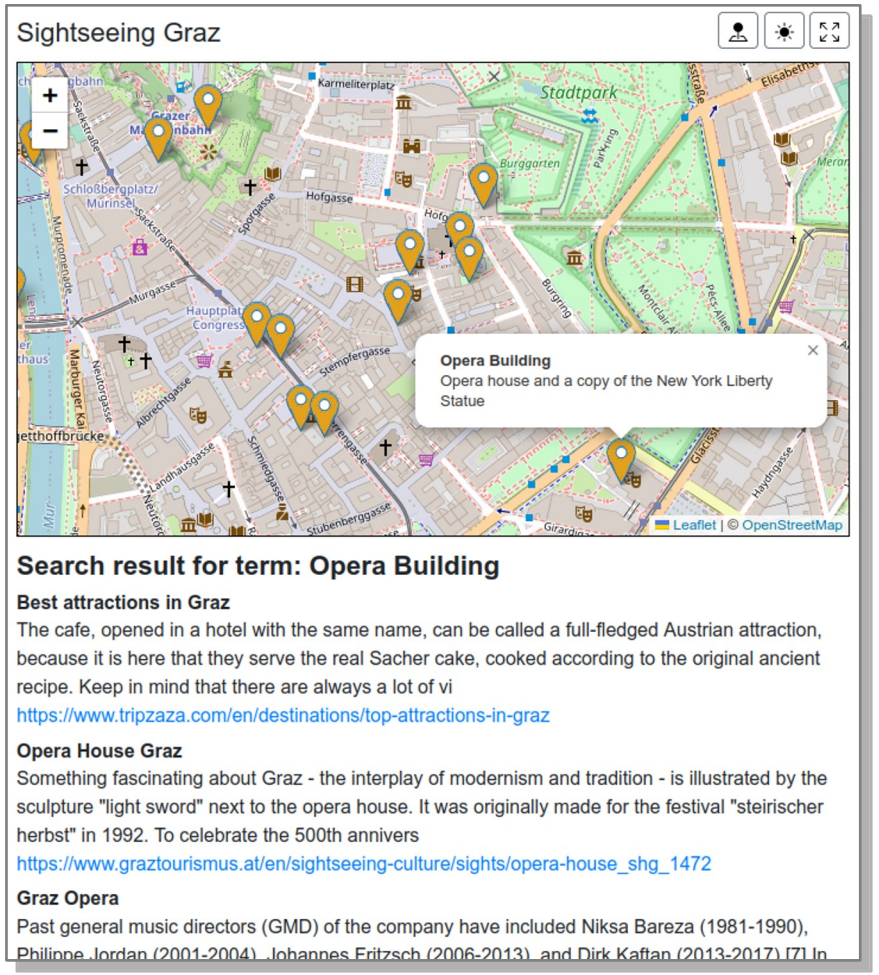
Demonstration 1: Basic Search

- Simple web interface to the Search Service
- Operates two indices
- Performs re-ranking and filtering
- Displays metadata
- Available at: https://qnode.eu/ows/prosa/front-end/

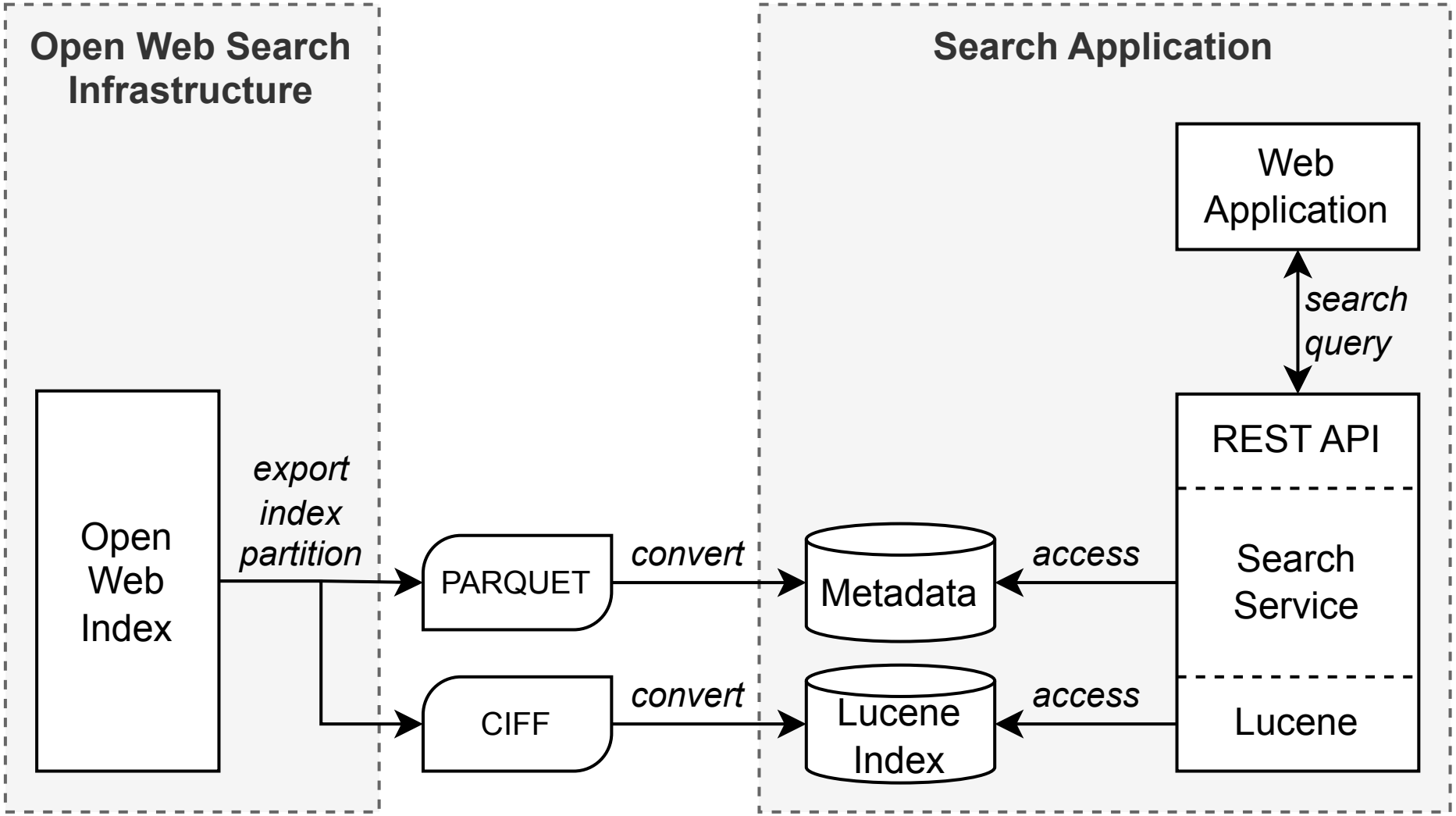


Demonstration 2: Sightseeing Application

- Integration into existing Sightseeing Application Spots
- Clicking on a sightseeing marker triggers a search with the title of the spot as search term
- Available at: https://qnode.eu/ows/prosa/spots/?mid=owsmeetinggraz



Conceptual Design of the Prototype Search Application



Index Partition: CIFF/Lucene and Parquet

- ➜ CIFF: Index exchange format
- ➜ Lucene: Inverted index
- ➜ Parquet: Tabular format

CIFF/Lucene

Term	List of (Document, Frequency)
Term 1	(docID A, 1) (docID B, 3) (docID C, 1)
Term 2	(docID D,2)
Term 3	(docID E, 4) (docID F, 2)
Term 4	(docID H, 2) (docID I, 1)

Parquet

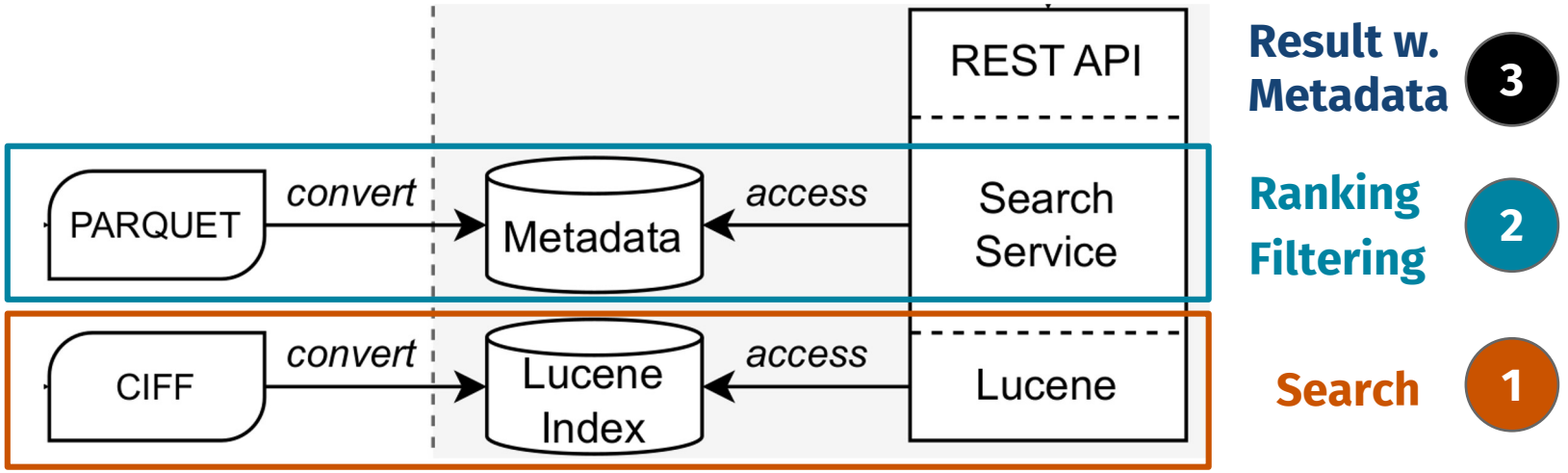
Document	Lang	Full text	WARC date	…
docID A	en	… text …	2023-07-01	
docID B	de	… text …	2023-07-01	
docID C	de	… text …	2023-07-01	
docID D	en	… text …	2023-07-01	

Parquet File / Metadata Fields

- id
- record_id
- title
- plain_text
- warc_date
- warc_ip
- language
- http_content_type
- http_server
- url
- url_scheme
- url_path
- url_params
- url_query
- url_fragment
- url_subdomain
- url_domain
- url_suffix

Search Process

- ➜ CIFF file is converted to a Lucene index that can be searched with the Lucene library
- ➜ Search result is filtered and ranked using metadata coming from the Parquet file



REST API

End point: `host:port/search?q=…`

Request parameters:

Parameter	Description
`q`	Search term(s)
`index`	Lucene index to be searched in
`lang`	Restrict searches to pages in language
`ranking`	Specify order of search result
`limit`	Set maximum number of returned results

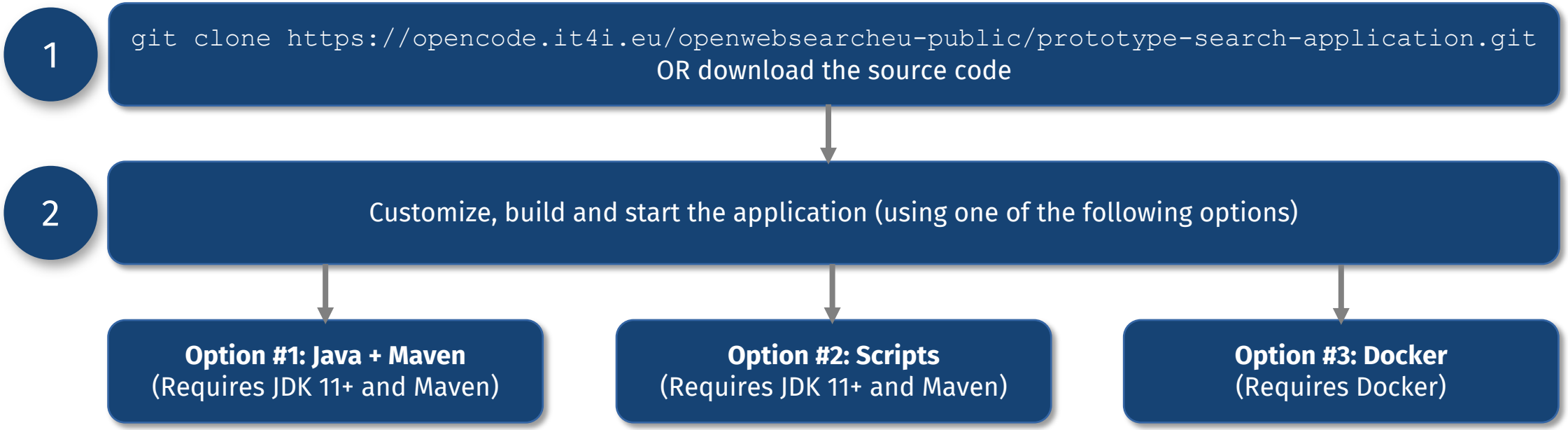
Example:
https://qnode.eu/ows/prosa/service/search?q=austria

Response data:

```
{
  results: [
    {
      id: …,
      url: …,
      title: ...,
      textSnippet: ...,
      language: ...,
      wordCount: ...,
      warcDate: ...,
    }, ...
  ]
}
```

Next Steps Towards Creating Your Own Search Application

➜ Find the source code of the Prototype Search Application on GitLab: https://opencode.it4i.eu/openwebsearcheu-public/prototype-search-application



Option #1: Java + Maven

➜ Prerequisites: Java (JDK 11+) + Maven

```
# change the directory
cd search-service
# clean the project and compile the source code
mvn clean compile
# build the executable
mvn package
# run the executable
java -jar target/app.jar -d <YOUR_DEFAULT_INDEX_NAME> -p <YOUR_API_PORT>
```

➜ Example: `java -jar target/app.jar -d demo-graz -p 8000`

Option #2: Scripts

➜ Prerequisites: Java (JDK 11+) + Maven

```
# change the directory
cd scripts
# convert indexes(optional), clean the project and create a packaged JAR
./build.sh
# run the executable
./start.sh
```

➜ Example: `./start.sh demo-graz 8000` (default index and port are optional)

➜ Batch files as alternative for Windows

Option #3: Docker

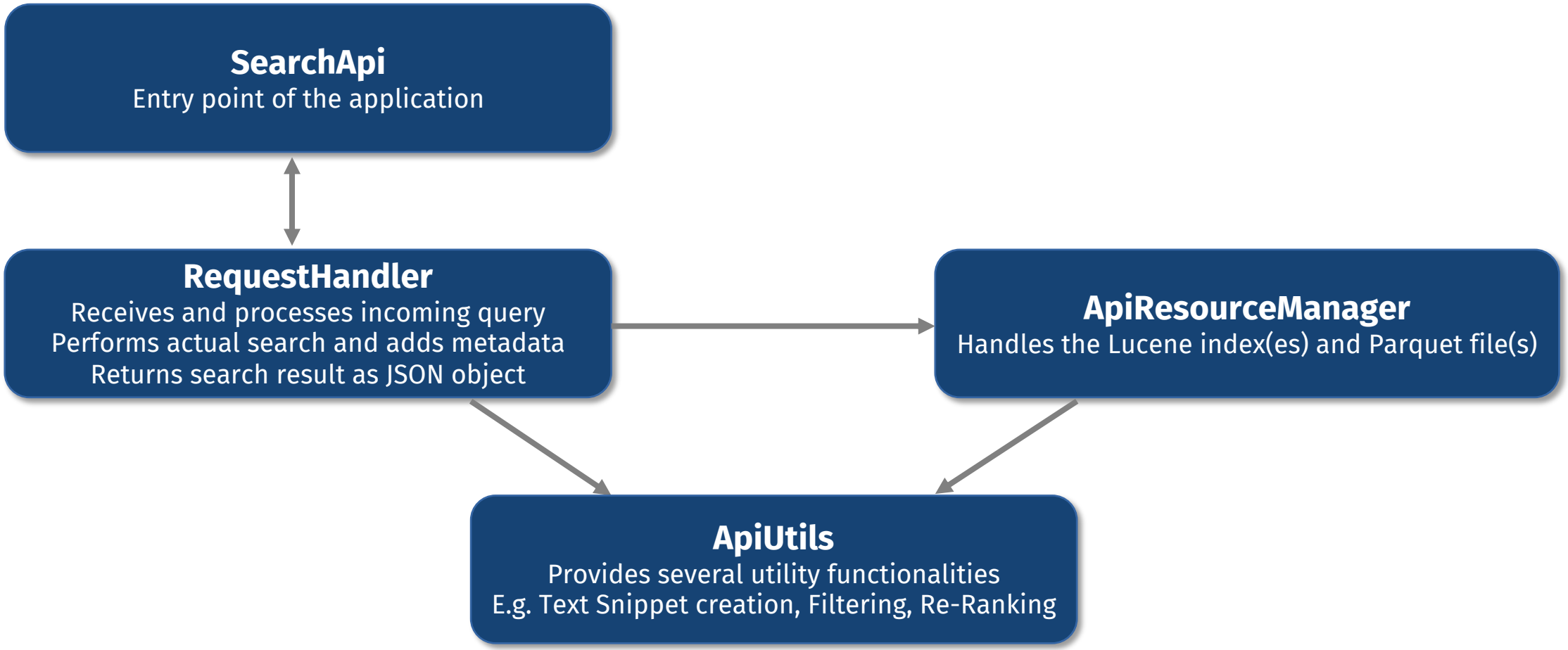
➜ Prerequisites: Docker

```
# create an image
cd scripts docker build -t prototype-search-application .
# start a container
docker run -p 8000:8000 prototype-search-application \
    -d <YOUR_DEFAULT_INDEX_NAME> -p <YOUR_API_PORT>
```

➜ Example: `docker run -p 8000:8000 prototype-search-application \ -d demo-graz -p 8000`

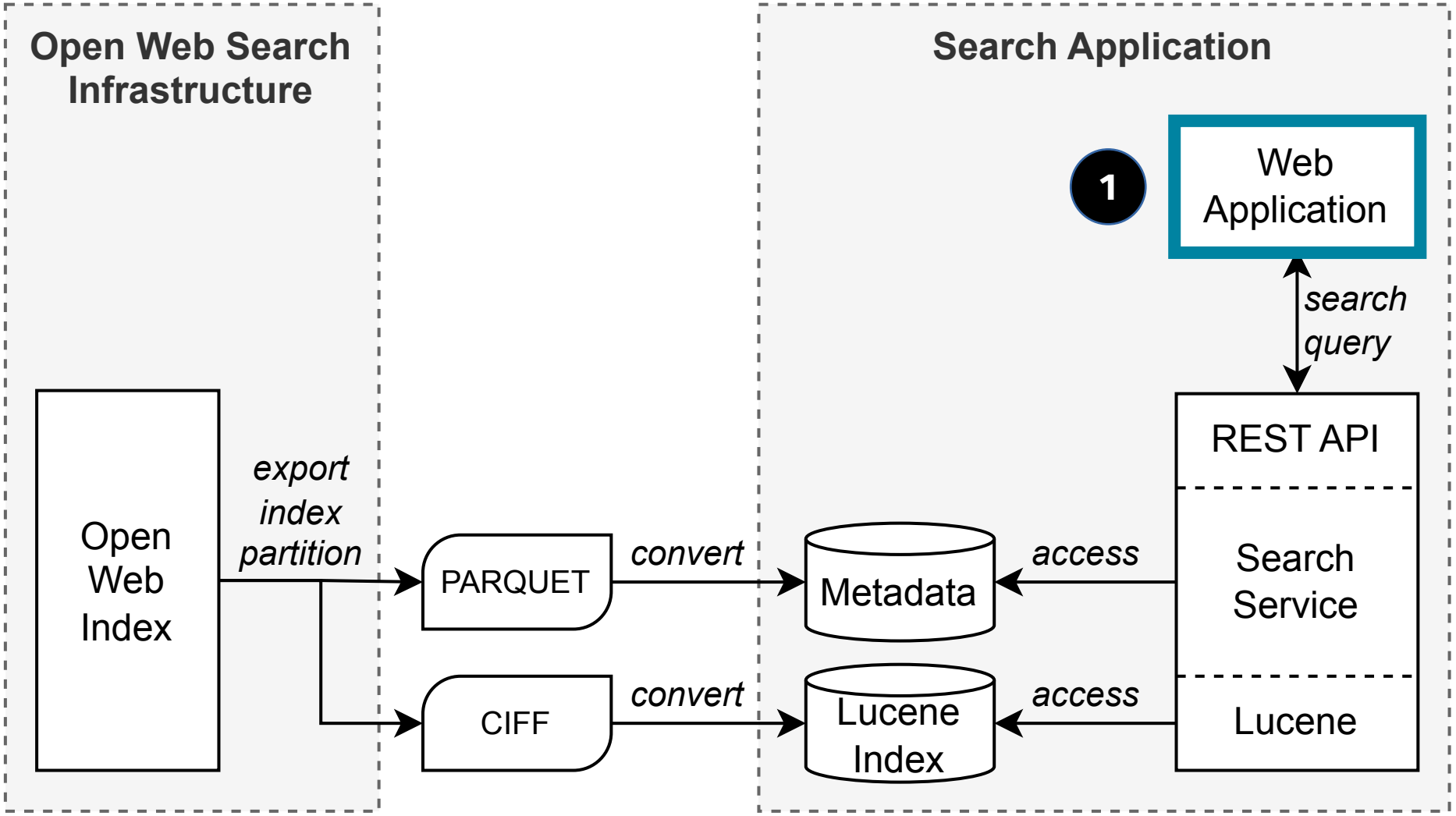
➜ Re-build of image necessary after every code change

Code Architecture of the Prototype Search Application



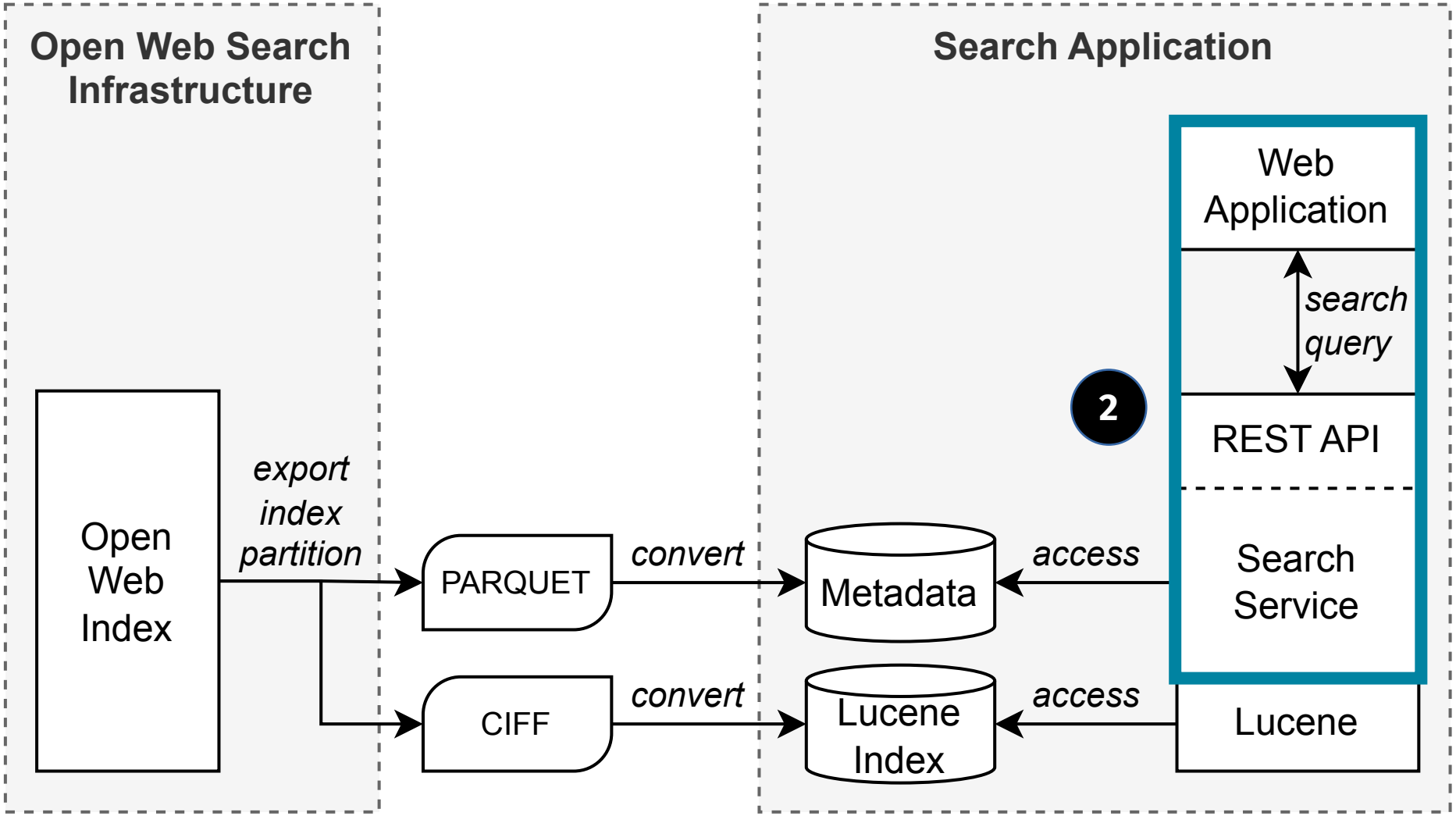
Ideas & Suggestions on What You Can Change

(1) Using the search service and development of a web interface



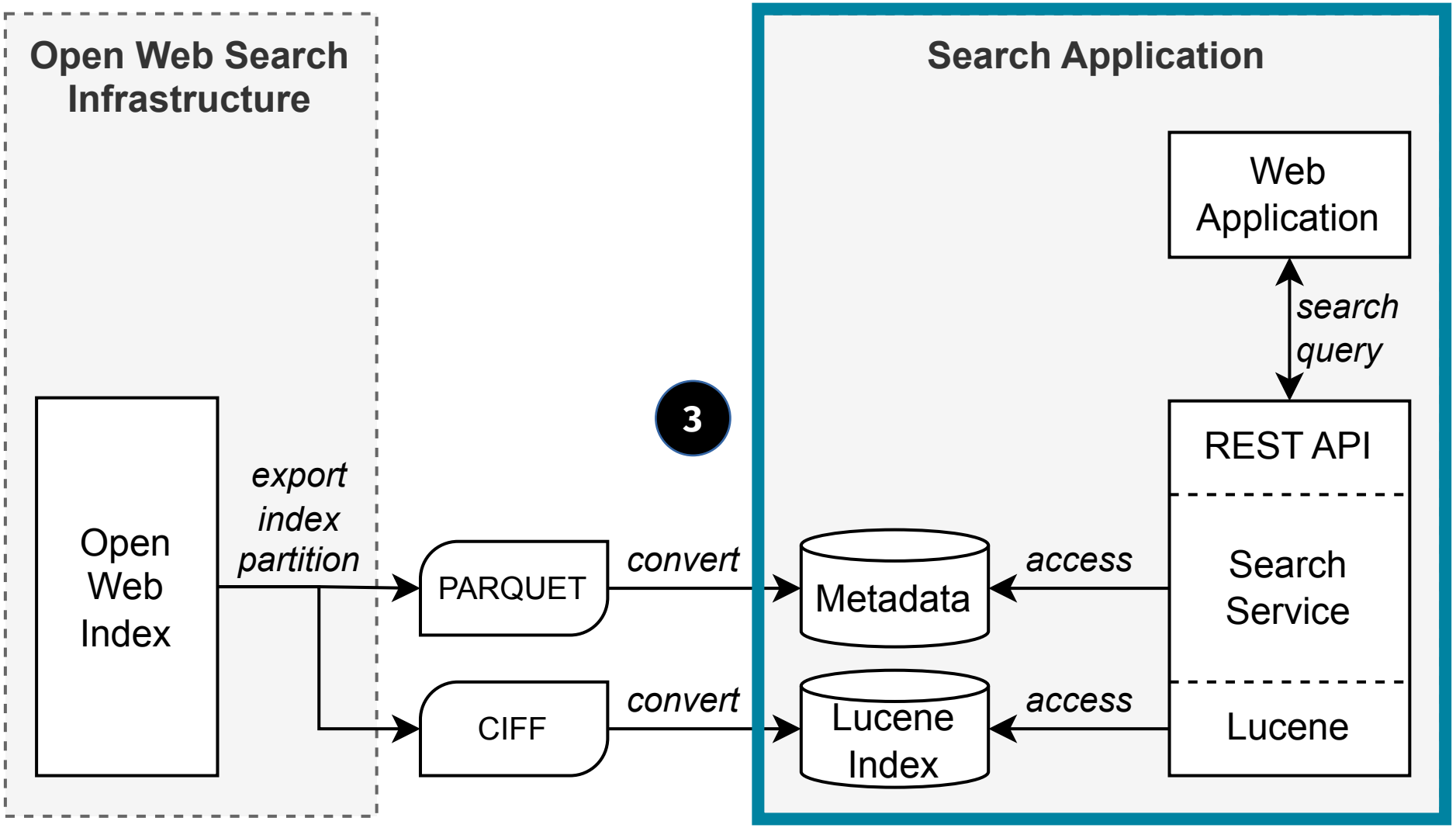
Ideas & Suggestions on What You Can Change

(2) Using the concept and development of the whole application



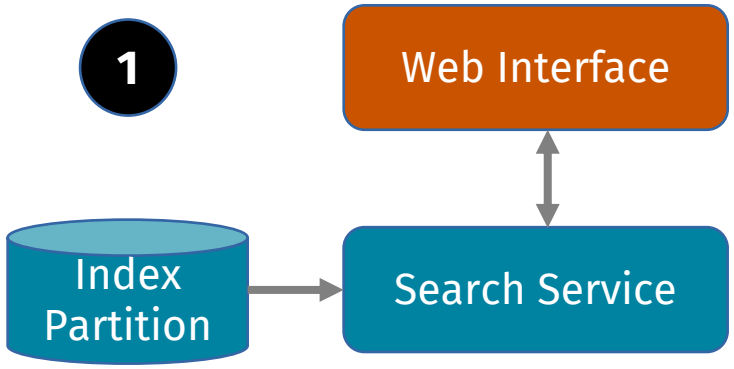
Ideas & Suggestions on What You Can Change

(3) Using the concept to explain the architecture of search applications



Ideas & Suggestions on What You Can Change

(1) Using the search service and development of a web interface

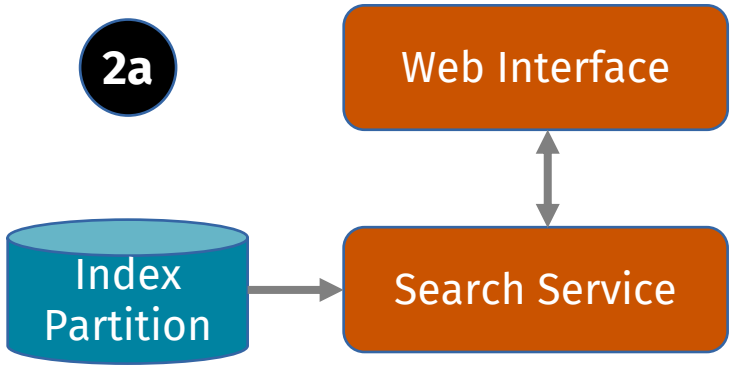


Web / User Interface:

- ➜ Change the existing web search interface
- ➜ Create a new web search interface from scratch
- ➜ Example: Replace list representation of search results
- ➜ Example: Re-build a sightseeing application for Passau using OpenStreetMap

Ideas & Suggestions on What You Can Change

(2) Using the concept and development of the whole application



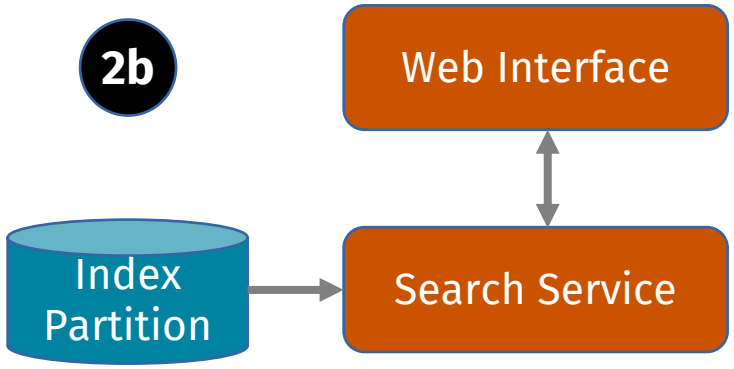
API Parameters

➜ Add additional parameters or change the functionality of existing parameters

➜ Example: Filter by Top-Level-Domains (as API parameter and in web interface)

Ideas & Suggestions on What You Can Change

(2) Using the concept and development of the whole application

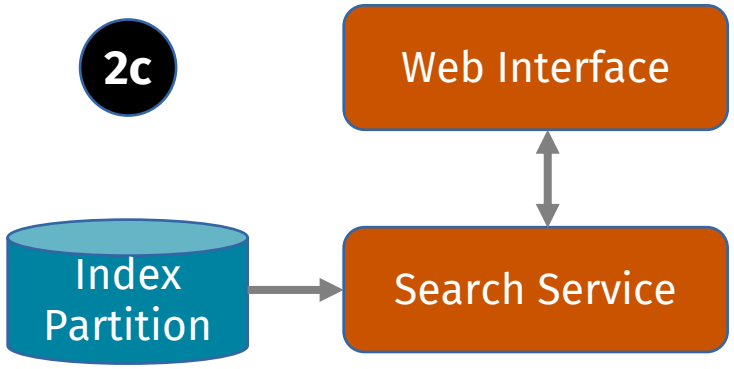


Re-Ranking

- ➜ Perform re-ranking not (only) based on word count of the document
- ➜ Example: Add additional re-ranking mechanism(s) that can be selected for re-ranking in web search interface
- ➜ Example: Further analysis of full text (e.g., perform a sentiment analysis)

Ideas & Suggestions on What You Can Change

(2) Using the concept and development of the whole application



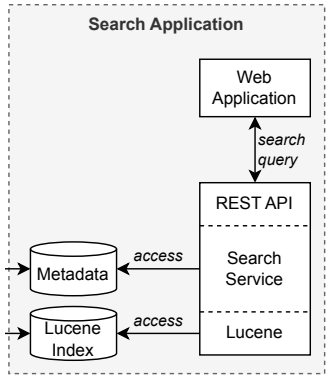
Text Snippet

- ➜ Use a more meaningful text snippet than just the beginning of the document
- ➜ Example: Use a text snippet from the full text that contains the queried term at least once
- ➜ Example: Automatically generate a summary of the full text

Ideas & Suggestions on What You Can Change

(3) Using the concept to explain the architecture of search applications

3a



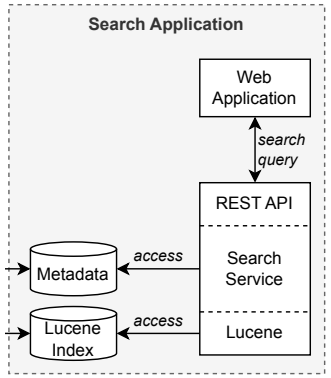
Conceptual Design (Software Architecture)

➜ Design an alternative concept based on the existing software architecture or from scratch

Ideas & Suggestions on What You Can Change

(3) Using the concept to explain the architecture of search applications

3b



Conceptual Design (Use Cases)

➜ Design (and implement) a concept of a use case

➜ Example: Conversational Search Application using an LLM API (e.g., ChatNoir Chat [1], OWLer GPT [2])

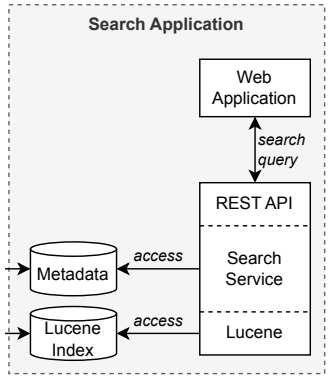
[1] https://chat.web.webis.de/

[2] https://owlergpt.pads.fim.uni-passau.de/

Ideas & Suggestions on What You Can Change

(3) Using the concept to explain the architecture of search applications

3c



Conceptual Design (Use Cases)

- ➜ Design (and implement) a concept of a use case
- ➜ Example: Geo-annotation based on full text (e.g., using the API of geonames.org)
- ➜ Example: Topic extraction based on full text (e.g., search and/or filter by topic)

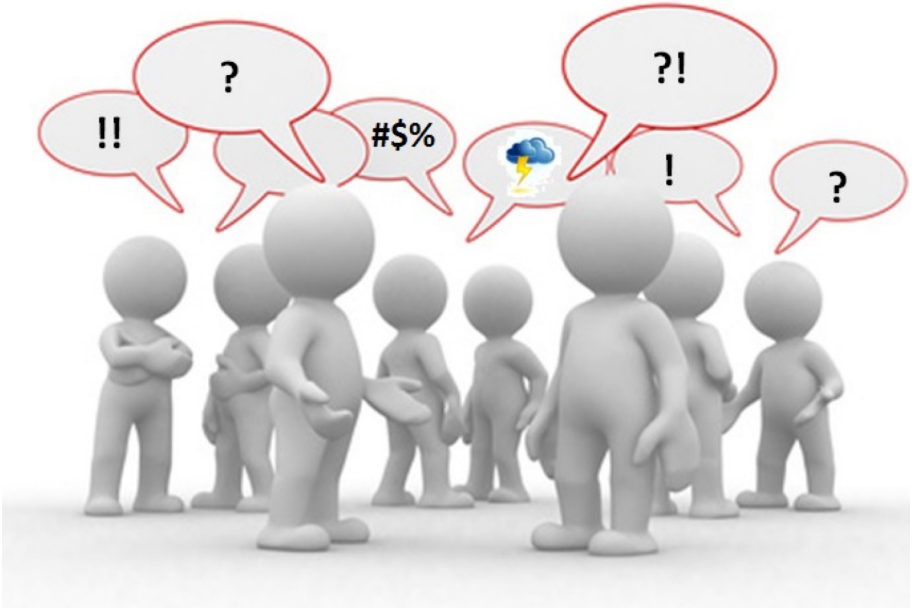
Entry Points In The Code

- MARKER-PARAMETERS
- MARKER-LUCENE-SEARCH
- MARKER-QUERY-PARSING
- MARKER-METADATA-ENRICHMENT
- MARKER-LANGUAGE-FILTER
- MARKER-RERANKING
- MARKER-TEXT-SNIPPET

```
/**********************
* MARKER-<ID>         *
* Some description.  *
**********************/
```

A description for each marker can be found in the source code

Any Questions?



Contact:

- ➜ Sebastian Gürtl <sebastian.guertl@tugraz.at>
- ➜ Alexander Nussbaumer <alexander.nussbaumer@tugraz.at>
- ➜ Graz University of Technology, Austria