

# Search Engine

ULPGC

Authors: Kimberly Casimiro Torres, Víctor Gil Bernal, Jacob Jażdżyk and María Alonso León

Big Data

October 2024

Data Science and Engineering





- 1. Introduction
- 0

Conclusions



- 2. Objective
- 0

6. Future Work



- Modules
- ()

4. Tests

6

 $\equiv$ 

A powerful book search engine designed to optimize performance and handle large datasets efficiently using cutting-edge data structures.









### Performance Optimization

 We implemented improvements in query handling, enabling faster search in large data volumes

#### Appropriate Data Structures

 By comparing File System, MongoDB, and Neo4j, we selected the most efficient structure for each type of query (inverted index vs. metadata)

### Modular Design with SOLID Principles

 By comparing File System, MongoDB, and Neo4j, we selected the most efficient structure for each type of query (inverted index vs. metadata)

### Extensive Testing

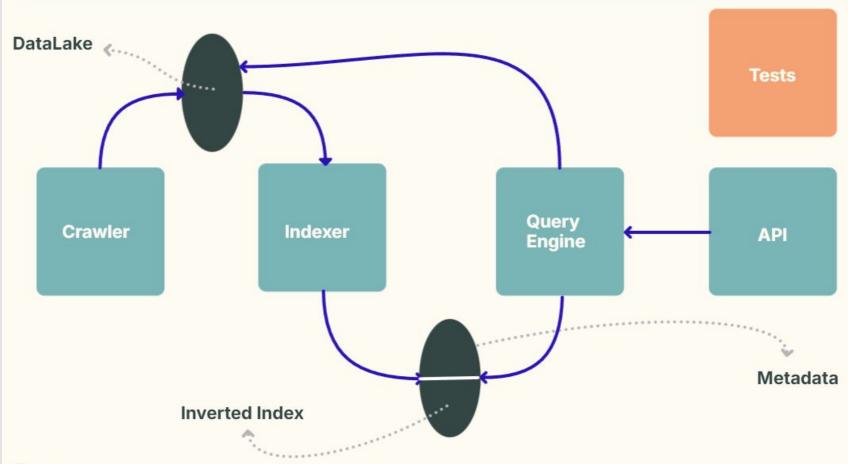
 We conducted performance tests to identify bottlenecks and optimize processing and query times

### Enhanced User Experience

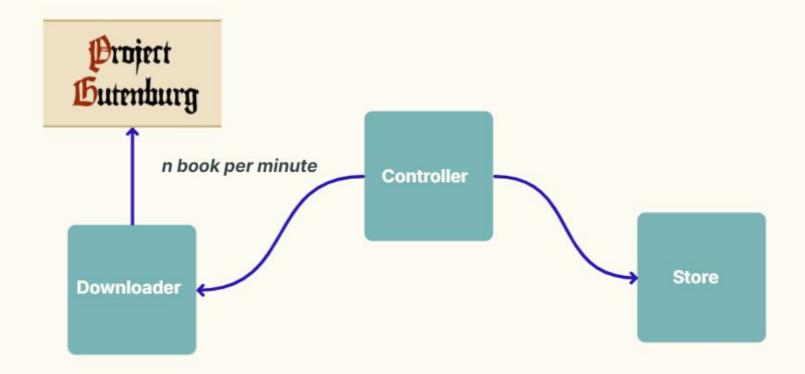
 We developed a user-friendly and intuitive interface, improving the interaction with queries and results



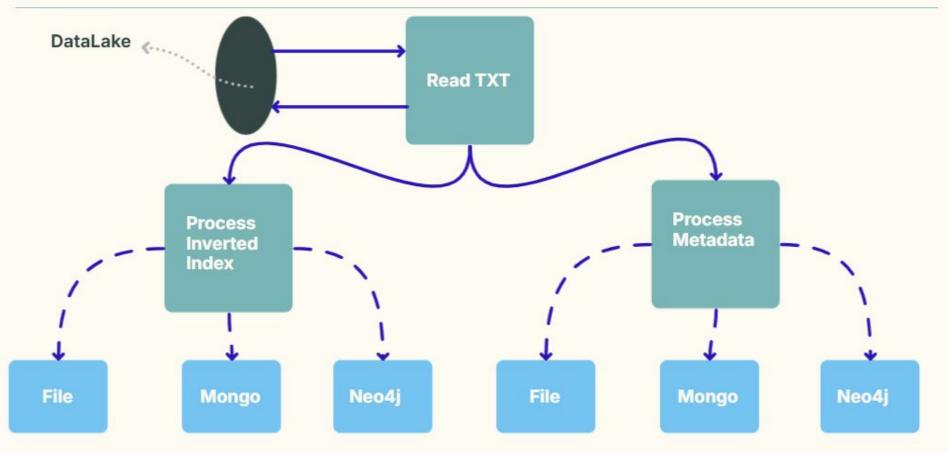








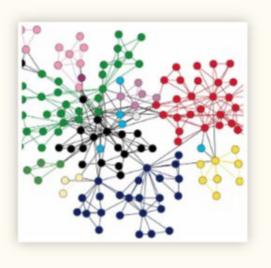










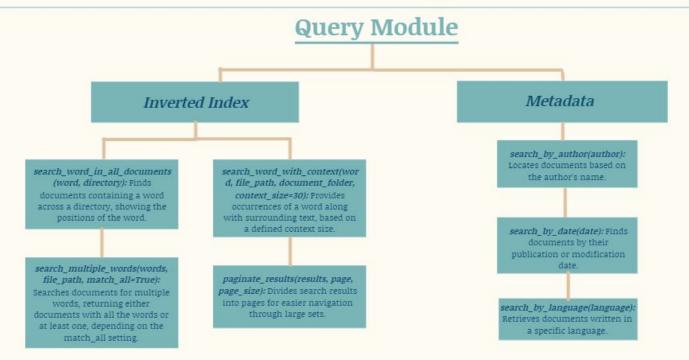


File System

Mongo DB

Neo4j







```
=
```

```
127.0.0.1:8000/files/search/?word=bill
         "results": {
    "bill": {
                   "book_4": {
    "positions": [
                             287,
                             1495,
1577,
                             1639,
10
                             1652,
11
                             1783,
12
                             2298,
13
                             2513
14
                         "count": 8
15
16
17
18
19 }
```

```
127.0.0.1:8000/neo4j/search/?word=bill
        "results for 'bill'": [
                  "book_4",
                       287.
                       1495,
                      1577.
                      1639,
10
                      1652,
11
                      1783,
12
                      2298,
13
                       2513
14
15
16
17
18 }
```

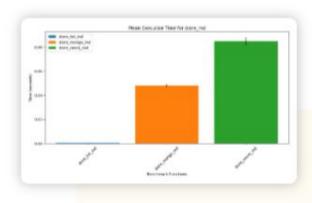
```
127.0.0.1:8000/mongodb/search/?word=bill
1 2 3 4
        "results for 'bill'": [
                  "book_id": "book_4",
                  "positions": [
                               287,
                               1495,
10
                               1577,
11
                               1639,
                               1652,
13
                               1783,
14
15
16
17
18
                               2298,
                               2513
19
20
21
22 }
```



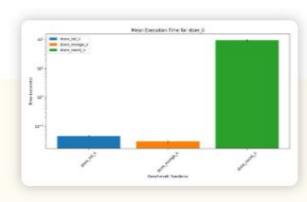
### 3.4 API Module - Context Search



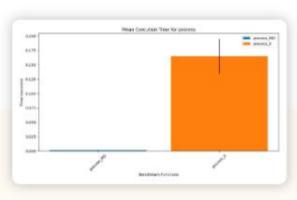
# **Time Results**



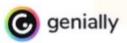
Graph 1



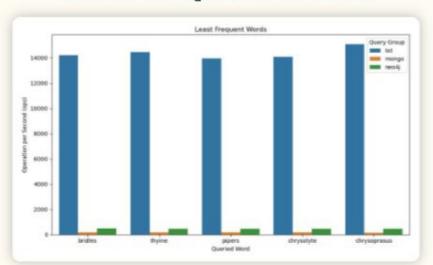
Graph 2

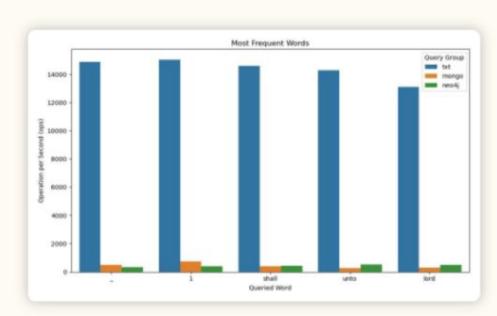


Graph 3



# Least Frequent Words



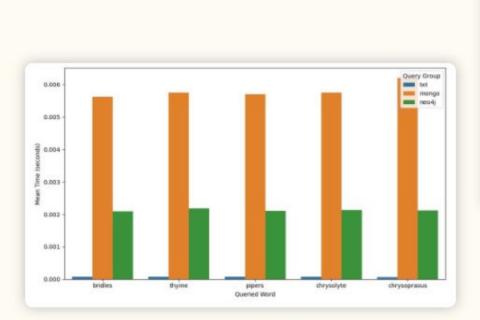


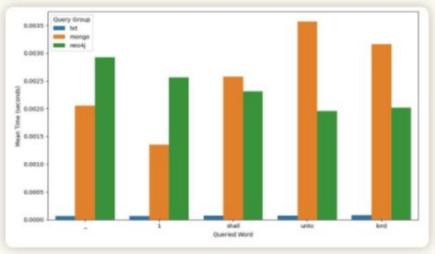
**Most Frequent Words** 



### $\equiv$

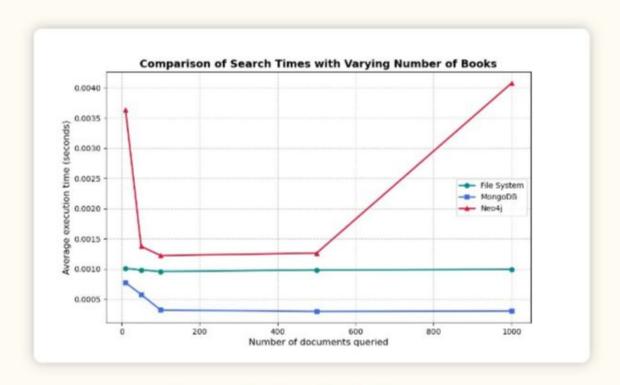
# **Most Frequent Words**





**Least Frequent Words** 





**Search Scalability** 





Performance Differences

## Metadata Processing Eficiency





Scalability with Increased Data





#### Java

Implementing the system in Java to improve performance and better follow SOLID principles.

### Docker

Using Docker to streamline deployment, ensure consistency, and enable easier scaling.

### Creating the UI

Creating a UI will make the application more user-friendly.

### Context Query

Enhancing the context query feature to make it more user-friendly and intuitive, similar to Google's search results.





TOTAXXX CLUAXOLICOBEDECOGESECOFOR COECE BOLLE APPEDOUGLE LONGO

TITE 1 AST. NET TOP ? CProfessional cortexille decepaling a cortex

