CME LIVE OPEN ACCESS

Engineering and Applied Sciences Journal

Research Article

ISSN: 3067-8005

An Information Retrieval System for E-Learning

Abdelmadjid BOUDINA[1*] and Ahmed MELILI[2]

[1]*Department of Continuing Education and Audio-visual, CERIST (Scientific and Technical Information Research Center), 05, rue des freres Aissou, Ben Aknoun, 16029, Algiers, Algeria.*

[2]*ADEX Cloud, Adex Technology, 2eme etage, Batiment Multilocataire, Cyberparc Sidi Abdellah, Rahmania, 16121, Zeralda, Algeria.*

Corresponding Author: Abdelmadjid BOUDINA, Department of Continuing Education and Audio-visual, CERIST (Scientific and Technical Information Research Center), 05, rue des freres Aissou, Ben Aknoun, 16029, Algiers, Algeria.

Received: 2025 Dec 15 **Accepted:** 2025 Dec 25 **Published:** 2026 Jan 13

Abstract

The amount of information available on the Internet is growing. Finding information has become extremely challenging in an ever-growing volume of data. This is caused by a number of factors, chief among them being indexing issues and the way the search engine behaves when looking for pertinent content. In this paper, we present an information retrieval system (IRS) for e-learning, and we have implemented several strategies to improve the system's accuracy and relevance while also decreasing its response time. In essence, our method is predicated on a fragmentation of the global data index. In fact, this index is broken up into multiple pieces, the number of which corresponds to the number of basic learning entities. To do this, every search query is run concurrently across all of the previously specified index pieces. Additionally, micro-services are distributed via network ports via an APIs gateway to ensure flawless synchronization between the database and the search engine. Moreover, in order to improve the score calculation and the field weight adjustment to produce more reasonable results, a weighting policy for the terms was implemented. Positive outcomes were achieved.

Keywords: Information Search, Web Database E-Learning, Search Engine

1. Introduction

In an E-Learning system, solutions must be found to store and structure the large mass of information, so that quick and relevant searches can be carried out in response to learners' queries. It is therefore a question of efficiently managing the volume of search requests made by users in order to propose an information search system (IRS) robust in terms of availability and flexible with regard to expansion. Historically, information retrieval is connected to information science and librarianship, which seek to present documents to extract information, through the development of indexes. The computer science evolution has enabled the development of tools to process information and establish the representation of documents during their indexing. Subsequently, a synchronization is carried out with the search engine as a method of information extraction. Therefore, information retrieval (IR) consists of finding among the large volume of available documents, those that best meet the needs of users in a minimum of time as a response to a user request. Thus, with the ever-increasing demand for access to information, the speed of response is becoming an increasingly pressing factor. Thus, developers are constantly looking for techniques to achieve faster indexes and faster query response times. Finally, the quality of the search task is greatly affected by the user's interaction with the system. Therefore, a better understanding of user behavior will affect the design and deployment of new information search strategies. This paper focuses on the development of an information retrieval system dedicated to E-Learning that meets the requirements of users and administrators of the platform in terms of both index construction and search relevant information in response to user needs. Section 2 presents the general concepts relating to E-Learning and its interactions as well as information retrieval techniques and underlines the important contribution of an information retrieval system (IRS) in the proper functioning of a platform of E-learning. Section 3 will be devoted to the study of our design of the search engine, in which it is explained in detail its implementation in a way adapted for E-Learning. Section 4 is devoted to the implementation part where the technologies and tools deployed for the needs of the implementation of our system will be discussed [1,2]. We finally conclude in section 5 with a conclusion and outlooks.

2. E-Learning and its interactions

It is a new form of online and distance learning using the Internet and new digital technologies to improve the learning process. E Learning draws its appeal from the fact that it gives the learner the opportunity to learn at their own pace, on

their computer, educational content on various subjects. Organized in sessions or modules, with assessment tests. The training can be completely self-managed and followed via a dashboard that lists each of the learner's advances. An E-learning platform contains courses and documents in all areas, thus offering users the possibility of making a search on the entire mass of documents stored in a database. Therefore, this new situation invites us to overcome the challenges we will face such as: How to save the huge number of documents, how to structure the E-learning content and How to make the search fast enough and relevant to the users.

2.1. Information Retrieval

Information retrieval (IR) concerns the representation, retrieval, and manipulation of large collections of electronic text and data in human language. IR systems and services are widespread today, and millions of people depend on them daily [1,3]. Web search engines are by far the most popular and widely used IR services. They provide access to up-to-date technical information, locate people and organizations, and summarize news and events. There are also other use cases such as digital library systems that help academic users learn about new journal articles and conference presentations related to their research areas [4].

2.2. Search Engine

The search engine is a necessary tool in the E-Learning knowledge and information system. It makes it possible to manage in the best possible way the backup, the search and the presentation of the data. Its components as well as its operation are designed and modelled in such a way as to properly manage indexing and search operations [5].

2.3. Web Search

Regular users of web search engines expect to receive precise and almost instantaneous answers to their queries through a search interface. This simple and intuitive interface is based on clusters of computers, comprising thousands of machines, which work in cooperation to generate a ranked list of web pages likely to meet the information need expressed in the request. These machines identify a set of web pages containing the query terms, calculate a score for each page, eliminate duplicate and redundant pages, generate summaries of the remaining pages, and finally return the summaries and links to the user so that he can exploit them. In its initial design, the web allowed a user to retrieve information. Nevertheless, the operations of sorting, classifying and extracting what is relevant to him were done manually. This task generally required a lot of hard work, whereas a search engine can produce thousands of results automatically [6].

2.4. Difference between Search Engine and Database

The search engine and the database are data persistence tools; however, one cannot replace the other, each having its own use case and its own specifications. While the data in the database (SQL) is stored as rows in structured tables, the search engine uses storage spaces where the data is saved in sometimes-unstructured documents. In addition, the entities stored in the database are linked to each other by means of foreign keys, facilitating the transition of an entity to other entities to which it is linked. In the storage space dedicated to the search engine, on the other hand, the notion of foreign keys is absent.

2.5. Structure and operation of a Search Engine

A search engine is a set of entities and tools that interact with each other to accomplish the indexing of new resources as well as the search for data already indexed as illustrated in figure 1 below.

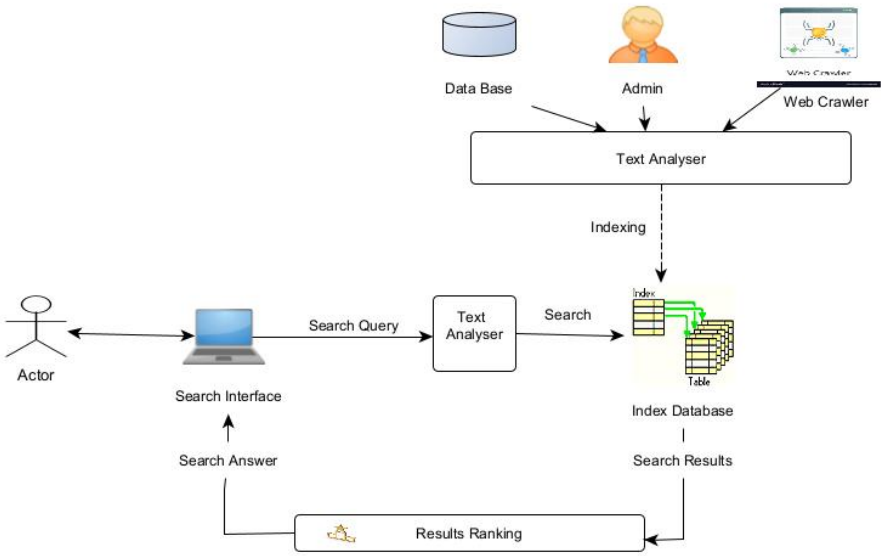

Figure 1: Structure of a Search Engine

2.5.1. Information Gathering

The addition of new data to the space dedicated to the search engine is done, either manually by an administrator or automatically by a "WEB Crawler" for example, or simply by extracting data from an already existing database [7].

2.5.2. Indexing of New Data

Indexing is an indispensable and very important operation in the ecosystem in which the search engine operates. After the introduction of new data, a pre-processing of these data must be carried out by a Text-Analyzer and then weighted and finally indexed in the index database [8].

2.5.3. Text-Analyzer and the Human Language Processing Process

It aims to analyze texts in order to extract machine-readable facts. The objective is to create structured data from free textual content. The process consists of cutting piles of unstructured and heterogeneous documents into data elements that are easy to manage and interpret. Text analysis is performed in several steps as follow [8,9].

- **Step 1.** Tokenization (Segmentation): Extraction of terms, these are the indexes used during the search [10].
- **Step 2.** Removal of "empty words" (Stop Word): Words with a very high frequency of appearance, having little information on the content, they will be compared to a Stop-List [11].
- **Step 3.** Standardization (Normalization): Process transforming all the words of the same family into a normal or canonical form in order to be able to make the pairing between the terms of the index and those of query [12].

2.5.4. Weighting

Here are the most important approaches for calculating the weight of a term.

- **Approach Based on the Frequency of Occurrences of the Term in the Document - Term Frequency (TF):** Simple and classic, it consists of choosing the words represented according to their frequency of occurrences [13].
- **Approach Based on Logarithmic Normalization:** In this approach, the raw frequency can be logarithmically normalized to dampen deviations [14].
- **Approach based on the TF-IDF:** The TF-IDF approach combines the two criteria, namely the frequency of appearance of the term, the "TF" and the frequency of the term in the collection by Inverted Document Frequency, we measure whether a term distinguishes well a document of the other documents [15].

2.5.5. Inverted Index

Once the weights of the terms have been calculated, the final phase of indexing follows, i.e. the saving of the weighted terms pointed to their documents in the index database. For this purpose, the most common technique is to use the reverse file [16]. As shown in Table 1 bellow, in a text search system, the system's reverse file is provided by a Word List and postings file where.

List of words (Word List), index file or Vocabulary = set of all distinct words; Postings (Occurrences) = lists containing all the necessary information on each word of the vocabulary.

Word List				Positing		
Word	Number of Documents	Global Frequency	Pointer	Document	Frequency	Position
Blue	3	5	@1	1	2	15

Table 1: Example of Inverted Index

2.6. Search Query

To perform a search query on previously indexed data, each query is processed by a Text-Analyzer, usually the same one used in the indexing phase, to extract all the search keywords. Then, the search engine will create an internal representation for a document or for a query based on these terms. Then, define a method of comparison between a document representation and a query representation in order to determine their degree of correspondence (or similarity). This is defined as our information seeking model [17,18]. An example is given in figure 2 bellow.

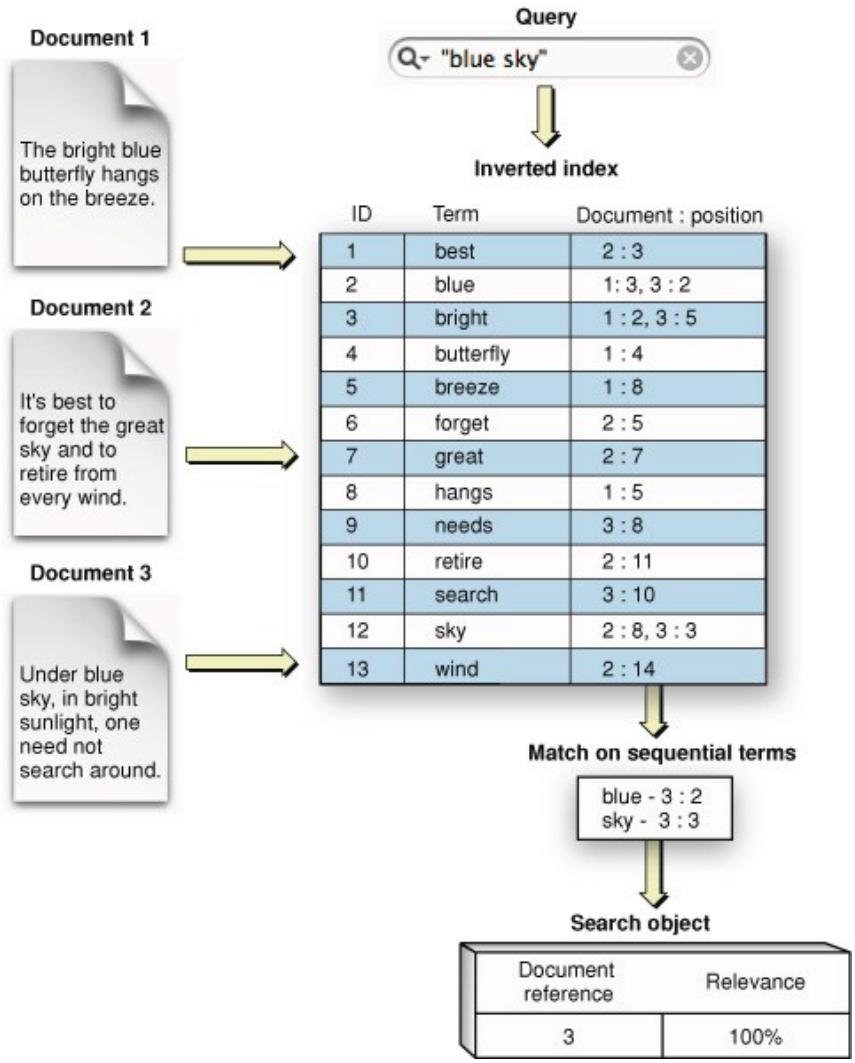

Figure 2: Illustration of an Indexing Operation with a Search Query

2.7. Information Seeking Model

There are three basic processes an information retrieval system has to support the representation of the content of the documents, the representation of the user's information need, and the comparison of the two representations. The processes are visualised in Figure 3 bellow [19].

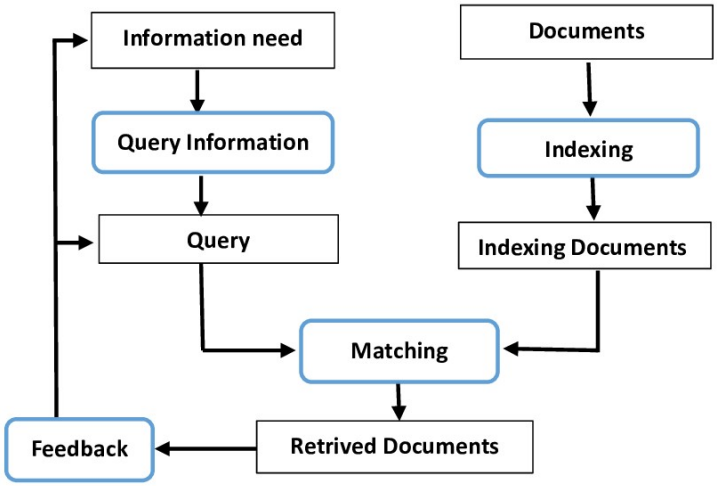

Figure 3: Information Retrieval Processes

In this figure, squared boxes represent data and rounded boxes represent processes. An Information Retrieval models are distinguished by the following matching principle
[19].

• **Exact Matching:** the query precisely specifies the search criteria, all the documents that exactly match the query are selected, but not ordered;

• **Approximate Matching (Best Matching):** the query describes the criteria sought in a document, the documents are selected according to a degree of relevance (similarity).

There are three main IR (Information Retrieval) models, namely : The Boolean models, also known as set models, The vector model or Vector Space Retrieval Model (VSM) and the probabilistic model [20-24].

2.8. Evaluation of Information Retrieval Systems

The validation of a new SRI is based on the experimental evaluation of its performance. In recent years, this evaluation has been a very active field of research; it makes it possible to estimate the impact of each of the characteristics of a system and to provide objective parameters for comparison between the various existing systems. The evaluation can cover several criteria of efficiency and effectiveness, which are generally constructed from the judgments expressed by users or by experts. We can mention: the relevance of the results, the quality of the presentation of the results and the performance, which in turn affects several criteria concerning the consumption of resources, such as response time, memory space, load capacity, etc. The most important criterion that undoubtedly interests the user the most is the one that measures the ability of an IRS to satisfy his information need. It is about the relevance of the results returned by this system [25]. Consequently, information retrieval (IR) is centered on this notion of relevance, which defines the degree of correspondence between a search query and the documents in the index. In response to this query, the list of results can be divided into four sets: relevant or not (the user says Yes/No), selected or not (the system says Yes/No). These different situations are summarized in the following table, called the contingency table (see Table 2 below).

	Relevant	Not Relevant
Selected	True positive	False positive
Not Selected	False Negative	True Negative

Table 2: The Contingency Table

From these values, different metrics can be calculated, making it possible to evaluate the relevance of the results. The two most commonly used metrics in IR are; the precision rate and the recall rate. The first one attempts to answer the question: how many elements are relevant among those selected and measures the system's ability to reject all irrelevant documents, while the second one attempts to answer the question: how many relevant elements are selected and measures the system's ability to select all relevant documents.

$$Precision\,rate(P) = \frac{true\,positive}{True\,positive + False\,positive} = \frac{Number\,of\,relevant\,documents\,selected}{Number\,of\,all\,relevant\,documents} \quad (1)$$

Where :
True positive = number of documents containing the search term selected.
False positive = number of documents not containing the search term selected.

$$Recall\,rate(R) = \frac{true\,positive}{True\,positive + False\,Negative} = \frac{Number\,of\,relevant\,documents\,selected}{Number\,of\,all\,relevant\,documents} \quad (2)$$

Where :
True positive = number of documents containing the search term selected.
False negatives = number of documents containing the search term not selected.

Recall is not a sufficient measure, as it does not show that there are irrelevant documents among the results. These two metrics are dependent on each other, when one increases, the other decreases. In other words, the greater the noise, the lower the accuracy, and vice versa. Various studies have shown that the use of these two measures alone is insufficient. This criticism concerns the inadequacy of the binary evaluation on which these measures are based (relevant and irrelevant); the authors believe that certain documents found may be partially relevant. Work has attempted to improve these measures and others have proposed new ones. Let us note, in this respect, the ESL (Expected search length) measure proposed by [26].

Which corresponds to the number of irrelevant documents that the user must browse through in the list of results before accessing a relevant document [3]. In the same vein, another rank-based measure has been proposed, often used in question-answer systems, it is the MRR (Mean Reciprocal Rank) metric. It is used to evaluate the average rank of the first relevant document in the list of results, calculated over all the queries. It is defined as follows.

$$MRR = \frac{1}{|Q|}\sum_{i=1}^{Q}\frac{1}{rank_i} \quad (3)$$

Where :
MRR = Mean Reciprocal Rank.
$|Q|$ = The total number of queries (questions or searches). rank i = The position (rank) of the first relevant result for query number i.
The value of MRR is high for a system that gives relevant results at the top of the returned list, and it is zero if no relevant documents were found.

3. Conception

In addition to the mechanisms adopted in the modeling of our approach to improve the relevance and accuracy and decrease the response time of the information retrieval system when processing a user query, we have also implemented other techniques related to typo tolerance, word suggestion, synonyms and abbreviations management as well as the manipulation of empty words or "Stop Word"

3.1. E-Learning Content Structure

In our E-learning system, the structure is defined according to the following fundamental entities.
• **Field:** groups together courses of the same specialty, e.g. medicine, IT, etc.
• **Course:** the main entity described by a name, a level (beginner, intermediate, advanced), and a description.
• **Chapter:** each course is composed of one or more chapters described by a Title and a content.
• **File:** each chapter can contain one or more files, described by name, download link, extension and content.

3.2. Platform Structure Design

Since the search engine is not suited to operations involving the addition, modification or reading of relations between entities, it is more rational to have two storage entities, which, moreover, must be permanently synchronized. While the BDD (database) is intended to carry out this type of operation, the search engine will be oriented mainly to searching for information. Consequently, every add or modify request will be applied first to the database, then to the storage space dedicated to the search engine as illustrated in figure 4 bellow.

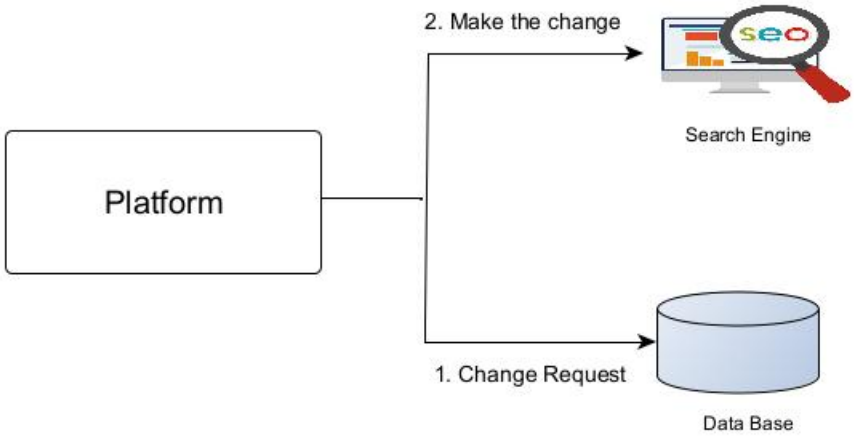

Figure 4: Platform Structure

3.3. Defining user roles

We distinguish two types of users
• **The platform administrator (ADMIN):** responsible for modifying the content present at the database level (adding, deleting, modifying, etc.).
• **The Visitor: (learner or consultant of the courses):** can just carry out search requests on the courses present at the level of the database and consult them.

3.4. Database Design

It is described below by the use case, class, and sequence diagrams.

3.4.1. Use case diagram

This is the main form of system/software requirements that we relied on to specify the expected behavior of our system. This is an effective technique for communicating system behavior in user terms by specifying all externally visible system behavior as illustrated in figure 5 bellow [27,28].

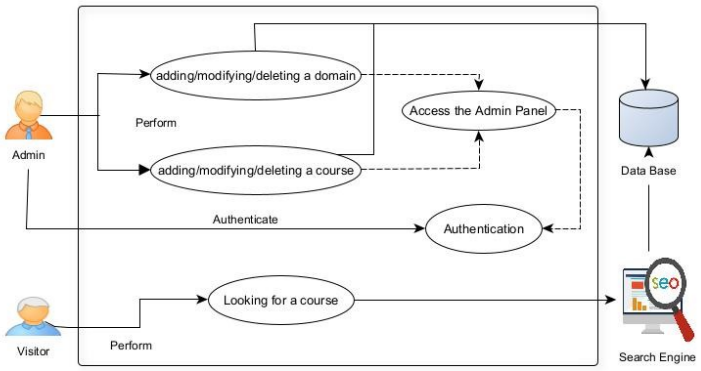

Figure 5: Use Case Diagram

3.4.2. Class Diagram

Used in the design and modeling of our system to describe classes and their relationships. Class diagrams allow us to model software at a high level of abstraction and without having to look at the source code as illustrated in Figure 6 bellow [27].

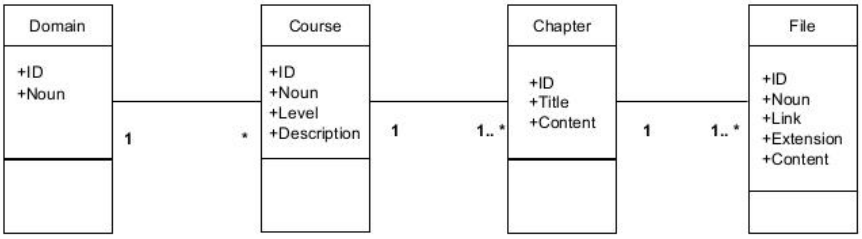

Figure 6: E-learning Class Diagram

3.4.3. Sequence Diagram

Used to describe the sequence of inter-object messages in an interaction. It consists of a group of objects represented by lifelines, and messages exchanged over time during the interaction as shown in figure 7 [27,28].

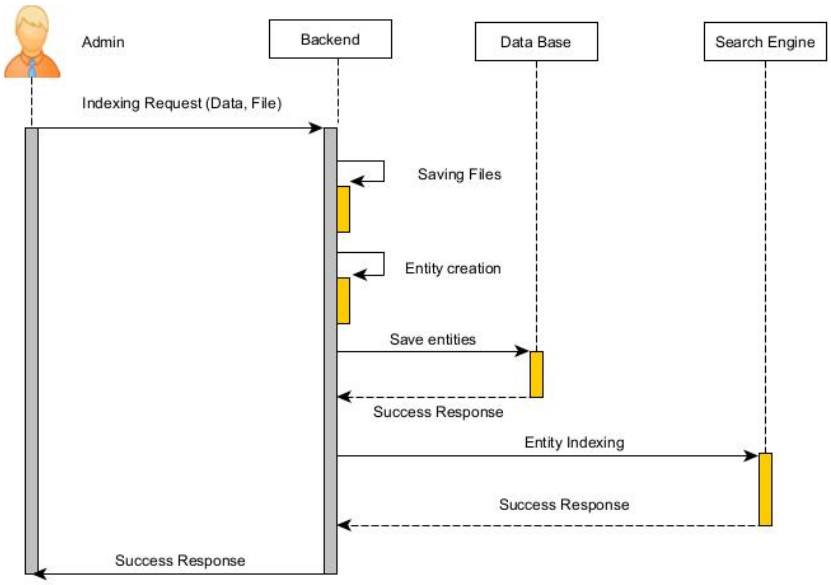

Figure 7: Sequence Diagram for Indexing Operation

3.4.4. Process

ADMIN sends a request for indexing data and files following mandatory authentication, the request is sent to other entities. Starting with the BACKEND, which will be responsible for recording and storing all the data, contained in the request, it first saves the files in its space, and then it creates the entities of the course and saves them in the Database. Once the backup has been successfully completed in the database, an operation success response is returned to the BACKEND, which completes the process by sending an indexing request to the search engine.

3.5. Search engine adapted for E-learning

- The design of a search engine for E-learning must meet several requirements, namely.
- A search query must be performed on all entities containing answers related to the search query.
- Answers must be structured and classified in a hierarchical form to represent relationships between course elements.
- Ensuring better use of hardware resources, in particular the consumption and management of storage space.
- The ability to index data when creating in several different formats means that the files associated with the chapter are in PDF, DOCX, PPT, etc.

3.5.1. Data Index

Considering the above, it is clear that as CRUD (Create - Read - Update - Delete) operations evolve, the index volume will increase with each new indexing, causing a lack of space at the level of the storage entity. In addition, the search engine may also encounter difficulties in its search for relevant information and its attempt to structure and classify the information in a hierarchical form to represent the relationships between the elements of the course. For this purpose, we thought that the fragmentation of the global index solves this problem by dividing it into four (04) indexes so that each of the 04 indexes is linked to an entity of the general E-Learning structure as illustrated by figure 8.

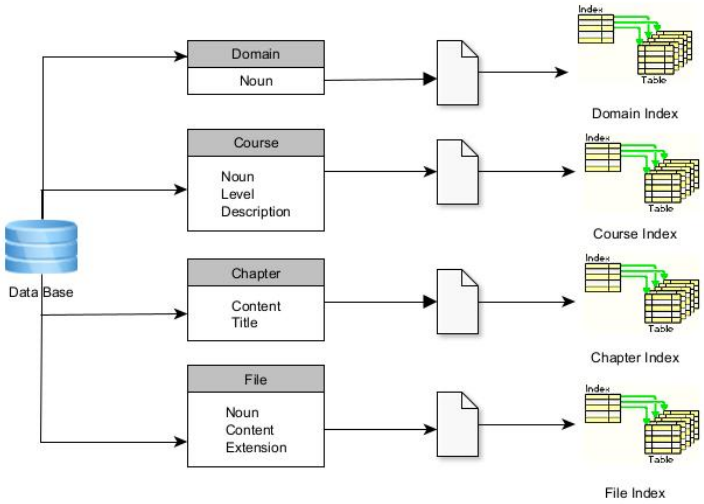

Figure 8: Creating and Extracting Indexes from the Database

Therefore, the documents associated with each index carry the same attributes as the tables in the database. Thus, each of them contains a subset of data from the global index and is in itself fully functional and independent. As such, the 04 indexes correspond to the different E-Learning entities previously defined, namely the Domain, the Course, the Chapter and the File. Each of the four (04) indexes contains only the data of the entity to which it refers. This structuring of the index allowed us to divide and therefore evolve the volumes of data without the risk of being confronted with a "bottleneck". In addition, each search query is launched simultaneously on the four (04) index fragments. At the end of the operation, the responses related to the course sought are merged to be presented to the user. In terms of time savings, the use of four (04) indexes proved to be more efficient than the use of a single index. Indeed, by using four (04) search processes on the four indexes simultaneously, the query processing operation will be faster than when the system launches a single search operation on a single index knowing that the single index must contain all platform data.

3.5.2. Data Indexing Process

Upon receipt of an indexing request bearing the integrity of the information concerning a given course (course data, chapter and files associated with each chapter), the system extracts the data from each entity to form index documents in order to index them (see Figure 9).

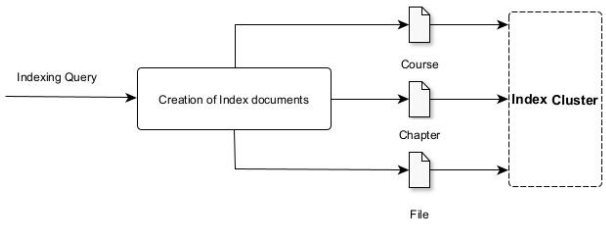

Figure 9: Search Request Process

3.5.4. Score Calculation and Term Weight Adjustment

In this regard, an adjustment is made to obtain more logical and rational results. This amounts to calculating the weight of each term appearing in the fields of the different E-Learning entities. In our design, we have made it possible, depending on the E-Learning entity involved, to assign more weight to a given term when it appears in one field than in another. For example, to calculate the weight of a term appearing in the "chapter" entity, the weight assigned to a given term appearing in the chapter title is greater than that assigned to the same term appearing in the content, since at this level, this same term may not have a direct relationship with the subject of the article. Adjustments have been made to that effect.

Course score: for courses, the search is carried out on two (2) fields: the name and the description. Here, an attribute present in the name of the course will have more weight than when it is present in the description.

$$Score(Course, term) = Coeficient * ScoreField(Nom, term) + ScoreField(Description, term) \quad (4)$$

After a number of trials with different values in order to obtain the most coherent and convincing result possible, we finally opted for the following values:

Name coefficient = 2 and description coefficient = 1

Chapters score: for chapters, the search is carried out on two (2) fields: the title and the content. Here, an attribute present in the chapter title will have more weight than when it is present in the content.

$$Score(Chapter, term) = Coeficient * ScoreChamp(Titre, term) + ScoreField(Content, term) \quad (5)$$

Using the same evaluation method as above, we have chosen the following values:
Title coefficient = 3 and content coefficient = 1

Files Score

For files, the search is carried out on two (2) fields: the name and the content. Here, an attribute present in the name of the course will have more weight than when it is present in the content.

$$Score(File, term) = Coeficient * ScoreField(name, term) + ScoreField(Content, term) \quad (6)$$

In a similar way, we have chosen the following values here:
Name coefficient = 2 and description coefficient = 1

3.6. Search Enhancement

To improve the search process, several methods and techniques have been implemented to make the search task easy and user-guided to give approximate results if none is found.

3.6.1. Typo Tolerance

To find terms similar to search terms that may contain typos, the search must be typo-tolerant at a certain threshold, such as.
Deletion of a character, example : (vision ⇒ visio)
Modifying a character, example : (web ⇒ wep)
The insertion of a character, example : (computer ⇒ computeir)
Permutation between two (2) adjacent characters, example : (act ⇒ cat)

3.6.2. Suggested Terms

For an optimal user experience, we can provide suggestions for terms close to those present in the search query. The usefulness of this technique is to bring the user closer to an existing term found in the index if the terms sought do not exist. The term suggestion implementation follows the same strategy as typo tolerance for finding terms to suggest (see figure 11).

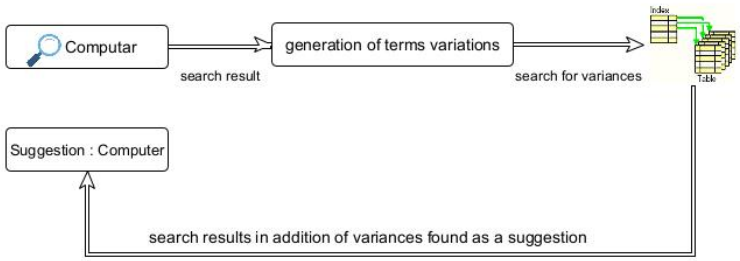

Figure 10: Suggested Terms

3.6.3. Management of Synonyms and Abbreviations

The search engine must manipulate synonyms and abbreviations in search queries. In addition, a user can perform a search query that does not necessarily have the same indexed terms. For this purpose, the search term may be an abbreviation or synonym, such as.

isr ⇒ Information Retrieval System pc ⇒ computer nature ⇒ bio/organic to manage this case, a list of synonyms has been created by the ADMIN user, which list is provided to the Text-Analyzer so that then during the indexing and research phase the system will be able to detect whether it is the search word itself or one of these synonyms or abbreviations.

3.6.4. The manipulation of Stop Words

Each term in a query has a cost. For example, a search for the sequence: "The Mathematical logic" requires three queries; one for each of the terms "the", "logic", and "mathematical", all of which are performed on all documents in the index. The query for "The" is likely to match many documents and therefore has a much lower impact on relevance than the other two terms. Previously, the solution to this problem was to ignore high-frequency terms. By treating "The" as an empty word. Because of this, we reduce the size of the index and the number of term queries to run. However, even if stop words have a small impact on relevance, they are still important; by removing them, we will lose precision. The solution to this problem is to create a new request by dividing the terms of the request into two groups: the most important (the low-frequency terms) and the least important (the high-frequency terms that would previously have been stop words). This solution first searches for documents that match the most important terms appearing in fewer documents and having a greater impact on relevance. Then it runs a second query for the less important terms - appearing frequently and having little impact on relevance. But instead of calculating the relevance score for all matching documents, it only calculates the score of documents already matched by the first query. In this way, high-frequency terms can improve the relevance calculation without paying the price for poor performance. If a query has only high frequency terms, it is executed as an AND (conjunction) query, i.e. all terms are required. Even though each individual term corresponds to many documents, the combination of terms reduces the set of results to the most relevant [11].

4. Implementation

In our conception, we have made sure to combine the different technologies as well as the methods and techniques implemented in order to set up a platform that satisfies the needs of learners while taking care of the problems mentioned above. We have also integrated other tools to monitor the different entities of the system in order to be able to evaluate and optimize its reliability and performance.

4.1. Tools and Technologies Used

We implemented the system as a web application by deploying several web technologies (see figure 12) [29,30]. As such, we have implemented micro-services through network ports through an APIs portal. Each one is deployed in a port by opening an APIs portal allowing the technologies implemented to communicate with each other [31].

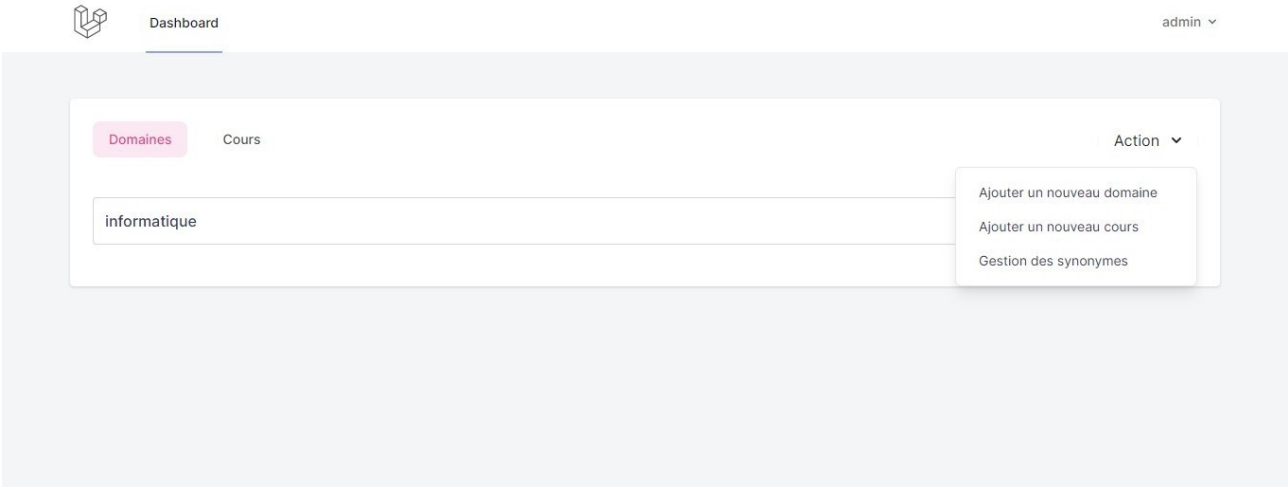

Figure 11: Application Interface

4.1.1. The Tools Used

• **Elasticsearch:** A distributed search and analysis engine based on Apache Lucene, Known for its simple REST APIs, distributed nature, speed and scalability. It is the central component of the Elastic stack; it is a set of open tools for data management, enrichment, storage, analysis and visualization [32,33].
• **Kibana:** A front-ended, open application located on top of the Elastic stack. Provides data search and visualization capabilities for indexed data in Elastic search. Kibana also acts as a user interface for monitoring, managing, and securing an Elastic Stack cluster, as well as a centralized hub for integrated solutions developed on the Elastic stack [34,35].
• **File Beat:** A lightweight service that allows the transmission and centralisation of log data. Installed as an agent on servers, File beat monitors log files or locations that we specify, collects log events and forwards them to Elasticsearch for indexing [36,37].
• **Metric Beat:** A lightweight service that periodically collects metrics from the operating system and services running on the server. Metric beat sends the metrics and statistics it collects to a specific output [36,38].
• **Laravel:** A PHP framework. It follows a model-view-controller design pattern. Laravel reuses existing components from different frameworks, making it easy to build a web application. The web application thus designed is more structured and pragmatic [39,40].
• **React:** React.js is a JavaScript library, used to create user interfaces specifically for single-page applications. Also used to manage the visualization layer of web and mobile applications. React also allows us to create reusable user interface components. Fast, scalable and simple, only works on the application's user interfaces. This corresponds to the view in the MVC model [41,42].
• **Flas:** A lightweight WSGI web application Framework, designed to allow quick and easy handling, while offering the possibility to evolve into complex applications. Flask offers suggestions but does not impose any project dependencies or layouts [43].
• **PostgreSQL:** A powerful object-relational database system that uses and extends the SQL language combined with many features to securely store and scale the most complex data workloads [44].

4.2. The adopted IR Model

Elasticsearch implements a probabilistic model, the BM25 model in this case [45].

4.2.1. BM25

BM25 is a ranking function that ranks a set of documents based on the query terms appearing in each document, regardless of the interrelationship between query terms within a document (e.g., their relative proximity). This is not a single function, but in fact, a whole family of scoring functions, with slightly different components and parameters. It is used by search engines to rank matching documents according to their relevance to a given search query and is often referred to as "Okapi BM25", because the Okapi information retrieval system was the first system to implement this function [46]. The BM25 search formula belongs to the BM (Best Match) family of search models, i.e. the weight of a term "t" in a document "d" is calculated according to the formula.

$$Score(D,q) = \sum_{i}^{n} .IDF(q,i.\frac{f(qi,D).(k1+1)}{f(qi,D)+k1.(1-b+b.\frac{fieldLength}{avg\ fieldLength})}) \qquad (7)$$

Where
Score(D, q): the relevance score of document D with respect to query q. i ∈ q : we sum over each term i of the query.
IDF(i) : Inverse Document Frequency.
f(qi,D): frequency of term i in document D (i.e. how many times it appears). k1: frequency saturation parameter = 1.2. b: document length normalization parameter = 0.75. field Length: length of the document D (i.e. total number of words).

Avg Field Length: Average length of documents in the collection.
We can see some known components like qi, IDF (qi), f (qi, D), and something about field lengths. Here is what each of these components corresponds to:
"qi" is the i-th query term.
IDF (qi) is the inverse document frequency of the i-th query term, calculated as follows.

$$ln(1+\frac{Number\ of\ Document - f(qi)+0.5}{f(qi)+0.5}) \qquad (8)$$

Where
– IDF(qi): measures the importance of the term qi in all documents.
– Number of Documents: total number of documents in the corpus. – f(qi): number of documents containing the term qi.

The variable b which appears in the denominator and which is multiplied by the ratio of the length of the field, which we have just discussed. If b is larger, the effects of document length relative to average length are more magnified. To understand this, we can imagine that if we set b to zero, the effect of the length ratio would be completely nullified, and the length of the document would have no impact on the score. By default, b has a value of 0.75 in Elasticsearch. "k1" is a variable that helps determine the saturation characteristics of term frequency. That is, it limits the influence of a single query term on the score of as given document. figure 13 below shows a comparison of term score calculation using the BM25 and TF-ID methods.

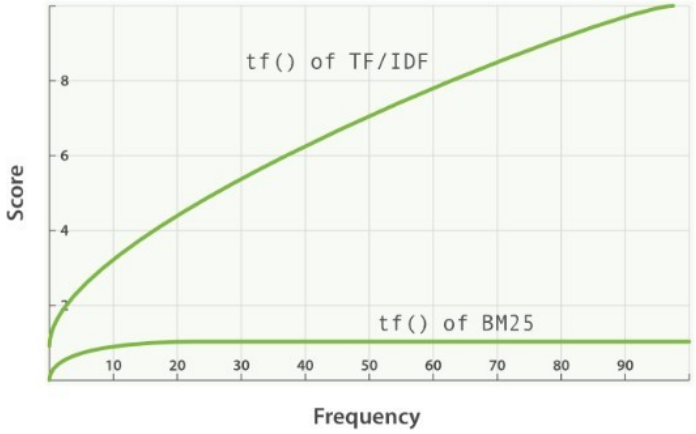

Figure 12: Comparison Between BM25 and Tf-IDF in Term Score Calculation

4.3. Implementation of the Solution

The system as we have conceived it is illustrated by the (see figure 14) bellow.

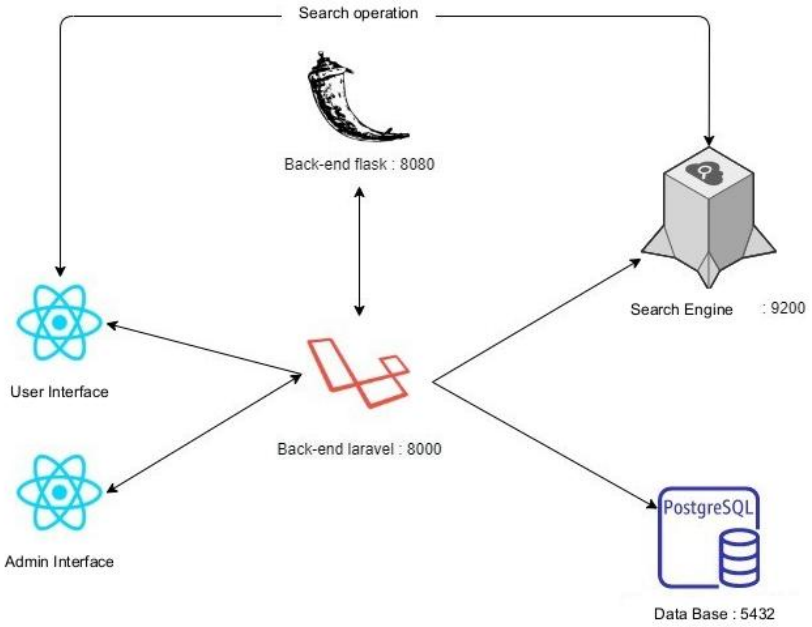

Figure 13: Solution Structure

4.3.1. The Role Performed by Each Entity (Micro-Service)

• **Backend – Laravel:** It manages both interfaces (User/ Administrator) when executing an administrator request. LARAVEL synchronizes between the search engine and the database with each new addition of information, modification or deletion. The operation is initially performed on the database then on the search engine to keep the synchronization between the two entities.

• **Backend – Flask:** Since Laravel (PHP) is not powerful enough in file processing, we used a second backend, FLASK, in this case, which will handle and extract information from the files added to each new course.

• **Search Engine:** This part contains all the indexed data allowing it to perform all search operations.

• **Data Base:** Where data is saved and CRUD operations (Create, Read, Update and Delete) are performed.

• **User and Administrator Interface:** the application interface for user and administrator (ADMIN) is entirely designed with ReactJS using more than forty (40) JavaScript utility libraries. Following the concept of React components, the interface has been cut into sub-components, which makes page handling fast and efficient. This allowed us to do

the search in real time. Search operations do not go through the Laravel Backend but directly from the interface to the search engine, due to concerns of saving execution time and reducing the load on the Backend part, as illustrated by the figure 15.

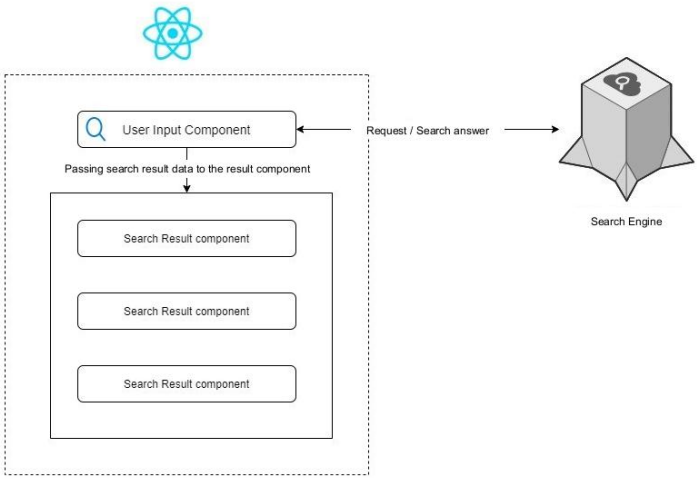

Figure 14: Real-Time Search Data Flow

4.4. Synchronization Between Database and Search Engine

For a good synchronization between the database and the search engine from the Backend, the **Laravel Scout** tool was used to provide the driver and the configuration of **Elasticsearch**. Laravel Scout provides a simple driver-based solution for adding a search on Laravel models. Using model observers, Scout will automatically synchronize search engine indexes with database records [47].

4.5. Security

Requests for creation, modification and deletion on all indexes must be secured to ensure that no request has been received from a non-administrative user. For this, a middleware is set up to control all requests received at the Backend level. It checks whether the request bears the authorization token in its header, which token is transmitted to the user during authentication. In which case, the middleware allows it to run otherwise an error message is returned.

4.6. Monitoring

The Administrator needs to have a global view of the operation of the system. For this, he must know in real time the statistics relating to the various entities in order to be able to intervene, if necessary, to correct the breakdowns, which can occur, and to make the system more efficient and accurate. These statistics can be summarized in terms of the exploitation of hardware resources such as the use of RAM, the processor, the consumption of hard disk storage space as well as the detection of errors found in entities collected in files. As such, **File beat** and **Metric beat** are practical tools for collecting logs created by the system and containing errors. File beat and Metric beat give the possibility to build Dashboards that contains statistics in **Kibana** from the collected data, as illustrated by figure 16.

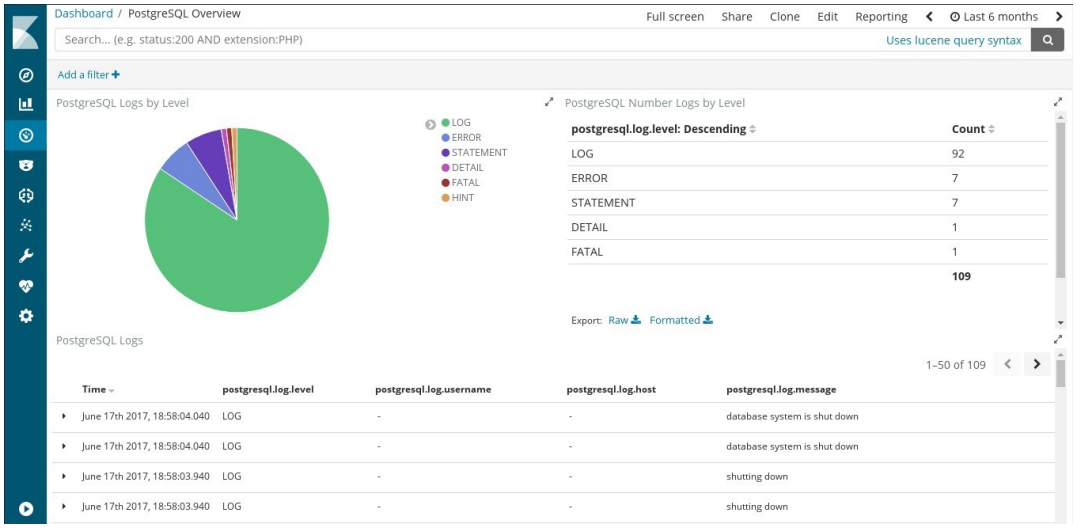

Figure 16: Kibana Dashboard

4.7. System Evaluation (Benchmarking)

During the progress of our application, we established a comparison between PostgreSQL and Elasticsearch and were able to observe, as illustrated in Figure 17 below, the significant gain in terms of response time to user requests due to the speed of indexing and response in Elasticsearch.

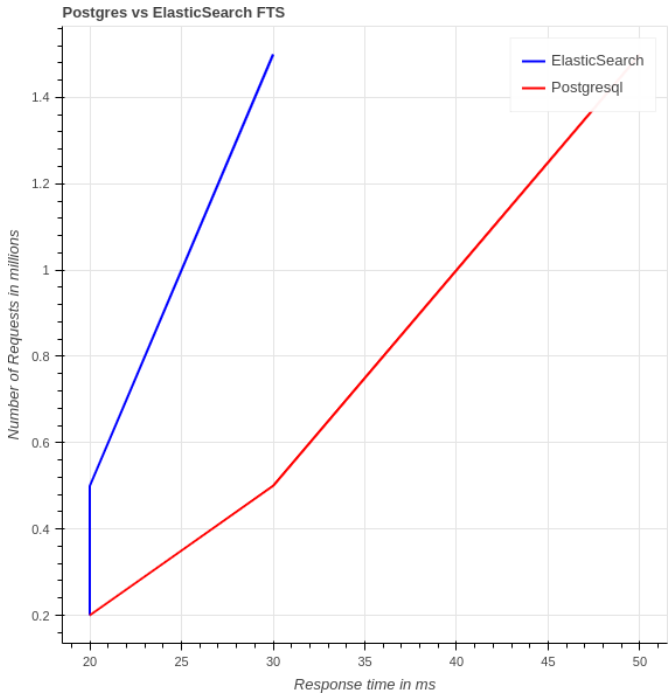

Figure 17: Difference in Response Time Between PostgreSQL and Elasticsearch in Relation to the Number of Queries

4.8. Results Obtained

Using the notion of precision to evaluate a search engine

$$Precision(P) = \frac{True\ Positive}{True\ Positive + FalsePositive} = \frac{Number\ of\ relevant\ documents\ selected}{Number\ of\ documents\ selected} \quad (9)$$

And after several tests, we obtained an approximate precision of 0.8 with the selection of 10 documents for each request.

5. Conclusion and outlook

In this article, we have sought to present a method that addresses the issue of information retrieval by suggesting an approach based on a particular indexing of data, associated with the establishment of mechanisms likely to promote optimal synchronization between the database system and the search engine. The major contribution of this project consists in the design and development of an SRI dedicated to E-Learning to facilitate the search for information within a growing quantity of data. The difficulties encountered by the learner in his search for appropriate information in an acceptable time, encourage us to improve the relevance and precision and reduce the response time of the information retrieval system. The advantage of the recommended approach consists in the adoption of several mechanisms as listed below. The global data index is fragmented into several sections, the number of which is proportional to the number of fundamental E-Learning entities defined. Also, each search request is executed simultaneously on all the predetermined index fragments; Deployment of micro-services through network ports using an API portal to ensure perfect synchronization between the search engine and the database; Implementation of a term weighting policy to improve score calculation and adjustment of field weights in order to obtain more logical results. The recommended approach brings a certain efficiency in the process of IR in relevance thanks to the indexing technique adopted as well as by its way of synchronizing between the database and the search engine via ports using APIs as well as the policy adopted for calculating the weight of scores and adjustment of field weights. A prototype, dedicated to E-Learning, concretizing this approach is developed. Our database is made up of four fundamental entities (”domain”, ”course”, ”chapter” and ”file”). It includes a single indexed area, namely computer science, comprising 53 courses, 142 chapters and 302 files. For this purpose, the modules with the following names have been indexed in the database and successfully tested. These are: bio-inspired computing, artificial intelligence, computer vision, network technology, advanced algorithm and web development. Queries have been completed on all courses. The tests carried out

locally (on an intranet) are quite satisfactory and the results obtained show that the initial objective has been achieved. Indeed, our approach has made it possible to return relevant documents while reducing the response time, estimated at 0.5 seconds on average. In the future, we intend to improve this work by first implementing certain extensions that were developed during the development. We can certify, by way of example, that the search engine called "document surfing" that we have developed will be able to authorize wildcard characters. As such, the learner, during his search for information, can introduce a "programming *" type request to find programming courses, algorithms, computer-assisted applications, the era of computers, computer circuits and many other subjects related to the programming theme. Moreover, in the system we have developed, only one user has the right to authenticate and create his profile, this is the administrator ADMIN, while the other users are anonymous. This is not without handicapping the system to access the user's search histories; therefore, we will have no information on similar preferences and themes that the system can suggest. Also, to solve this problem, we can allow users to create their own profiles and the searches will thus become compatible with the themes of the old searches.

Declarations

• **Ethics Approval:** Our study did not include any human and/ or animal studies. All datasets used in the paper are publicly available for research purposes.
• **Conflict of Interests:** The authors declare that they have no conflict of interest.
• **Data Availability:** All datasets used in this paper to support the findings are publicly available. Links are reported in the bibliography.
• **Funding:** This study was conducted without any financial support.

Author contribution

A.B. conceived the idea presented. All authors developed the theory and performed the calculations. All authors verified the analytical methods. All authors wrote the article.

Reference

1. Kelly, D. (2009). Methods for evaluating interactive information retrieval systems with users. *Foundations and Trends® in Information Retrieval, 3*(1–2), 1-224.
2. Loughran, T., & McDonald, B. (2016). Textual analysis in accounting and finance: A survey. *Journal of accounting research, 54*(4), 1187-1230.
3. Bollmann, P., & Raghavan, V. V. (1988, May). A utility-theoretic analysis of expected search length. In *Proceedings of the 11th annual international ACM SIGIR conference on Research and development in information retrieval* (pp. 245-256).
4. Krichel, T. (2007). Information retrieval performance measures for a current awareness report composition aid. *Information processing & management, 43*(4), 1030-1043.
5. Palanisamy, R. (2013). Evaluation of search engines: a conceptual model and research issues. *International Journal of Business and Management, 8*(6), 1.
6. Almukhtar, F., Mahmoodd, N., & Kareem, S. (2021). Search engine optimization: a review. *Applied computer science, 17*(1), 70-80.
7. Menshchikov, A., Komarova, A., Gatchin, Y., Korobeynikov, A., & Tishukova, N. (2017, April). A study of different web-crawler behaviour. In *2017 20th Conference of Open Innovations Association (FRUCT)* (pp. 268-274). IEEE.
8. Basile, P., De Gemmis, M., Gentile, A., Iaquinta, L., Lops, P., & Semeraro, G. (2008). META-MultilanguagE Text Analyzer. In *Proceedings of the Language and Speech Technnology Conference-LangTech* (pp. 28-29).
9. Cunningham, H. (2005). Information extraction, automatic. *Encyclopedia of language and linguistics,, 3*(8), 10.
10. Singh, V., & Saini, B. (2014). An effective tokenization algorithm for information retrieval systems. *Departement of Computer Engineering, National Institute of Technology Kurukshetra, Haryana, India.*
11. Kaur, J., & Buttar, P. K. (2018). Stopwords removal and its algorithms based on different methods. *International Journal of Advanced Research in Computer Science, 9*(5), 81-88.
12. Aliero, A. A., Bashir, S. A., Aliyu, H. O., Tafida, A. G., Bashar, U. K., & Nasiru, M. D. (2023). Systematic review on text normalization techniques and its approach to non-standard words.
13. Ibrahim, O. A. S., & Landa-Silva, D. (2016). Term frequency with average term occurrences for textual information retrieval. *Soft Computing, 20*(8), 3045-3061.
14. Jian, F., Huang, J. X., Zhao, J., & He, T. (2018, June). A new term frequency normalization model for probabilistic information retrieval. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval* (pp. 1237-1240).
15. Ramos, J. (2003, December). Using tf-idf to determine word relevance in document queries. In *Proceedings of the first instructional conference on machine learning* (Vol. 242, No. 1, pp. 29-48).
16. Kaushik, R., Krishnamurthy, R., Naughton, J. F., & Ramakrishnan, R. (2004, June). On the integration of structure indexes and inverted lists. In *Proceedings of the 2004 ACM SIGMOD international conference on Management of data* (pp. 779-790).
17. Christoforaki, M., He, J., Dimopoulos, C., Markowetz, A., & Suel, T. (2011, October). Text vs. space: efficient geo-search query processing. In *Proceedings of the 20th ACM international conference on Information and knowledge management*

(pp. 423-432).
18. Long, X., & Suel, T. (2005, May). Three-level caching for efficient query processing in large web search engines. In *Proceedings of the 14th international conference on World Wide Web* (pp. 257-266).
19. Hiemstra, D. (2009). Information retrieval models. *Information Retrieval: searching in the 21st Century,* 1-19.
20. Lashkari, A. H., Mahdavi, F., & Ghomi, V. (2009, April). A boolean model in information retrieval for search engines. In *2009 International Conference on Information Management and Engineering* (pp. 385-389). IEEE.
21. Prajuli, R. (2008). *Boolean operator in logical and efficient information retrieval* (Doctoral dissertation, Central Department of Library and Information Science Faculty of Humanities and Social Sciences Tribhuvan University, Kirtipur, Katmandu, Nepal).
22. Castells, P., Fernandez, M., & Vallet, D. (2006). An adaptation of the vector-space model for ontology-based information retrieval. *IEEE transactions on knowledge and data engineering, 19*(2), 261-272.
23. Wong, S. K. M., Ziarko, W., Raghavan, V. V., & Wong, P. C. (1987). On modeling of information retrieval concepts in vector spaces. *ACM Transactions on Database Systems (TODS), 12*(2), 299-321.
24. Wen, J. R., Lao, N., & Ma, W. Y. (2004, July). Probabilistic model for contextual retrieval. In *Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval* (pp. 57-63).
25. Zuva, K., & Zuva, T. (2012). Evaluation of information retrieval systems. *International journal of computer science & information technology, 4*(3), 35.
26. Cooper, W. S. (1968). Expected search length: A single measure of retrieval effectiveness based on the weak ordering action of retrieval systems. *American documentation, 19*(1), 30-41.
27. Aquino, E. R., de Saqui-Sannes, P., & Vingerhoeds, R. A. (2020, February). A methodological assistant for UML and SysML use case diagrams. In *International Conference on Model-Driven Engineering and Software Development* (pp. 298-322). Cham: Springer International Publishing.
28. Swain, S. K., Mohapatra, D. P., & Mall, R. (2010). Test case generation based on use case and sequence diagram. *International Journal of Software Engineering, 3*(2), 21-52.
29. Al-Shaikh, A. A., & Sleit, A. (2017, May). Evaluating IndexedDB performance on web browsers. In *2017 8th International Conference on Information Technology (ICIT)* (pp. 488-494). IEEE.
30. Saroni, M. I. N., & Mulyanti, B. (2020, April). Hypertext preprocessor framework in the development of web applications. In *IOP Conference Series: Materials Science and Engineering* (Vol. 830, No. 2, p. 022096). IOP Publishing.
31. Patni, S.: Pro RESTful APIs. Springer (2017)
32. Olsson, J. (2019). Using Elasticsearch for full-text searches on unstructured data.
33. Kathare, N., Reddy, O. V., & Prabhu, V. (2020). A comprehensive study of elasticsearch. *International journal of science and research (IJSR).*
34. Robles-Gómez, A., Ros, S., Martínez-Gámez, A., Hernández, R., Tobarra, L., Pastor, R., ... & Cano, J. (2017). Using Kibana and ElasticSearch for the Recommendation of Job Offers to Students. In *LASI-SPAIN* (pp. 93-99).
35. Shah, N., Willick, D., & Mago, V. (2022). A framework for social media data analytics using Elasticsearch and Kibana. *Wireless networks, 28*(3), 1179-1187.
36. Mardianto, I., Sugiarto, D., & Ashari, K. A. (2021). The elastic stack ability test to monitor slowloris attack on digital ocean server. *Ultimatics: Jurnal Teknik Informatika, 13*(2), 120-126.
37. Hussein, M. A., & Hamza, E. K. (2022). Secure Mechanism Applied to Big Data for IIoT by Using Security Event and Information Management System (SIEM). *International Journal of Intelligent Engineering & Systems, 15*(6).
38. Calderon, G., Del Campo, G., Saavedra, E., & Santamaria, A. (2021, August). Management and monitoring IoT networks through an elastic stack-based platform. In *2021 8th International Conference on Future Internet of Things and Cloud (FiCloud)* (pp. 184-191). IEEE.
39. BOUDINA, A., & MELILI, A. (2025). An information retrieval system for e-learning.
40. He, R. Y. (2015, January). Design and implementation of web based on Laravel framework. In *2014 International Conference on Computer Science and Electronic Technology (ICCSET 2014)* (pp. 301-304). Atlantis Press.
41. Srivastava, A. K., Laxmi, V., Singh, P., Pratima, K., & Kirti, V. (2022). React JS (Open Source JavaScript Library). INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH IN TECHNOLOGY (IJIRT), 8(9).
42. Poudel, J.: Library management system with react. js (2023)
43. Kamble, A., Gagan, A., Wagh, P., Aziz, U., & Rajput, M. R. (2022). Flask Web Framework Based News Summarizer: Web Application. *Department of Computer Science and Engineering, PES College of Engineering, Dr. Babasaheb Ambedkar Technological University, Lonere, Raigad (MH), India, 5*(6), 953-957.
44. Senellart, P., Jachiet, L., Maniu, S., & Ramusat, Y. (2018). ProvSQL: Provenance and probability management in PostgreSQL. *Proceedings of the VLDB Endowment (PVLDB), 11*(12), 2034-2037.
45. Rostami, L. (2022). Investigating Search Algorithms for Shorter Documents: A study on how to search for titles.
46. Svore, K. M., & Burges, C. J. (2009, November). A machine learning approach for improved BM25 retrieval. In *Proceedings of the 18th ACM conference on Information and knowledge management* (pp. 1811-1814).
47. Hassen, F., & Amel, G. T. (2017). An efficient synchronous indexing technique for full-text retrieval in distributed databases. *Procedia computer science, 112*, 811-821.