الجـــمهوريـــة الجزائريـــة الديمـــقراطيـــــة الشـعبيـة
ⵜⴻⵣⴰⵢⵔⵉⵜ ⵜⴰⵣⴷⴰⵢⵔⵜ ⵜⴰⵎⴰⵣⵉⵖⵜ ⵜⴰⵡⵙⵙⵓⵔⵜ
République Algérienne Démocratique et Populaire
وزارة التعليـــم العـــالي والبـــحث العلمـــي
ⴰⵢⵏⵉⵎ ⵏ ⵜⵙⵙⵏⵎⵓⵔ ⵜⵉⵏⵣⵡⴰⵏ ⴰ ⵜⵙⴰⵏⵉ ⵜⵙⵙⵏⴰ
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

ECOLE NATIONALE SUPÉRIEURE D'INFORMATIQUE

المدرسة الوطنية العليا للإعلام الآلي
ⴰⵢⵜⵙⵙⴰⵙ ⴰⵢⵜⵏⵏⴰⵡ ⵜⵉⵏⵣⵡⴰⵏ ⵏ ⵜⵙⵜⵏⵜⵜⵓⵔⵉⵜ
École nationale Supérieure d'Informatique

1CS **2023/2024**

Article Harbor

Intro to Software Engineering ArticleHarbor: Scientific Articles Search Web Application Using Elastic Search

Made by:

BROUTHEN Kamel
MADANI Mohamed Mehdi
BENGHERBIA Abdelkarim
AKEB Abdelaziz
RAIS Mohamed Malek
FELLAH Mahdi

Supervised by:

Mr. BATATA Sofiane
Mrs. OUFAIDA Houda

1. Project Overview

For long centuries in science history, collecting information was one of the main concerns of a science practitioner, for the reason that it takes enormous time o assemble a decent database that the scholar can study and build upon. But this has changed to other extremes in recent times, there is just “too much” information to consume, and what institutes and presses get out in one day exceeds what a human can consume in 20 years of his life. But researching and studying without building on what others did is a failed mission, so how can the modern scholar gets exactly what he needs in this flood of shiny and tangled maze of knowledge?
We introduce **ArticleHarbor**, a web application solution that aims to help the researcher to navigate through an enormous and renewed database of scientific papers, with specific search queries varying from simple keywords to a complete advanced search through authors and institutions, performed on extracted text with high precision and fast results.

2. Project Functionalities

2.1. Authentication

- Client types: User, Moderator, Admin
- JSON Web Tokens (JWT) authentication

2.2. Administration

- Scientific articles upload (local PDFs / from URL)
- Moderators activation management

2.3. Moderation

- Correct/Delete inaccurately extracted scientific articles

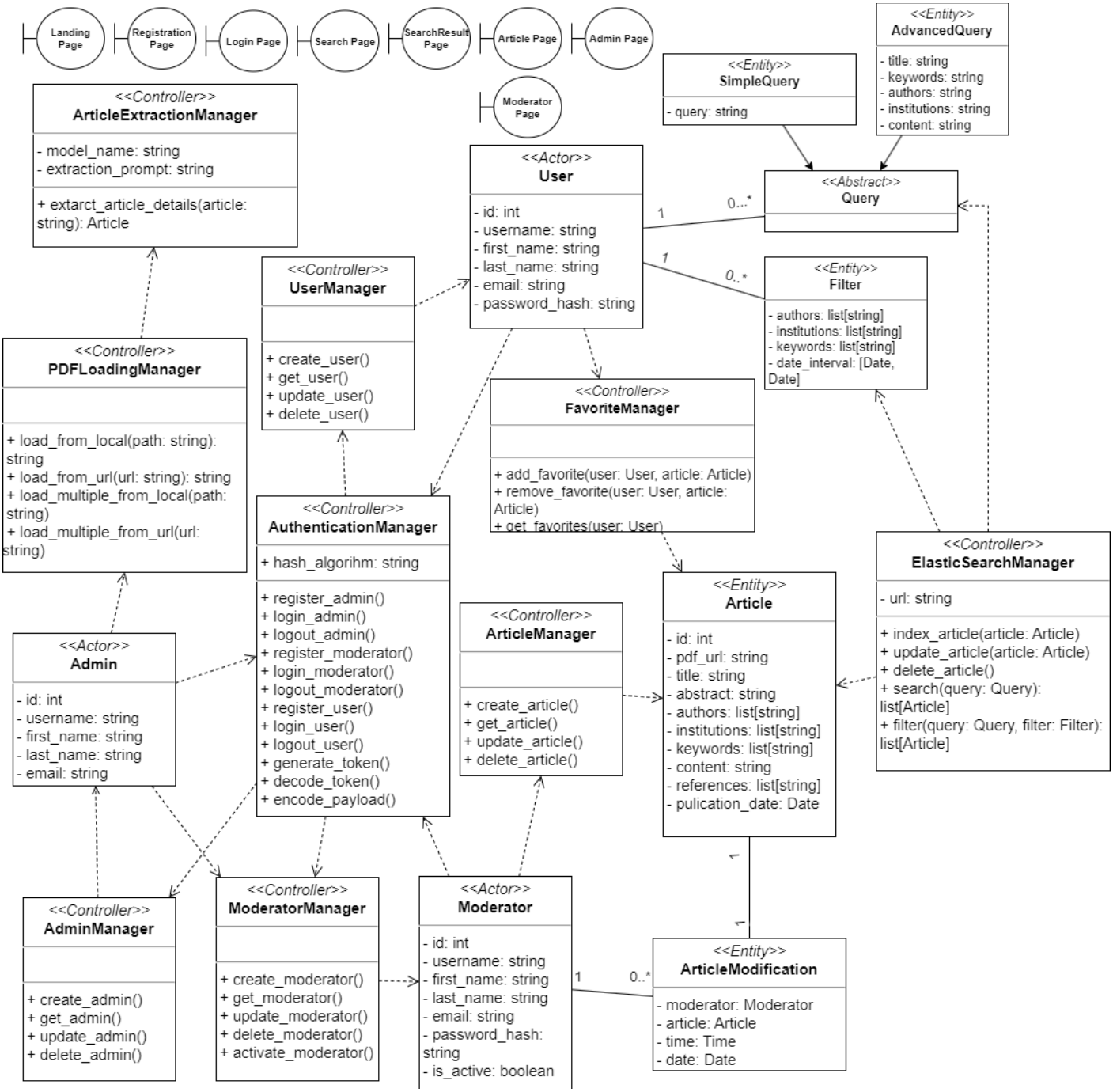
2.3. Usage

- Search scientific articles through simple/advanced queries
- Filter search results
- Save favoured scientific articles
- Display scientific articles details

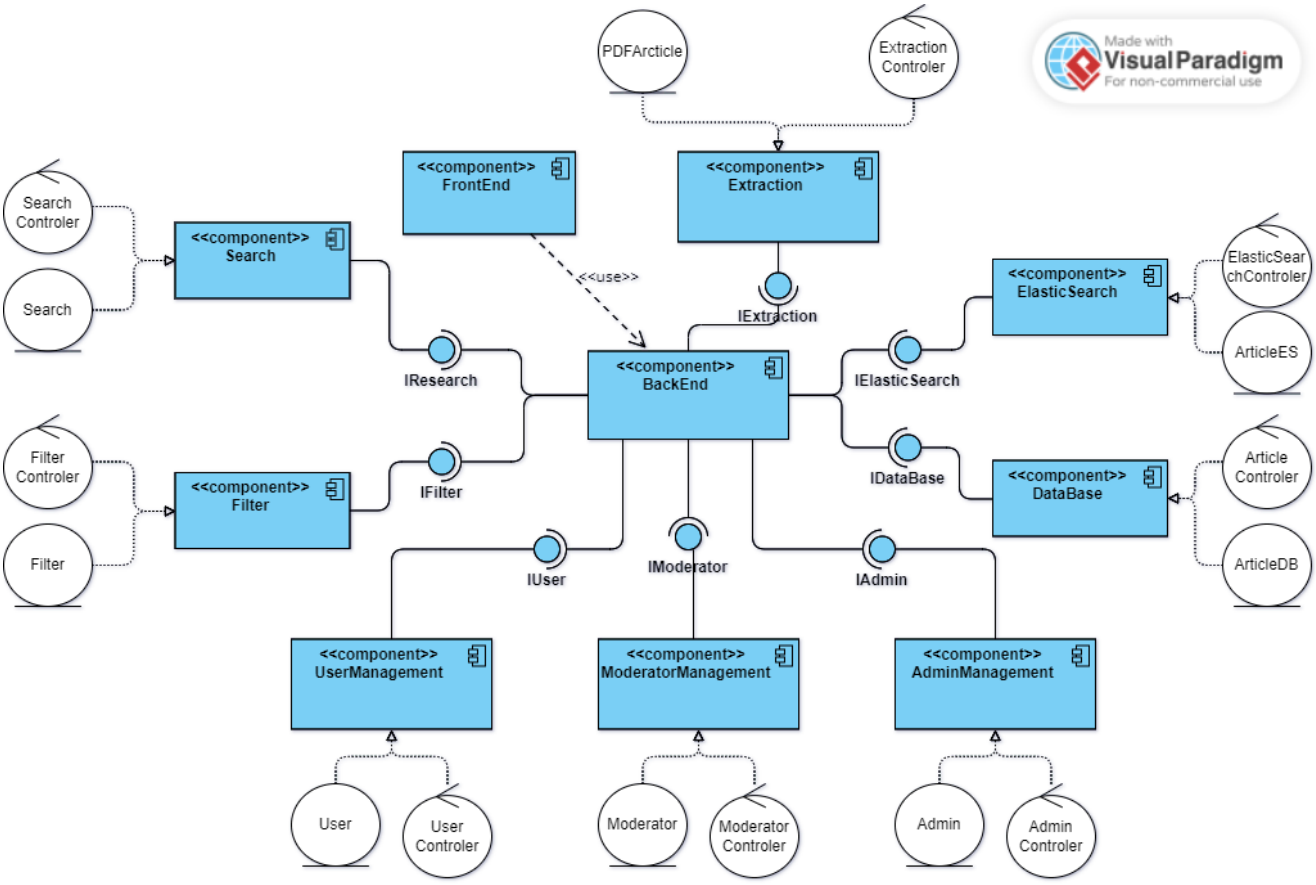
2.4. Extraction

- Combine LLMs and Regex for accurate scientific articles details extraction

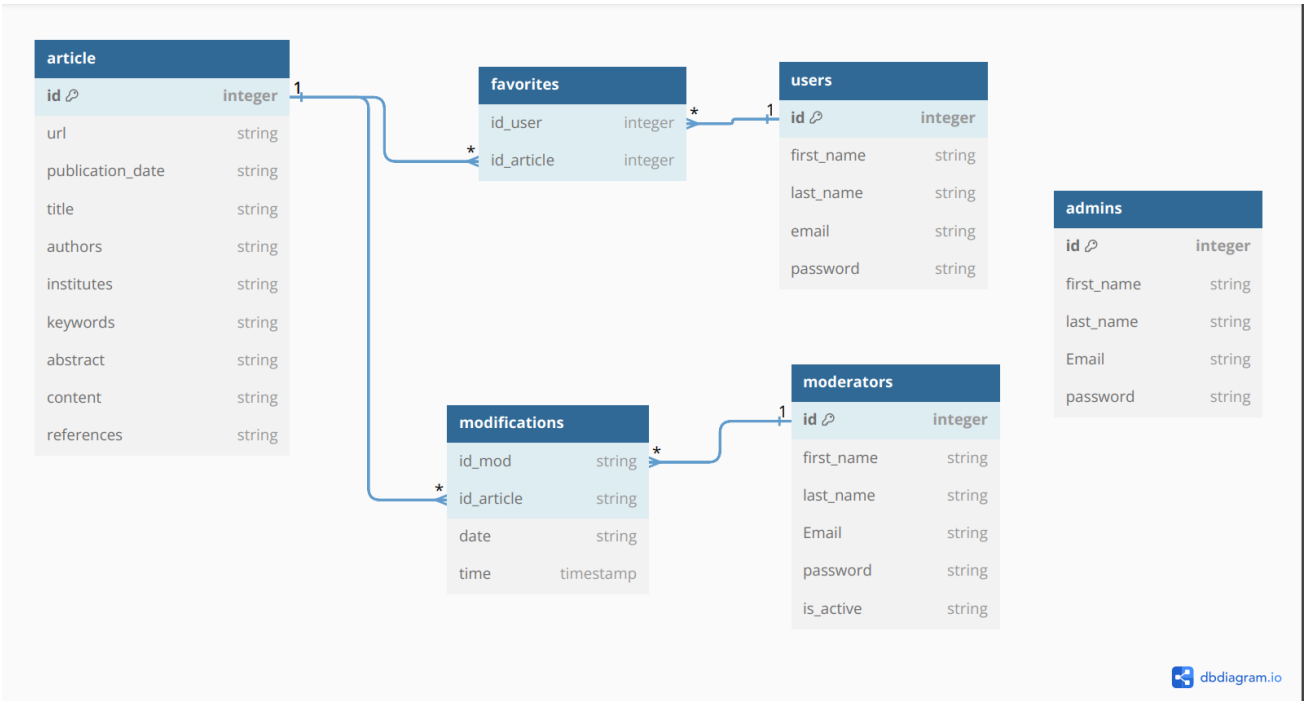
3. Class Diagram



4. Components Diagram (Architecture)



5. Database Schema



6. General Workflow and Development Environment

The development of ArticleHarbor has been centred around the principle components: the **interface** (front-end) , the **API** (back-end) and the **search-engine** (elastic-search)

6.1. Front-End

- **ReactJS:** JavaScript library primarily user in front-end development to create reusable components.

The application's interface, developed with ReactJS, allows users/admins/moderators to explore the library's philosophy through reusable **components**. Requests and interactions are transmitted to the back end for processing.

6.2. Back-End

- **FastAPI:** Python back-end framework, Known for its simplicity for crafting RESTful APIs with his responsive documentation ,automatic validation, and annotating types.
- **PostgreSQL:** A popular and open source SGBD solution, ideal for quick interaction, and effective querying
- **ChatGPT API:** versatile tool for general-purpose applications, ideal for use cases involving high precision text manipulation and extreme flexibility and integration.

The system employs a **service-oriented** architecture, with separate APIs for distinct functionalities such as authentication. The articles, handled by an upload service, undergo text manipulation using **OpenAI's ChatGPT API** to extract relevant information. The structured article data is then stored in a **PostgreSQL** database, along with actor information like emails and passwords.

6.3. Search Engine

- **ElasticSearch:** Open-source search engine, with high capabilities in indexing and querying large datasets.

Queries are directed to a **Python** service utilizing a deployed **ElasticSearch** service, which handles a large dataset extracted from articles. The responses, adhering to a specific model, are then sent to the front-end for display.

6.4. Infrastructure

- **Docker:** Containerisation solution that has been used for maintaining consistent deployments purposes by isolating application dependencies into images and running them as containers.

The chosen deployment strategy involves **Railways** for hosting both the front-end and back-end in **dockerised** environments. For ElasticSearch, the decision is to use **Bonsai**, a free host of ElasticSearch clusters and nodes, suitable for initial application needs. The database is hosted on **AlwaysData**, with secure access through **PHPpgadmin** gateway, allowing tables to be viewed and accessed at any time, ensuring data persistence.

7. Collaboration Environment

7.1. Team Structure

Human and skill resources management was key to ensuring smooth collaborative workflow and information flow among the team. The team was basically lead by **Kamel Brouthen** setting the ground for the project execution plan, tasks scheduling and monitoring and acting as a bridge between different teams. The team was organized in the following structure:

- **UX/UI Team** lead by **Abdelkarim Bengharbia** designing the experience and interface of our website.
- **Front-End Team** lead by **Abdelkarim Bengharbia** alongside **Malek Rais** and **Mahdi Fellah** turning the delivered UX/UI into a seamless website while integrating different backend functionalities.
- **Backend Team** lead by **Mehdi Madani** alongside **Kamel Brouthen** and **Aziz Akeb** implementing several server functionalities as authentication, data management, article's information extraction and search...etc

7.2. Development Methodology

In modern software development age, this project was a great opportunity to explore agile methodologies as we opt for **SCRUM** methodology. Having a clear vision about the different project functionalities and requirements was of great benefit for acquiring a well structured and planned product backlog allowing us to effectively run our sprints throughout the past two months. In that context, as a team, we went experienced different agile driven activities as: running multiple **stand-up** meetings either daily or bi-daily depending on team's availability and project's current phase, sprint tasks planning cooperatively and synchronously among different teams, and most importantly holding a high flexibility, agility and adaptation spirit throughout the project journey.

7.3. Information, management and communication

As software development success is mainly driven by collaboration and communication, we established a **GitHub Organisation** which made our information circuits centralised and well-organised which was key both on individual level where it becomes so intuitive, easy and straightforward process getting and communicating updates and on a collective level where we effectively took advantage of **GitHub Projects** which are of similar utility to other tasks planning software, yet it is direct integration with code, especially **GitHub Issues** and **Pull Requests** which can all be assembled in one place. Speaking of communication, as a way to set more of a quick and direct communication channels, we relied on **Discord** where we set our server accounting for more detailed and casual reach-outs whether it is general announcements, different teams workspace or any kind of urgent update.