МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

МАТЕМАТИЧНИЙ ФАКУЛЬТЕТ
Кафедра комп'ютерних наук

КВАЛІФІКАЦІЙНА РОБОТА МАГІСТРА

на тему: **«ЗАСТОСУВАННЯ АЛГОРИТМУ СПРЯМОВАНОГО РОЗПОВСЮДЖЕННЯ ЗАПИТУ ДО ПОШУКУ У БАЗІ ДАНИХ SCIENCEDIRECT»**

Виконав: студент 2 курсу, групи 8.1222
спеціальності 122 комп'ютерні науки
(шифр і назва спеціальності)
освітньої програми комп'ютерні науки
(назва освітньої програми)
Камінський С.П.
(ініціали та прізвище)
Керівник викладач кафедри комп'ютерних наук, к.т.н. Добровольський Г.А.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)
Рецензент завідувач кафедри фундаментальної та прикладної математики, професор, д.т.н. Гребенюк С.М.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Запоріжжя – 2023

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет математичний

Кафедра комп`ютерних наук

Рівень вищої освіти магістр

Спеціальність 122 комп'ютерні науки
(шифр і назва)

Освітня програма комп'ютерні науки

ЗАТВЕРДЖУЮ
Професор кафедри комп`ютерних наук,
д.т.н., доцент

_____ Шило Г.М.
(підпис)

" 01 " 05 2023 р.

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТОВІ

Камінському Сергію Павловичу
(прізвище, ім`я та по-батькові)

1. Тема роботи Застосування алгоритму спрямованого розповсюдження запиту до пошуку у базі даних ScienceDirect

керівник роботи Добровольський Геннадій Анатолійович, к.т.н., викладач
(прізвище, ім`я та по-батькові, науковий ступінь, вчене звання)

затверджені наказом ЗНУ від « 01 » травня 2023 року № 643-с

2. Строк подання студентом роботи

3. Вихідні дані до роботи 1. Постановка задачі.
2. Перелік літератури.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
1. Постановка задачі, аналіз предметної області.
2. Проектування.
3. Реалізація та тестування.

5. Перелік графічного матеріалу (з точним зазначенням обов`язкових креслень)
презентація за темою докладу

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання 01.05.2023

К А Л Е Н Д А Р Н И Й П Л А Н

№	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1.	Розробка плану роботи.	02.05.2023	
2.	Збір вихідних даних.	10.05.2023	
3.	Обробка методичних та теоретичних	15.06.2023	
	джерел.		
4.	Розробка першого та другого розділу.	20.08.2023	
5.	Розробка третього розділу.	01.10.2023	
6.	Оформлення та нормоконтроль	28.11.2023	
	кваліфікаційної роботи бакалавра.		
7.	Захист кваліфікаційної роботи.		

Студент _____ С. П. Камінський
(підпис) (ініціали та прізвище)

Керівник роботи _____ Г.А. Добровольський
(підпис) (ініціали та прізвище)

Нормоконтроль пройдено

Нормоконтролер _____ О. Г. Спиця
(підпис) (ініціали та прізвище)

РЕФЕРАТ

Кваліфікаційна робота магістра «Застосування алгоритму спрямованого розповсюдження запиту до пошуку у базі даних ScienceDirect»: 71 с., 31 рис., 12 джерел, 4 додатки.

DJANGO, ELSAPY, REST API, SCIENCE DIRECT, UML.

Об`єкт дослідження – ScienceDirect, алгоритм спрямованого розповсюдження запиту до пошуку у базі даних.

Мета роботи – розробити веб-додаток для реалізації алгоритму спрямованого розповсюдження запиту до пошуку у базі даних ScienceDirect.

Методи дослідження – моделювання, проектування, програмний, аналітичний.

Алгоритм спрямованого розповсюдження запиту представляє значний науковий інтерес у контексті сучасних вимог до обробки великих обсягів даних, ефективності пошуку інформації та оптимізації комунікаційних процесів в мережах. Актуальність цієї теми зростає у зв'язку з розвитком технологій біг дата та Інтернету речей (IoT), де виникає потреба у розширенні можливостей пошукових систем та алгоритмів для швидкої та точної обробки запитів.

Основна значимість алгоритмів спрямованого розповсюдження запиту полягає у їх здатності ефективно маршрутизувати запити до відповідних вузлів у мережі, що мінімізує час обробки та зменшує навантаження на мережеві ресурси. Це особливо важливо у великих розподілених системах, де традиційні методи розповсюдження запиту, такі як повне охоплення мережі, можуть бути неефективними та ресурсомісткими.

Отже, в результаті нашої роботи було створено зручний та ефективний веб-додаток для пошуку у базі даних ScienceDirect.

SUMMARY

Master`s qualifying paper «Application of the Focused Query Routing Algorithm to ScienceDirect Search»: 71 pages, 31 figures, 12 references, 4 supplements.

DJANGO, ELSAPY, REST API, SCIENCE DIRECT, UML.

The object of the study is the ScienceDirect, an algorithm for targeted query distribution for database searches.

The aim of the study to develop a web application for implementing the algorithm of directed query propagation in the ScienceDirect database.

The methods of research are modeling, design, programming, analytical.

The algorithm of directed query propagation is of considerable scientific interest in the context of modern requirements for processing large amounts of data, information retrieval efficiency, and optimization of communication processes in networks. The relevance of this topic is growing due to the development of big data and Internet of Things (IoT) technologies, where there is a need to expand the capabilities of search engines and algorithms for fast and accurate query processing.

The main significance of directed query propagation algorithms lies in their ability to efficiently route queries to the appropriate nodes in the network, which minimizes processing time and reduces the load on network resources. This is especially important in large distributed systems where traditional request propagation methods, such as full network coverage, can be inefficient and resource-intensive.

As a result of our work, we have created a convenient and efficient web application for searching the ScienceDirect database.

ЗМІСТ

ВСТУП

Алгоритм спрямованого розповсюдження запиту є важливою темою в сучасних дослідженнях в області комп'ютерних наук, зокрема у сферах розподілених систем та обчислювальних мереж. Актуальність даної теми обумовлюється кількома ключовими факторами, які визначають її значимість у швидко розвиваючомуся світі цифрових технологій.

Перш за все, із зростанням розмірів та складності розподілених систем, таких як хмарні обчислення, Інтернет речей (IoT), та великі дата-центри, виникає необхідність ефективного управління запитами в мережі. Спрямоване розповсюдження запитів дозволяє оптимізувати процес пошуку та обміну даними в розподілених системах, що в свою чергу забезпечує підвищення продуктивності та ефективності обчислень. Крім того, розвиток алгоритмів спрямованого розповсюдження запиту сприяє зменшенню навантаження на мережеві ресурси. Це особливо важливо у контексті великих об'ємів даних та високих вимог до пропускної здатності мережі, що є характерними для сучасних інформаційних систем.

Зазначений алгоритм може забезпечити кращу масштабованість та надійність системи, враховуючи динамічну природу розподілених мереж. Спрямоване розповсюдження запитів може автоматично адаптуватися до змін у мережі, забезпечуючи стійкість до відмов та коливань у мережевих умовах.

В сфері безпеки даних, спрямоване розповсюдження запитів може сприяти підвищенню захисту та приватності, оскільки воно дозволяє контролювати шляхи розповсюдження запитів та даних в мережі, обмежуючи доступ до чутливої інформації. Таким чином, дослідження в області алгоритмів спрямованого розповсюдження запиту має важливе значення для розвитку та оптимізації сучасних інформаційних систем, забезпечуючи більш ефективне та безпечне використання ресурсів розподілених мереж.

Адаптований до пошуку документів у ScienceDirect алгоритм розповсюдження запиту дозволяє знаходити пов'язані, але несхожі статті, тим самим збільшуючи повноту та різноманітність результатів пошуку. У цьому випадку вузлами мережі вважаються статті, дугами - посилання із однієї статті на іншу. Шлях розповсюдження запиту задається із інтерфейса користувача.

З огляду на це, можна виділити наступні цілі і задачі нашого дослідження:

Мета: розробити веб-додаток для реалізації алгоритму спрямованого розповсюдження запиту до пошуку у базі даних ScienceDirect.

Задачі:

1) Оглянути рішення та інструменти для реалізації системи;
2) Спроектувати та побудувати архітектуру системи;
3) Реалізувати веб-додаток для реалізації алгоритму спрямованого розповсюдження запиту до пошуку у базі даних ScienceDirect;
4) Протестувати роботу системи.

Об`єкт дослідження: ScienceDirect, алгоритм спрямованого розповсюдження запиту до пошуку у базі даних.

Предмет дослідження: фреймворки веброзробки.

Методи дослідження: моделювання, проектування, програмний, аналітичний.

Перший розділ присвячено збору та аналізуванню дотичної до теми літератури та досліджень, а також огляду інструментів розробки.

У другому розділі розглянуто етапи проектування веб-додатку, наведено детальний опис прецедентів.

Третій розділ присвячено реалізації та тестуванню роботи веб-додатку, наведено детальне керівництво користувача, яке описує процес роботи з веб-додатком пошуку у базі даних ScienceDirect.

1 ОГЛЯД РІШЕНЬ ТА ІНСТРУМЕНТІВ

1.1 ScienceDirect

ScienceDirect – це повнотекстова база даних, яка пропонує журнальні статті та розділи книг із понад 2500 рецензованих журналів і 11 000 книг. Старіші матеріали (старі файли) доступні як архіви за додаткову плату. Цільовою аудиторією є бібліотекарі та дослідники в галузі медицини, наук про життя, фізичних наук та інженерії. Він також містить чудове доповнення до матеріалів із соціальних наук і бізнес-темат. Він сумісний з ПК з Internet Explorer 7, 8 і 9, а також з Mozilla Firefox і Google Chrome. Браузерами, сумісними з Mac, є Google Chrome, Safari та Firefox.

Ця база даних використовує механізм об’єднаного пошуку, який є єдиним пошуковим порталом для пошуку та отримання результатів з кількох електронних ресурсів одночасно. Мета-пошук або широкомовний пошук є іншими термінами, які зазвичай використовуються для цього механізму пошуку. ScienceDirect використовує пошук природною мовою, подібний до пошуку Google. Він не має контрольованого словника, як у PubMed’s Medical Subject Headings (MeSH). Функція розширеного пошуку дозволяє здійснювати пошук за автором, назвою, томом, випуском і сторінкою, подібно до Single Citation Matcher PubMed. Пошук природною мовою за допомогою логічних операторів можна фільтрувати за темою та датами публікації. Пошук можна обмежити журналами, книгами, зображеннями та довідниками. Чітко описані підказки щодо підстановки, скорочення та пошуку на відстані. Підручники доступні англійською, іспанською, французькою, німецькою, португальською мовами.

Використання особистого облікового запису активує всі можливості цієї бази даних. Дійсно прості канали синдикації (RSS) і сповіщення налаштовуються та доступні через функцію персоналізованого облікового запису. Власник облікового запису може отримати доступ до спеціалізованих безкоштовних і

платних програм (наприклад, рак, подкаст тощо). Увімкнено швидкі посилання на улюблені книги чи журнали користувача. За допомогою сторінки «Керування налаштуваннями» користувачі RefWorks можуть автоматично входити в систему та експортувати цитати до RefWorks. Доступ до минулих пошуків здійснюється через персоналізований обліковий запис.

Налаштування мобільних додатків для пристроїв iPhone, iPad, Android здійснюється шляхом реєстрації через особистий кабінет. Безкоштовний мобільний додаток доступний у магазині iTunes. Дисплей чистий і його легко читати. У мобільному додатку підтримується пошук за ключовими словами, авторами та журналами, а також пошук за логічними значеннями, символом підстановки та скороченим пошуком. Відновлені статті містять анотацію та план. Повний текст і цифри відображаються, якщо заклад має передплату. Реєстрацію можна використовувати максимум на трьох пристроях, таких як iPhone, iPad та iPod. Статті, збережені на мобільному пристрої, недоступні з облікового запису користувача ScienceDirect. Щоб отримати доступ, статті потрібно надсилати електронною поштою.

Отримані статті містять посилання на анотацію статті, посилання, пов’язані статті та додаткові матеріали. Якщо організація підписується на журнал, надається посилання на файл у форматі переносного документа (PDF). Експорт цитат у формати Reference Manager, ProCite, EndNote, RefWorks і BibTex. Отримані статті можна роздрукувати, надіслати електронною поштою або поділитися ними через соціальні мережі Facebook, Twitter і CiteULike.

Історія пошуку обмежена 500 елементами. Якщо це число перевищено, найстаріші цитати видаляються. Ця база даних належить Elsevier, і більша частина її вмісту обмежена журналами та книгами, опублікованими Elsevier. База даних орієнтована на статті та матеріали, опубліковані з 1995 року по теперішній час.

1.2 Алгоритм спрямованого розповсюдження запиту

Алгоритми пошуку в базах даних (БД) відіграють ключову роль у ефективному доступі до інформації. Один із таких алгоритмів — алгоритм розповсюдження запиту, який використовується для пошуку інформації в розподілених базах даних або в мережі.

Алгоритм розповсюдження запиту працює на принципі передачі запиту від одного вузла до інших вузлів у мережі. Цей процес може бути реалізований різними способами, залежно від структури мережі та вимог до системи. Основна мета цього алгоритму – знайти та отримати відповідну інформацію, мінімізуючи навантаження на мережу та час відповіді.

Ось декілька ключових аспектів алгоритму розповсюдження запиту:

Оптимізація маршруту запиту: Алгоритм може визначати найбільш ефективний шлях для передачі запиту, уникаючи непотрібних повторень та зайвого навантаження на систему.

Фільтрація даних: На кожному етапі передачі запиту можуть застосовуватися фільтри для забезпечення того, що передається лише релевантна інформація.

Балансування навантаження: Ефективне розподілення запитів між вузлами допомагає уникнути перевантаження окремих вузлів.

Запобігання зацикленню: Механізми для запобігання повторній передачі запиту до вже відвіданих вузлів є критично важливими для запобігання зацикленням.

1.3 Інструменти розробки

1.3.1 Django

Django – високорівневий Python веб-фреймворк, який був розроблений з основною ідеєю прискорення процесу розробки веб-додатків, забезпечуючи розробникам потужний інструментарій для створення надійних, масштабованих та ефективних веб-додатків. Завдяки його архітектурі MVT (Model-View-Template), Django дозволяє чітко розділити логіку додатку на взаємопов’язані компоненти, що сприяє більшій організованості коду та полегшує його розширення та підтримку. Модель відповідає за взаємодію з базою даних і бізнес-логіку, вид керує взаємодією з користувачем, а шаблон забезпечує презентацію даних, тим самим уніфікуючи процес розробки та забезпечуючи високий рівень повторного використання коду.

Одним з ключових аспектів Django є його ORM (Object-Relational Mapping) система, яка дозволяє розробникам взаємодіяти з базою даних вищого рівня, використовуючи Python об’єкти, замість прямого написання SQL-запитів. Це не тільки спрощує процес розробки, але й забезпечує більшу безпеку та стабільність роботи з даними. Крім того, Django включає в себе вбудовану систему адміністрації, яка дозволяє швидко створювати інтерфейси для управління моделями додатку, значно економлячи час на розробку та тестування.

Ще однією важливою особливістю Django є його підтримка безпеки веб-додатків. Фреймворк включає в себе ряд механізмів для захисту від загальновідомих веб-атак, таких як cross-site scripting (XSS), cross-site request forgery (CSRF) та SQL-ін’єкцій, забезпечуючи базовий рівень безпеки без додаткових зусиль з боку розробників. Також, Django підтримує механізми кешування та масштабованості, що є критично важливим для великих та високонавантажених веб-додатків.

Завдяки своїй широкій функціональності, гнучкості та великій спільноті розробників, Django продовжує бути одним з найпопулярніших веб-

фреймворків. Його активне використання в проектах різного рівня складності, від простих веб-сайтів до складних веб-сервісів і корпоративних додатків, свідчить про його універсальність та ефективність як інструменту веб-розробки.

Django не лише забезпечує розробників потужними інструментами для створення високоякісних веб-додатків, але й сприяє більшій організованості, безпеці та ефективності розробки, що є ключовими чинниками в сучасному швидкозмінному світі веб-технологій.

1.3.2 Elsapy

Elsapy, репрезентуючи собою інноваційну Python-бібліотеку, розроблену з метою оптимізації і автоматизації взаємодії з комплексним і багатогранним API Elsevier, представляє собою значний прогрес у сфері наукових досліджень, забезпечуючи дослідникам інструмент, який дозволяє значно спростити та прискорити процес пошуку, аналізу та витягу наукових даних з обширної бази наукових публікацій Elsevier.

Ця бібліотека, використовуючи гнучкість та мощі RESTful веб-сервісів Elsevier, дозволяє дослідникам здійснювати глибокий та різноплановий пошук у величезній кількості наукових документів, зокрема, шляхом формування запитів з використанням специфічних ключових слів, авторів, назв публікацій, а також надає доступ до деталізованої інформації про академічні журнали та конференції, тим самим значно підвищуючи продуктивність наукових досліджень.

Окрім того, Elsapy відкриває широкі можливості для інтеграції з іншими аналітичними та науковими інструментами, що сприяє створенню комплексних наукових рішень, де дані, отримані через Elsevier API, можуть бути легко вбудовані в різноманітні додатки, забезпечуючи більш глибокий і всебічний аналіз.

Використання Elsapy, однак, вимагає обережного ставлення до питань ліцензування та дотримання політики використання даних Elsevier, адже неправомірне використання авторських матеріалів може призвести до юридичних наслідків, що є важливим аспектом при впровадженні цього інструменту в наукову практику.

1.3.3 REST API

REST API, являючи собою архітектурний стиль веб-сервісів, що був розроблений Роєм Філдінгом у його докторській дисертації 2000 року, став основоположним елементом сучасних веб-додатків, оскільки він пропонує стандартизований та інтуїтивно зрозумілий спосіб взаємодії з різними веб-ресурсами. Основна ідея REST (Representational State Transfer) полягає в використанні безстанових запитів з використанням стандартних HTTP методів, таких як GET, POST, PUT та DELETE, для доступу та маніпуляції представленнями ресурсів, які ідентифікуються за допомогою URI (Uniform Resource Identifier). Кожен з цих методів відповідає типовим операціям CRUD (Create, Read, Update, Delete), що дозволяє виконувати стандартизовані операції з даними.

REST API вирізняється своєю безстановістю, що означає, що кожен запит містить всю необхідну інформацію для його обробки, без потреби в зберіганні стану сесії на сервері, що спрощує архітектуру сервера та покращує масштабованість систем. Також, REST підтримує кешування відповідей, що може підвищити продуктивність та ефективність веб-сервісів. Завдяки своїм характеристикам, REST API став вибором для багатьох публічних та корпоративних веб-сервісів, забезпечуючи легку інтеграцію між різними системами та платформами.

Однією з ключових переваг REST API є його універсальність у представленні даних, оскільки він не накладає жорстких обмежень на формат

даних, таким чином дозволяючи використовувати JSON, XML, HTML або будь-який інший зручний формат. Це робить REST API особливо привабливим для розробки гнучких веб-додатків, які можуть взаємодіяти з різними системами та сервісами. Крім того, REST API сприяє створенню легко масштабованих та високопродуктивних систем, оскільки він дозволяє клієнтам та серверам розвиватися та змінюватися незалежно один від одного, не порушуючи загальну функціональність системи.

Як висновок, REST API відіграє важливу роль у сучасній веб-розробці, надаючи розробникам потужний інструмент для створення масштабованих, ефективних та гнучких веб-сервісів і додатків, які є критично важливими у швидкозмінному цифровому світі.

1.4 Огляд та рецензія статей

ScienceDirect

Science Direct платформа, яка була запущена в 1997 році та доступна через Elsevier, представляє собою обширну бібліографічну базу даних, охоплюючу широкі галузі наукових, медичних та гуманітарних дисциплін, відіграючи таким чином фундаментальну роль у поширенні наукових знань і сприянні академічним дослідженням. Забезпечуючи доступ до понад 18 мільйонів статей і розділів книг, Science Direct стає центром для більш ніж 2500 рецензованих журналів, а також надає користувачам можливість ознайомитися з понад 42 000 електронних книг, що робить його одним з найбільших та найбільш комплексних ресурсів у світі наукових публікацій [10].

Великий обсяг контенту, що охоплюється Science Direct, включає не тільки дослідницькі статті та розділи книг, але й енциклопедії, детальну інформацію про наукові конференції, офіційне листування, патенти, програмне забезпечення, а також відеоматеріали, створюючи тим самим універсальний інформаційний простір для дослідників, студентів та академіків. Ця різноманітність форматів та

типів публікацій забезпечує міждисциплінарний підхід до досліджень, дозволяючи користувачам інтегрувати знання з різних галузей науки для більш комплексного розуміння досліджуваних проблем [9].

Контент на платформі Science Direct систематизовано та упорядковано в чотири основні сфери, що відображають ключові напрямки сучасних наукових досліджень та академічних зусиль, включаючи фізичні науки та інженерію, науки про життя, науки про здоров'я, а також соціально-гуманітарні науки, тим самим забезпечуючи комплексний підхід до наукового знання та його практичного застосування. У межах цих значних сфер контенту Science Direct розрізняє 24 підкатегорії, які включають, але не обмежуються такими спеціалізованими галузями, як хімічна інженерія, матеріалознавство, математика, біохімія, екологія, медицина, фармакологія, ветеринарія та медичні науки, мистецтво та гуманітарні науки, бізнес, економіка та соціальні науки.

На платформі Science Direct реалізовано можливості базового та розширеного пошуку, що відкривають широкі горизонти для наукових дослідників, студентів та академіків, які прагнуть здійснювати точний та ефективний пошук величезної кількості наукових публікацій. Основні функції пошуку включають пошук за ключовими словами та зіставлення цитат для журнальних статей або назв книг, враховуючи такі деталі, як томи та номери сторінок, що дозволяє користувачам легко знаходити конкретну інформацію. Розширені функції пошуку надають можливість шукати за анотаціями, ключовими словами, наданими автором, іменами авторів, їх приналежностями, посиланнями, а також за ISSN або ISBN, тим самим забезпечуючи гнучкість та глибину у дослідженнях [9].

Для ініціації пошуку на Science Direct достатньо заповнити лише одне поле, не вимагаючи від користувачів заповнення всіх доступних полів, що значно спрощує процес пошуку. Особливістю платформи є те, що весь пошук базується на ключових словах, тому користувачам не потрібно покладатися на предметні заголовки або бути знайомими з будь-якими іншими основними заголовками

бази даних, як, наприклад, MeSH чи Emtree, що робить Science Direct доступним для широкого кола користувачів.

Інтерфейс Science Direct характеризується простотою з обмеженими можливостями, однак дотримується стандартних логічних значень (І, АБО, НІ) у верхньому регістрі під час пошуку, що забезпечує точність у відборі результатів. Рекомендується використовувати лапки для фразування, а розділові знаки в фразах ігноруються, при цьому варіанти множини та написання шукаються автоматично, що підвищує зручність та ефективність пошуку. Після запуску пошуку користувачі можуть уточнити його за допомогою фільтрів, включаючи дату публікації, тип статті, назву публікації, тематику та тип доступу (відкритий доступ або відкритий архів), що дозволяє здійснювати більш цілеспрямований пошук. Однак, тип статті може включати різні форми публікацій, такі як розділи книг, статті енциклопедії та інші, що вимагає від користувачів уважності при відборі матеріалів [5].

Science Direct, як передова платформа для наукових досліджень, виконує значну роль у спрощенні процесу пошуку та доступу до наукових ресурсів, надаючи користувачам, які знайомі з базами даних, очікувані результати пошуку, що відображаються у форматі заголовок/цитата та автори. Результати цих пошуків можуть бути сортовані за релевантністю або датою, забезпечуючи користувачам можливість організувати отриману інформацію за їхніми уподобаннями та потребами. Крім того, вміст з Science Direct може бути експортований у різноманітні програми керування цитуваннями, включаючи Mendeley і RefWorks, із опціями RIS, BibTeX, CSV або як простий текст, що полегшує інтеграцію отриманої інформації у наукові роботи та дослідження.

Science Direct також надає доступ до PDF-файлів для завантаження, з подальшим переходом користувача до додаткового вікна, що забезпечує гнучкість у використанні наукових матеріалів. Користувачі можуть виявляти подібні статті через список рекомендованих статей, цитувати статті та ознайомлюватися з показниками статей, включаючи цитування, захоплення та соціальні мережі, а додаткова інформація доступна через PlumX. Завдяки

власному PDF-рідеру Elsevier, доступному через Chrome, Firefox і Safari, користувачі можуть посилатися на посилання та переглядати тези, що забезпечує глибше занурення в контент.

На Science Direct доступні такі додаткові функції, як сторінки авторів та тем, які користувачі можуть знаходити без необхідності самостійного пошуку. При відкритті статті в новому вікні, імена афілійованих авторів представлені у вигляді гіперпосилань, що ведуть на сторінку автора в іншому продукті Elsevier - Scopus. Профіль автора на Scopus забезпечує додаткову інформацію та зв'язок, однак доступ до повного профілю вимагає наявності доступу до Scopus.

Функція «Теми» на Science Direct також є важливою, де користувачі можуть здійснювати пошук на сторінці тем двома способами: на сторінці покажчика тем або читаючи статті. Тематичні сторінки містять огляд, гіперпосилання на відповідні терміни та пов'язані розділи книг, що надає користувачам можливість глибшого дослідження специфічних тем [10].

Крім того, як і в інших базах даних, Science Direct дозволяє створювати облікові записи для віддаленого доступу через настільні комп'ютери, ноутбуки або мобільні пристрої. Функція збереження пошуків та отримання сповіщень і оновлень включає інформацію про зміст журналу та інші рекомендовані публікації, що забезпечує користувачам персоналізований досвід використання платформи.

Science Direct прагне відповідати вимогам доступності веб-контенту, підтримуючи більшість критеріїв, встановлених у Рекомендаціях щодо доступності веб-вмісту W3C та відповідно до розділу 508 стандартів США, з певними обмеженнями у підтримці слухових та зорових можливостей.

Взаємозв'язок вмісту є однією з сильних сторін Science Direct, де різні функції, такі як сторінки авторів, тематичні сторінки та рекомендований вміст, працюють разом для надання користувачам більш глибокого та ширшого огляду наукових матеріалів, що сприяє більш об'ємному та інтегрованому підходу до наукових досліджень.

Алгоритм спрямованого розповсюдження запиту

В області комп'ютерних наук, оптимізація запитів бази даних є процесом вдосконалення швидкості та ефективності доступу до даних у системах управління базами даних. Цей процес включає ряд стратегій та технік, спрямованих на мінімізацію ресурсів, необхідних для виконання запитів, та оптимізацію загальної продуктивності системи. Основними компонентами цього процесу є переписування запитів, індексація, кешування, а також оптимізація дискового вводу/виводу та паралельної обробки [2].

Vivek Basavegowda Ramu зосереджується на важливості стратегій оптимізації для підвищення продуктивності баз даних, зокрема включає такі аспекти як оптимізація виконання запитів, ефективне управління ресурсами, та забезпечення найкращого можливого досвіду користувачів. Дослідження охоплює методи та техніки, такі як переписування запитів, індексація та кешування, для мінімізації часу відповіді на запити та підвищення загального пропуску системи. Стаття також розглядає стратегії для ефективного управління ресурсами, включаючи розподіл пам'яті, оптимізацію дискового вводу/виводу та паралельну обробку, а також вплив дизайну схеми бази даних на продуктивність і найкращі практики для оптимізації схеми [9].

Додаткові джерела, що згадуються у статті, включають дослідження Jingbo Shao і інших про оптимізацію продуктивності баз даних для SQL Server на основі моделі ієрархічної мережі очікування, роботу Khaled Saleh Maabreh про оптимізацію продуктивності запитів до бази даних за допомогою технік розділення таблиць та дослідження Jiangang Zhang про методи оптимізації продуктивності застосунків баз даних [6]. Публікація надає всебічний огляд стратегій та технік, необхідних для оптимізації продуктивності баз даних, пропонуючи організаціям можливості максимізувати ресурси та забезпечити ефективність виконання запитів (рис. 1.1).

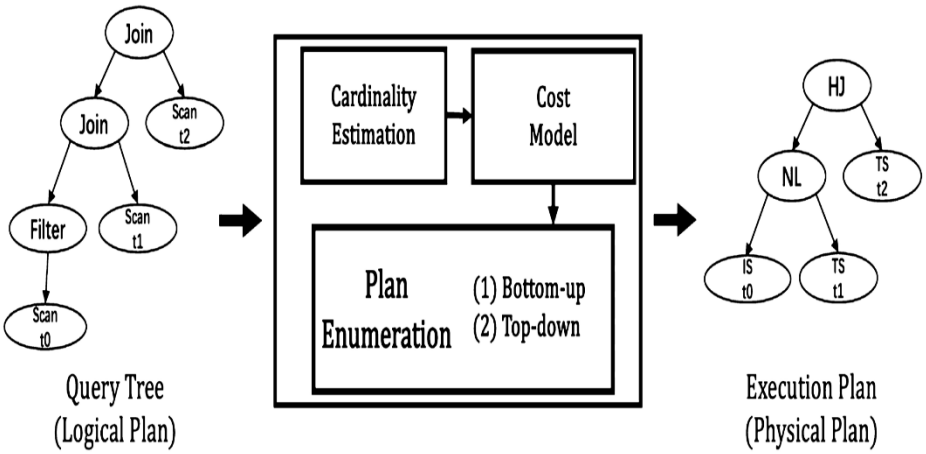

Рисунок 1.1 – Архітектура оптимізатора запитів

Автори Hai Lan, Zhifeng Bao та Yuwei Peng детально досліджують ключові компоненти, які впливають на процес оптимізації запитів, зосереджуючись на трьох основних елементах: оцінці кардинальності, моделі вартості та переліченні планів.

Оцінка кардинальності: Кардинальність у контексті баз даних відноситься до кількості рядків, які повертаються окремим запитом або оператором. Оцінка кардинальності відіграє вирішальну роль у виборі оптимального плану виконання запиту. Стаття детально розглядає різні підходи та методології, які використовуються для точної оцінки кардинальності, включаючи традиційні статистичні методи та більш сучасні техніки, засновані на машинному навчанні [4].

Модель вартості: Важливим аспектом оптимізації запитів є моделювання вартості різних планів виконання. Модель вартості допомагає визначити очікувані витрати на виконання конкретного плану, враховуючи різні фактори, такі як час виконання, використання пам’яті, витрати на ввод-вивод. Автори обговорюють різні стратегії та підходи до моделювання вартості, а також виклики, пов’язані з точністю таких оцінок.

Перелік планів виконання: План виконання запиту визначає, як база даних буде обробляти і виконувати запит. Оптимізатори запитів повинні оцінити та вибрати з різноманітності можливих планів. Стаття досліджує методи переліку

та порівняння планів, включаючи алгоритми пошуку та вибору оптимального плану з урахуванням ефективності та вартості [7].

В області мережевих технологій, зокрема у системах однорангових (P2P) мереж, важливим аспектом є розробка ефективних та надійних алгоритмів пошуку. Для широкого використання техніки пошуку у P2P-мережах необхідно враховувати ряд фундаментальних вимог, серед яких деякі є абсолютно необхідними, а інші класифікуються як бажані чи "добре мати". Сучасні схеми P2P та пов’язані з ними методи пошуку постійно вдосконалюються, але багато викликів все ще залишаються нерозв’язаними.

Основні вимоги до алгоритмів пошуку в P2P-мережах включають:

Гарантовані результати: Висока надійність алгоритмів пошуку є критичною, оскільки якщо шуканий об’єкт існує у мережі та запит виконаний з правильними параметрами, результати пошуку повинні бути завжди повернуті. Це має бути забезпечено незалежно від розміру або структури мережі, розміру чи типу запитуваного об’єкту та географічної віддаленості між вузлами.

Низьке використання ресурсів: Ефективність використання ресурсів є ключовою, оскільки надмірне навантаження на мережу може негативно впливати на загальну продуктивність системи. Пошукові запити повинні бути оптимізовані таким чином, щоб залучати мінімальну кількість ресурсів, як з точки зору обчислювальної потужності, так і мережевого трафіку.

Швидкий час відповіді: Швидкість відповіді є важливим параметром, оскільки користувачі очікують отримання результатів у найкоротші терміни. Оптимальні алгоритми пошуку повинні бути спроектовані таким чином, щоб мінімізувати кількість переходів запиту у мережі, що також сприятиме зменшенню використання ресурсів та відповідає вимозі ефективності.

Точність результатів: Надійність та точність результатів є невід’ємною частиною ефективної пошукової системи. Техніка пошуку повинна забезпечувати консистентність і точність результатів при кожному пошуку, навіть у разі зміни структури мережі чи інших умов.

Збір додаткової інформації: Інтелектуальні техніки пошуку повинні здійснювати збір корисної інформації під час процесу пошуку, що може бути корисним для вибору між різними варіантами або для прийняття рішення про завантаження об’єктів. Наприклад, інформація про мережеву вартість завантаження різних об’єктів може бути важливою для користувача.

Розширені методи пошуку для мереж P2P:

Метод випадкового блукання у неструктурованих однорангових (P2P) мережах, таких як Gnutella, представляє собою значний інтерес у контексті пошуку та виявлення ресурсів. Христос Гканцідіс та інші дослідники висвітлили важливість та ефективність випадкового блукання як альтернативного підходу до лавинної передачі, яка традиційно є домінуючою технікою пошуку в неструктурованих мережах P2P [5].

Специфіка методу випадкового блукання полягає у його здатності адаптуватися до специфічних характеристик топології мережі, особливо коли мережа організована на двох рівнях. Перший рівень представляє собою однорангову кластеризацію, де вузли в мережі формують кластери, що можуть відображати тематичне розділення або спільні інтереси. Другий рівень включає в себе з’єднувальні представники кожного кластера, такі як супервузли, які забезпечують глобальне підключення між різними кластерами. Ця дворівнева структура підвищує ефективність пошуку, оскільки випадкове блукання дозволяє локалізувати пошук в межах релевантних кластерів, зменшуючи марнотратство ресурсів на обробку запитів у несуттєвих областях мережі.

Крім того, ефективність випадкового блукання зростає у випадках, коли одні й ті ж пошукові запити виконуються неодноразово, що є типовою практикою у ситуаціях, коли користувачі шукають нових партнерів або оновлюють свої пошукові запити для виявлення нових ресурсів. У таких сценаріях, незважаючи на стабільність загальної топології мережі, випадкове блукання дозволяє ефективно ідентифікувати нові та релевантні ресурси без необхідності значного переформатування мережевої структури (рис. 1.2).

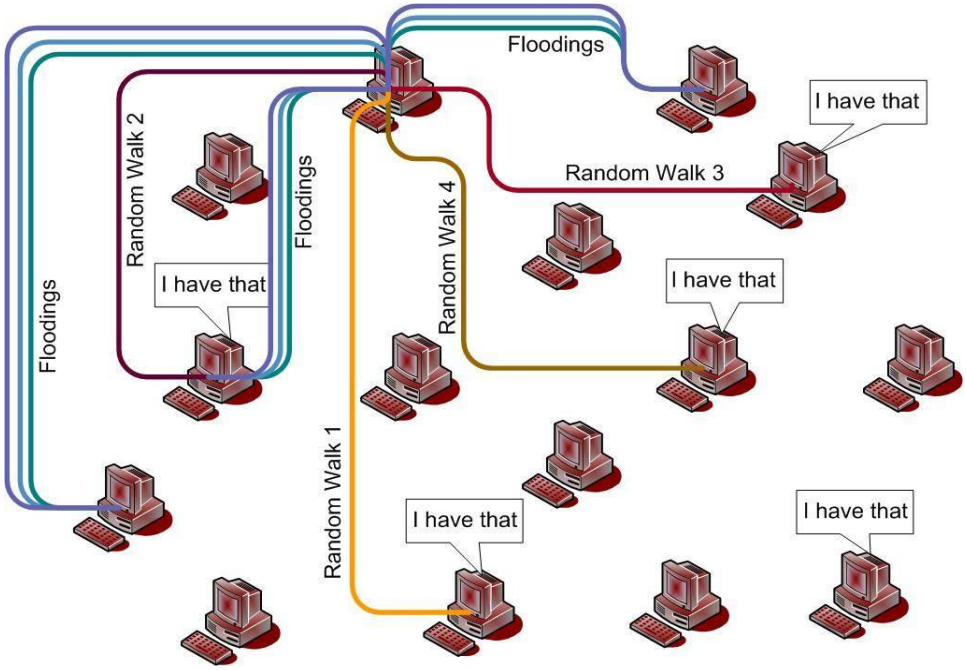

Рисунок 1.2 – Випадок повторних пошуків

Техніка перколяційного пошуку в однорангових (P2P) мережах займає значне місце у дослідженнях комп'ютерних мереж, особливо в контексті неструктурованих мереж, які характеризуються розподілом за степеневим законом (Power-Law, PL). Цей метод ефективно використовує структурні особливості таких мереж, де невелика кількість однорангових вузлів має високий ступінь з'єднань, тоді як більшість вузлів мають низький ступінь з'єднань [1].

У мережах з розподілом PL, вузол вважається високопідключеним, якщо його ступінь з'єднань дорівнює або перевищує половину максимального ступеня з'єднань у мережі. Перколяційний пошук оптимізує процес виявлення ресурсів, спрямовуючи запити до цих високопідключених вузлів, які через свої численні з'єднання мають велику ймовірність успішної відповіді на запит (рис. 1.3).

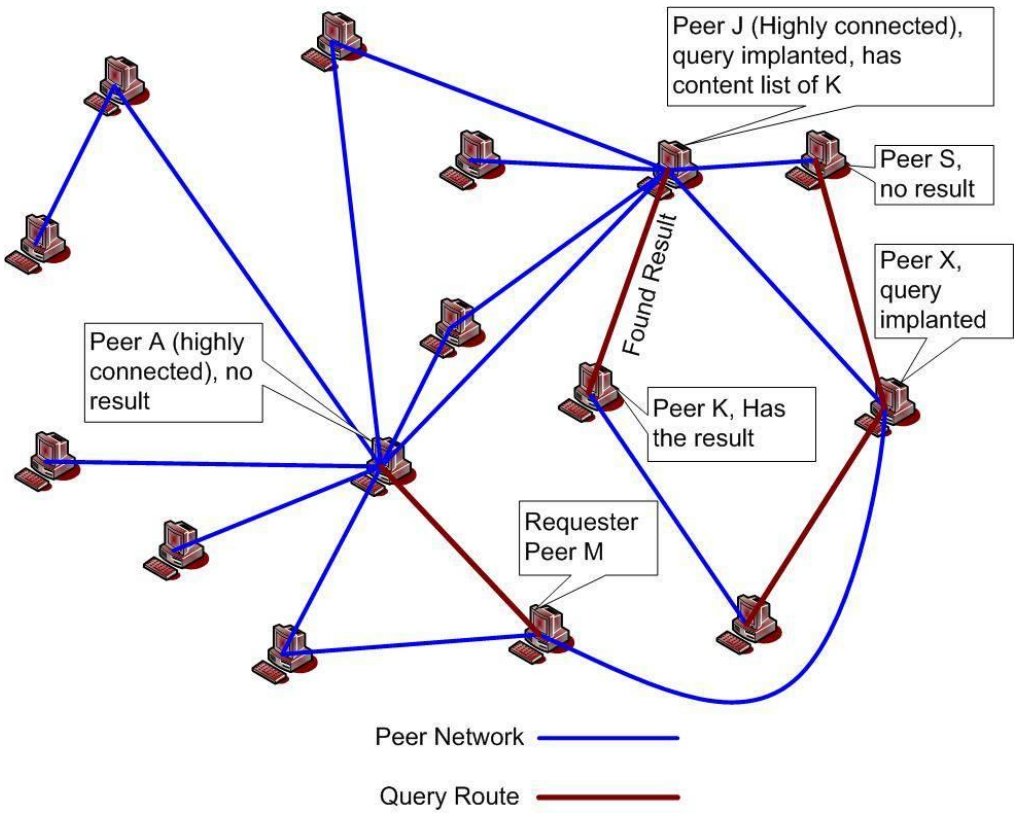

Рисунок 1.3 – Приклад, що показує фазу пошуку перколяції зв’язку

Процес перколяційного пошуку можна поділити на три основні фази:

Імплантація списку вмісту: Коли одноранговий вузол приєднується до мережі, він ініціює процес кешування або імплантації свого списку вмісту у вузлах мережі. Це досягається шляхом випадкового блукання через мережу, починаючи від самого себе, і дублювання свого списку вмісту на кожному кроці. Розмір цього випадкового блукання визначається як O(log N), де N - загальна кількість однорангових вузлів у мережі. Це забезпечує, що інформація про будь-який вміст буде доступна хоча б на одному високопідключеному вузлі.

Імплантація запиту: На цій стадії запит імплантується аналогічним чином, як і список вмісту, за допомогою механізму випадкового блукання. Це підвищує ймовірність того, що запит досягне вузлів із високим ступенем підключення.

Перколяційний пошук: Остання фаза полягає в активному пошуку, де запит пересилається до високопідключених вузлів, що забезпечує високу ймовірність отримання відповіді на запит через широкі мережеві зв’язки цих вузлів.

Ці фази перколяційного пошуку узгоджуються з загальною стратегією оптимізації використання ресурсів у неструктурованих P2P-мережах, забезпечуючи ефективне розповсюдження інформації та оптимізуючи швидкість та точність пошукових запитів.

2 ПРОЕКТУВАННЯ

2.1 Використання UML під час розробки системи

UML (Unified Modeling Language) є стандартом для візуалізації, специфікації, конструювання та документування програмних систем. Використання UML в розробці систем дозволяє команді розробників чітко розуміти вимоги, взаємодію та структуру системи.

Ось деякі ключові елементи використання UML під час розробки систем:

1) Вимоги і аналіз системи (Use Case Diagrams, Activity Diagrams):
   - Діаграми випадків використання (Use Case Diagrams): Дозволяють моделювати взаємодію між системою та її користувачами, ідентифікувати функціональність та визначати основні сценарії використання.
   - Діаграми активностей (Activity Diagrams): Використовуються для моделювання потоків роботи та дій в системі, щоб зрозуміти послідовність подій.
2) Аналіз об’єктів (Class Diagrams, Object Diagrams):
   - Діаграми класів (Class Diagrams): Дозволяють визначити структуру системи, класи, їх атрибути та методи, а також взаємозв’язки між класами.
   - Діаграми об’єктів (Object Diagrams): Моделюють конкретні екземпляри класів та їх стан в певний момент часу.
3) Динаміка системи (Sequence Diagrams, Collaboration Diagrams):
   - Діаграми послідовностей (Sequence Diagrams): Показують взаємодію об’єктів у конкретних сценаріях та порядок викликів методів.
   - Діаграми співпраці (Collaboration Diagrams): Відображають структуру об’єктів та їх взаємодію в момент часу.

4) Стан системи (Statechart Diagrams):
   - Діаграми станів (Statechart Diagrams): Використовуються для моделювання станів об’єктів та переходів між ними відповідно до подій.
5) Компоненти та розгортання системи (Component Diagrams, Deployment Diagrams):
   - Діаграми компонентів (Component Diagrams): Моделюють архітектуру системи з точки зору її компонентів та взаємодії між ними.
   - Діаграми розгортання (Deployment Diagrams): Показують фізичну архітектуру системи та взаємодію з навколишнім середовищем.
6) Взаємодія системи (Communication Diagrams):
   - Діаграми взаємодії (Communication Diagrams): Відображають структуру системи та обмін повідомленнями між її елементами.

Використання UML дозволяє розробникам взаємодіяти, співпрацювати та легко розуміти різні аспекти системи на ранніх етапах розробки, сприяючи створенню більш якісного та зрозумілого програмного продукту.

2.2 Діаграма варіантів використання

Діаграма варіантів використання (Use Case Diagram) є одним із ключових елементів UML і використовується для моделювання функціональних вимог до системи та взаємодії між системою та її акторами (користувачами чи іншими системами). Основна мета цієї діаграми - показати, як користувачі взаємодіють із системою та як система реагує на їхні дії.

Основні елементи діаграми варіантів використання:

1) Актори (Actors):
   - Актори представляють ролі, які взаємодіють із системою. Це може бути користувач, інша система або зовнішній компонент.

2) Варіанти використання (Use Cases):
   - Варіанти використання описують конкретні функціональні можливості або послуги, які система надає для своїх акторів. Кожен варіант використання представляє конкретний сценарій взаємодії.
3) Відносини між акторами та варіантами використання:
   - Відносини показують, які варіанти використання доступні для кожного актора. Наприклад, один актор може мати доступ до кількох варіантів використання, або кілька акторів можуть використовувати один і той самий варіант використання.
4) Система:
   - Система є центральним елементом, що оточує всі варіанти використання та акторів. Це показує, як взаємодіють всі складові системи.

Діаграма варіантів використання є ефективним засобом комунікації між розробниками, замовниками та іншими учасниками проекту. Вона допомагає зрозуміти функціональні вимоги до системи та визначити, як користувачі будуть взаємодіяти з системою на рівні високого рівня.

На рисунку 2.1 представлена діаграма варіантів використання.

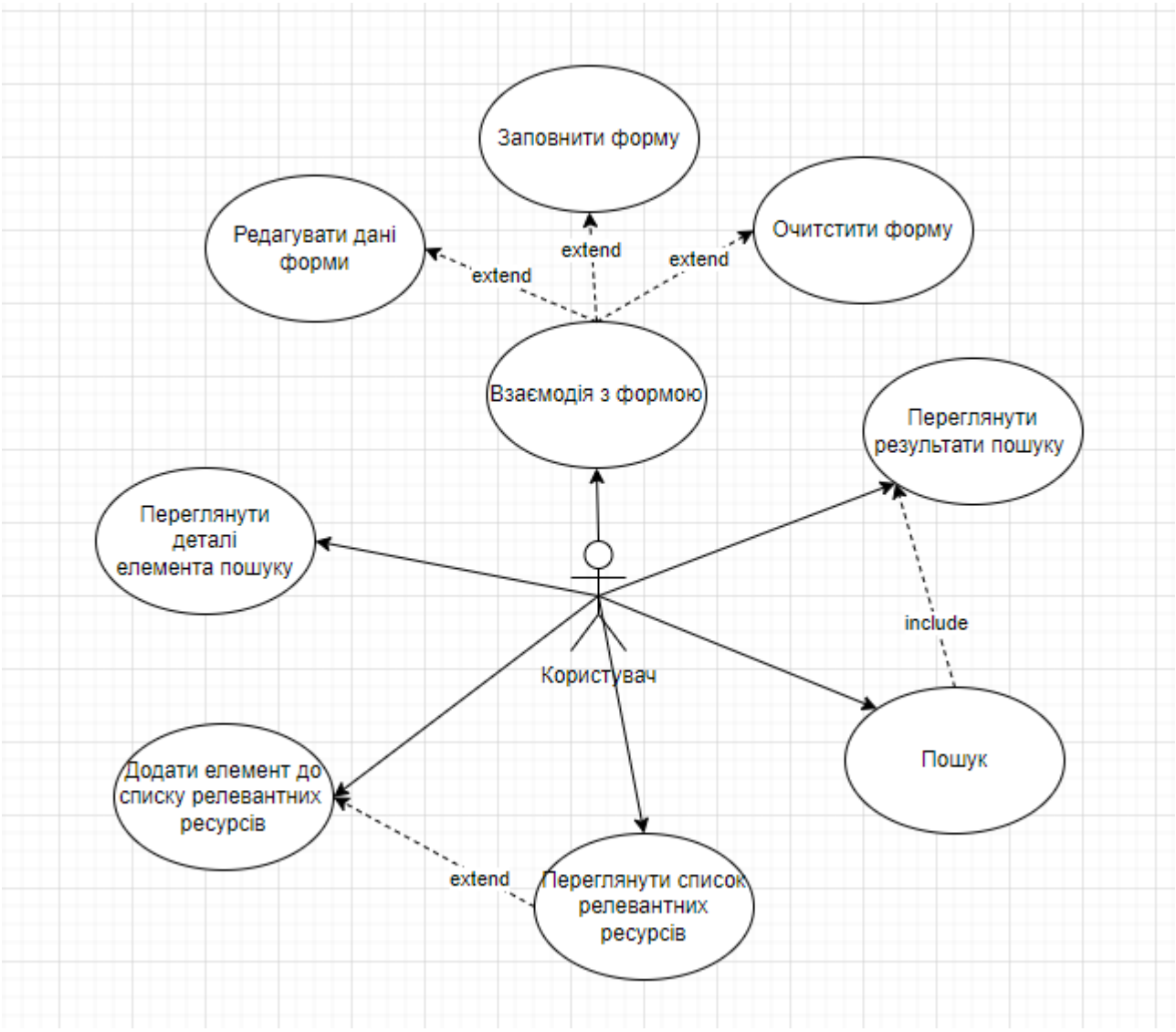

Рисунок 2.1 – Діаграма варіантів використання

На діаграмі представлено актора «Користувач», який відображає функціональні можливості користувачів системи.

Виділено 1 основний варіант використання – «Пошук». Після входу користувача на сайт системи пошуку, система відображає форму та поле результату, яке містить текст «Результати пошуку відсутні» Після того, як користувач заповнює поля форми (опціонально) та натискає на кнопку «Пошук», система відображає результат, що відповідає вказаним параметрам.

Варіанти використання визначають функціональні можливості. Кожен з них представляє певний спосіб використання. Таким чином, кожен варіант

використання відповідає послідовності дій для того, щоб користувач міг отримати певний результат.

2.2.1 Опис варіантів використання

Прецедент «Заповнити форму»

Призначення: даний варіант використання надає можливість користувачу вводити дані до полів форми.

Основний потік подій: даний варіант використання починає виконуватися, коли користувач вводить дані до відповідних полів форми. Система перевіряє валідність даних.

Альтернативний потік: користувач не ввів дані – система ніяк не реагує (блокування кнопок, обов'язкові поля), користувач може взаємодіяти з іншими елементами інтерфейсу.

Передумова: перед початком виконання даного варіанта використання користувач повинен знаходитися на веб-сторінці системи.

Прецедент «Редагувати дані форми»

Призначення: даний варіант використання надає можливість користувачу змінювати дані, які вже були введені в поля форми.

Основний потік подій: даний варіант використання починає виконуватися, коли користувач змінює значення полів форми. Система перевіряє валідність даних.

Передумова: перед початком виконання даного варіанта використання користувач повинен ввести дані в відповідне поле.

Прецедент «Очистити форму»

Призначення: даний варіант використання надає можливість користувачу очистити поля форми.

Основний потік подій: даний варіант використання починає виконуватися, коли користувач натискає на кнопку «Очистити». Система автоматично видаляє дані з усіх полів форми.

Альтернативний потік: форма не має даних – система ніяк не реагує (блокування кнопок), користувач може натиснути на кнопку «Очистити».

Передумова: перед початком виконання даного варіанта використання користувач повинен перейти на веб-сторінку системи.

Прецедент «Пошук»

Призначення: даний варіант використання надає можливість користувачу здійснювати пошук за обраними параметрами або без них.

Основний потік подій: даний варіант використання починає виконуватися, коли користувач заповнює необхідні поля форми (опціонально) та натискає на кнопку «Пошук». Система надсилає запит до БД засобами REST API та відображає результат.

Передумова: перед початком виконання даного варіанта використання користувач повинен перейти на веб-сторінку системи.

Виняткова ситуація 1: результати відсутні – система відображає повідомлення «Результати пошуку відсутні». Користувач може змінити дані та спробувати знов.

Виняткова ситуація 2: API-ключ невалідний – система відображає відповідне повідомлення, користувач повинен звернутися до адміністратора системи.

Виняткова ситуація 3: сервер не відповідає – система відображає відповідне повідомлення, користувач може спробувати здійснити пошук пізніше.

Прецедент «Переглянути результати пошуку»

Призначення: даний варіант використання надає можливість користувачу переглядати результати пошуку.

Основний потік подій: даний варіант використання починає виконуватися, коли користувач натискає на кнопку «Пошук» або переходить на/оновлює головну сторінку. Система відображає список елементів, що відповідають критеріям пошуку.

Альтернативний потік подій: якщо жодного елемента не знайдено, то система відображає повідомлення «Результати пошуку відсутні».

Передумова: перед початком виконання даного варіанта використання користувач повинен перейти на веб-сторінку системи або натиснути на кнопку «Пошук».

Прецедент «Переглянути деталі елемента пошуку»

Призначення: даний варіант використання надає можливість користувачу переглядати елемент пошуку, шляхом переходу за посиланням.

Основний потік подій: даний варіант використання починає виконуватися, коли користувач натискає на назву елемента пошуку, система переадресовує на сторінку опису елемента на ресурсі ScienceDirect.

Передумова: перед початком виконання даного варіанта використання у полі результату повинен бути щонайменше один елемент.

Прецедент «Додати елемент до списку релевантних ресурсів»

Призначення: даний варіант використання надає можливість користувачу додавати елементb до списку релевантних ресурсів.

Основний потік подій: даний варіант використання починає виконуватися, коли користувач натискає на кнопку «Додати до релевантного». Система додає ресурс до загального списку та відображає сторінку пошуку.

Передумова: перед початком виконання даного варіанта використання користувач повинен знаходитися на сторінці пошуку, яка повернула щонайменше один результат або на сторінці перегляду списку релевантних джерел, що містить щонайменше одне джерело. Також необхідною умовою є те, що джерело повинно посилатися на інші джерела або інші джерела мають посилатися на поточне.

Прецедент «Переглянути список релевантних ресурсів»

Призначення: даний варіант використання надає можливість користувачу переглядати список релевантних ресурсів.

Основний потік подій: даний варіант використання починає виконуватися, коли натискає на кнопку «Переглянути релевантні». Система відображає

сторінку зі списком релевантних джерел, а також джерел які посилаються на поточне та на які посилається поточне джерело.

Альтернативний потік подій: якщо жодного елемента не додано до списку релевантних джерел або відсутні джерела на які посилається поточне, або які посилаються на поточне, то система відображає відповідне повідомлення.

Передумова: перед початком виконання даного варіанта використання користувач повинен перейти на веб-сторінку системи.

Виняткова ситуація 1: результати відсутні – система відображає відповідне повідомлення. Користувач може повернутися до інтерфейсу пошуку.

Виняткова ситуація 2: API-ключ невалідний – система відображає відповідне повідомлення, користувач повинен звернутися до адміністратора системи.

2.3 Діаграма діяльності

Діаграма діяльності (Activity Diagram) входить в стандарт UML і використовується для моделювання послідовності дій або процесів у системі. Ця діаграма дозволяє візуалізувати, як об’єкти взаємодіють у конкретних сценаріях та як контролюється потік даних. Основна мета діаграми діяльності - представити логіку процесів та покращити розуміння, як вони виконуються.

Основні елементи діаграми діяльності:

1) Дія (Action):
   - Дії представляють конкретні операції або кроки, які виконуються в процесі. Наприклад, "створити користувача" або "вивести повідомлення".
2) Рішення (Decision):
   - Рішення вказують на різні шляхи або умови, які впливають на подальші кроки у процесі.
3) Об’єкт (Object):

– Об’єкти показують, які об’єкти беруть участь у діяльності та взаємодіють між собою.

4) Старт та Завершення (Initial and Final Nodes):
   – Початковий вузол (Initial Node) вказує початок діаграми, а Кінцевий вузол (Final Node) вказує завершення процесу.
5) Форк та Злиття (Fork and Join Nodes):
   – Форк (Fork Node) дозволяє одночасно виконувати кілька дій, а Злиття (Join Node) об’єднує паралельно виконані дії.
6) Потік керування (Control Flow):
   – Потік керування вказує напрямок послідовності виконання дій та з’єднання між вузлами діаграми.

Діаграма діяльності допомагає розробникам та іншим учасникам проекту визначити та зрозуміти послідовність подій та керування в рамках конкретної діяльності чи процесу.

Наведемо діаграму діяльності, що описує модель поведінки варіанта використання «Пошук» та механізм релевантних ресурсів. Діаграма варіанта використання «Пошук» представлена на рисунках 2.2 – 2.3.

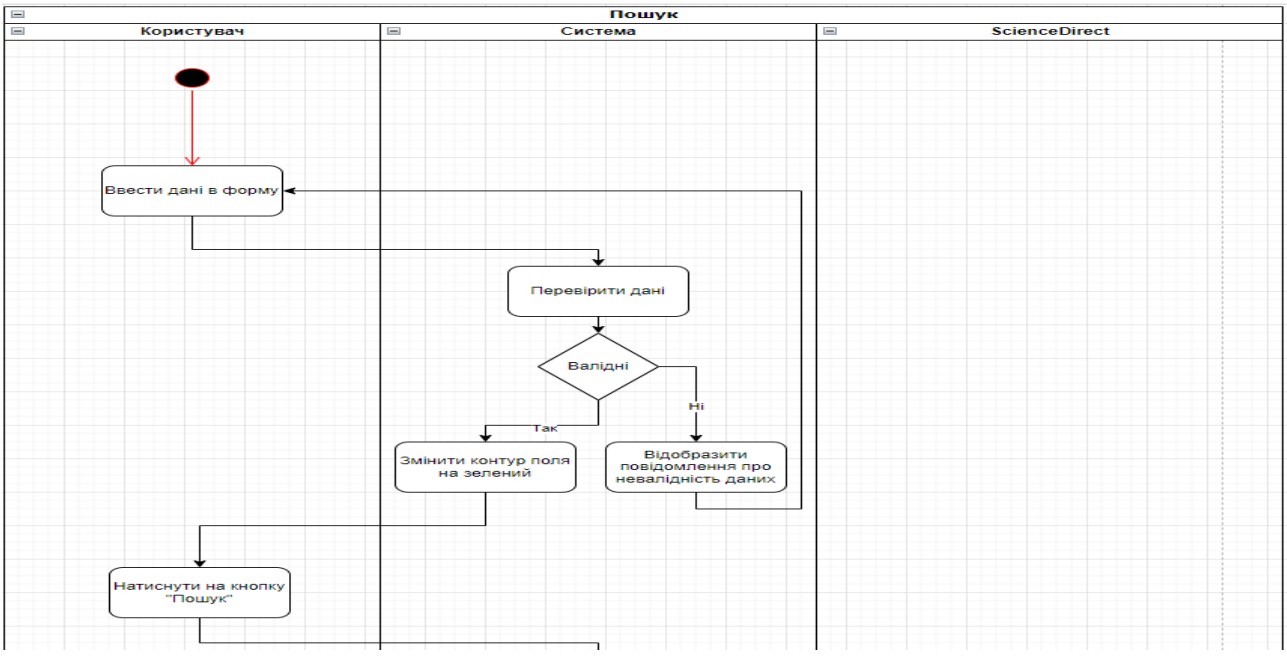

Рисунок 2.2 – Діаграма діяльності ДВВ «Пошук»

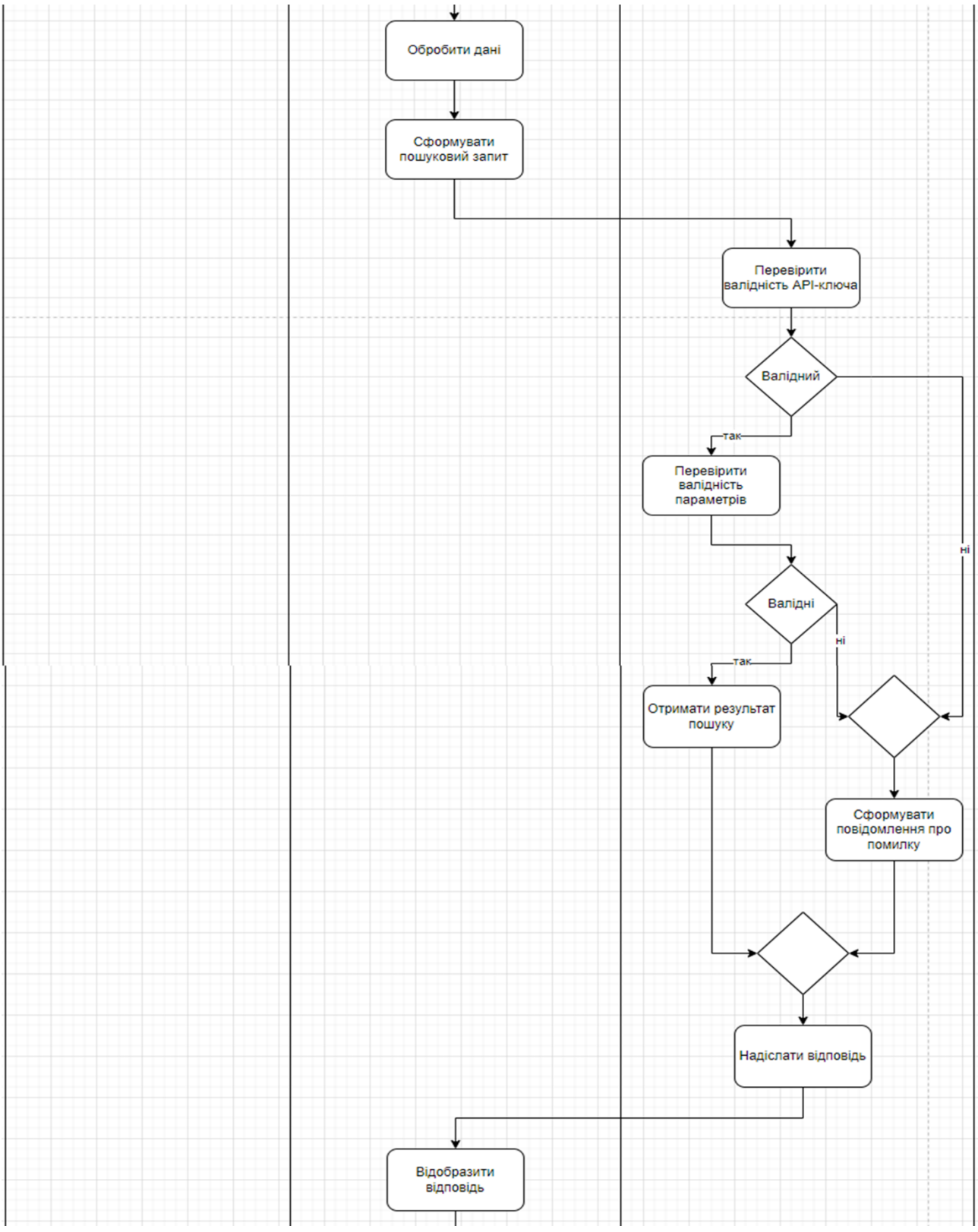

Рисунок 2.3 – Діаграма діяльності ДВВ «Пошук»

Так як основною задачею є застосування алгоритму розповсюдження запиту до пошуку у БД, то на рисунку 2.4 продемонструємо цей алгоритм на прикладі механізма релевантності.

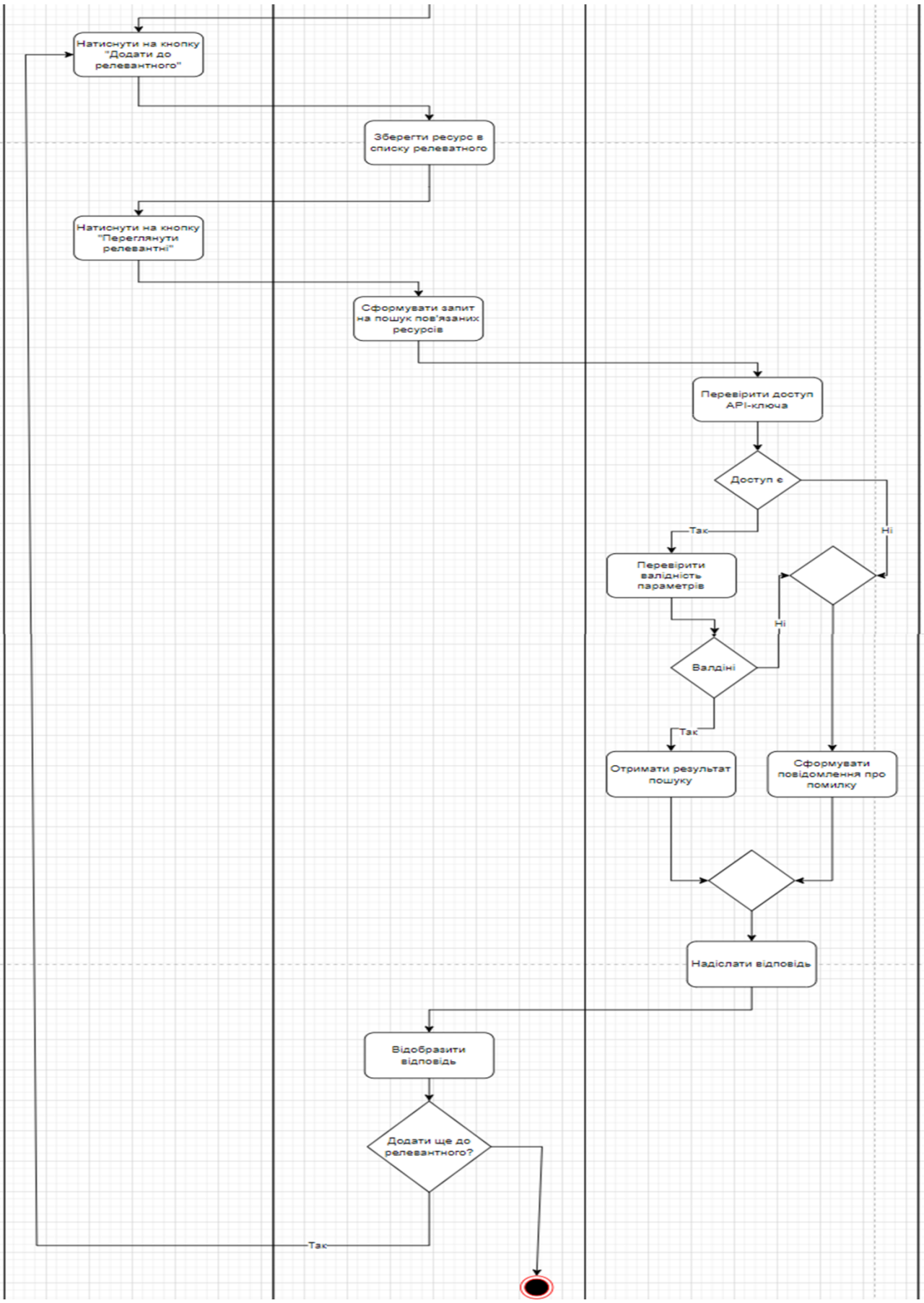

Рисунок 2.4 – Діаграма діяльності алгоритму

2.4 Діаграма послідовності

Діаграма послідовності (Sequence Diagram) входить в стандарт UML і використовується для моделювання взаємодії між об’єктами у конкретному сценарії. Ця діаграма відображає послідовність повідомлень, які передаються між об’єктами у конкретних умовах або подіях.

Основні елементи діаграми послідовності:

1) Об’єкти (Objects):
   - Об’єкти відображають ролі або класи, які беруть участь у взаємодії. Об’єкти зображуються великими прямокутниками з назвою класу або об’єкта.
2) Лінії життя (Lifelines):
   - Лінії життя вказують існування об’єкта протягом певного періоду часу та визначають область, де об’єкт може взаємодіяти.
3) Повідомлення (Messages):
   - Повідомлення показують обмін інформацією або виклик методу між об’єктами. Вони представлені стрілками, що вказують напрямок передачі повідомлення.
4) Фрейми (Frames):
   - Фрейми використовуються для виділення певних відсортованих частин діаграми та підкреслення певних аспектів взаємодії.
5) Засоби керування (Control Constructs):
   - Засоби керування, такі як умови та цикли, можуть бути додані для показу взаємодії в умовах або циклічних структурах.

Діаграма послідовності дозволяє ілюструвати порядок виконання операцій, передачу повідомлень та взаємодію об’єктів у конкретних сценаріях використання. Вона є ефективним інструментом для розуміння та комунікації логіки взаємодії між різними компонентами системи.

На рисунку 2.5 описана діаграма послідовності прецедента «Пошук» та механізму релевантності.

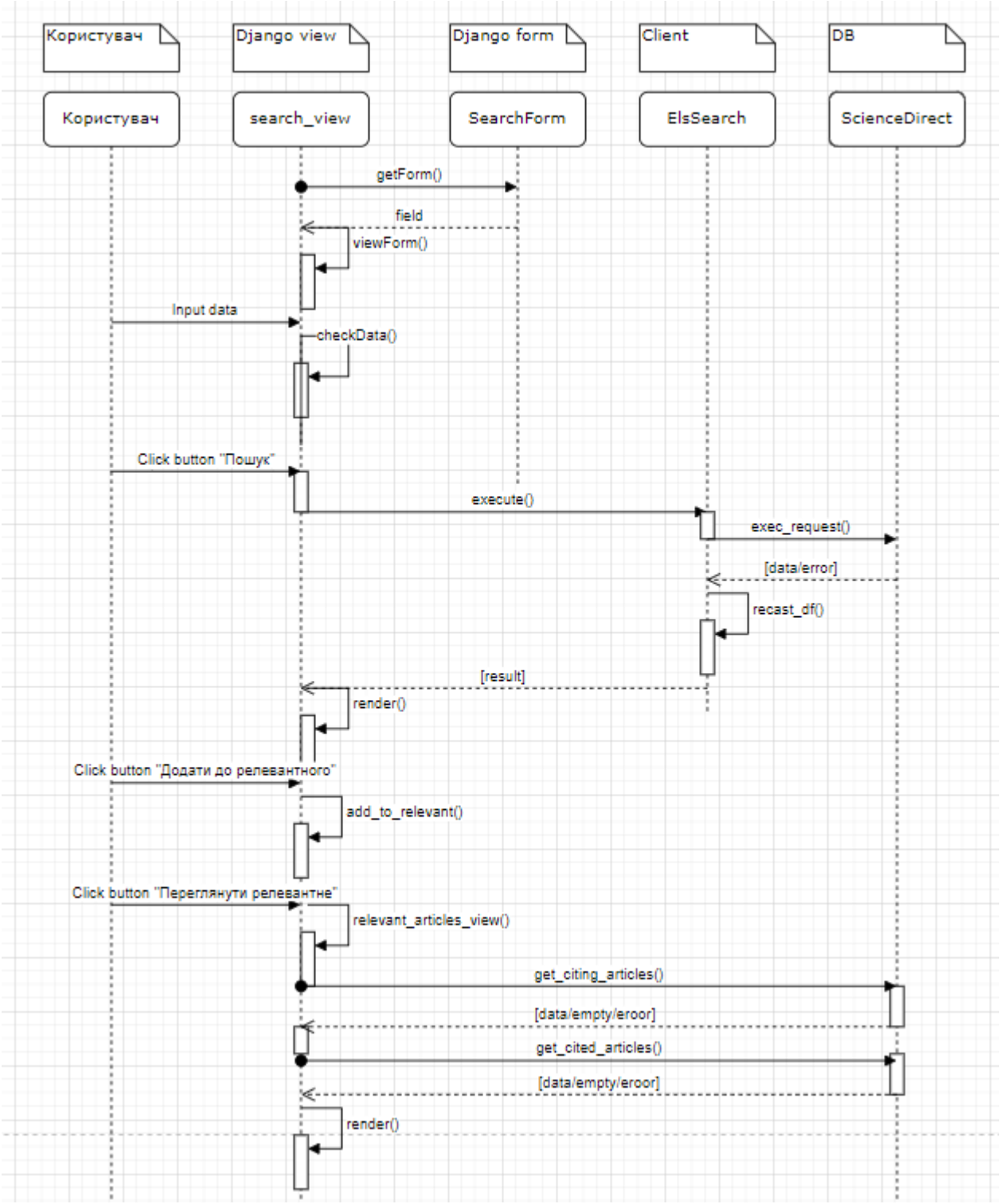

Рисунок 2.5 – Діаграма послідовності

2.5 Діаграма розгортання

Діаграма розгортання (Deployment Diagram) входить в стандарт Unified Modeling Language (UML) і використовується для моделювання фізичного розташування компонентів програмної системи та їх взаємодії на рівні апаратного забезпечення. Ця діаграма дозволяє представити архітектурну конфігурацію системи, визначити розташування обладнання та компонентів, а також показати комунікацію між ними.

Основні елементи діаграми розгортання:

1) Вузли (Nodes):
   - Вузли представляють фізичне або логічне обладнання, на якому виконуються компоненти системи. Це може бути сервер, комп’ютер, мобільний пристрій, тощо.
2) Артефакти (Artifacts):
   - Артефакти представляють програмні або інші файли, які розгортаються на вузлах. Це можуть бути виконувані файли, конфігураційні файли, бази даних, тощо.
3) Взаємодія між вузлами:
   - Відносини між вузлами показують, як вузли взаємодіють між собою, наприклад, які вузли здійснюють взаємодію через мережу.
4) Інтерфейси (Interfaces):
   - Інтерфейси вказують, як компоненти взаємодіють між собою через вузли.
5) Аспекти безпеки (Security Aspects):
   - Діаграма розгортання може включати аспекти безпеки для визначення заходів забезпечення на рівні розгортання системи.

Діаграма розгортання допомагає команді розробників та архітекторам системи розуміти фізичну архітектуру системи, включаючи розташування обладнання та розподіл компонентів.

На рисунку 2.6 наведено діаграму розгортання системи візуалізації даних.

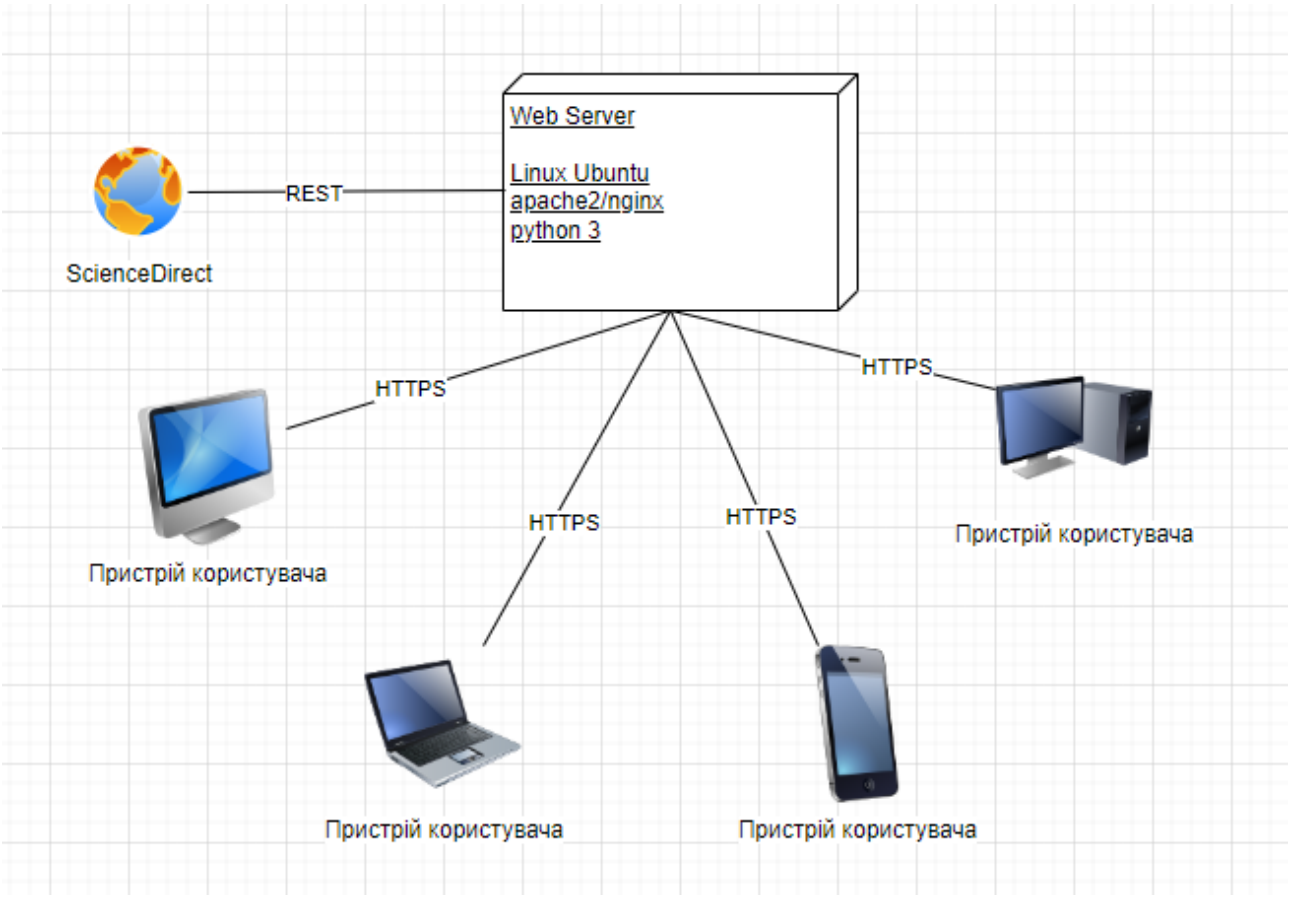

Рисунок 2.6 – Діаграма розгортання

3 РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ

3.1 Опис інструментів розробки

Для реалізації були використані Django та модуль elsapy.

Elsapy – модуль Python для використання з API Elsevier: Scopus, ScienceDirect, тощо.

Django – відкритий фреймворк для розробки веб-додатків, написаний на мові програмування Python.

3.2 Django view

Функція або клас, яка приймає HTTP-запит, обробляє його і повертає HTTP-відповідь. View визначає те, яка інформація повинна бути відображена на сторінці та як ця інформація повинна бути відформатована [12].

На рисунку 3.1 наведено приклад відображення:

```python
def search_view(request):
    search_results = None
    if request.method == 'POST':
        form = SearchForm(request.POST)
        if form.is_valid():
            search_query = build_search_query(form.cleaned_data)
            doc_search = ElsSearch(search_query, 'scopus')
            client = ElsClient('aae02033e85ba53199774d3e99b37ddb')
            doc_search.execute(client)
            search_results = doc_search.results
    else:
        form = SearchForm()
    result_data = []
    hide = False
    if search_results and not search_results[0].get('error'):
        hide = True
        for entry in search_results:
            result_data.append({
                'prism_url': entry.get('link')[2]["@href"],
                'dc_title': entry.get('dc:title'),
                'dc_creator': entry.get('dc:creator'),
                'prism_cover_display_date': entry.get('prism:coverDisplayDate'),
            })

    return render(request, 'search.html', {'form': form, 'search_results': search_results, 'result_data': result_data, 'hide': hide})
```

Рисунок 3.1 – Приклад відображення

Повний код відображення наведено в Додатку А.

3.3 Django form

Django форми використовуються для створення HTML-форм на стороні сервера, обробки введених даних та валідації користувацького вводу [12].

На рисунку 3.2 наведено приклад такої форми:

```
class SearchForm(forms.Form):
    author = forms.CharField(label='Пошук за автором', max_length=100, required=False)
    title = forms.CharField(label='Пошук за назвою', max_length=100, required=False)
    keywords = forms.CharField(label='Пошук за ключовими словами', max_length=100, required=False)
    year = forms.IntegerField(label='Пошук від вказанного до поточного року', required=False)
```

Рисунок 3.2 – Приклад форми

Повний код форми наведено в Додатку Б.

3.4 Django template

Django шаблони дозволяють створювати гнучкі та повторно використовувані шаблони для відображення різних частин веб-сайтів на основі вхідних даних з представлень [12].

На рисунку 3.3 наведено приклад шаблону:

```
<form method="post" class="scopus-form">
    {% csrf_token %}
    <div class="container">
        {{ form.as_p }}
    </div>
    <button type="submit" class="scopus-search-button">Search</button>
</form>

<!-- Результати пошуку -->
{% if result_data %}
<h3>Результати пошуку</h3>
{% for result in result_data %}
<div class="scopus-result">
    {% if result.prism_url %}
    <p class="scopus-title"><a href="{{ result.prism_url }}">{{ result.dc_title }}</a></p>
    {% endif %}


    {% if result.dc_creator %}
    <p class="scopus-creator">Автор: {{ result.dc_creator }}</p>
    {% endif %}

    {% if result.prism_cover_display_date %}
    <p class="scopus-cover-display-date">Дата публікації: {{ result.prism_cover_display_date }}</p>
    {% endif %}
</div>
```

Рисунок 3.3 – Приклад шаблону

Повний код шаблону наведено в Додатку В.

3.5 Структура відповіді ScienceDirect

Відповідь надається в форматі JSON та містить інформацію про конкретну наукову публікацію, включаючи її заголовок, автора, афіліацію, інформацію про публікацію, ідентифікатори та інше [3].

На рисунку 3.4 наведено приклад структури:

```
      '@href': "https://www.scopus.com/inward/citedby.uri?partnerID=HzOxMe3b&scp=85168732385&origin=inward"
    }
  ],
  'prism:url': 'https://api.elsevier.com/content/abstract/scopus_id/85168732385',
  'dc:identifier': 'SCOPUS_ID:85168732385',
  'eid': '2-s2.0-85168732385',
  'dc:title': 'Active Research Data Management with the Django Globus Portal Framework',
  'dc:creator': 'Saint N.',
  'prism:publicationName': 'PEARC 2023 - Computing for the common good: Practice and Experience in Advanced Research Computing',
  'prism:isbn': [
    {
```

Рисунок 3.4 – Приклад структури

Повний код структури наведено в Додатку Г.

3.6 Тестування проекту

3.6.1 Unit-тест

Unit-тест (тест одиниці) є видом тестування програмного забезпечення, в якому окремі частини програми, такі як функції, методи або класи, перевіряються для визначення того, чи працюють вони коректно.

Основна ідея unit-тестування полягає в ізоляції окремих одиниць коду та перевірці їхньої коректності ізольовано від інших частин програми.

Приклади такого тестування наведено на рисунках 3.5 – 3.6.

```python
class SearchViewTestCase(TestCase):
    def setUp(self):
        self.factory = RequestFactory()

    @patch('search.views.ElsSearch.execute')
    def test_search_view(self, mock_execute):
        # Setup request with POST data
        request = self.factory.post('/search/', {'author': 'John Doe', 'title': 'Sample Title'})
        request.session = {}
        form = SearchForm(request.POST)

        # Call the view function
        response = search_view(request)

        # Assert that ElsSearch.execute is called with the correct arguments
        mock_execute.assert_called_with('api_key')

        # Add more assertions based on the expected behavior of your view
        self.assertEqual(response.status_code, 200)
        # Add more assertions as needed
```

Рисунок 3.5 – Тестування view

```python
class SearchFormTestCase(TestCase):
    def test_valid_form(self):
        # Test the form with valid data
        data = {'author': 'John Doe', 'title': 'Sample Title', 'keywords': 'Keyword1, Keyword2', 'year': 2022}
        form = SearchForm(data=data)
        self.assertTrue(form.is_valid())

    def test_invalid_form(self):
        # Test the form with invalid data
        data = {'author': 'John Doe', 'title': '', 'keywords': 'Keyword1, Keyword2', 'year': 'invalid_year'}
        form = SearchForm(data=data)
        self.assertFalse(form.is_valid())
        self.assertEqual(len(form.errors), 1)
        self.assertIn('year', form.errors)
```

Рисунок 3.6 – Тестування form

3.6.2 Integration-тест

Integration-тест (тест інтеграцій) — це вид тестування програмного забезпечення, який спрямований на перевірку взаємодії між різними компонентами або модулями системи. Основна мета інтеграційних тестів полягає в тому, щоб виявити помилки або неправильності в процесах взаємодії між компонентами, які можуть виникнути при їхньому об’єднанні в одну систему.

Розглянемо приклад інтеграційного тесту (рис. 3.7).

```
class SearchViewIntegrationTest(TestCase):
    def setUp(self):
        self.client = Client()

    @patch('search.views.ElsSearch.execute')
    def test_search_view_integration(self, mock_execute):
        # Мокуємо ElsSearch.execute, щоб уникнути з'єднання з реальним API
        mock_results = [
            {'dc:title': 'Sample Title 1', 'dc:creator': 'Author 1', 'prism:coverDisplayDate': '2023-01-01'},
            {'dc:title': 'Sample Title 2', 'dc:creator': 'Author 2', 'prism:coverDisplayDate': '2023-01-02'},
        ]
        mock_execute.return_value = MagicMock(results=mock_results)

        # Виконуємо POST-запит на search_view
        response = self.client.post('/search/', {'author': 'John Doe', 'title': 'Sample Title'})

        # Перевіряємо, чи view повертає сторінку з кодом 200
        self.assertEqual(response.status_code, 200)

        # Перевіряємо, чи викликається ElsSearch.execute з очікуваними аргументами
        mock_execute.assert_called_with('api_key')

        # Перевіряємо, чи дані, які ми мокнули, використовуються для відображення сторінки
        self.assertContains(response, 'Sample Title 1')
        self.assertContains(response, 'Author 1')
```

Рисунок 3.7 – Інтеграційне тестування

3.7 Керівництво користувача

3.7.1 Рівень підготовки користувача

Користувач сайту повинен володіти певною кваліфікацією.

Навички користувача для роботи з ПК та навички роботи з web-браузером.

Знайомство з «Керівництвом користувача» та представлення про онлайн-коллекції.

3.7.2 Підготовка до роботи

Перед початком роботи, необхідно отримати API-ключ. Для цього потрібно зареєструватися на сайті https://dev.elsevier.com/ та у вкладці «My API Key» створити ключ доступу.

Приклад такого ключа наведено на рисунку 3.8.

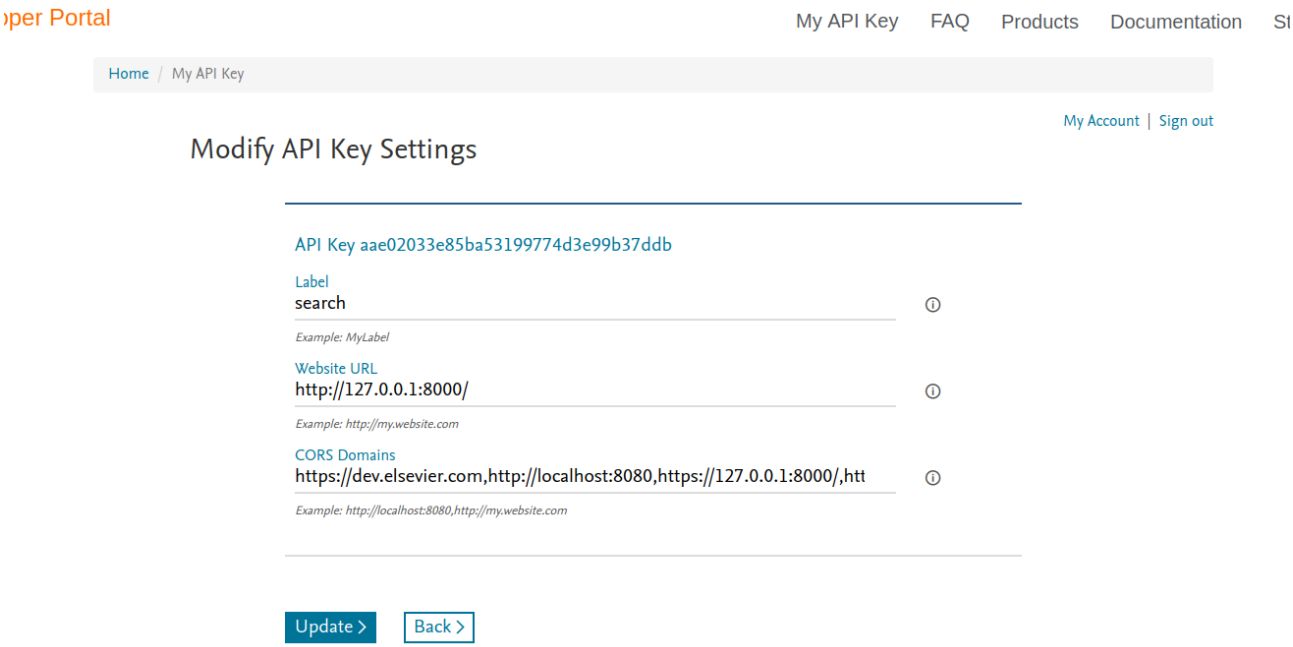

Рисунок 3.8 – Приклад API-ключа

3.7.3 Пошук

Після входу на сайт, система відображає форму для вибору параметрів пошуку та область відображення результатів. За замовчуванням область відображення результатів містить назву «Результати пошуку відсутні» (рис. 3.9).

Рисунок 3.9 – Форма параметрів пошуку

Користувач має можливість гнучкого налаштування параметрів, жодне поле форми не є обов’язковим, що дає можливість шукати за частковими значеннями, наприклад за назвою (рис. 3.10).

Рисунок 3.10 – Результати пошуку за назвою

Після обробки запиту, система повертає результат, який відображається у вигляді списку під формою пошуку. Користувач може взаємодіяти окремо з кожним елементом, для цього потрібно натиснути на назву елемента пошуку. Система переадресує на сторінку з описом ресурсу (рис. 3.11).

Рисунок 3.11 – Сторінка елемента

На цій сторінці користувач може ознайомитись з загальною інформацією (реферат, автори, ключові слова, тощо). Також можна шукати одночасно за двома чи трьома параметрами. Наприклад якщо ми хочемо знайти конкретний ресурс, то можемо в поле пошуку додати автора та ключові слова (див. рис. 3.12 – 3.13).

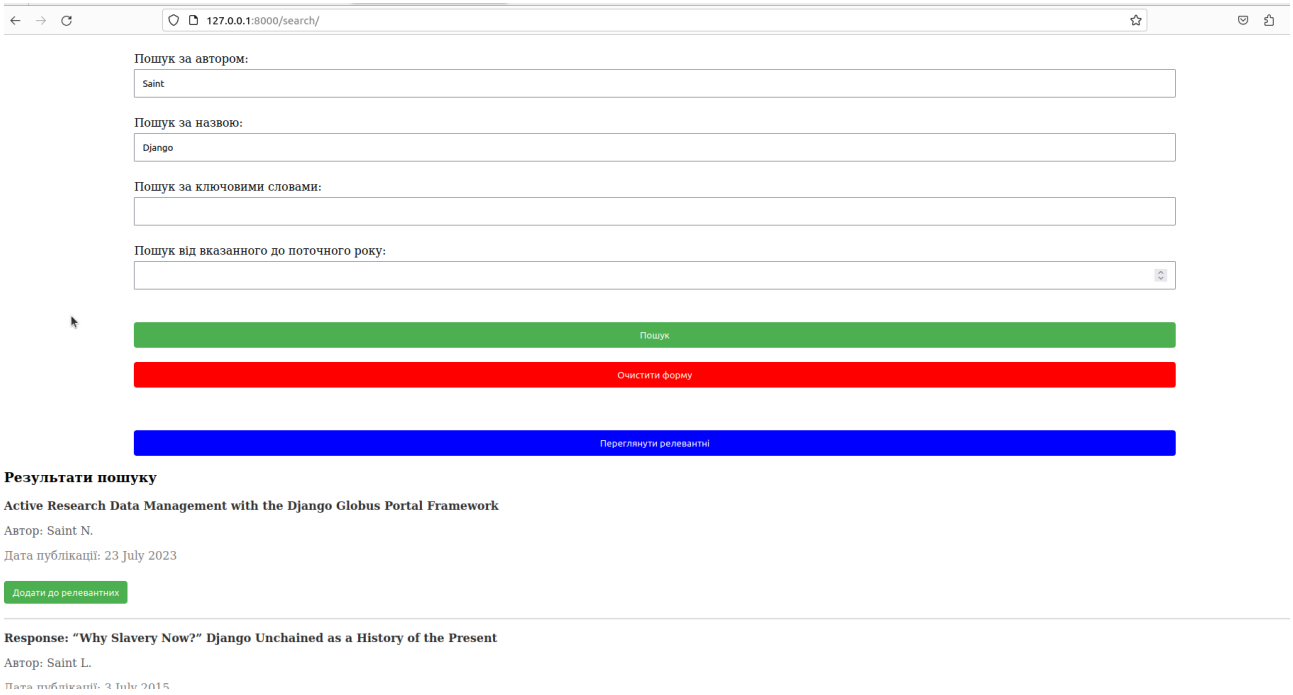

Рисунок 3.12 – Пошук за назвою та автором

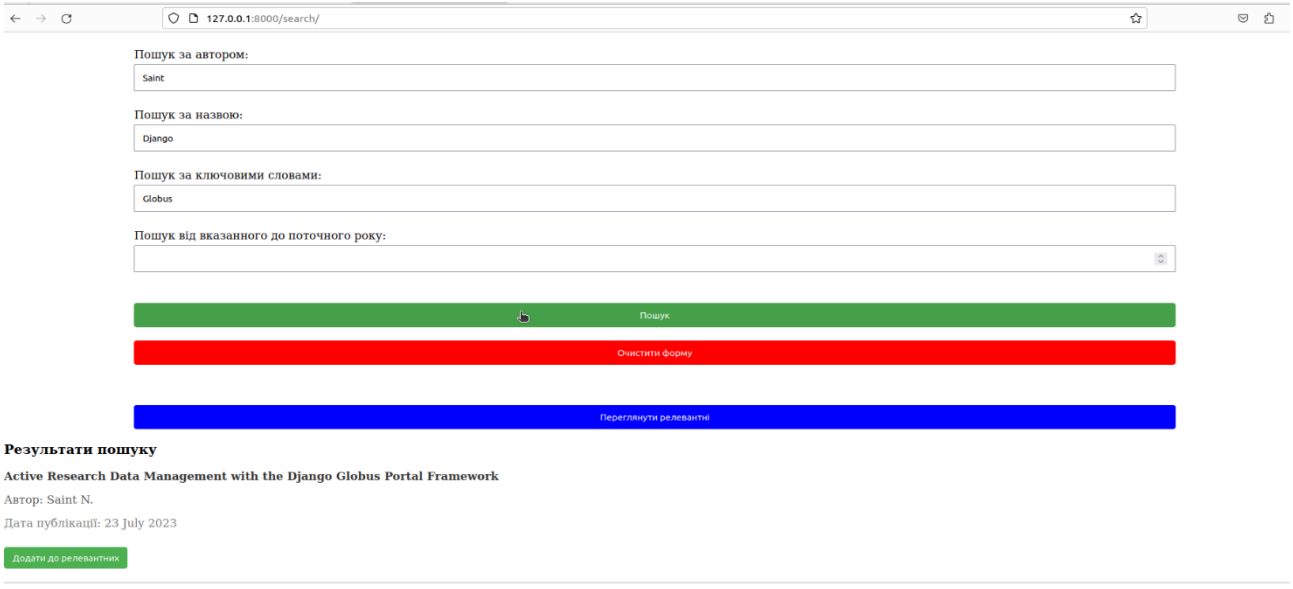

Рисунок 3.13 – Пошук за назвою, автором та ключовими словами

Зауважимо, що якщо параметри пошуку не відповідають ні одному елементу (ресурсу), то система відображає повідомлення «Результати пошуку відсутні» (рис. 3.14).

Рисунок 3.14 – Результати пошуку відсутні

Пошук за датою влаштований таким чином, що користувач вводить початкове значення, а кінцеве – поточний рік. Саме в вказаному діапазоні система буде шукати ресурси (див. рис. 3.15 – 3.16).

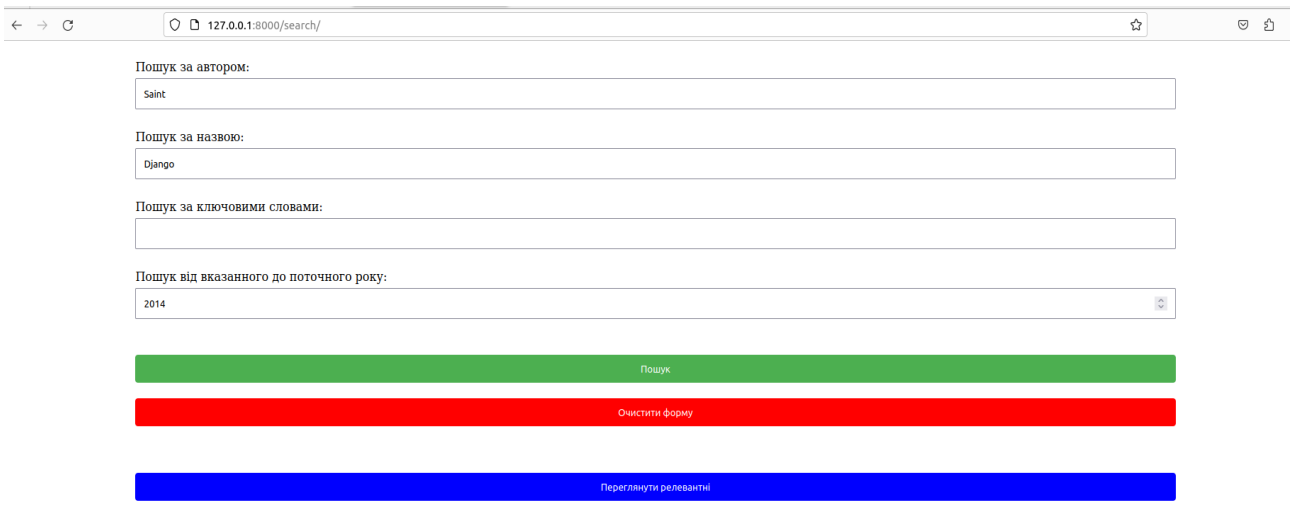

Рисунок 3.15 – Діапазон 2014-2023

Рисунок 3.16 – Діапазон 2016-2023

3.7.4 Списки релевантних ресурсів

Застосування алгоритму спрямованого розповсюдження запиту пошуку до БД реалізовано за допомогою механізму релевантності ресурсів. Користувачу потрібно натиснути на кнопку «Додати до релевантного» (рис. 3.17). Система зберігає інформацію про ресурс.

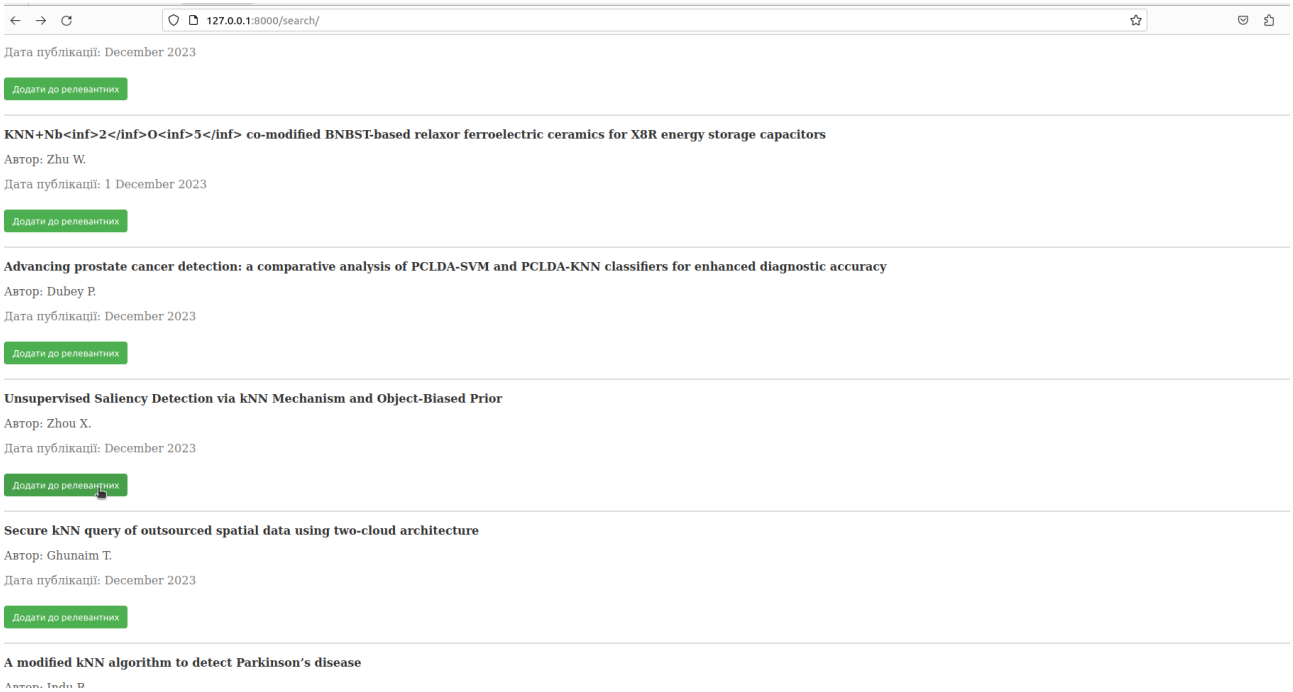

Рисунок 3.17 – Додавання до списку релевантних ресурсів

Для того щоб переглянути список, необхідно натиснути на кнопку «Переглянути релевантні» (рис. 3.18). Система відображає сторінку з переліком релевантних джерел (рис. 3.19).

Рисунок 3.18 – Кнопка переходу на сторінку зі списком релевантних ресурсів

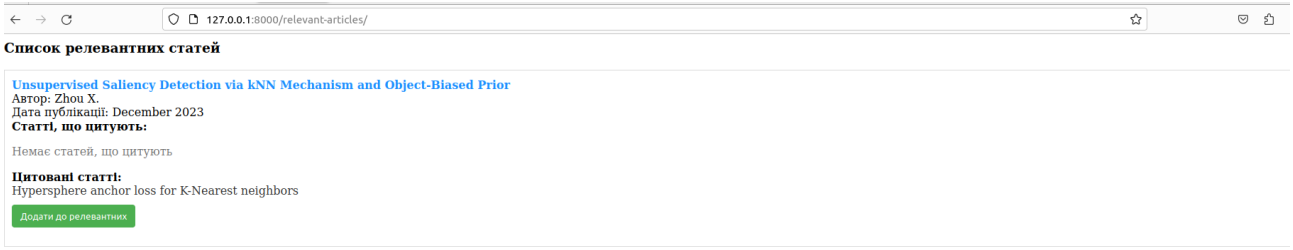

Рисунок 3.19 – Сторінка зі списком релевантних ресурсів

Можемо побачити, що у статті, яку ми додали раніше є цитовані статті, але відсутні статті, що цитують. З кожною статтею можна взаємодіяти шляхом натиснення на її назву, система переадресує на сторінку ресурсу (рис. 3.20). Також користувач може з цією сторінки додати статті до списку релевантних, для цього необхідно натиснути на кнопку «Додати до релевантного» під назвою необхідної статті (рис. 3.21 – 3.22).

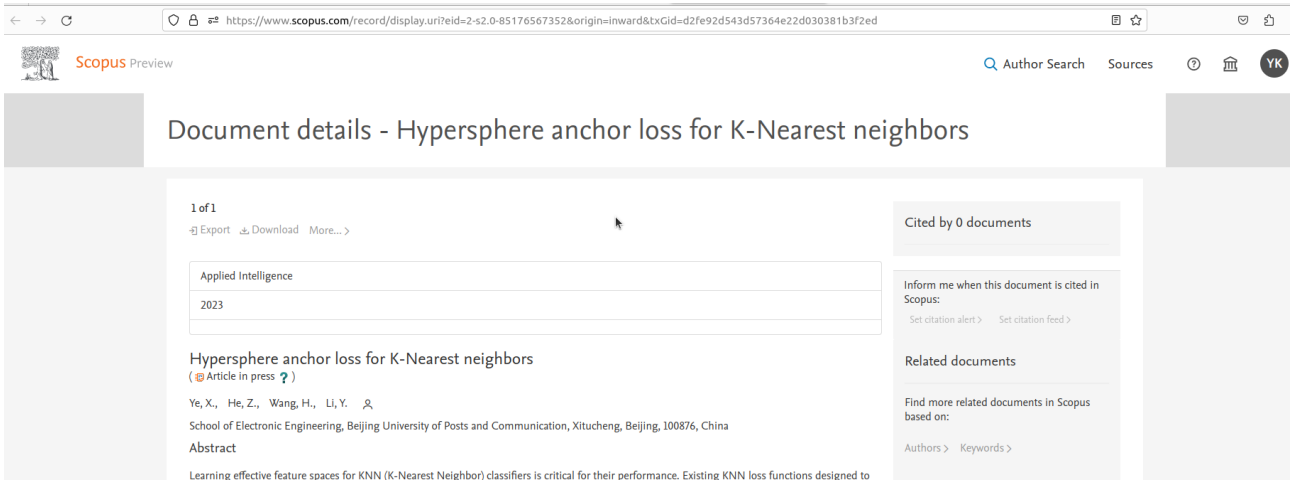

Рисунок 3.20 – Цитована стаття

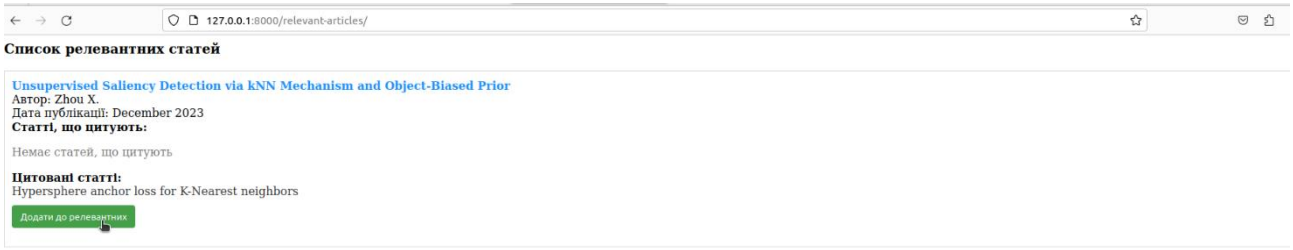

Рисунок 3.21 – Додавання цитованої статті до релевантного

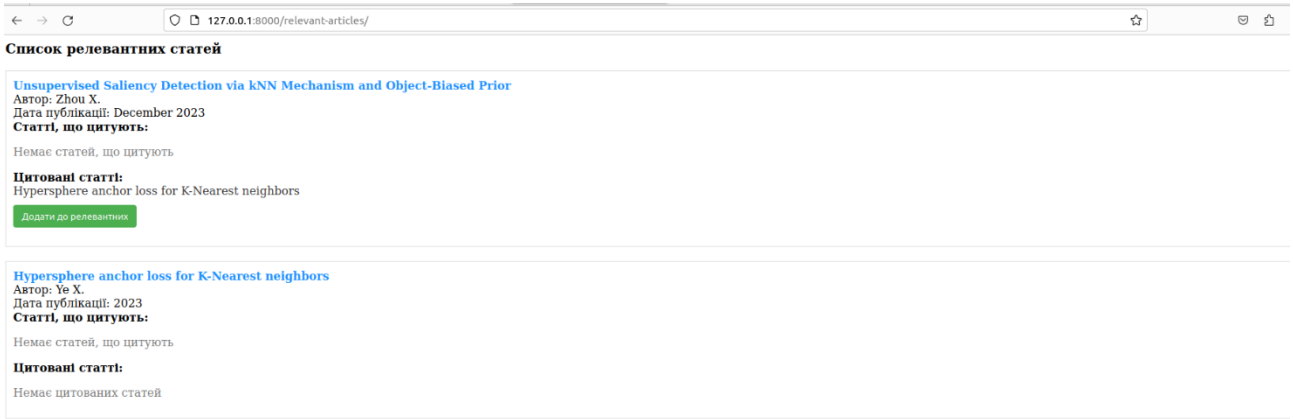

Рисунок 3.22 – Відображення доданої статті

ВИСНОВКИ

В результаті роботи було написано технічне завдання на розробку веб-додатку пошуку у базі даних ScienceDirect. Для створення цієї системи було обрано фреймворк Django як і зі сторони клієнта, так і зі сторони сервера, за його широкі можливості у сфері створення web-систем.

У відповідності з метою кваліфікаційної роботи було розроблено веб-додатку пошуку із застосуванням наступних технологій:

- Django для реалізації backend і frontend;
- Elsapy для зв’язку з ScienceDirect методами REST API.

У відповідності з поставленими задачами були виконані наступні етапи створення системи:

- зібрані, оглянуті та проаналізовані дотичні до теми література та дослідження. Також проведено огляд інструментів розробки;
- спроектована та побудована структура системи (побудовані діаграми прецедентів, діяльності, послідовності та розгортання; надано детальний опис прецедентів);
- реалізовано веб-додатку пошуку у базі даних ScienceDirect (наведена інструкція по створенню компонентів системи, надано керівництво користувача та структура проекту);
- протестована робота системи.

Як результат, створене у роботі програмне забезпечення дозволяє знаходити пов'язані, але несхожі статті, збільшуючи різноманітність результатів пошуку

ПЕРЕЛІК ПОСИЛАНЬ

1. A Survey of Advanced Search in P2P Networks. URL: https://www.medianet.cs.kent.edu/surveys/IAD06S-p2psearch-alok/index.html (дата звернення: 03.10.2023).
2. Database Performance Optimization for SQL Server Based on Hierarchical Queuing Network Model. URL: https://www.researchgate.net/publication/295862329_Database_Performance_Optimization_for_SQL_Server_Based_on_Hierarchical_Queuing_Network_Model (дата звернення: 23.11.2023).
3. Elsevier Developer Portal. URL: https://dev.elsevier.com/ (дата звернення: 15.09.2023).
4. Guide to ScienceDirect Usage Reports. ScienceDirect.com. URL: https://p.widencdn.net/uzrbqe/COUNTER_report_descriptions_ (дата звернення: 03.11.2023).
5. Optimizing Database Query Performance Using Table Partitioning Techniques. URL: https://ieeexplore.ieee.org/document/8672584 (дата звернення: 12.11.2023).
6. Optimizing Database Performance: Strategies for Efficient Query Execution and Resource Utilization. URL: https://ijcttjournal.org/archives/ijctt-v71i7p103 (дата звернення: 12.11.2023).
7. Random Walks in Peer-to-Peer Networks. URL: https://ieeexplore.ieee.org/abstract/document/1354487 (дата звернення: 11.10.2023).
8. Research on Database Application Performance Optimization Method. URL: https://www.atlantis-press.com/proceedings/mmebc-16/25859012 (дата звернення: 12.11.2023).
9. Science, health and medical journals, full text articles and books. URL: https://www.sciencedirect.com/ (дата звернення: 23.11.2023).

10. Survey on Advancing the DBMS Query Optimizer: Cardinality Estimation, Cost Model, and Plan Enumeration. URL: https://doi.org/10.1007/s41019-020-00149-7 (дата звернення: 03.09.2023).

11. The ScienceDirect accessibility journey: A case study. URL: https://onlinelibrary.wiley.com/doi/10.1002/leap.1142 (дата звернення: 23.11.2023).

12. William S. Vincent. Django for Beginners: Build websites with Python and Django. : WelcomeToCode, 2020. 221 p.

ДОДАТОК А

Django view

```
from django.shortcuts import render, redirect
from .forms import SearchForm
from elsapy.elsclient import ElsClient
from elsapy.elssearch import ElsSearch
from elsapy.elsdoc import AbsDoc
import requests


def search_view(request):
    search_results = None
    if request.method == 'POST':
        form = SearchForm(request.POST)
        if form.is_valid():
            search_query = build_search_query(form.cleaned_data)
            doc_search = ElsSearch(search_query, 'scopus')
            client = ElsClient('aae02033e85ba53199774d3e99b37ddb')
            doc_search.execute(client)
            search_results = doc_search.results
            request.session['search_results'] = search_results
    else:
        form = SearchForm()

    result_data = []
    hide = False
    if search_results and not search_results[0].get('error'):
        hide = True
        for entry in search_results:
            result_data.append({
                'prism_url': entry.get('link')[2]["@href"],
                'dc_title': entry.get('dc:title'),
                'dc_creator': entry.get('dc:creator'),
                'prism_cover_display_date': entry.get('prism:coverDisplayDate'),
```

```
                'scopus_url': entry.get('prism:url'),
                'eid': entry.get('eid'),
            })

    return render(request, 'search.html',
                  {'form': form, 'search_results': search_results, 'result_data': result_data, 'hide': hide})


def build_search_query(data):
    query_parts = []
    if data['author']:
        query_parts.append(f"AUTHNAME({data['author']})")
    if data['title']:
        query_parts.append(f"TITLE({data['title']})")
    if data['keywords']:
        query_parts.append(f"AUTHKEY({data['keywords']})")
    if data['year']:
        query_parts.append(f"IS({data['year']})")
    return ' AND '.join(query_parts)


def relevant_articles_view(request):
    relevant_articles = request.session.get('relevant_articles', [])
    extended_relevant_articles = []

    for article in relevant_articles:
        cited_articles_container = []
        # Получаем цитирующие и цитируемые статьи для каждой статьи
        citing_articles = get_citing_articles(article['scopus_url'])
        cited_articles = get_cited_articles(article['scopus_url'])

        # Обновляем информацию о статье
        article['citing_articles'] = citing_articles
        # for cited in article:
        #     cited_articles_container.append()
        article['cited_articles'] = cited_articles

        extended_relevant_articles.append(article)
```

```
    return render(request, 'relevant_articles.html', {'relevant_articles': extended_relevant_articles})


def add_to_relevant(request):
    if request.method == 'POST':
        relevant_articles = request.session.get('relevant_articles', [])
        relevant_articles.append({
            'prism_url': request.POST.get('prism_url'),
            'dc_title': request.POST.get('dc_title'),
            'dc_creator': request.POST.get('dc_creator'),
            'prism_cover_display_date': request.POST.get('prism_cover_display_date'),
            'scopus_url': request.POST.get('scopus_url'),
            'eid': request.POST.get('eid'),
        })
        print(relevant_articles)
        request.session['relevant_articles'] = relevant_articles
    return redirect('search_view')


def get_scopus_id(prism_url):
    # Виділіть Scopus ID (EID) із URL
    parts = prism_url.split('/')
    return parts[-1]


def get_citing_articles(prism_url):
    scopus_id = get_scopus_id(prism_url)
    api_key = 'aae02033e85ba53199774d3e99b37ddb'  # Замініть на свій фактичний ключ API

    # Запит до API для отримання цитуючих статей
    url = f'https://api.elsevier.com/content/search/scopus?query=refeid({scopus_id})&apiKey={api_key}'
    headers = {'Accept': 'application/json'}

    response = requests.get(url, headers=headers)

    if response.status_code == 200:
        citing_articles_data = response.json()
```

```
        # Отримання інформації про цитуючі статті
        citing_articles = []
        for item in citing_articles_data.get('search-results', {}).get('entry', []):
            article_title = item.get('dc:title', '')
            citing_articles.append({'title': article_title})

        print(f"Цитуючі статті: {citing_articles}")

        # Повертаємо результати або додатково обробляємо їх
        return citing_articles
    else:
        print(f"Не вдалося отримати цитуючі статті за Scopus ID. Код відповіді: {response.status_code}")
        return None


def get_cited_articles(prism_url):
    scopus_id = get_scopus_id(prism_url)
    api_key = 'aae02033e85ba53199774d3e99b37ddb'  # Замініть на свій фактичний ключ API

    # Запит до API для отримання цитованих статей
    url = f'https://api.elsevier.com/content/search/scopus?query=cite({scopus_id})&apiKey={api_key}'
    headers = {'Accept': 'application/json'}

    response = requests.get(url, headers=headers)

    if response.status_code == 200:
        cited_articles_data = response.json()

        # Отримання інформації про цитовані статті
        cited_articles = []
        for item in cited_articles_data.get('search-results', {}).get('entry', []):
            article_title = item.get('dc:title', '')
            cited_articles.append({'title': article_title})

        print(f"Цитовані статті: {cited_articles}")

        # Повертаємо результати або додатково обробляємо їх
```

```
        return cited_articles
    else:
        print(f"Не вдалося отримати цитовані статті за Scopus ID. Код відповіді: {response.status_code}")
        return None
```

ДОДАТОК Б

Django form

```
from django import forms


class SearchForm(forms.Form):
    author = forms.CharField(label='Пошук за автором', max_length=100, required=False)
    title = forms.CharField(label='Пошук за назвою', max_length=100, required=False)
    keywords = forms.CharField(label='Пошук за ключовими словами', max_length=100, required=False)
    year = forms.IntegerField(label='Пошук від вказанного до поточного року', required=False)
```

ДОДАТОК В

Django template

```
<!-- Форма пошуку -->
<style>
    a {
      text-decoration: none;
      color: #333;
    }

    .scopus-form {
        display: flex;
        flex-direction: column;
        align-items: center;
        margin-bottom: 20px;
        padding-bottom: 40px;
    }

    .scopus-form .container {
        width: 80%;
        margin-bottom: 20px;
    }

    .scopus-form label {
        display: block;
        text-align: left;
        margin-bottom: 5px;
    }

    .scopus-form input[type="text"], .scopus-form input[type="number"]  {
        width: 100%;
```

```
    padding: 10px;
    margin-bottom: 10px;
}

.scopus-search-button, .scopus-reset-button, .scopus-button {
    color: white;
    padding: 10px;
    border: none;
    border-radius: 4px;
    cursor: pointer;
    width: 80%;
}

.scopus-search-button {
  background-color: #4CAF50;
}

.scopus-reset-button {
  margin-top: 20px;
  background-color: red;
}

.scopus-button {
  display: block;
  background-color: blue;
  margin: 0 auto;
}

.scopus-search-button:hover {
    background-color: #45a049;
}

/* Результати пошуку */
.scopus-result {
```

```
    margin-bottom: 20px;
}

.scopus-prism-url a {
    color: #1e90ff;
    text-decoration: none;
}

.scopus-prism-url a:hover {
    text-decoration: underline;
}

.scopus-title {
    font-weight: bold;
    color: #333;
}

.scopus-creator {
    color: #555;
}

.scopus-cover-display-date {
    color: #777;
}

.scopus-hr {
    border: 0.5px solid #ddd;
}

.add-to-relevant-button {
background-color: #4CAF50;
color: white;
padding: 8px 12px;
border: none;
```

```
    border-radius: 4px;
    cursor: pointer;
    margin-top: 10px;
  }

  .add-to-relevant-button:hover {
    background-color: #45a049;
  }

</style>



{% block content %}
<form method="post" class="scopus-form">
    {% csrf_token %}
    <div class="container">
        {{ form.as_p }}
    </div>
    <button type="submit" class="scopus-search-button">Пошук</button>
    <input type="reset" class="scopus-reset-button" value="Очистити форму">
</form>
<button type="button" class="scopus-button" onclick="redirectToRelevantArticles()">Переглянути релевантні</button>


<script>
    function redirectToRelevantArticles() {
      // Переход на сторінку за адресою relevant-articles/
      window.location.href = "/relevant-articles/";
    }
</script>
<!-- Результаты поиска -->
{% if result_data %}
<h3>Результати пошуку</h3>
{% for result in result_data %}
```

```
<div class="scopus-result">
    <p class="scopus-title"><a href="{{ result.prism_url }}">{{ result.dc_title }}</a></p>
    <p class="scopus-creator">Автор: {{ result.dc_creator }}</p>
    <p class="scopus-cover-display-date">Дата публікації: {{ result.prism_cover_display_date }}</p>
    <!-- Добавить кнопку "Добавить в релевантные" с соответствующим action -->
    <form method="post" action="{% url 'add_to_relevant' %}">
        {% csrf_token %}
        <input type="hidden" name="prism_url" value="{{ result.prism_url }}">
        <input type="hidden" name="dc_title" value="{{ result.dc_title }}">
        <input type="hidden" name="dc_creator" value="{{ result.dc_creator }}">
        <input type="hidden" name="prism_cover_display_date" value="{{ result.prism_cover_display_date
}}">
        <input type="hidden" name="scopus_url" value="{{ result.scopus_url }}">
        <input type="hidden" name="eid" value="{{ result.eid }}">
        <button class="add-to-relevant-button" type="submit">Додати до релевантних</button>
    </form>
</div>

<hr class="scopus-hr">  {# Опціонально: Додати горизонтальну лінію між кожним набором
результатів #}
{% endfor %}
{% endif %}
{% endblock %}

{% if not hide %}
<div class="scopus-result">
    <p class="scopus-cover-display-date">Результати пошуку відсутні</p>
</div>
{% endif %}
```

ДОДАТОК Г

Структура відповіді ScienceDirect


```
{
 ‘@_fa’: ‘true’,
 ‘link’: [
  {
   "@_fa": ‘true’,
   ‘@ref’: ‘self’,
   ‘@href’: "https://api.elsevier.com/content/abstract/scopus_id/85168732385"
  },
  {
   ‘@_fa’: ‘true’,
   ‘@ref’: ‘author-affiliation’,
   ‘@href’: ‘https://api.elsevier.com/content/abstract/scopus_id/85168732385?field=author,affiliation’
  },
  {
   ‘@_fa’: ‘true’,
   ‘@ref’: ‘scopus’,
   ‘@href’:
‘https://www.scopus.com/inward/record.uri?partnerID=HzOxMe3b&scp=85168732385&origin=inward’
  },
  {
   ‘@_fa’: ‘true’,
   ‘@ref’: ‘scopus-citedby’,
   ‘@href’:
"https://www.scopus.com/inward/citedby.uri?partnerID=HzOxMe3b&scp=85168732385&origin=inward"
  }
 ],
 ‘prism:url’: ‘https://api.elsevier.com/content/abstract/scopus_id/85168732385’,
 ‘dc:identifier’: ‘SCOPUS_ID:85168732385’,
 ‘eid’: ‘2-s2.0-85168732385’,
 ‘dc:title’: ‘Active Research Data Management with the Django Globus Portal Framework’,
 ‘dc:creator’: ‘Saint N.’,
```

‘prism:publicationName’: ‘PEARC 2023 - Computing for the common good: Practice and Experience in Advanced Research Computing’,
‘prism:isbn’: [
{
‘@_fa’: ‘true’,
‘$’: ‘9781450399852’
}
],
‘prism:pageRange’: ‘43-51’,
‘prism:coverDate’: ‘2023-07-23’,
‘prism:coverDisplayDate’: ‘23 July 2023’,
‘prism:doi’: ‘10.1145/3569951.3593597’,
‘citedby-count’: ‘0’,
‘affiliation’: [
{
‘@_fa’: ‘true’,
‘affilname’: ‘The University of Chicago’,
‘affiliation-city’: ‘Chicago’,
‘affiliation-country’: ‘United States’
}
],
‘prism:aggregationType’: ‘Conference Proceeding’,
‘subtype’: ‘cp’,
‘subtypeDescription’: ‘Conference Paper’,
‘source-id’: ‘21101185801’,
‘openaccess’: ‘0’,
‘openaccessFlag’: False
}