МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

ІНЖЕНЕРНИЙ НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ
ІМ. Ю.М. ПОТЕБНІ

КАФЕДРА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ
АВТОМАТИЗОВАНИХ СИСТЕМ

Кваліфікаційна робота

другий (магістерський)
(рівень вищої освіти)

на тему Дослідження пошукових систем на прикладі Elasticsearch

Виконав: студент 2 курсу, групи 8.1210-іпз
спеціальності 121 Інженерія програмного забезпечення
(код і назва спеціальності)
освітньої програми Інженерія програмного забезпечення
(код і назва освітньої програми)
Л. В. Ваганов
(ініціали та прізвище)
Керівник доцент, А. І. Безверхий
(посада, вчене звання, науковий ступінь, підпис, ініціали та прізвище)
Рецензент директор ТОВ «Дісітел»
П. О. Лютий
(посада, вчене звання, науковий ступінь, підпис, ініціали та прізвище)

Запоріжжя
2021

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

ІНЖЕНЕРНИЙ НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ
ІМ. Ю.М. ПОТЕБНІ

Кафедра програмного забезпечення автоматизованих систем
Рівень вищої освіти другий (магістерський)
Спеціальність 121 Інженерія програмного забезпечення
(код та назва)
Освітня програма Інженерія програмного забезпечення
(код та назва)

ЗАТВЕРДЖУЮ
Завідувач кафедри В.Г. Вербицький
"01" вересня 2021 року

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТОВІ

Ваганову Леоніду Володимировичу
(прізвище, ім'я, по батькові)

1.Тема роботи Дослідження пошукових систем на прикладі Elasticsearch

керівник роботи Безверхий Анатолій Ігорович, доцент
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом ЗНУ від "30" червня 2021 року № 974-с

2. Строк подання студентом кваліфікаційної роботи 30.11.2021
3. Вихідні дані магістерської роботи
- комплект нормативних документів ;
- технічне завдання до роботи.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
- огляд та збір літератури стосовно теми кваліфікаційної роботи;
- огляд та аналіз існуючих рішень та аналогів;
- дослідження проблеми розпізнавання мов та розробка методів її вирішення;
- створення програмного продукту та його опис;
- перелік вимог для роботи програми;
- дослідження поставленої проблеми та розробка висновків та пропозицій.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)
слайдів презентації

6. Консультанти розділів магістерської роботи

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

7. Дата видачі завдання 01.09.2021

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів магістерської роботи	Строк виконання етапів магістерської роботи	Примітка
1	Аналіз предметної області	01.09-10.09.21	виконано
2	Формулювання основної задачі дипломної роботи та узгодження її з науковим керівником	11.09-12.09.2021	виконано
3	Аналіз існуючих методів рішення	13.09-17.09.21	виконано
4	Дослідження області розробки крос-платформних застосунків	18.09-24.09.21	виконано
5	Узгодження подальших дій з науковим керівником	25.09-26.09.21	виконано
6	Аналіз теоретичних відомостей	27.09-30.09.21	виконано
7	Проектування інтерфейсу веб-застосунку	01.10-12.10.21	виконано
8	Узгодження інтерфейсу з науковим керівником	13.10-14.10.21	виконано
9	Реалізація функціоналу для роботи з пошуковим двигуном	15.10-28.10.21	виконано
10	Представлення отриманих результатів науковому керівнику і узгодження плану подальшого дослідження	29.10-30.10.21	виконано
11	Реалізація функціоналу для аналітичного порівняння пошукових двигунів	31.10-13.11.21	виконано
12	Проведення аналізу роботи з базами даних розробленого програмного застосунку	14.11-18.11.21	виконано
13	Оформлення звіту	19.11-30.11.21	виконано

Студент _____ Л.В. Ваганов
(підпис) (прізвище та ініціали)

Керівник роботи _____ А.І. Безверхий
(підпис) (прізвище та ініціали)

Нормоконтроль пройдено

Нормоконтролер _____ І.А. Скрипник
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Сторінок: 90

Таблиць: 2

Джерел: 27

Ваганов Леонід Володимирович. Дослідження пошукових систем на прикладі Elasticsearch.

Кваліфікаційна робота для здобуття ступеня вищої освіти магістра за спеціальністю 121 — Інженерія програмного забезпечення, науковий керівник А. І. Безверхий. Інженерний навчально-науковий інститут Запорізького національного університету.

Метою завдання було розпочати роботу зі створення сайту, який відображає великі за обсягом слів тексти. Такими текстами можуть слугувати наукові статті, рецензії на книжки, фільми та серіали, короткі розповіді або цілі блоги. Але щоб швидко, якісно та точно опрацьовувати інформацію такого великого розміру бажано використовувати пошуковий двигун. У випадку із цим програмним забезпеченням обрано пошуковий двигун Elasticsearch.

У процесі дослідження розглянуто проблему швидкості використання бази даних для веб-сайтів і якість UX під час роботи з сайтами. Для цього створено веб-сайт мовою PHP.

Ключові слова: *користувач, інтерфейс, elasticsearch, laravel, eloquent, пошуковик, кластер, шард*.

SUMMARY

Pages: 93

Tables: 2

Sources: 27

Vahanov Leonid. Search engine research on the example of Elasticsearch Qualification work for a master's degree in specialty 121 - Software Engineering, research supervisor Anatolii Bezverkhyi. Engineering Educational and Scientific Institute of Zaporizhzhia National University.

The aim of the task was to start work on creating a site that displays large texts. Such texts can be scientific articles, reviews of books, movies and TV series, short stories or entire blogs. But in order to quickly, efficiently and accurately process information of such a large size, it is desirable to use a search engine. In the case of this software, the Elasticsearch search engine was chosen.

The study examined the problem of the speed of using the database for websites and the quality of UX when working with sites. A PHP website has been created for this purpose.

Keywords: *user, interface, elasticsearch, laravel, eloquent, cluster, shard*.

ЗМІСТ

ВСТУП

Актуальність теми

Майже половина всіх користувачів Інтернету зараз користуються пошуковими системами в типовий день

Відсоток користувачів Інтернету, які користуються пошуковими системами в типовий день, постійно зростав з приблизно однієї третини всіх користувачів у 2002 році до нового максимуму - трохи менше половини (49%). З цим збільшенням кількість тих, хто користується пошуковою системою в типовий день, все більше наближається до 60% користувачів Інтернету. Тому якість, повнота та точність результатів пошуку стає більш важливою щоденно.

Мета і завдання дослідження

Метою завдання було розпочати роботу зі створення сайту, який відображає великі за обсягом слів тексти. Такими текстами можуть слугувати наукові статті, рецензії на книжки, фільми та серіали, короткі розповіді або цілі блоги. Але щоб швидко, якісно та точно опрацьовувати інформацію такого великого розміру бажано використовувати пошуковий двигун. У випадку із цим програмним забезпеченням було обрано пошуковий двигун Elasticsearch.

Сайт повинен бути написаним мовою PHP та за допомогою фреймворку Laravel, що має підтримку об’єктно-орієнтовного мапперу Eloquent, який буде підтримувати базу даних MySQL.

Особисте виконання цього завдання зробить внесок в остаточну версію дипломної роботи за темою «Дослідження пошукових систем на прикладі Elasticsearch».

Задачею виробничої практики було створення сайту для пошуку книг за тегами, жанрами, автором або назвою та підтримувати здатність до еластичного пошуку для пошуку за синонімами. Сайт повинен мати такі можливості:

- Додавання власних постів до бази даних сайту;
- Реєстрація/авторизація користувачів;

- Пошук існуючої інформації за тегами;
- Пошук існуючої інформації за жанром;
- Пошук існуючої інформації за автором;
- Пошук існуючої інформації за назвою;
- Підтримка пошуку схожого за змістом матеріалу.

Об’єкт дослідження

Об’єкт дослідження – пошукові системи, пошукові двигуни і їх складові.

Предмет дослідження

Предмет дослідження — швидкість роботи пошукових двигунів, причини прискорення роботи з базами даних і можливість використання пошукових двигунів у системі веб-сайту для пошуку великорозмірних даних.

Наукова новизна одержаних результатів

Сучасний споживач не може працювати в інтернеті без використання пошукових двигунів, які надихають власною різноманітністю, тому його увагу все складніше залучити до найкращих пошукових систем. Програмний продукт з використанням пошукового двигуна має багато переваг за рахунок своєї інтерактивності.

Методи дослідження

Для вирішення поставленої проблеми будуть використовуватись наступні методи дослідження:

1. Аналіз аналогів програмного забезпечення та існуючих рішень для проблеми прискорення роботи із базами даних.
2. Порівняння бібліотек для використання у процесі створення програмного забезпечення.
3. Аналіз статей, що будуть зберігатись у базі даних.

4. Аналіз методів пошуку по базі даних, наявних у даному пошуковому двигуні.
5. Синтез отриманих результатів під час дослідження пошукового двигуна заради вирішення проблем роботи з великими обсягами даних.
6. Тестування з використанням різних статей для порівняння швидкості роботи із базою даних.
7. Тестування зі зміною параметрів роботи розробленої системи.

Практичне значення одержаних результатів

Все більше провідних компаній використовують пошукові двигуни, щоб залучити аудиторію споживачів швидкістю роботи із даними. Пошукові двигуни дозволяють поглянути на звичні всім речі по-новому, а також пропонує нові продукти брендів в новому світлі, як би «оживляючи» їх. Це дає можливість взаємодіяти зі споживачами або ж потенційними покупцями зовсім інакше. Інтерактив сам по собі привабливіше, ніж статичні об'єкти, тому реклама або ж інші продукти з використанням технологій доповненої реальності мають всі шанси стати популярними і успішними.

Апробація одержаних результатів

Результати дослідження були представлені на науково-практичній конференції студентів, аспірантів, докторантів і молодих вчених Запорізького національного університету «Молода наука-2021» [1], а також на науково-технічній конференції студентів, аспірантів, магістрантів і викладачів Інженерного навчально-наукового інституту Запорізького національного університету [2].

Глосарій

- *Пошукова система* (або скорочено пошукови́к) — онлайн-служба (апаратно-програмний комплекс з вебінтерфейсом) або програмне забезпечення, що надає можливість пошуку інформації в базі даних або Інтернеті.

- *Фреймворк* (англ. Framework, каркас, платформа, структура, інфраструктура) —інфраструктура програмних рішень, що полегшує розробку складних систем. Спрощено дану інфраструктуру можна вважати своєрідною комплексною бібліотекою, але при цьому вона має ряд обмежень, що задають правила створення структури проекту та написання коду.
- *Бібліотека* (від англ. library) — збірка об'єктів чи підпрограм для вирішення близьких за тематикою задач.
- *Laravel* — безкоштовний, з відкритим кодом PHP-фреймворк, створений Тейлор Отвел (англ. Taylor Otwell) і призначений для розробки веб-додатків відповідно до шаблону model-view-controller (MVC).
- *Фронтенд* (англ. front-end) — клієнтська сторона інтерфейсу користувача до програмно-апаратної частини сервісу.
- *Бекенд* (англ. back-end) — програмно-апаратна частина сервісу, що відповідає за функціонування його внутрішньої частини.
- *Моде́ль–вигляд–контролер* (MVC, Модель–представлення–контролер, англ. Model-view-controller) — архітектурний шаблон, який використовується під час проектування та розробки програмного забезпечення, що передбачає поділ системи на три взаємопов'язані частини: модель даних, вигляд (інтерфейс користувача) та модуль керування.

РОЗДІЛ 1 ДОСЛІДЖЕННЯ ПРОБЛЕМАТИКИ СТВОРЕННЯ ДОДАТКІВ З ВИКОРИСТАННЯМ ПОШУКОВИХ ДВИГУНІВ

1.1 Поширення проблеми використання пошукових двигунів

Підкреслюючи різке збільшення з часом відсоток користувачів Інтернету, які здійснюють пошук у звичайний день, зріс на 69% із січня 2002 р., коли Pew Internet & American Life Project вперше відстежив цю діяльність, до травня 2008 р., коли були зібрані поточні дані. Протягом того ж шестирічного періоду використання електронної пошти в типовий день зросло з 52% до 60% за темпів зростання лише 15%.

Ці нові цифри сприяють подальшому пошуку поза межею, набагато випереджаючи інші популярні дії в Інтернеті, такі як перевірка новин, що робить 39% користувачів Інтернету в типовий день, або перевірка погоди, що робить 30% у звичайний день .

Для більшості людей середній день включає велику кількість електронних листів (60% користувачів Інтернету), загальний пошук (49%) та читання новин (39%), якщо вони взагалі в мережі (30% користувачів Інтернету перебувають поза мережею звичайного дня).

Ті, хто користується пошуковими системами в середньому на день, частіше виявляються соціально висококласними, бо принаймні певна освіта в коледжі та доходи перевищують 50 000 доларів на рік. Вони, швидше за все, будуть користувачами Інтернету, що мають щонайменше шість років досвіду роботи в Інтернеті, і їх будинки будуть підключені до мережі для швидкого з’єднання з Інтернетом. Молодші користувачі Інтернету частіше, ніж користувачі старшого віку, здійснюють пошук звичайного дня. Чоловіки частіше за жінок шукають у звичайний день. Ось цифри:

Освіта: Користувачі Інтернету з вищим рівнем освіти частіше користуються пошуком у звичайний день, причому ті, хто має хоч якусь коледжну

освіту, значно частіше це роблять, ніж користувачі з меншою освітою. Ось відсотки:

- Випускник коледжу +: 66%.
- Деякі коледжі: 49%.
- Випускник середньої школи або менше: 32%.

Дохід: Інтернет-користувачі, які живуть у домогосподарствах з високим рівнем доходу, частіше користуються пошуком у звичайний день, причому ті, у кого дохід перевищує 50 000 доларів на рік, мають значно більшу ймовірність, ніж користувачі з меншими доходами. Ось відсотки: 2

- 75 000 дол. США +: 62%.
- 50 000 - 74 999 доларів США: 56%.
- 30 000 - 49 999 доларів США: 34%.
- <30000 $: 36%.

Широкосмугове використання: ті, хто використовує широкосмугові зв’язки вдома, набагато частіше, ніж ті, хто використовує комутований доступ, взагалі коли-небудь пробували користуватися пошуковими системами, на 94% - 80%. Вони значно частіше шукають у звичайний день, і ця різниця зберігається, коли інші фактори, такі як вік та освіта, залишаються незмінними. Це відсотки відповідно до типу підключення до Інтернету для тих, хто шукає в типовий день:

- Широкосмугове підключення вдома: 58%.
- Модемний зв'язок вдома: 26%.

Вік: Молодші користувачі Інтернету постійно частіше здійснювали пошук звичайного дня протягом останніх п’яти років досліджень. Ось відсоток шукачів у різних вікових групах, які здійснюють пошук у типовий день:

- 18-29 років: 55%;
- 30-49 років: 54%;
- 50 - 64 роки: 40%;
- 65 років і старше: 27%.

Стать: Хоча приблизно однакова кількість чоловіків (91%) і жінок (88%) повідомляє, що взагалі коли-небудь користувалась пошуковими системами, чоловіки значно частіше, ніж жінки, здійснюють пошук звичайного дня.

- Чоловіки: 53%.
- Жінки: 45%.

Дані, зібрані з 2002 року, показують, що чоловіки, які користуються Інтернетом, частіше, ніж жінки, частіше інтегрують пошук у своє повсякденне життя. Відсоток чоловіків, які здійснюють пошук в звичайний день, постійно зростав з 33% у 2002 році до 53% в даний час. Відсоток жінок також зріс, збільшившись з 25% у 2002 році до 45%.

Дані минулих опитувань також свідчать про те, що чоловіки загалом більше займалися пошуком.3 Чоловіки в мережі кажуть, що шукали частіше; вони висловили більшу впевненість у своїх пошукових можливостях (хоча жінки повідомляють, що вони однаково успішно отримують задовольняючі результати пошуку). Чоловіки також більше, ніж жінки, були обізнані з деякими суперечливими проблемами, пов’язаними з пошуком, наприклад, про існування платних проти неоплачених результатів пошуку та різницю між ними.

Хоча кількість користувачів Інтернету, які здійснюють пошук в типовий день, неухильно зростає, це вже другий раз з тих пір, як Інтернет-проект Pew почав відстежувати використання пошукової системи, і ми спостерігаємо очевидний стрибок у кількості. Перший відбувся наприкінці 2005 р., Коли відсоток користувачів, які здійснювали пошук в типовий день, зріс з приблизно 30% (у червні 2004 р.) До приблизно 40% (у вересні 2005 р.). Тоді ми спекулювали про кілька можливих причин зростання, вказуючи на те, що це був час значного висвітлення у ЗМІ та галасу про компанії пошукових систем, включаючи IPO Google.

Зараз відсоток користувачів, які здійснюють пошук у звичайний день, знову зріс, приблизно з 40% до 49%. Що змінилося у світі пошуку, що могло б спричинити це збільшення?

Однією з ймовірних причин є те, що користувачі тепер можуть розраховувати на високоефективну пошукову систему, специфічну для сайту, майже на кожному багатому вмістом веб-сайті, який вартий свого та бажає конкурувати із іншими веб-сайтами у власній галузі. Зі зростанням маси веб-вмісту з блогів, новинних сайтів, архівів зображень та відео, персональних веб-сайтів тощо, користувачі Інтернету мають можливість звертатися не лише до основних пошукових систем, а й до пошукових систем на окремих сайтах як транспортних засобів. щоб отримати інформацію, яку вони шукають.

Інша причина може бути пов'язана з тим, що повністю 55% американських будинків мають високошвидкісний Інтернет. З усіх аналізованих нами демографічних змінних присутність домашнього широкосмугового зв'язку найсильніше пов'язана зі схильністю користувача до використання пошукових двигунів у звичайний день. Попередні дослідження показали, що коли користувач Інтернету переходить на домашню широкосмугову мережу, він чи вона частіше звертається до мережі у будь-який момент, наприклад коли у нього або неї виникає якесь питання – і тепер користувач буде все частіше відвідувати пошуковий двигун заради того, щоб знайти відповідь.

Нарешті, може бути так, що загальні сайти пошукових систем стали настільки корисними та добре налаштованими, що люди звертаються до них з метою знайти відповідь на все більш широке коло питань.

1.2 Проблеми аналізу великих даних

Завдяки технологічному прогресу Big Data стали доступними в різних науково-технічних галузях, включаючи соціальні науки та керування. На основі Big Data можна провести дослідження та аналіз для виявлення людських намірів, почуттів та думок, що дозволить нам ідентифікувати не лише окремих людей, а й наміри спільнот та суспільства в цілому. Однак аналіз великих даних відрізняється від традиційних методів узагальнення.

Одним з основних питань, що виникають при аналізі великих даних, є розрив між об'єктом спостереження та об'єктом аналізу. Наприклад, якщо об’єктами спостереження є облікові записи користувачів, не завжди очевидно, кого представляє кожен обліковий запис. Також обліковим записом можуть користуватися члени сім'ї, друзі чи сторонні особи. При аналізі великих даних зазвичай робляться деякі припущення щодо природи об'єкта спостереження, які часто порушуються на практиці. Verstrepen та Goethals розглядають проблеми системи рекомендацій, що застосовується до спільних облікових записів користувачів. Іншою проблемою, яка призводить до різниці між об'єктом спостереження та об'єктом аналізу, є компроміс між інформацією та конфіденційністю. Оскільки конфіденційність обмежує доступ до даних на індивідуальному рівні, аналіз проводиться на основі непрямих даних.

Іншим важливим питанням при аналізі великих даних є можливий хибний висновок про те, що об'єктом спостереження можна вважати середнього представника ширшої сукупності, ніж фактично охоплену вибірку. Наприклад, ви можете отримати хибний висновок, якщо аналізуєте дані з Інтернет-джерел у країнах, де менше половини населення має доступ до Інтернету.

Важливим питанням, яке виникає під час збору загальнодоступних даних з веб-сайтів компаній або агрегаторів, є узагальнення. Не завжди є інформація про те, які групи людей представляють наявні Великі дані, як вони були відібрані, чи були вони попередньо оброблені тощо. Такі дані також з меншою ймовірністю включають детальну демографічну інформацію з міркувань конфіденційності. Отже, висновок від аналізу великих даних до ширших груп, ніж ті, з яких ці дані були отримані, є невизначеним. Більше того, сама можливість такого висновку ставиться під сумнів. Наприклад, хоча Twitter є популярним джерелом великих даних серед дослідників, навіть якщо ми можемо розглядати дані Twitter як випадкову вибірку більшого набору твітів, користувачі Twitter відрізняються від середнього представника населення в цілому — вони схильні бути молодшими та мати спеціаліста або вищу освіту.

Узагальнення також є серйозною проблемою, коли великі дані отримуються за допомогою веб-скрапінгу. Хоча вишкрібання забезпечує більший контроль над процесом збору, існує багато невідомих про взаємозв'язок між інформацією, доступною на веб-сайті, та інформацією, яку власник веб-сайту не надає. Крім того, проблеми із сервером, завантаження мережі, політика оновлення веб-сайтів, поганий дизайн веб-сторінок та невипадковий характер результатів пошуку - це також лише деякі фактори, що призводять до помилок вибірки під час збору великих даних.

Після визначення джерела великих даних необхідно вказати на ряд проблем, пов'язаних безпосередньо з їх аналізом. Найпоширенішими методами статистичного аналізу для визначення та кількісної оцінки причинно-наслідкового зв'язку за експериментальними даними є аналіз дисперсійних та регресійних моделей. Моделі регресії також надзвичайно популярні в спостережних дослідженнях, які перевіряють причинно-наслідкові гіпотези. Тим не менш, підходи до поведінкових висновків, які ефективні в невеликих вибірках, стикаються з проблемами при аналізі великих даних. Перевага Великих даних полягає в їх багатстві з точки зору різноманітності: вони, швидше за все, містять інформацію про рідкісні меншини, ніж малі зразки або дані менших розмірів. Однак, застосовуючи ці статистичні моделі до великих даних, рідкісні меншини або відфільтровуються (наприклад, їх вважають викидами), або їх вплив нівелюється шляхом усереднення з більшістю. Наприклад, для дуже великих вибірок вплив малих меншин та викидів на коефіцієнти регресії та статистичні тести дуже малі [3].

Іншою проблемою, пов’язаною з агрегацією та неоднорідністю, є парадокс Сімпсона: явище в статистиці, коли за наявності двох груп даних, у кожній з яких існує однаково спрямована залежність, напрямок залежності змінюється, коли групи об’єднуються . Важливо визначити, чи проявляється парадокс Сімпсона у наборі даних, що використовується для прийняття рішень. Враховуючи розмір і різноманітність великих даних, ймовірність парадоксу

Сімпсона виникає, коли аналіз значно вищий, ніж при роботі з малими вибірками. Шмуелі та Яхав запровадили метод, який використовує класифікаційні та регресійні дерева для автоматизованого виявлення потенційних парадоксів Сімпсона в наборах даних з малою чи багатьма потенційно незрозумілими змінними, яка масштабується до великих вибірок. Якщо дослідник зацікавлений в аналізі поведінки підгруп чи осіб на додаток до середньої оцінки, то підходи прогнозного моделювання та валідації можуть бути корисними [4-5]. Наприклад, причинно-наслідкова статистична модель може бути використана для формування прогнозів для обмеженого набору спостережень. Тоді прогнози та їх помилки можна порівняти в різних підгрупах або відсортувати, щоб визначити підгрупи, для яких ефекти суттєво відрізняються від середньої більшості. Наведені приклади використання прогностичного тестування для вдосконалення причинно-наслідкових досліджень. Таким чином, дослідники повинні розуміти, що класичне статистичне причинно-наслідкове моделювання, засноване на експериментальних або спостережних даних, спрямоване на виявлення загального причинного ефекту в аналізованій вибірці. Тому необхідно з обережністю інтерпретувати результати, отримані за допомогою цих методів, щоб не помилитися висновками, отриманими для вибірки в цілому, як застосовними на індивідуальному рівні.

Ще однією проблемою, що виникає при аналізі великих даних, є корисність статистичної значущості та висновку через великі вибірки та багаторазове тестування. Зокрема, у дуже великих зразках навіть незначні ефекти є статистично значущими, і тому перевірка гіпотез із використанням Р-значення складна. Альтернативи використанню значень Р включають такі методи, як використання оцінки замість тестування або прийняття байєсівського підходу.

Проблема багаторазового тестування викладена у роботах Агарваля та Чена, присвячених розробці алгоритмів обчислювальної реклами та рекомендацій щодо вмісту. Він базується на багатовимірному характері результатів у різноманітному контексті з різними цілями. Така множинність, використову-

ючи статистичний висновок, призводить до перевірки багатьох гіпотез, які ризикують помилковими висновками. Наприклад, при тестуванні кількох незалежних гіпотез, якщо кожна гіпотеза перевіряється на заданому рівні значущості, то ймовірність помилкового висновку принаймні на одному з тестів зростає експоненціально щодо їх кількості.

Щодо аналізу великих даних, які, як правило, неоднорідні та мають велику розмірність, вищезазначені проблеми змушують нас шукати компроміс між ефектами після тестування для різних підгруп для виявлення різнорідних ефектів (наприклад, за статтю, віком, місцем місце проживання та інші змінні та їх поєднання) та ризик неправдивих висновків через багаторазове тестування. Беручи до уваги описані проблеми, можна зробити висновок, що для здійснення перевірки осіб на основі аналізу великих даних щодо їх взаємодії з різними інформаційними ресурсами першочерговим є завдання оптимальної сегментації (кластеризації) аналізованих даних [6].

1.3 Дослідження проблематики

Процедура кластеризації спрямована на розподіл даних на групи подібних об'єктів відповідно до критерію максимізації подібності між об'єктами в одній групі та мінімізації подібності між об'єктами в різних групах. З постійним збільшенням обсягу даних традиційні методи кластеризації досягли своїх меж, що призвело до розробки методів паралельної кластеризації.

Найвідоміша модель обробки великих даних - MapReduce. Ця модель була запропонована Діном та Гемаватом у Google, де вона була успішно використана для різних цілей. Сильні сторони цієї моделі пов’язані з тим, що вона допускає автоматичний паралелізм та розподіл. На додаток до відмовостійкого механізму, який допомагає подолати несправності, він також надає інструменти для управління станом, моніторингу та балансування навантаження. Оптимізація розподілу даних забезпечується шляхом їх зберігання на локальних дисках, щоб уникнути надмірного споживання пропускної здатності мережі.

Кластерний аналіз за допомогою MapReduce складається з двох етапів: «Карта» та «Зменшення». На етапі «Карта» дані фільтруються та сортуються, тоді як на етапі «Зменшити» підсумовуються результати попереднього кроку. Функція Map бере записи із вхідних файлів як пари ключ-значення та створює проміжні пари key/value. Функція зменшення працює зі значеннями певного проміжного ключа і виробляє одне остаточне значення для того самого ключа.

Існує кілька програмних реалізацій моделі MapReduce. Найпопулярніший фреймворк — Hadoop, реалізований на мові Java. Цей проект, розроблений Apache Software Foundation, включає набір модулів з відкритим кодом, що забезпечує надійні та масштабовані розподілені обчислення. Завдяки таким характеристикам Hadoop, як організована архітектура, масштабованість, економічність, гнучкість та стійкість, фреймворк Hadoop MapReduce є найбільш переважною платформою для вирішення даної проблеми. Деякі найпоширеніші алгоритми кластеризації на основі MapReduce:

PKMeans — це реалізація алгоритму k-означає на основі MapReduce. Він розроблений з одним завданням MapReduce, в якому функція Map відповідає за віднесення кожного зразка до найближчого центру, а функція Reduce — за оновлення нових центрів.

MR-DBSCAN - це реалізація відомого алгоритму DBSCAN на основі MapReduce. Його паралельний метод складається з чотирьох кроків. На першому кроці узагальнюється розмір та загальний просторовий розподіл усіх записів, а потім для наступного кроку формується перелік розмірних індексів, що вказують приблизний розподіл сітки. На другому кроці виконується основний процес DBSCAN для кожного підпростору, розділеного на профіль розділу. Третій крок вирішує транскордонні проблеми при об’єднанні підпросторів. В кінці створюється відображення ідентифікатора кластера, від локальних кластерів до глобального, для всього набору даних на основі списків пар, зібраних з попереднього кроку. Нарешті, локальні ідентифікатори змінюються на глобальні для точок з усіх розділів, щоб отримати єдиний результат.

DBCURE-MR — паралельна версія нового алгоритму кластеризації на основі щільності, який називається DBCURE, який реалізований за допомогою моделі програмування MapReduce. DBCURE діє подібно до DBSCAN, повторюючи два кроки. На першому кроці в наборі даних вибирається невидима точка, яка вважається насінням і вставляється в набір насіння. На другому етапі отримуються всі точки, до яких можна отримати щільність із насіння. Цей процес створює кластери по одному і зупиняється, коли насіннєвий набір стає порожнім, на відміну від своєї паралельної версії, яка знаходить кілька кластерів одночасно, обробляючи кожну основну точку паралельно через чотири етапи. Перший крок відповідає за оцінку матриць коваріації сусідства, і він виконується за допомогою двох алгоритмів MapReduce. Другий крок виконує обчислення еліпсоїдних τ-сусідств і виконується за допомогою двох інших алгоритмів MapReduce. Третій крок виявляє основні кластери, що робиться за допомогою одного алгоритму MapReduce. Нарешті, останній крок відповідає за об’єднання основних кластерів, і він виконується за допомогою одного алгоритму MapReduce.

PMR-Transitive — це нова паралельна евристика, заснована на моделі програмування MapReduce нещодавно з’явився методу, а саме, Transitive heuristic. У цій евристиці кластери отримують шляхом розподілу категоріальних великих наборів даних відповідно до підходу реляційного аналізу. Підхід реляційного аналізу забезпечує математичний формалізм, де проблема кластеризації має форму лінійної програми з n2 цілочисельними атрибутами (де n - кількість екземплярів). Евристика є найбільш зручним рішенням для отримання задовільних результатів кластеризації в найкоротші терміни, особливо в контексті великих даних, де кількість екземплярів велика, а час відгуку є критичним фактором. Оскільки оригінальна евристика є послідовною, її потрібно пристосувати до моделі MapReduce. У цій роботі представлений детальний опис нового дизайну, заснованого на ключових методах моделі

MapReduce, а саме Map and Reduce. І переважно, що більшість етапів, які спричиняють великі обчислювальні витрати, пов’язані з транзитивною евристикою, можуть бути оброблені паралельно [7].

Завдання перевірки користувача вирішується у два етапи. Сегментація (кластеризація) дозволяє групувати непрямі дані про поведінку несанкціонованих користувачів у мережі таким чином, що кожна група (сегмент або кластер) відповідає конкретній особі. Після цього дані в цих сегментах можна шукати, отримувати на вимогу, аналізувати та візуалізувати. Для цих завдань використовуються спеціальні інструменти; Найпоширеніші з них такі.

Sphinx — це повнотекстовий пошуковий механізм з відмінною рисою високої індексації та швидкості пошуку, а також інтеграції з існуючими системами управління базами даних (MySQL, PostgreSQL) та API для загальних мов веб-програмування (офіційно підтримує PHP, Python, Java; це реалізовані спільнотою API для Perl, Ruby, .NET та C ++). Підтримує розширені можливості пошуку, включаючи ранжирування та вивчення російської та англійської мов, розподілену підтримку пошуку та кластеризації. Для великих обсягів даних схема індексу Delta може бути використана для прискорення індексації. Крім того, Sphinx підтримує індекси реального часу, фільтрування та сортування результатів пошуку та пошук умов підстановки.

Apache Solr — це розширювана пошукова система для повнотекстового пошуку з відкритим кодом, заснована на проекті Apache Lucene. Його особливістю є те, що це не просто технічне рішення для пошуку, а платформа, яку можна легко розширити, змінити та налаштувати під різні потреби - від звичайного повнотекстового пошуку на веб-сайті до розподіленої системи для зберігання, прийому та аналізу текстові та інші дані з потужною мовою запитів. На відміну від Sphinx, документи зберігаються повністю, і їх не потрібно дублювати в базі даних. Основні особливості Solr — повнотекстовий пошук, виділення результатів, фасетний пошук, динамічна кластеризація, інтеграція з базами даних, обробка документів у складному форматі (наприклад, Word, PDF). Оскільки Solr має можливість розподіленого пошуку та реплікації, він

дуже масштабований. Xapian - це бібліотека пошукової системи. Пакети доступні для Ubuntu та Red Hat, можуть бути скомпільовані для OSX, а також можуть працювати під управлінням Windows через CygWin. Xapian є менш поширеним і гнучким, ніж згадані вище пошукові системи. Він не має морфології, але існує для низки мов (включаючи російську та українську). Інші реалізовані функції включають перевірку орфографії в пошукових запитах, інкрементний індекс, оновлюваний паралельно пошуку, оперування кількома індексами та індекси в пам'яті для невеликих баз даних.

Elasticsearch спочатку розроблявся як система для повнотекстового пошуку у великих обсягах неструктурованих даних. В даний час Elasticsearch — це повноцінна аналітична система з різними можливостями. Дані в Elasticsearch зберігаються у перевернутому форматі індексу на основі Apache Lucene. Apache Lucene — найвідоміша пошукова система, спочатку орієнтована спеціально на вбудовування в інші програми. Lucene — це бібліотека для високошвидкісного повнотекстового пошуку, написана на Java. Він забезпечує розширені можливості пошуку, хорошу систему побудови та зберігання індексів, яка може одночасно додавати, видаляти документи та виконувати оптимізацію разом з пошуком, а також паралельний пошук за набором індексів, що поєднують результати. Недоліком є порівняно низька швидкість індексації (особливо в порівнянні зі Sphinx), а також відсутність API (про що дбає Elasticsearch).

Elasticsearch дозволяє розподіляти дані між кількома машинами, що дозволяє підтримувати операції з високою продуктивністю. Частини, між якими поділяються дані, називаються шардами. Шарди бувають двох типів - майстер і репліки. Майстер дозволяє як операції читання, так і запису, тоді як репліка є лише для читання і є точною копією майстра. Така структура забезпечує стабільність системи, оскільки у випадку відмови майстра репліка стає ведучою. Оскільки репліки — це точні копії основного файлу, різні запити можуть одночасно оброблятися як із головного, так і з репліки. Таким чином, запити клієнтів на індекс виконуються паралельно на всіх шардах, після чого результати

кожного осколка збираються і відправляються назад клієнту. Це значно підвищує продуктивність системи [8-10].

Існує багато інших бібліотек для повнотекстового пошуку, таких як MySQL fulltext, PostgreSQL Textsearch, CLucene, Lucene ++ та інші, але більшість з них застосовні лише в системах з певною базою даних або мовою програмування і не підходять для загального рішення завдання, про яке йдеться.

Sphinx забезпечує дуже швидкий пошук та індексацію, але повільно оновлюється через те, що не існує механізму автоматичного оновлення індексу. Істотним недоліком є те, що він працює лише з MySql та Postgres. Він не підходить для вирішення розглянутої задачі, оскільки не може оновити або видалити документи в індексі (працює лише доповнення).

Apache Solr забезпечує дуже високу швидкість індексації та пошуку, його розмір індексу є одним із найменших та має високу розширюваність. Він також може виконувати роль сховища. Solr включає безліч додаткових функцій, таких як неточний пошук та можливість масштабування з коробки. Недоліком є те, що це Java-сервер у контейнері сервлетів, реалізований як веб-служба з інтерфейсами XML / JSON / CSV. Elasticsearch (на основі Apache Lucene) має дещо нижчу швидкість індексації та пошуку в порівнянні зі Sphinx, але він пропонує не тільки пошук та зберігання, але також містить інші інструменти (візуалізація, колектор журналів, система шифрування тощо). Він здатний масштабувати та забезпечує вибірку дуже складних форм, що робить його хорошим вибором для аналітичної платформи. Цей двигун не найпростіший у використанні, але він містить масу додаткових функцій. Велика перевага полягає в тому, що цей механізм використовує дуже мало пам'яті, а інкрементальне індексування відбувається так само швидко, як індексування кількох документів одночасно.

Xapian забезпечує відносно швидкий пошук, але значно повільніше індексування. Він також має дуже великий розмір індексу. Як перевагу можна також виділити, що він має безліч інтерфейсів для різних мов (C++, Java, Perl, Python, PHP, C#, Ruby, Lua). Тим не менше, для розглянутого завдання Xapian

абсолютно недоречний через велику кількість даних та часте індексування. Таким чином, пошукова машина та повнотекстова система пошуку Elasticsearch найкраще підходить для завдання пошуку та візуалізації у великих наборах кластерних даних, що відповідає взаємодії користувачів з різними інформаційними ресурсами.

1.4 Система повнотекстового пошуку

Центральною складовою дослідження пошукових систем є Elasticsearch. Elasticsearch — це пошукова система, заснована на бібліотеці Lucene. Він забезпечує розподілену повнотекстову пошукову систему з підтримкою кількох арендаторів, веб-інтерфейсом HTTP та документами JSON без схем. Elasticsearch розроблений на Java і має подвійну ліцензію відповідно до загальнодоступної публічної ліцензії на стороні сервера та ліцензії Elastic, тоді як інші частини підпадають під власну (доступну для джерела) ліцензію Elastic. Офіційні клієнти доступні на Java, .NET (C #), PHP, Python, Apache Groovy, Ruby та багатьох інших мовах. Згідно з рейтингом DB-Engines, Elasticsearch є найпопулярнішою пошуковою системою для підприємств.

Що саме використовує Elasticsearch:

- Повнотекстовий пошук — Elasticsearch відомий своїми потужними можливостями повнотекстового пошуку. Його швидкість походить від перевернутого індексу в основі, а потужність - від налаштованого оцінювання релевантності, розширеного DSL-запиту та широкого спектру функцій, що покращують пошук.
- Перевернутий індекс — Elasticsearch використовує структуру, звану інвертованим індексом, яка призначена для дуже швидкого повнотекстового пошуку. Інвертований індекс складається зі списку всіх унікальних слів, які містяться в будь-якому документі, і для кожного слова, списку документів, у яких він з’являється. Щоб створити інвертований індекс, ми спочатку розділяємо поле вмісту кожного документа на окремі слова

(які ми називаємо термінами або лексемами), створюємо відсортований список усіх унікальних термінів, а потім перелічуємо, в якому документі з’являється кожен термін.

- Поле виконання — це поле, яке обчислюється під час запиту (схема при читанні). Поля середовища виконання можуть бути введені або змінені в будь-який час, у тому числі після індексації документів, і можуть бути визначені як частина запиту. Поля середовища виконання піддаються запитам з тим самим інтерфейсом, що і індексовані поля, тому поле може бути полем часу виконання в деяких індексах потоку даних та індексованим полем в інших індексах цього потоку даних, і запити не повинні цього знати. Хоча індексовані поля забезпечують оптимальну продуктивність запитів, поля середовища виконання доповнюють їх, вводячи гнучкість для зміни структури даних після індексації документів.

Запропонована система повнотекстового пошуку та візуалізації має такий склад:

- Програмний модуль, відповідальний за отримання зібраних даних, інтеграцію сторонніх послуг та клієнтів та аналіз отриманих даних.
- Програмний модуль, відповідальний за структурування отриманих даних, подальшу обробку даних та підготовку до завантаження або візуального відображення.
- Програмний модуль, відповідальний за зберігання отриманої інформації в базі даних.
- Програмне забезпечення, відповідальне за резервне копіювання даних, яке є частиною інструменту зберігання даних - нереляційна база даних Elasticsearch. Усі зібрані дані зберігаються у відмовостійкому кластері, де зберігання та доступність даних забезпечуються вбудованими механізмами Elasticsearch.

Система Elasticsearch також забезпечує масштабування даних. Коли з'являється новий сервер баз даних Elasticsearch, його внутрішні механізми забезпечують організацію зберігання даних на ньому автоматично.

- Програмний модуль, відповідальний за передачу даних, зібраних від інтегрованих сторонніх служб та клієнтів. Шифрування переданих даних здійснюється за допомогою стандартної технології безпеки Інтернету SSL. Шифрування на стороні користувача виконується браузером, а дані передаються за протоколом https. Шифрування даних на системній стороні виконується за допомогою веб-сервера системи.
- Допоміжні програмні модулі — модуль журналу транзакцій, модуль захисту даних та балансування навантаження. Журнал транзакцій ведеться за допомогою системного журналу, а також ведення журналу операцій, що виконуються в кожному модулі за допомогою самого модуля. Балансування навантаження даних здійснюється за допомогою бази даних Elasticsearch. Балансування потоку вхідних запитів здійснюється за рахунок збільшення кількості серверів у інтерфейсному рівні (кількості використовуваних веб-серверів). Кожен веб-сервер має власну ip-адресу. Щоразу, коли веб-сайт отримує доступ за його назвою, він призначається наступному серверу інтерфейсного рівня відповідно до порядку IP-адрес. Таким чином виконується балансування навантаження, а також автоматично підтримується працездатність інтерфейсного серверного рівня на випадок, якщо один із них зазнає несправності.

Повнотекстова система пошуку та візуалізації розроблена з використанням моделі mvc (моделі, контролери та подання):

- Клас моделі описує доступ до даних, необхідних для роботи сторінок веб-сайту програми.
- Клас контролера описує логіку управління сторінками веб-сайту програми.
- Клас перегляду описує інтерфейс користувача.

Головною перевагою використання моделі mvc є свобода поєднання її компонентів. Кожну частину програми можна змінювати незалежно від інших програмних модулів.

Вхідні дані для повнотекстової системи пошуку та візуалізації включають:

- Дані, отримані з бази даних Elasticsearch — у форматі JSON.
- Дані з бази даних служби з інформацією про користувачів та робочі місця - у форматі SQL.

Вихідні дані системи включають:

- Дані для візуального відображення — у форматі HTML.
- Дані для друку — у форматі csv.

Дані зберігаються в нереляційній базі даних Elasticsearch, основною метою якої є швидкий пошук у великих наборах даних.

Внутрішнє представлення даних Elasticsearch, що використовуються системою, має таку ієрархічну структуру:

- Об’єктний індекс.
- Тип документа.
- Набір значень даних.

Дані служби, пов'язані з управлінням користувачами та робочими станціями користувачів, зберігаються у допоміжній базі даних MySQL. Апаратне забезпечення запропонованої повнотекстової системи пошуку та візуалізації включає набір серверів (інтерфейсний та внутрішній рівні серверів), робочі станції адміністраторів та розробників системи та периферійне обладнання, включаючи зовнішні накопичувачі та архіви.

Фронтальні сервери отримують запити від користувачів. Отримавши запит, вони подають відповідний запит на обробку на фоновий рівень.

Вся статична інформація, яка використовується для відображення даних веб-сайту, зберігається та доставляється веб-сервером. Така статична інформація включає зображення, медіа-файли тощо. Усі ці дані обробляються файловою системою ОС. Доставка цих статичних даних на сервери здійснюється під час розгортання або оновлення системного програмного забезпечення. Це гарантує, що всі статичні дані ідентичні на всіх серверах на зовнішньому рівні.

Очікувана ефективність обробки даних системою повнотекстового пошуку та візуалізації складає щонайменше 10000 запитів на секунду загалом із часом затримки відповіді менше 1 секунди [11-14].

1.5 Висновки з розділу 1

Проблема швидкості інтернету досить популярна і буде популярна, бо від цього залежить якість користування веб-сайтом. Швидкість доступу до інтернету здебільшого залежить від швидкості роботи з базами даних веб-сайтів, а швидкість роботи з базами даних залежить від обсягу самої бази, об’єму одиниці даних та швидкості пошуку по базі даних.

Досить важливо мати уявлення про велику різноманітність альтернатив до пошукового двигуна Elasticsearch заради розуміння причин чому він є найпопулярнішим на даний момент і як можна покращити і далі цей пошуковий двигун.

Розуміння переваг пошукових двигунів допоможе у розробці більш якісних баз даних заради поліпшення роботи із веб-сайтом або будь-яким іншим програмним забезпеченням.

РОЗДІЛ 2 ТЕОРЕТИЧНІ ВІДОМОСТІ РОБОТИ ПОШУКОВИХ ДВИГУНІВ

2.1 Вступ до аналізу пошукових систем

Всесвітня павутина сьогодні є інформаційним носієм, який швидко зростає. Великі загальні пошукові системи є основними інструментами пошуку ресурсів. Послуги каталогів, пошукові системи, списки та "чутки" (вивчення URL-адрес від інших людей чи інших типів засобів масової інформації) охоплюють лише незначну частину Мережі. Раніше було показано, що загальні пошукові системи не є ідеальними. Їх охоплення низьке. Останнє дослідження показує, що спільні зусилля найбільших пошукових систем дають лише 16% охоплення індексованої Мережі. Перекриття показників пошукових систем напрочуд невелике. Ще одним показником, який вивчався досить широко, є точність роботи пошукових систем . У цих дослідженнях вивчали лише перші десять або двадцять звернень для кожної пошукової системи, оскільки користувачі рідко виходять за межі перших двадцяти результатів. Оскільки точність є суб’єктивною мірою, були зроблені різні висновки. Тематичне дослідження, що вивчає весь набір результатів (6681 URL-адрес, отриманих шістьма пошуковими системами), з використанням технічного визначення точності (перевірка того, чи в документі містяться пошукові терміни), показало, що точність пошукових систем була цілком задовільною (BarIlan, 1998). Однією з причин пошуку неточних документів та низького охоплення пошукових систем є динамічний характер самої мережі. Документи зникають, зазнають змін, і, отже, індекс пошукової системи більше не відображає зміст документа. Нові відповідні веб-сторінки постійно додаються, і "чекають" ще деякий час, щоб їх виявили пошукові системи.

Ми взяли за мету дослідити зміни, що відбуваються в Інтернет-літературі, що містить пошукові терміни «інформація» та «інформативна». Пошуки проводились з інтервалом в один місяць протягом п’яти місяців між січнем і

червнем 1998 р. Запит «інформація OR інформативна» був представлений шести пошуковим системам (в алфавітному порядку); AltaVista, Excite, Hotbot, Infoseek, Lycos та Northern Light. Всі документи, на які вказували пошукові системи, отримувались та зберігались локально кожного разу, коли проводився пошук. Документи були ретельно вивчені та охарактеризовано зміни, що відбулися в URL-адресах, що з'являються в кількох раундах пошуку (Bar-Ilan & Peritz, 1999). У червні 1999 року було проведено додатковий обшук. Обшуки проводились один раз на місяць (у пошукових турах) протягом п'яти місяців. Під час вищезазначеного дослідження ми з нашим великим подивом зрозуміли, що результати пошукових систем не стабільні, URL-адреси, які пошукова система отримувала в даному раунді, не отримували їх у наступному раунді, навіть якщо інші пошукові системи отримували його , а зміст цих сторінок продовжував бути актуальним для нашого пошуку. Вивчаючи ступінь вищезазначеного явища, ми зрозуміли, що ми маємо справу не лише з кількома сторінками (як пропонується AltaVista у своїх таблицях довідки (AltaVista, 1998)): Excite впав за один раз 72,3% (!) URL-адрес, продовжували існувати та бути актуальними, Lycos 34,5%, AltaVista 29,3%, Hotbot 25,9%, Infoseek 22,7% та Northern Light 5,5% URL-адрес. AltaVista та Hotbot (Hotbot, 1998) - єдині пошукові системи, які натякають у своїх таблицях довідки про те, що деякі URL-адреси можуть зникнути з їх баз даних. Excite навіть не згадує про проблему.

Можливим поясненням зникаючих URL-адрес може бути той факт, що фізичний документ може мати кілька різних адрес (наприклад: searchenginewatch.com та www.searchenginewatch.com) (Notess, 1997), про що пошукові системи знають і відображають у результат встановлює лише одну з URL-адрес, кожен раз іншу. Оскільки у нас не було можливості вирішити, чи це так, ми подали наші "забуті" URL-адреси на надзвичайно суворий тест: для кожної пошукової системи ми порівняли вміст кожної "забутої" URL-адреси із вмістом усіх інших URL-адрес, які були знайдені пошуковою системою в тій

формі, URL-адреса була втрачена Набір URL-адрес, для яких не знайдено збігів, - це набір, де відбулася реальна втрата інформації: механізм не знайшов жодної URL-адреси, яка б містила точно таку ж інформацію, що і "забуті" URL-адреси. Навіть за цим м’яким визначенням (ми поблажливо ставимося до пошукових систем) результати дивують: 57,9% із охоплених URL-адрес для Excite, 23,8% для AltaVista, 21,8% для Hotbot, 20,2% для Lycos, 12,1% для Infoseek та лише 1,6% для Північного світла належить до цих наборів. Наскільки нам відомо, ця проблема раніше не формулювалась і не вивчалась у цій обстановці. Коливання та зміни кількості результатів пошуку за заданими запитами з часом досліджувались раніше (наприклад, Петерсон, 1997; Notess, 1999; Aguillo, 1999; Rousseau, 1999). Однак ці дослідження враховували лише кількість результатів пошуку, а не самі результати, як це було зроблено в поточному дослідженні. Ця різниця в методології виявляється досить важливою для вивчення стабільності пошукової системи. Наприклад, пошукова система Excite повідомляла про більш-менш однакову кількість результатів пошуку в кожному раунді. Однак він отримував зовсім інший набір URL-адрес кожного разу, коли здійснювався пошук.

Користувачі очікують, що пошукові системи будуть надійними та зможуть знайти URL-адресу, яку пошукова система отримала також у майбутньому, якщо URL-адреса продовжує містити відповідну інформацію за темою пошуку. Мета даної статті — попередити користувачів Інтернету про проблеми, що виникають через непослідовну поведінку пошукових систем з часом. Ми вирішили висловити свою думку, виконавши тематичне дослідження, яке дозволило нам ретельно вивчити кожен отриманий документ.

2.2 Методологія

Використовувались пошукові терміни "інформаторія або інформативна". Обшуки проводились шість разів протягом п'яти місяців між 3 січня 1998 року та 7 червня 1998 року. Обшук проводився у першу неділю кожного місяця

(січень, лютий, березень, квітень, травень та червень). Кожної з цих дат (так званих раундів пошуку) ми подавали запит до шести основних пошукових систем (Салліван-1) в Інтернеті (в алфавітному порядку).

Додатковий раунд, який отримав назву порівняльного раунду, відбувся роком пізніше, 20 червня 1999 р. На цей час також були переглянуті всі URL-адреси, визначені в початкових раундах. Спочатку було збережено весь список результатів кожного двигуна. Далі URL-адреси та заголовки були відфільтровані зі сторінок, повернутих пошуковими системами за допомогою програми Visual Basic. Результатом цього процесу стала таблиця зі стовпцями для URL-адреси та заголовка. Ці таблиці були завантажені в Microsoft Excel. Ми запустили модуль Visual Basic в Excel, щоб створити список унікальних URL-адрес, що повертаються пошуковими системами в даному раунді пошуку (порівняння URL-адрес було чутливим до регістру). На останньому етапі кожного раунду пошуку було отримано кожне з цих посилань і документи збережено на нашому локальному жорсткому диску. Цей етап здійснювався "грубою силою", тобто шляхом паралельного збереження документів на декількох комп'ютерах. Для кожного раунду пошуку весь процес збору даних займав близько десяти годин. Аналіз результатів проводився шляхом побудови частотних та перехресних таблиць, використання інструмента фільтрації Microsoft Excel. Витягнуті документи були перевірені вручну на відповідність. Порівняння між різними файлами проводилося програмою, написаною на PHP.

2.3 Аналіз пошукових систем

Спочатку подаються узагальнені результати збору даних: кількість URL-адрес за кожен раунд пошуку (так звані загальні URL-адреси), кількість технічно точних URL-адрес за кожен раунд пошуку та відсоток точних URL-адрес у раунді пошуку із загальних URL-адрес у той раунд. У пошуковому раунді документ із заданою URL-адресою вважається технічно точним, якщо в ньому

з’являється один із пошукових термінів («інформаметричний» чи «інформаційний»). Набір неточних документів включає ті, які не включали пошукові терміни, ті, які не були знайдені (помилка 404), і ті, які не вдалося отримати через проблеми зі зв’язком (сервер не працює тощо). URL-адреса вважається технічно точною, якщо вона була класифікована як така у всіх пошукових раундах, в яких вона з’явилася. Загальна технічна точність набагато нижча за точність у будь-якому окремому раунді пошуку, оскільки вимога полягає в тому, що URL-адреса повинна бути технічно точною у всіх турах пошуку, в яких вона знаходилася. Технічна точність окремих пошуків була дуже високою, незважаючи на поширену думку, що пошукові системи видають багато шуму. Можливо, наше визначення технічної точності, яке не базується на суб'єктивному вимірі релевантності, є однією з причин високих цифр. Пошукові системи здійснюють пошук у вільному тексті, тому не можна і не слід очікувати вгадування контексту, в якому зацікавлений шукач. Подальший розвиток штучного інтелекту може дозволити двигунам робити кращі судження, але до того часу, на наш погляд, точність повинна вимірюватися лише появою пошукових термінів у документі. Технічна точність об'єктивна, і вимірювання можуть проводитися автоматично, без залучення експертів, судження яких є суб'єктивним.

Далі ми розглянули, як різні пошукові системи виконували результати в пошукових турах. Відсотки від загальної кількості різних URL-адрес та загальної кількості різних технічно точних URL-адрес, розташованих у даному раунді пошуку за всіма шістьма пошуковими системами двигуни. Ці відсотки вказують на частину URL-адрес (загальних та технічно точних), які кожна пошукова система охоплює у кожному пошуковому раунді із загальної кількості URL-адрес (загальних та технічно точних), зібраних у цьому раунді. В останньому рядку представлена кількість URL-адрес (загальних і технічно точних) кожної пошукової системи, розміщених протягом усього періоду пошуку. Значення в цьому рядку не є сумами стовпців над ними, оскільки кожна URL-адреса підраховується лише один раз, хоча вона, можливо, була розташована в

декількох раундах пошуку. В останньому рядку відсотки не відповідають загальному та загальному технічно точному відповідно. З цього моменту ми будемо використовувати точний як скорочення для технічно точного.

*% від загальної кількості в раунді

+% від загальної точності в раунді

Зверніть увагу на надзвичайно високу точність Excite, у четвертому раунді 168 із 170 документів були точними (98,8%!). Загальна точність Excite також є найвищою, 535 URL-адрес із 590 (90,7%), тоді як загальна точність Northern Light є найнижчою (80,9%).

Замість того, щоб розглядати кожен раунд окремо, ми порівнюємо кількість URL-адрес, знайдених кожною пошуковою системою в раунді в середньому (так званий середній раунд), із кількістю різних URL-адрес, визначених ними за всі шість раундів. На рисунку 1 розглядаються лише точні URL-адреси. Цей показник наочно показує величезну різницю між продуктивністю пошукових систем за один момент часу та за весь період. Дві крайності - це Northern Light та Excite. Що стосується Північного сяйва, різниця між двома стовпцями незначна, оскільки Northern Light знаходило майже однаковий набір URL-адрес у кожному з шести раундів. З іншого боку, Excite розмістив у три рази більше URL-адрес, ніж у будь-якому одному раунді. Очікується, що кількість URL-адрес, розташованих пошуковою системою протягом усього періоду, перевищує кількість URL-адрес, розташованих у будь-якому окремому раунді. Мережа є динамічним середовищем - з’являються нові сторінки, старі повністю видаляються, а інші змінюють свій вміст і перестають бути актуальними для даного запиту. Список URL-адрес за весь період складається з усіх URL-адрес, які були технічно точними при кожному отриманні URL-адреси. Цей список містить URL-адреси, які були, наприклад, лише у першому та другому раундах (приклад URL-адреси, яка перестала існувати або бути актуальною), а також URL-адреси, які з’явилися лише один раз в останньому раунді (приклад "нова" URL-адреса). Незважаючи на наші обгрунтовані очікування, для Northern Light існує лише дуже незначна різниця між кількістю URL-адрес,

визначених у середньому раунді, та кількістю URL-адрес, розташованих загалом. Здається, що її база даних (принаймні сторінки, на яких з'явилися терміни "інформатики" або "інформаметрика") є досить статичною, хоча графік, що з'являється на сайті "Пошукова система спостереження" про розміри пошукової системи (Салліван-2), показує, що під час за період пошуку розмір індексу Northern Light виріс з 55 мільйонів URL-адрес до 65 мільйонів URL-адрес. У попередній версії сторінки Саллівана (доступ до якої було отримано в серпні 1998 р.) Зазначалося, що між листопадом 1997 р. І серпнем 1998 р. Північне світло припинило повзати - твердження, яке більш відповідає нашим висновкам.

Очікується, що зміни в Інтернеті відображатимуться більш-менш однаково кожною з пошукових систем. Однак це не так. Давайте розглянемо результати пошуку з іншої точки зору. Ми розрахували середнє відносне охоплення кожної пошукової системи. Відносне охоплення пошукової машини за кожен раунд пошуку — це кількість точних URL-адрес, отриманих даною пошуковою системою, поділена на загальну кількість точних URL-адрес, отриманих у даному раунді пошуку. Оскільки значення для різних раундів пошуку були дуже схожими для кожної пошукової системи, ми вирішили відобразити середнє відносне охоплення, яке є середнім значенням відносного покриття за пошукові раунди. Ці середні показники можна порівняти із загальним відносним охопленням кожної пошукової системи (кількість точних URL-адрес, розташованих пошуковою системою у всіх раундах, поділена на загальну кількість точних URL-адрес, розташованих усіма пошуковими системами протягом усіх раундів). Зверніть увагу, що середнє відносне охоплення вимірює відносне охоплення пошукової системи в даний момент часу ("знімок"), тоді як загальне відносне охоплення вимірює охоплення пошукової системи протягом певного періоду часу. Результати показують, що продуктивність пошукових систем з часом значно відрізняється від їхньої продуктивності в даний момент часу. Наведені попередні результати базуються на пошуках, проведених у листопаді 1997 р. Обидва статті давали оцінки розміру Мережі та перекриття між

індексами пошукових систем, як побічний продукт вони також обчислювали значення, які відповідають нашим поняття відносного охоплення. Зазначимо, що Бхарат і Бродер вивчали характеристики лише чотирьох двигунів. Ці попередні роботи вимірювали відносне охоплення в даний момент часу на основі великої кількості запитів.

Порівнюючи загальне відносне покриття із середнім відносним покриттям, випадок Excite є найбільш вражаючим: у кожному окремому раунді пошуку він знайшов майже однакову кількість точних URL-адрес (від 148 до 168) і охоплює від 17,4% до 21,8 % точних URL-адрес, розташованих у даному раунді, однак, поєднавши результати шести пошуків, йому вдалося охопити 50,5% (!) від загальної кількості точних URL-адрес. Як це можливо?

Вивчивши список URL-адрес, ми виявили, що пошукові системи не тільки виявляють нові URL-адреси, вони також забувають URL-адреси, які вони знали раніше. Забуту URL-адресу ми визначаємо як точну URL-адресу, яка була знайдена даною пошуковою системою в певному раунді (який називається round_located), і вона все ще існує як точний документ в Інтернеті в одному з наступних раундів (званий round_forgotten), але не є отримано більше пошуковою системою, що обговорюється. Ми знаємо, що сторінка все ще існує і є релевантною, оскільки її було отримано одним із інших двигунів у round_forgotten та перевірено нами. При вивченні ступеня забудькуватості беруться до уваги лише точні URL-адреси, оскільки саме ці URL-адреси пошукова система повинна зберігати від одного раунду пошуку до іншого.

Ми майже не знайшли жодної згадки про проблему в літературі, лише AltaVista (1998) та Hotbot (1998) згадують про існування проблеми на своїх довідкових сторінках. Єдиним посиланням, що обговорює проблему зникнення сторінок, є Салліван (Салліван-3, 1998), "в зоні, лише для передплатників" його веб-сайту searchenginewatch.com (передплата не безкоштовна). За його словами, це проблема з 1997 року, однак ці сторінки "зазвичай з'являються протягом Excite приблизно через три тижні після їх зникнення". Це також проблема для Hotbot, і в цьому випадку також зниклі сторінки "повинні

автоматично з'явитися під час наступного оновлення" (протягом місяця). Пропоновані пояснення цих проблем із Hotbot - це проблеми із сервером та затримки мережі під час сканування; пояснення для Excite не дається. Таким чином, ми вирішили перевірити, чи забуті сторінки знову з’являться пізніше у списку сторінок, отриманих пошуковою системою. Ми розглядали точні URL-адреси лише протягом перших чотирьох раундів, щоб «дати шанс» пошуковій системі скинути URL-адресу (пізніше у п’ятому раунді) та заново її відкрити (пізніше у шостому раунді). Відсотки обчислюються із загальної кількості точних URL-адрес, які кожен пошуковий механізм отримував у перші чотири раунди.

Ці результати досить дивовижні. Усі пошукові системи видаляли URL-адреси зі своїх баз даних. Північне світло та Excite - це дві крайності, Northern Light навряд чи забув будь-які URL-адреси, тоді як Excite забув 70,9% (!) URL-адрес, які колись знаходив. У кожному з пошукових раундів Excite представляв нам зовсім іншу картину того, що Інтернет містить у нашому запиті, хоча він отримував майже однакову кількість URL-адрес у кожному раунді пошуку. Два додаткові механізми, AltaVista та Hotbot, не відновили значну частину скинутих URL-адрес протягом періоду пошуку.

У попередніх абзацах ми вивчали забуті та відновлені URL-адреси. В ході нашого дослідження ми застосували два додаткові заходи для кращого розуміння явища. Як згадувалось у вступі, один і той же сервер може мати кілька псевдонімів, і якщо пошукові системи знають це (наприклад, через DNS), він може отримати ту саму URL-адресу як у round_located, так і round_forgotten, але відображати результати під різними іменами в різні раунди. Оскільки ми не мали можливості вирішити, чи вказують дві URL-адреси на одну і ту ж фізичну адресу, ми розробили наступний тест (який охоплює вищезазначений випадок): для кожної пошукової системи ми порівняли вміст кожної забутої URL-адреси в round_forgotten.

URL-адреси, для яких не знайдено збігів, називаються повністю забутими. Це визначення досить широке та поблажливе щодо пошукових систем, оскільки воно охоплює набагато більше, ніж різні імена сервера (дублікати

дійсно існують у Мережі). З іншого боку, відсоток повністю втрачених URL-адрес вказує на втрату інформації, спричинену пошуковою системою протягом п’яти місяців, що розглядаються. Сторінки в цьому наборі - це сторінки, що існують в Інтернеті, пошукова машина їх знайшла один раз, і десь пізніше ці сторінки були вилучені зі списку отриманих результатів пошуковою системою з ясних незрозумілих причин. Механізм не лише отримав саму URL-адресу, але й жодну іншу URL-адресу, яка містила б точно таку ж інформацію. Ми говоримо не лише про декілька сторінок, як видно з таблиці 6, а про значну частину URL-адрес для всіх пошукових систем, за винятком Infoseek та Northern Light. Поняття бути повністю забутим досліджує зміст URL-адрес. Оскільки в Інтернеті є навмисні дублікати, ми придумали третє визначення, яке враховує існування таких дублікатів. Визначення втраченої URL-адреси базується на ідеї, що якщо сама пошукова система не "думала" в round_located, що дві різні URL-адреси з абсолютно однаковим вмістом є дублікатами, то якщо вони все ще існують в Інтернеті та є релевантними , вони обидва повинні з’являтися у його списку URL-адрес у round_forgotten. Згідно з цим визначенням, ми дозволяємо пошуковій системі вирішувати, що таке дублікати. Точніше, URL-адреса втрачається, якщо вона була забута, і будь-яка інша отримана URL-адреса в round_forgotten не містить точно тієї ж інформації, або якщо вона є, вона також була отримана в round_locate обговорюваним механізмом. Зверніть увагу, що за будь-яким із наведених вище визначень представлені цифри цілком можуть бути заниженими, оскільки пошукові системи можуть втратити деякі додаткові URL-адреси, але ці URL-адреси не перевірялись. Відсотки обчислюються із загальної кількості точних URL-адрес, отриманих механізмом у перших п’яти раундах (оскільки URL-адреси, виявлені лише в останньому раунді, не можна забувати). Зверніть увагу, що кількість URL-адрес на пошукову систему в таблиці 6 відрізняється від кількості в таблиці 5, де ми розглядали лише перші чотири раунди. AltaVista, Hotbot та Lycos в середньому втратили 21,9% інформації. «Північне світло» та «Infoseek» зробили значно краще за другим визначенням (повністю забутим), ніж за першим.

Результати за третім визначенням подібні до результатів першого, але в середньому менші на 5%.

Щоб ще більше підкреслити нашу думку, для кожної пошукової системи та кожної точної URL-адреси, виявленої нею, ми обчислили кількість раундів, в яких пошукова система знаходила URL-адресу. Також було підраховано кількість пошукових раундів, в яких знаходилася URL-адреса (незалежно від пошукової системи). Відсотки не відповідають загальній кількості точних URL-адрес, знайдених кожною пошуковою системою (загальна кількість подано в дужки під назвою механізму). У стовпці "Усього" відображаються спільні зусилля всіх шести пошукових систем. Результати свідчать про самостійне накладання пошукових систем. Знову ж таки, спостерігається разюча різниця між Excite та об'єднаними зусиллями всіх двигунів. Спільними зусиллями всіх шести пошукових систем було створено 484 документи (45,7% від загальної кількості), які з'явились у всіх шести пошукових раундах, тоді як лише 0,6% документів, отриманих Excite, знаходились усі шість разів. Північне світло представляє іншу крайність: його база даних була майже стабільною протягом періоду пошуку, майже не відображаючи змін, що відбуваються в Інтернеті з часом.

Останній раунд пошуку відбувся 20 червня 1999 року, через рік після початкового експерименту. Цього разу, окрім процедури пошуку та збору даних, проведеної у всіх пошукових раундах, були переглянуті всі URL-адреси, розташовані в початкових раундах. Невелика кількість результатів у Hotbot, ймовірно, була спричинена змінами у відображенні результатів, які були введені приблизно в той час, коли відбувся цикл порівняння, і не були пояснені в документації Hotbot. Таким чином, результати для Hotbot не аналізуються. Результати наочно показують, що тенденція "забудькуватості" продовжується, а результати пошукових систем не настільки стабільні, як можна було очікувати.

Пошукові системи залишаються популярними — і користувачі як ніколи задоволені якістю результатів пошуку — але багато хто стурбований збором

особистої інформації пошуковими системами та іншими веб-сайтами. Більшість користувачів пошуку не схвалюють особисту інформацію, яка збирається для результатів пошуку або для цільової реклами.

Опитування Pew Internet & American Life у лютому 2012 року включало кілька запитань, які досліджували, як респонденти ставляться до пошукових систем та інших веб-сайтів, які збирають інформацію про них та використовують її для формування результатів пошуку або націлення на них реклами [5]. Очевидно, що більшість користувачів Інтернету та пошуку не схвалюють цю практику у всіх контекстах, які ми досліджували. Зокрема, опитування поставило наступний вибір для користувачів пошукових систем:

- 65% стверджують, що це погано, якщо пошукова система збирає інформацію про ваші пошуки, а потім використовує її для ранжирування ваших майбутніх результатів пошуку, оскільки це може обмежити інформацію, яку ви отримуєте в Інтернеті, і які результати пошуку ви бачите;
- 29% кажуть ДОБРО, якщо пошукова система збирає інформацію про ваші пошукові запити, а потім використовує її для ранжирування ваших майбутніх результатів пошуку, оскільки вона дає вам результати, які є більш доречними для вас;

Всім користувачам Інтернету було запропоновано наступний вибір щодо цільової реклами:

- 68% кажуть я не погоджуюсь з цільовою рекламою, бо мені не подобається відстежувати та аналізувати мою поведінку в Інтернеті;
- 28% говорять я погоджуюсь з цільовою рекламою, оскільки це означає, що я бачу рекламу та отримую інформацію про речі, які мене насправді цікавлять.

Загальні погляди на ефективність роботи пошукової системи є дуже позитивними. Більше десяти років дані Інтернету Pew постійно показують, що використання пошукової системи є однією з найпопулярніших видів діяльності в Інтернеті. У січні 2002 року 52% всіх американців користувалися пошу-

ковими системами. У лютому 2012 року цей показник зріс до 73% усіх американців. Будь-якого дня на початку 2012 року більше половини дорослих, які користуються Інтернетом, користуються пошуковою системою (59%). Це вдвічі більше, ніж 30% користувачів Інтернету, які користувались пошуковими системами в звичайний день 2004 року. І частота користування пошуковими системами різко зросла.

Більше того, користувачі повідомляють загалом про хороші результати та відносно високу впевненість у можливостях пошукових систем:

- 91% користувачів пошукових систем кажуть, що завжди або більшу частину часу знаходять інформацію, яку вони шукають, коли користуються пошуковими системами;
- 73% користувачів пошукових систем кажуть, що більшість або вся інформація, яку вони знаходять, користуючись пошуковими системами, є точною та надійною;
- 66% користувачів пошукових систем кажуть, що пошукові системи є справедливим та неупередженим джерелом інформації;
- 55% користувачів пошукових систем кажуть, що з їх досвіду якість результатів пошуку з часом покращується, тоді як лише 4% кажуть, що вона погіршилася;
- 52% користувачів пошукових систем кажуть, що результати пошукових систем з часом стали більш актуальними та корисними, тоді як лише 7% повідомляють, що результати стали менш актуальними.

Ці висновки є фоном для постійних політичних дискусій щодо конфіденційності, збору особистої інформації в Інтернеті та ентузіазму щодо цілеспрямованого пошуку та цільової реклами серед компаній. Вони також виникають, коли Google впроваджує нову політику конфіденційності, в якій інформація про поведінку користувачів в Інтернеті під час входу в програми Google може бути зібрана та об’єднана у згуртований профіль користувача. Сюди входять матеріали пошукової системи Google, соціальної мережі Google+, вебсайту YouTube для обміну відео та Gmail.

Більшість користувачів Інтернету кажуть, що не знають, як обмежити інформацію, яку збирає про них веб-сайт. Тільки 38% користувачів Інтернету кажуть, що вони загалом обізнані про те, як вони самі можуть обмежити кількість інформації про них, яку збирає веб-сайт. Серед цієї групи одна загальна стратегія, яку люди використовують для обмеження збору персональних даних, - це видалення своєї історії веб-пошуку: 81% тих, хто знає способи управління захопленням своїх даних, роблять це. Близько 75% цієї групи використовують налаштування конфіденційності веб-сайтів, щоб контролювати те, що про них зафіксовано. А 65% змінюють налаштування браузера, щоб обмежити інформацію, яка збирається.

Загалом користувачі пошуку впевнені у своїх силах. Більшість користувачів пошуку кажуть, що впевнені у власних пошукових можливостях і знаходять те, що шукають, більшу частину часу. Більше половини користувачів пошуку (56%) заявляють, що дуже впевнені у своїх пошукових можливостях, тоді як лише 6% заявляють, що не надто або не всі впевнені. І переважна більшість користувачів пошуку повідомляють, що вони завжди можуть знайти те, що шукають (29%) або більшу частину часу (62%). Позитивний досвід пошуку частіше, ніж негативний досвід

На запитання про різний досвід роботи з пошуковими системами, більше користувачів повідомляють про позитивний досвід, ніж про негативний. Вони заявили, що користуючись пошуковими системами, вони мали:

- дізналися щось нове або важливе, що дійсно допомогло їм або збільшило їх знання (86% користувачів пошуку мали такий досвід);
- знайшли справді неясний факт чи інформацію, яку, на їх думку, не вдалося знайти (50%);
- отримали суперечливу інформацію в результатах пошуку і не змогли зрозуміти, що правильно (41%);
- отримали стільки інформації в наборі результатів, що ви відчуваєте себе пригніченими (38%);

- виявив, що в результатах пошуку відсутня важлива інформація (34%) Google продовжує залишатися найпопулярнішою пошуковою системою, з великим відривом Google продовжує домінувати у списку найбільш часто використовуваних пошукових систем. На запитання, яку пошукову систему вони використовують найчастіше, 83% користувачів пошуку відповідають Google. Наступною найбільш цитованою пошуковою системою є Yahoo, про яку згадують лише 6% користувачів пошуку. Коли ми востаннє задавали це запитання в 2004 році, розрив між Google і Yahoo був набагато вужчим: 47% користувачів пошуку вважали, що Google є їхнім двигуном на вибір, а 26% посилалися на Yahoo.

Такі результати опитування, проведеного з 20 січня по 19 лютого 2012 року серед 2253 дорослих людей віком від 18 років, у тому числі 901 інтерв'ю на мобільний телефон. Співбесіди проводились англійською та іспанською мовами. Похибка для повної вибірки становить плюс-мінус 2 відсоткові пункти.

Опитування в Інтернеті від лютого 2012 р. Виявило, що 91% дорослих людей в Інтернеті використовують пошукові системи для пошуку інформації в Інтернеті, порівняно з 84% у червні 2004 р., Коли ми востаннє проводили розширену групу запитань щодо опитування людей, які користуються пошуковою системою. У будь-який день в Інтернеті 59% тих, хто користується Інтернетом, користуються пошуковими системами. У 2004 році ця цифра становила лише 30% користувачів Інтернету. Ще в 2002 році більше восьми з десяти дорослих користувачів Інтернету користувались пошуковими системами, і, як ми зазначали у звіті від серпня 2011 р., Пошук відповідає лише електронною поштою як у загальному відсотку користувачів Інтернету, які беруть участь у цій діяльності, так і у відсотках Інтернет-користувачі роблять це в певний день. У наведеній нижче таблиці показано, як пошук порівнюється в часі з деякими іншими популярними мережевими заходами.

2.4 Висновки з розділу 2

Розвиток пошукових двигунів приносить користь для всіх можливих користувачів мережі інтернет, бо неможливо уявити собі користування інтернетом без використання таких пошукових двигунів як Google, Yahoo та Bing. На даний момент пошукові двигуни досить якісно виконують свої обов’язки, але ще досить далеко до ідеальних рівнів пошуку. Зважаючи на те, що кількість користувачів інтернету буде лише зростати, пошуковим двигунам необхідний постійний та безперервний розвиток.

На даний момент вже майже кожен веб-сайт повинен розглядати можливість використання як розроблених стороно пошукових двигунів, так і розроблених власно. Це покращить якість користування веб-сайтом і допоможе як користувачам, так і веб-розробникам, що працюють над цим сайтом. Чим більший обсяг даних над яким необхідно працювати користувачу, тим більша необхідність у використанні пошукових двигунів.

РОЗДІЛ 3 РОЗРОБКА КОМП'ЮТЕРНОЇ СИСТЕМИ ДЛЯ ДОСЛІДЖЕННЯ ПРИСКОРЕННЯ ПОШУКУ ІНФОРМАЦІЇ У БАЗІ ДАНИХ

3.1 Налаштування середовища для розробки програмного забезпечення

Перед початком роботи необхідно встановити і налаштувати програмне забезпечення:

1. Мова програмування: PHP версії 8.0.13, для цього треба завантажити пакет встановщик з офіціального сайту
2. Elasticsearch версії 7.13.2

Після цього треба встановити фреймворк Laravel. Laravel — це фреймворк веб-додатків з виразним, елегантним синтаксисом. Веб-фреймворк забезпечує структуру та вихідну точку для створення вашого додатку, що дозволяє зосередитись на створенні чогось дивовижного, поки ми поглинаємо деталі.

Laravel прагне забезпечити дивовижний досвід розробника, надаючи потужні функції, такі як ретельне введення залежностей, виразний рівень абстракції бази даних, черги та заплановані завдання, тестування модулів та інтеграції тощо.

Чому саме Laravel? Існує безліч інструментів та фреймворків, доступних для вас під час створення веб-програми. Однак Laravel - найкращий вибір для створення сучасних повнофункціональних веб-додатків.

Laravel є "прогресивною" структурою. Під цим мається на увазі, що Ларавель росте. Для цього фреймворка вже наявна величезна бібліотека документації, посібників та відеоуроків Laravel допоможе будь-кому вивчитись не перевантажуючись. І це лише початок, бо ця бібліотека продовжує розвиватись і далі.

Для старших розробників Laravel пропонує надійні інструменти для введення залежностей, модульного тестування, черг, подій у реальному часі

тощо. Laravel досконало налаштований на створення професійних веб-додатків і готовий витримати корпоративні робочі навантаження.

Laravel неймовірно масштабований. Завдяки зручності масштабування PHP та вбудованій підтримці Laravel швидких розподілених кеш-систем, таких як Redis, горизонтальне масштабування за допомогою Laravel - це легкий бриз. Насправді програми Laravel легко масштабуються для обробки сотень мільйонів запитів на місяць, що робить цей фреймворк дуже корисним у створенні сайту, що працює з «важкими» даними.

Laravel поєднує в собі найкращі пакети в екосистемі PHP, щоб запропонувати найбільш надійну та зручну для розробників структуру. Крім того, тисячі талановитих розробників з усього світу внесли свій внесок у фреймворк.

Laravel включає в себе Eloquent, об’єктно-реляційний картограф (ORM), що робить легкою взаємодію з базою даних. При використанні Eloquent кожна таблиця бази даних має відповідну «модель», яка використовується для взаємодії з цією таблицею. Окрім отримання записів із таблиці бази даних, моделі Eloquent дозволяють також вставляти, оновлювати та видаляти записи з таблиці.

Так як операційною системою, на якій буде працювати програмне забезпечення, є операційна система Microsoft Windows 10, фреймворк Laravel повинен бути встановлений за інструкцією для Windows, що наявна в документації на офіційному сайті.

Перш ніж створити нову програму Laravel на вашому комп’ютері з Windows, переконайтеся, що ви встановили Docker Desktop. Далі ви повинні переконатися, що підсистема Windows для Linux 2 (WSL2) інстальована та активована. WSL дозволяє запускати двійкові виконувані файли Linux у Windows 10. Інформацію про те, як встановити та ввімкнути WSL2, можна знайти в документації середовища розробника Microsoft.

Після встановлення та ввімкнення WSL2 ви повинні переконатися, що Docker Desktop налаштовано на використання серверної частини WSL2.

Далі ви готові створити свій перший проект Laravel. Запустіть термінал Windows і розпочніть новий термінальний сеанс для вашої операційної системи WSL2 Linux. Далі ви можете використовувати просту команду терміналу, щоб створити новий проект Laravel.

Фреймворк Laravel та бібліотека JQuery були використані у вигляді прив'язок (bindings) для мови програмування PHP. Найголовнішою причиною використання такого стеку технологій є кросплатформність, оскільки розроблена система повинна працювати на мікрокомп'ютері NVIDIA Jetson Nano, який працює під операційною системою Linux [15].

3.2 Програмна архітектура

Для архітектури обрана архітектура MVC (model-view-controller), на якій базується сам фреймворк Laravel. Для здебільшого покращення якості структури та архітектури проекту був використаний набір схем і принципів розроблений Еріком Евансом, відомий як DDD (domain-driven design, предметно-орієнтоване проектування). Застосування предметно-орієнтованого проектування дозволяє знизити ризик невдач при розробці програмного забезпечення за рахунок покращення комунікації між бізнес-замовником і командою розробників. Практики DDD поділяються на дві групи: стратегічну та тактичну.

Стратегічні інструменти DDD використовуються для прийняття архітектурних проектних рішень високого рівня: декомпозиції систем на компоненти та визначення способів їх інтеграції на основі вибудовування загального розуміння, як система, що розробляється, забезпечуватиме задоволення потреб бізнесу або замовника.

Тактичні патерни DDD дозволяють нам писати код таким чином, щоб він відображав предметну область, відповідав її цілям та розмовляв мовою бізнесу.

Значний час приділяється розгляду реальних прикладів та вирішення практичних завдань, що дає можливість учасникам закріпити отримані знання

на практиці та відпрацювати навички застосування стратегічного дизайну та тактичного моделювання.

3.3 Реалізація основного функціоналу

Реалізація почалась з розробки фронтенд частини веб-сайту, створення HTML-сторінок. Найпершою сторінкою розроблено сторінку index.html. Ця сторінка відповідає домашній сторінці веб-сайту, з якої користувач розпочинає роботу із програмним забезпеченням і має доступ до сторінок перегляду статтей, авторизації, реєстрації, сторінки з описом сайту, сторінки-блогу та інш. Лістинг 1 демонструє вигляд коду сторінки.

Лістинг *1 Код основної сторінки*

```
<header>
<div class="header-area ">
<div id="sticky-header" class="main-header-area">
<div class="container-fluid ">
<div class="header_bottom_border">
<div class="row align-items-center">
<div class="col-xl-3 col-lg-2">
<div class="logo">
<a href="/">
</a>
</div>
</div>
<div class="col-xl-6 col-lg-7">
<div class="main-menu  d-none d-lg-block">
<nav>
<ul id="navigation">
<li><a href="/">home</a></li>
```

```
    <li><a href="jobs">Browse</a></li>
    <li><a href="#">pages <i class="ti-angle-
down"></i></a>
    <ul class="submenu">
    <li><a href="candidate">Authors </a></li>
    <li><a href="job_details">Details </a></li>
    <li><a href="elements">elements</a></li>
    </ul>
    </li>
    <li><a href="#">blog <i class="ti-angle-
down"></i></a>
    <ul class="submenu">
    <li><a href="blog">blog</a></li>
    <li><a href="single-blog">single-blog</a></li>
    </ul>
    </li>
    <li><a href="contact">Contact</a></li>
    </ul>
    </nav>
    </div>
    </div>
    <div class="col-xl-3 col-lg-3 d-none d-lg-block">
    <div class="Appointment">
    <div class="phone_num d-none d-xl-block">
    <a href="#">Log in</a>
    </div>
    <div class="d-none d-lg-block">
    <a class="boxed-btn3" href="#">Post an Article</a>
    </div>
    </div>
```

```
    </div>
    <div class="col-12">
    <div class="mobile_menu d-block d-lg-none"></div>
          </div>
         </div>
        </div>
       </div>
      </div>
     </div>
    </header>
    <div class="slider_area">
    <div class="single_slider  d-flex align-items-
center slider_bg_1">
    <div class="container">
    <div class="row align-items-center">
    <div class="col-lg-7 col-md-6">
    <div class="slider_text">
    <h3 class="wow fadeInLeft" data-wow-duration="1s"
data-wow-delay=".2s">4000+ Articles Available</h3>
    <p class="wow fadeInLeft" data-wow-duration="1s"
data-wow-delay=".4s"></p>
    <div class="sldier_btn wow fadeInLeft" data-wow-
duration="1s" data-wow-delay=".5s">
    <a href="#" class="boxed-btn3">Write an Article</a>
          </div>
        </div>
       </div>
      </div>
     </div>
    </div>
```

```
    <div class="ilstration_img wow fadeInRight d-none
d-lg-block text-right" data-wow-duration="1s" data-wow-
delay=".2s">
    <img src="resources/img/banner/illustration.png"
alt="">
     </div>
    </div>
    <div class="catagory_area">
    <div class="container">
    <div class="row cat_search">
    <div class="col-lg-3 col-md-4">
    <div class="single_input">
    <input type="text" placeholder="Search keyword">
     </div>
    </div>
    <div class="col-lg-3 col-md-4">
    <div class="single_input">
    <select class="wide" >
    <option data-display="Location">Topic</option>
    <option value="1">Dhaka</option>
    <option value="2">Rangpur</option>
    <option value="4">Sylet</option>
    </select>
     </div>
    </div>
    <div class="col-lg-3 col-md-4">
    <div class="single_input">
    <select class="wide">
    <option data-display="Category">Category</option>
    <option value="1">Category 1</option>
```

```
    <option value="2">Category 2</option>
    <option value="4">Category 3</option>
    </select>
    </div>
    </div>
    <div class="col-lg-3 col-md-12">
    <div class="job_btn">
    <a href="#" class="boxed-btn3">Find an Article</a>
    </div>
    </div>
    </div>
    <div class="row">
    <div class="col-lg-12">
    <div class="popular_search d-flex align-items-
center">
    <span>Popular Search:</span>
    <ul>
    <li><a href="#">Design & Creative</a></li>
    <li><a href="#">Marketing</a></li>
    <li><a href="#">Administration</a></li>
    <li><a href="#">Teaching & Education</a></li>
    <li><a href="#">Engineering</a></li>
    <li><a href="#">Software & Web</a></li>
    <li><a href="#">Telemarketing</a></li>
          </ul>
        </div>
       </div>
      </div>
     </div>
    </div>
```

```
<div class="featured_candidates_area">
<div class="container">
<div class="row">
<div class="col-lg-12">
<div class="section_title text-center mb-40">
<h3>Featured Authors</h3>
</div>
</div>
</div>
<div class="row">
<div class="col-lg-12">
<div class="candidate_active owl-carousel">
<div class="single_candidates text-center">
<div class="thumb">
<img src="resources/img/candiateds/1.png" alt="">
</div>
<a href="#"><h4>Markary Jondon</h4></a>
<p>Software Engineer</p>
</div>
<div class="single_candidates text-center">
<div class="thumb">
<img src="resources/img/candiateds/2.png" alt="">
</div>
<a href="#"><h4>Markary Jondon</h4></a>
<p>Software Engineer</p>
</div>
<div class="single_candidates text-center">
<div class="thumb">
<img src="resources/img/candiateds/3.png" alt="">
</div>
```

```
<a href="#"><h4>Markary Jondon</h4></a>
<p>Software Engineer</p>
</div>
<div class="single_candidates text-center">
<div class="thumb">
<img src="resources/img/candiateds/4.png" alt="">
</div>
<a href="#"><h4>Markary Jondon</h4></a>
<p>Software Engineer</p>
</div>
<div class="single_candidates text-center">
<div class="thumb">
<img src="resources/img/candiateds/5.png" alt="">
</div>
<a href="#"><h4>Markary Jondon</h4></a>
<p>Software Engineer</p>
</div>
<div class="single_candidates text-center">
<div class="thumb">
<img src="resources/img/candiateds/6.png" alt="">
</div>
<a href="#"><h4>Markary Jondon</h4></a>
<p>Software Engineer</p>
</div>
<div class="single_candidates text-center">
<div class="thumb">
<img src="resources/img/candiateds/7.png" alt="">
</div>
<a href="#"><h4>Markary Jondon</h4></a>
<p>Software Engineer</p>
```

```
    </div>
    <div class="single_candidates text-center">
    <div class="thumb">
    <img src="resources/img/candiateds/8.png" alt="">
    </div>
    <a href="#"><h4>Markary Jondon</h4></a>
    <p>Software Engineer</p>
    </div>
    <div class="single_candidates text-center">
    <div class="thumb">
    <img src="resources/img/candiateds/9.png" alt="">
    </div>
    <a href="#"><h4>Markary Jondon</h4></a>
    <p>Software Engineer</p>
    </div>
    <div class="single_candidates text-center">
    <div class="thumb">
    <img src="{{asset('img/candiateds/4.png')}}"
alt="">
    </div>
    <a href="#"><h4>Markary Jondon</h4></a>
    <p>Software Engineer</p>
    </div>
    <div class="single_candidates text-center">
    <div class="thumb">
    <img src="{{asset('img/candiateds/10.png')}}"
alt="">
    </div>
    <a href="#"><h4>Markary Jondon</h4></a>
    <p>Software Engineer</p>
```

```
    </div>
    <div class="single_candidates text-center">
    <div class="thumb">
    <img src="{{asset('img/candiateds/3.png')}}"
alt="">
    </div>
    <a href="#"><h4>Markary Jondon</h4></a>
    <p>Software Engineer</p>
    </div>
    <div class="single_candidates text-center">
    <div class="thumb">
    <img src="{{asset('img/candiateds/4.png')}}"
alt="">
    </div>
    <a href="#"><h4>Markary Jondon</h4></a>
    <p>Software Engineer</p>
    </div>
    </div>
    </div>
    </div>
    </div>
    </div>
<script src="{{asset('js/vendor/modernizr-
3.5.0.min.js')}}"></script>
        <script src="{{asset('js/vendor/jquery-
1.12.4.min.js')}}"></script>
        <script
src="{{asset('js/popper.min.js')}}"></script>
        <script
src="{{asset('js/bootstrap.min.js')}}"></script>
```

```
        <script
src="{{asset('js/owl.carousel.min.js')}}"></script>
        <script
src="{{asset('js/isotope.pkgd.min.js')}}"></script>
        <script src="{{asset('js/ajax-
form.js')}}"></script>
        <script
src="{{asset('js/waypoints.min.js')}}"></script>
        <script
src="{{asset('js/jquery.counterup.min.js')}}"></script>
        <script
src="{{asset('js/imagesloaded.pkgd.min.js')}}"></script
>
        <script
src="{{asset('js/scrollIt.js')}}"></script>
        <script
src="{{asset('js/jquery.scrollUp.min.js')}}"></script>
        <script
src="{{asset('js/wow.min.js')}}"></script>
        <script src="{{asset('js/nice-
select.min.js')}}"></script>
        <script
src="{{asset('js/jquery.slicknav.min.js')}}"></script>
        <script src="{{asset('js/jquery.magnific-
popup.min.js')}}"></script>
        <script
src="{{asset('js/plugins.js')}}"></script>
        <script
src="{{asset('js/gijgo.min.js')}}"></script>
```

Після створення HTML-файлів вони були перетворені у файли blade.php, що підтримуються фреймворком Laravel для відображення сторінок. Під час перетворення деякий функціонал сторінки необхідно змінити спираючись на стандарти написання фронтенд частини для цього фреймворку, що несе назву view.

Можна також звернути увагу на CSS стилі, що будуть використовуватись по всій частині проекту. Одним із розроблених та використаних стилів є стиль style.css. Код стилю приведений у лістингу 2.

Лістинг *2 Код основного стилю*

```
.flex-center-start {
  display: -webkit-box;
  display: -ms-flexbox;
  display: flex;
  -webkit-box-align: center;
  -ms-flex-align: center;
  align-items: center;
  -webkit-box-pack: start;
  -ms-flex-pack: start;
  justify-content: start;
}

body {
  font-family: "Roboto", sans-serif;
  font-weight: normal;
  font-style: normal;
}

.img {
  max-width: 100%;
```

```
  -webkit-transition: 0.3s;
  -moz-transition: 0.3s;
  -o-transition: 0.3s;
  transition: 0.3s;
}

.button {
  -webkit-transition: 0.3s;
  -moz-transition: 0.3s;
  -o-transition: 0.3s;
  transition: 0.3s;
}

a:focus,
.button:focus, button:focus {
  text-decoration: none;
  outline: none;
}

a:focus {
  text-decoration: none;
}

a:focus,
a:hover,
.portfolio-cat a:hover,
.footer -menu li a:hover {
  text-decoration: none;
}
```

```
p {
  font-size: 16px;
  font-weight: 400;
  line-height: 28px;
  color: #7A838B;
  margin-bottom: 13px;
  font-family: "Roboto", sans-serif;
}

.overlay::before {
  position: absolute;
  content: "";
  background-color: #040E27;
  top: 0;
  left: 0;
  width: 100%;
  height: 100%;
  z-index: -1;
  opacity: .5;
}

.overlay2::before {
  position: absolute;
  content: "";
  background-color: #191d34;
  top: 0;
  left: 0;
  width: 100%;
  height: 100%;
  z-index: -1;
```

```
      opacity: 0.6;
    }

    .overlay_03::before {
      position: absolute;
      width: 100%;
      height: 100%;
      left: 0;
      top: 0;
      background: #191d34;
      opacity: .6;
      content: '';
      z-index: -1;
    }

    .bradcam_overlay::before {
      position: absolute;
      content: "";
      background: -moz-linear-gradient(left, #5db2ff 0%,
#65b4f9 24%, rgba(124, 185, 233, 0) 96%, rgba(125, 185,
232, 0) 100%);
      background: -webkit-linear-gradient(left, #5db2ff
0%, #65b4f9 24%, rgba(124, 185, 233, 0) 96%, rgba(125,
185, 232, 0) 100%);
      background: linear-gradient(to right, #5db2ff 0%,
#65b4f9 24%, rgba(124, 185, 233, 0) 96%, rgba(125, 185,
232, 0) 100%);

      filter:
progid:DXImageTransform.Microsoft.gradient(
```

```
startColorstr='#5db2ff',
endColorstr='#007db9e8',GradientType=1 );
      top: 0;
      left: 0;
      width: 100%;
      height: 100%;
      z-index: -1;
      opacity: 1;
    }
    .owl-carousel .owl-nav div {
      background: transparent;
      height: 50px;
      left: 0px;
      position: absolute;
      text-align: center;
      top: 50%;
      -webkit-transform: translateY(-50%);
      -ms-transform: translateY(-50%);
      transform: translateY(-50%);
      -webkit-transition: all 0.3s ease 0s;
      -o-transition: all 0.3s ease 0s;
      transition: all 0.3s ease 0s;
      width: 50px;
      color: #707070;
      background-color: transparent;
      -webkit-border-radius: 50%;
      -moz-border-radius: 50%;
      border-radius: 50%;
      left: 50px;
      font-size: 15px;
```

```
  line-height: 50px;
  border: 1px solid #4D6174;
  left: 150px;
  color: #fff;
}

.boxed-btn {
  background: #fff;
  color: #131313;
  display: inline-block;
  padding: 14px 44px;
  font-family: "Roboto", sans-serif;
  font-size: 14px;
  font-weight: 400;
  border: 0;
  border: 1px solid #00D363;
  text-align: center;
  color: #00D363 !important;
  text-transform: uppercase;
  cursor: pointer;
}

.boxed-btn3 {
  background: #00D363;
  color: #fff;
  display: inline-block;
  padding: 13px 29px 13px 29px;
  font-family: "Roboto", sans-serif;
  font-size: 16px;
  font-weight: 500;
```

```
  border: 0;
  border: 1px solid transparent;
  -webkit-border-radius: 5px;
  -moz-border-radius: 5px;
  border-radius: 5px;
  text-align: center;
  color: #fff !important;
  text-transform: capitalize;
  -webkit-transition: 0.3s;
  -moz-transition: 0.3s;
  -o-transition: 0.3s;
  transition: 0.3s;
  cursor: pointer;
}
.boxed-btn4 {
  display: inline-block;
  padding: 11px 29px 13px 29px;
  font-family: "Roboto", sans-serif;
  font-size: 16px;
  font-weight: 500;
  -webkit-border-radius: 5px;
  -moz-border-radius: 5px;
  border-radius: 5px;
  text-align: center;
  color: #fff !important;
  text-transform: capitalize;
  -webkit-transition: 0.3s;
  -moz-transition: 0.3s;
  -o-transition: 0.3s;
  transition: 0.3s;
```

```
  cursor: pointer;
  color: #00D363 !important;
  border: 1px solid #00D363;
  background: transparent;
}
.boxed-btn3-line {
  color: #F91842 !important;
  display: inline-block;
  padding: 14px 31px;
  font-family: "Roboto", sans-serif;
  font-size: 15px;
  font-weight: 500;
  border: 0;
  border: 1px solid #F91842;
  -webkit-border-radius: 30px;
  -moz-border-radius: 30px;
  border-radius: 30px;
  text-align: center;
  text-transform: capitalize;
  -webkit-transition: 0.5s;
  -moz-transition: 0.5s;
  -o-transition: 0.5s;
  transition: 0.5s;
  cursor: pointer;
}
```

Цей стиль використовується в усіх сторінках веб-сайту і несе в собі велику кількість покращень якості використання цього сайту.

Іншими файлами що використовуються при роботі із фронт-частиною сайту є файли типу .js, що несуть в собі функції написані мовою JavaScript і використовуються на усіх сторінках.

Лістинг *3 Код файлу main.js*

```
(function ($) {
"use strict";
$(window).on('scroll', function () {
var scroll = $(window).scrollTop();
if (scroll < 400) {
    $("#sticky-header").removeClass("sticky");
    $('#back-top').fadeIn(500);
} else {
    $("#sticky-header").addClass("sticky");
    $('#back-top').fadeIn(500);
}
});

$(document).ready(function(){

var menu = $('ul#navigation');
if(menu.length){
menu.slicknav({
    prependTo: ".mobile_menu",
    closedSymbol: '+',
    openedSymbol:'-'
});
};

var slider = $('.slider_active');
```

```
    if(slider.length) {
      slider.owlCarousel({
        loop:true,
        margin:0,
      items:1,
      autoplay:true,
      navText:['<i      class="ti-angle-left"></i>','<i
class="ti-angle-right"></i>'],
        nav:true,
      dots:false,
      autoplayHoverPause: true,
      autoplaySpeed: 800,
        responsive:{
            0:{
                items:1,
                nav:false,
            },
            767:{
                items:1,
                nav:false,
            },
            992:{
                items:1,
                nav:false
            },
            1200:{
                items:1,
                nav:false
            },
            1600:{
```

```
                items:1,
                nav:true
            }
        }
      });
    }

    var candidate = $('.candidate_active');
    if(candidate.length){
      candidate.owlCarousel({
      loop:true,
      margin:30,
      autoplay:true,
      navText:['<i      class="ti-angle-left"></i>','<i
class="ti-angle-right"></i>'],
      nav:true,
      dots:false,
      autoplayHoverPause: true,
      autoplaySpeed: 800,
        responsive:{
            0:{
                items:1,
                dots:false,
                nav:false,
            },
            767:{
                items:3,
                dots:false,
                nav:false,
            },
```

```
            992:{
                items:4,
                nav:true
            },
            1200:{
                items:4,
                nav:true
            },
            1500:{
                items:4
            }
        }
      });
    }

      var $grid = $('.grid').isotope({
        itemSelector: '.grid-item',
        percentPosition: true,
        masonry: {
          columnWidth: 1
        }
      });

      $('.portfolio-menu').on('click',        'button',
function () {
        var filterValue = $(this).attr('data-filter');
        $grid.isotope({ filter: filterValue });
      });
```

```
      $('.portfolio-menu  button').on('click',  function
(event) {

$(this).siblings('.active').removeClass('active');
        $(this).addClass('active');
        event.preventDefault();
    });

      new WOW().init();

      $('.counter').counterUp({
        delay: 10,
        time: 10000
      });

    $('.popup-image').magnificPopup({
    type: 'image',
    gallery: {
      enabled: true
    }
    });

    $('.img-pop-up').magnificPopup({
    type: 'image',
    gallery: {
      enabled: true
    }
    });

    /* magnificPopup video view */
```

```
$('.popup-video').magnificPopup({
type: 'iframe'
});

  var brand = $('.brad_active');
  if(brand.length){
    brand.owlCarousel({
    loop:true,
    autoplay:true,
    nav:false,
    dots:false,
    autoplayHoverPause: true,
    autoplaySpeed: 800,
      responsive:{
          0:{
              items:2,
              nav:false
          },
          767:{
              items:4
          },
          992:{
              items:5
          }
      }
    });
  }

if (document.getElementById('default-select')) {
  $('select').niceSelect();
```

```
    }

    $('.details_active').owlCarousel({
      loop:true,
      margin:0,
    items:1,
    navText:['<i         class="ti-angle-left"></i>','<i
class="ti-angle-right"></i>'],
    nav:true,
    dots:false,
      responsive:{
          0:{
              items:1,
              nav:false

          },
          767:{
              items:1,
              nav:false
          },
          992:{
              items:1,
              nav:false
          },
          1200:{
              items:1,
          }
      }
    });
    });
```

```
$(document).ready(function() {
$('.popup-with-form').magnificPopup({
    type: 'inline',
    preloader: false,
    focus: '#name',

    callbacks: {
        beforeOpen: function() {
            if($(window).width() < 700) {
                this.st.focus = false;
            } else {
                this.st.focus = '#name';
            }
        }
    }
});
});

function mailChimp() {
  $('#mc_embed_signup').find('form').ajaxChimp();
}
mailChimp();
        // Search Toggle
        $("#search_input_box").hide();
        $("#search").on("click", function () {
            $("#search_input_box").slideToggle();
            $("#search_input").focus();
        });
        $("#close_search").on("click", function () {
```

```
            $('#search_input_box').slideUp(500);
        });
        // Search Toggle
        $("#search_input_box").hide();
        $("#search_1").on("click", function () {
            $("#search_input_box").slideToggle();
            $("#search_input").focus();
        });
        $(document).ready(function() {
          $('select').niceSelect();
        });
```

Доступ до бази даних відбувається через систему Eloquent, що вбудована у фреймворк Laravel. Eloquent дозволяє доступ до бази даних через концепт моделей, у які перетворюються таблиці БД з якими потрібно працювати розробнику. Прикладом моделі є модель User.

Лістинг *4 Код моделі User*

```
class User extends Authenticatable
{
    use HasApiTokens, HasFactory, Notifiable;

    /**
     * The attributes that are mass assignable.
     *
     * @var string[]
     */
    protected $fillable = [
        'name',
        'email',
```

```
            'password',
        ];

        /**
         * The attributes that should be hidden for
serialization.
         *
         * @var array
         */
        protected $hidden = [
            'password',
            'remember_token',
        ];

        /**
         * The attributes that should be cast.
         *
         * @var array
         */
        protected $casts = [
            'email_verified_at' => 'datetime',
        ];
    }
```

Із цього коду можна побачити які поля має таблиця Users:

- name;
- email;
- password;
- remember_token;
- email_verified_at.

Зважаючи на те, що розробка ведеться за принципами DDD, доступ до моделей досягається через ланцюг контроллер-сервіс-репозиторій, тобто обробка даних не проводиться прямо у контроллері при отриманні запиту, а передається у сервіс, що належить окремій моделі.

Лістинг *5 Частина коду сервісу UserService*

```
class UserService extends UserServiceInterface
{
    /**
     * @var UserRepositoryInterface
     */
    private $userRepository;

    function getUser(int $userId) {
        return
$this->userRepository->getUser($userId);
}
}
```

3.4 Висновки з розділу 3

Курс розробки програмного забезпечення почався з розробки архітектури та вибору стороннього програмного забезпечення, що необхідно для використання у розробці. Для створення концепції архітектури та її подальшої втілення в програмному забезпеченні були обрані перевірені принципи та структури, які підвищують шанс створення успішного проекту.

РОЗДІЛ 4 ДОСЛІДЖЕННЯ РЕЗУЛЬТАТІВ РОБОТИ КОМП'ЮТЕРНОЇ СИСТЕМИ З ВИКОРИСТАННЯМ ПОШУКОВОГО ДВИГУНА ELASTICSEARCH

4.1 Результати використання пошукового двигуна Elasticsearch для роботи з базою даних

Дослідницькі зусилля в області швидкості роботи пошукових двигунів зосереджені на максимізації кількості позитивних ідентифікацій результатів при мінімізації часу від початку запиту до отримання результату. Одним із існуючих рішень проблеми швидкості запиту є кластерізація даних, що розбиває базу даних на окремі частини і дозволяє одночасний пошук інформації по всім кластерам. Це рішення має найбільший вплив на швидкість роботи із великими обсягами даних, якщо база даних не займає багато місця на жорсткому диску або сервері система кластерів не несе помітного для користувача зменшення часу відповіді.

Набір даних для тестування — це статті за різною тематикою, від новин у світі технологій до рецензій на кінофільми. Ці дані відрізняються за контекстом, кількістю слів, стилем написання та авторами. Вбудований в розроблений програмний застосунок пошук використовує ці критерії для покращення точності відповіді користувачу. Якщо розглядати набір даних як дані для тестування то важливо мати велику різноманітність тексту для того, щоб знайти найбільшу можливу кількість наявних помилок у результатах пошуку.

4.2 Покращення швидкості пошуку в базі даних

З великою кількістю даних, які можуть бути згенеровані під час експерименту, проблемним місцем у більшості конвеєрів аналізу даних є кількість обчислювального часу, необхідного для обробки даних. Два фактори, які мо-

жуть вплинути на час пошуку в базі даних, – це кількість критеріїв, які потрібно шукати, і кількість окремих результатів, які оцінюються за кластер. Оскільки толерантність до різниці по критерію для пошуку в базі даних знижується у вищевказаних методах пошуку, кількість результатів, які знаходяться у масиві відповідей, зменшується [16-17].

Щоб оцінити ефективність різних рівнів точності запитів, ми виміряли середній час, необхідний для кожного пошуку в базі даних, щоб визначити, чи можна отримати значне скорочення часу пошуку. Для набору даних пошук статтей відбувався з великим вікном допуску точності за критеріями. Цей пошук мав найдовший час виконання — 154,62 секунди. Велика кількість результатів пошуку була пов’язана з нездатністю розрізняти статті, внаслідок чого майже кожен окремий блок тексту лише віддалено нагадував про сутність запиту. На відміну від цього пошуки, виконані з використанням більш точних критеріїв або заголовку файлу, шукалися лише за визначеними користувачем рамками для вихідних даних, що значно зменшило кількість пошуків, виконаних у кожному кластері.

4.3 Розгляд отриманої інформації

Elasticsearch має програмні засоби, які здатні призначати інформативні теги до статтей і використовувати їх для підвищення точності результатів. Оскільки ці інструменти працюють на основі даних після отримання, вони мають доступ до інших критеріїв отриманих даних. Додаткові точки даних збільшують кількість підходящих елементів, які можна ідентифікувати, та підвищують впевненість у тому, що виявлені статті належать темі, і покращують точність.

Фільтрація після пошуку допомогла покращити кількість позитивних результатів, які можуть бути отримані за запитом, проте було дві помітні переваги пошуку в базі даних, які були обмежені вузькими допусками за критеріями. По-перше, масовий обмежений пошук у базі даних мав перевагу значного

скорочення обчислювального часу для виконання пошуку. По-друге, обмежений пошук у базі даних дозволив шукати дані за кількома критеріями, що відрізнялися лише на кілька відсотків. Через унікальні переваги кожного методу, найкращий метод для використання має визначатися наявністю обчислювальних ресурсів і складністю аналізованого зразка [18-22].

Масова точність за критеріями, що були задані користувачем при запиті, є лише однією відмінною ознакою, яка дозволяє розрізняти правильні та неправильні результати. Інші характеристики, такі як повнота відповіді запиту, можуть бути використані для розрізнення неправильних результатів, і при використанні в поєднанні з точністю вони повинні призвести до найкращих результатів.

Типовий життєвий цикл оцінки ефективності починається з визначення досліджуваної системи та цілей дослідження разом із специфікацією показників ефективності, які слід враховувати. У цій роботі ми зосередимося на багаторівневих архітектурах, які є типовими для сучасних веб-сайтів електронної комерції. Основними компонентами, які слід враховувати при оцінці продуктивності, є клієнт користувача, мережа, що з'єднує клієнтів і сервери (як правило, Інтернет), веб-сервер, сервер веб-додатків, система управління БД, а також мережа, що з'єднує веб-сервер, сервер веб-додатків і систему БД, якщо ці системні компоненти не знаходяться на одній машині.

Перші показники продуктивності, які цікавлять, це час відповіді, який визначається як час між поданням запиту від користувача та завершенням відповіді (тобто момент часу, коли клієнт отримав відповідь). Цей показник є критичним фактором при оцінці веб-сервісів. З емпіричних досліджень відомо, що якщо час відповіді перевищує 8 секунд, користувачі, як правило, припиняють сеанс і залишають сайт. Такий прив’язаний набір для показника продуктивності називається «угодою про рівень обслуговування». Другим важливим критерієм є пропускна здатність, яка визначається як кількість запитів, які система може виконати за одиницю часу. Типове питання планування пропу-

скної здатності веб-сервісів полягає в тому, щоб визначити максимальну пропускну здатність, якої може досягти система при заданому рівні обслуговування [23].

Обидва показники пов'язані зі «швидкістю», з якою запити можуть бути виконані, як з точки зору користувачів (час відповіді), так і з точки зору системи (пропускна здатність). Інші критерії включають масштабованість, яка визначається як здатність системи впоратися зі збільшенням навантаження, і доступність, яка визначається як відношення часу, протягом якого система обслуговувала запити, до загального часу (часи підвищення і зупинки).

Додаток, який використовується для перевірки швидкості, масштабованості та надійності різних компонентів динамічних веб-сайтів, є власне розроблений веб-сайт, зроблений у вигляді блогу. Мета веб-сайту — пропонувати результати у вигляді статей на різноманітні теми в режимі реального часу онлайн через HTTP. Хоча ці послуги надаються клієнтам безкоштовно, архітектура сайту також є типовою для комерційних послуг і тому служить репрезентативним тестовим стендом. Причина вибору сайту такого типу полягає в тому, що для такого сайту можна спостерігати систему пошуку під реальним важким навантаженням, а саме подавати кілька тисяч запитів до бази даних одночасно як під час перегляду в мережі Інтернет, так і за допомогою сторонніх програм для тестування пропускної здатності.

Далі ми коротко обговоримо різні компоненти цього тестового майданчика.

4.4 Тестування продукту

Щоб стандартизувати тест і застосувати однаковий баланс для реалізації веб-сайту необхідно було уважно ознайомитися з файлами журналів попередніх змагань, де навантаження на сервер було значним. Після аналізу поведінку середнього користувача можна звести до шести різних типів інформації, яку запитує користувач:

- Знайдіть ім’я автора в стандартному інтерфейсі (наприклад, усі автори, чиє ім’я «Майкл»).
- Коли потрібну особу знайдено в отриманому списку, запитується пошук початкового номера цієї особи (наприклад, Початковий номер = 1201).
- На третьому кроці відвідувач ближче подивиться на результати вибраного автора та перейде на сторінку з детальними результатами, де відображаються середня кількість постів та результати.
- Наступний запит буде подано за допомогою форми детального пошуку для всіх статей з певної теми (наприклад, Topic = TECH).
- Ще один момент, який представляє зосереджений інтерес, полягає в тому, щоб подивитися на перші кілька позицій за рейтингом (наприклад, Position < 100).
- Нарешті, відвідувач може зацікавитися найпопулярнішими статтями і може запитати список усіх статей, які належать до певної групи за кількістю слів у статті (наприклад, Words < 3500).

Між запитами середній час обдумування користувача (тобто час між відображенням запитаних результатів і видачею нового запиту) було встановлено на дві секунди.

4.5 Поступові тести

У першому тестовому середовищі метою було випробувати програми веб-сервера в послідовному тесті, тобто запитувати один і той самий запит 100 разів один за одним, щоб побачити, як ці три програми порівнюються один з одним. Крім того, розмір бази даних був змінений, щоб отримати уявлення про кількість часу, який витрачається на запит даних та обробку даних. Веб-сервер (Apache 1.3.22, Tomcat 4.0.1, Cocoon 2.0.1) і СУБД (Mysql 3.23.47) були встановлені на одній системі (Intel Pentium II, процесор 399 МГц, 256 МБ RAM) під керуванням SUSE Linux 7.3. Запити були подані з іншої машини, яка виступала в якості клієнта.

Щоб уникнути ефектів кешування, рядок пошуку в першому запиті користувача (пошук імені) вибирається випадковим чином з пулу різних рядків, щоб представляти більш реалістичне навантаження.

Результати перших послідовних тестів були не зовсім несподівані [24-27]. Запит за допомогою Elasticsearch мав бути найшвидшим за часом відгуку, PHP — другим. Як і очікувалося, програма за допомогою Elasticsearch найкраще вела себе під час стресу, продуктивність істотно не знижувалася зі збільшенням розміру бази даних. Однак PHP, який мав хороший старт, збільшив загальний час відповіді швидше, ніж зростання бази даних, що є показником поганої масштабованості.

Щоб зібрати більше інформації про поведінку з ще більшим навантаженням проведено кілька поглиблених тестів, щоб визначити ємність системи, тобто з'ясувати, яке навантаження можуть витримувати три програми та як вони працюють при зростаючій кількості запитів. Тести проведені на однакових комп'ютерах (тобто з ідентичним апаратним і програмним забезпеченням).

У першій серії тестів httperf використовувався для порівняння системи, а також для оцінки та порівняння часу відповіді трьох різних методів сервера додатків.

Таблиця 1

Пошуковик	*Середній час відповіді*
Elasticsearch	5,6 сек
Пошуковик на мові PHP	9,4 сек
XSP	11,3 сек

Як і передбачалося, пошуки з Elasticsearch показують безперечно найкращу продуктивність порівняно з іншими технологіями, пошук за допомогою

вбудованого пошуковика мови PHP займає друге місце з майже подвійним часом відповіді (деталі: кількість одночасних сеансів збільшено на 10. Тести проводилися без перерв між одиничними тестами щоб не надавати полегшення серверу, тобто запуск окремого тесту, наприклад, із 200 одночасними сеансами призведе до іншого результату, як показано тут). Найгіршу продуктивність відстежено під час тестів XSP, на рівні близько 150 одночасних сеансів система зазнала збою і залишила сервер у неконтрольованому стані. Після того, як тест було повторено кілька разів із незмінним результатом, я вирішив залишити XSP для подальших тестів через його неконкурентоспроможність щодо продуктивності під великим навантаженням.

У другій серії тестів використовувався http_load, який запускає кілька HTTP-вибірок паралельно, щоб перевірити пропускну здатність веб-сервера. Однак, на відміну від більшості таких тестових клієнтів, він виконується в одному процесі, тому не загрожує клієнтській машині. Отримані результати були подібними до результатів, отриманих за допомогою httperf. Крім того, проведено кілька контрольних послідовностей з різними наборами обладнання. Машина, на якій працює tomcat і модуль apache – PHP, залишилися незмінними, головною метою було з'ясувати, якою мірою продуктивність бази даних вплине на загальний час відповіді. Протестовано три різні апаратні установки:

- Веб-сервер повинен отримати таблиці результатів бази даних з однієї машини mysql зі слабким обладнанням (Pentium II 200 МГц, 256 Мб ОЗУ).
- Веб-сервер повинен отримати таблиці результатів бази даних з однієї машини mysql з потужним апаратним забезпеченням (Pentium IV 2000 МГц, 512 Мб RAM).
- Веб-сервер повинен отримати таблиці результатів бази даних з трьох різних машин (Pentium II 200 МГц, 256 Мб ОЗУ; Pentium III 500 МГц, 256 Мб ОЗУ; Pentium IV 2000 МГц, 512 Мб ОЗУ) і керувати підключенням до всіх трьох машин mysql.

Тест із використанням http_load проводився кілька разів під час кожного налаштування обладнання. Цей тест використовувався для перевірки доступності підрахунку кількості спостережуваних помилок при обслуговуванні запитів.

Таблиця 2

Пошуковик	*Час*	*Помилок*
Elasticsearch	3,3 сек	0%
Пошуковик на мові PHP	5,9 сек	23%

Знову ж таки, система Elastisearch показала найкращу продуктивність. Це вказує на використання потужного обладнання для машини баз даних, а 3db представляє результати трьох апаратних баз даних. Реалізація додатку Elasticsearch взагалі не викликає жодних помилок (і, здається, не залежить від швидкості бази даних) до приблизно 700 запитів в секунду, досягаючи рівня запитів в секунду, коли програма стає недоступною (потрібен перезапуск сервера) . Однак PHP починає видавати помилки з самого початку тесту, частота помилок збільшується із запитами в секунду і залежить від швидкості з якою обслуговуються таблиці результатів бази даних.

4.6 Висновки з розділу 4

Розглянуті методи кластеризації добре підходять для попередньої обробки великих даних з метою класифікації несанкціонованих користувачів Інтернету на основі непрямих даних та поведінкових характеристик, отриманих в результаті аналізу мобільного трафіку та взаємодії користувачів з мобільними пристроїв. Результати обробки даних можна проаналізувати за допомогою різних програмних засобів, які найбільш підходять для вирішення певної проблеми.

Система Elasticsearch в цілому є найбільш підходящою для завдань повнотекстового пошуку та візуалізації даних (безкоштовний, з відкритим кодом, простий інтерфейс, веб-обробка даних).

Пропонується використовувати можливості Elasticsearch для організації інтерфейсу для роботи з великими даними (пошук та візуалізація), тоді як для попередньої обробки та завдань сегментації даних та перевірки користувачів на основі непрямих даних може використовуватися модель MapReduce, і зокрема новий ПМР-транзитивний підхід.

Запропонована повнотекстова система пошуку та візуалізації може покрити попит на сучасну інноваційну платформу перевірки користувачів програмного забезпечення, яка може виконувати завдання деанонімізації користувачів та збільшувати залучення користувачів до економічної онлайн-моделі. Доступні методи авторизації та ідентифікації користувачів не справляються із завданням отримання актуальної та достовірної інформації про користувачів, а методи авторизації з використанням таких ключових параметрів, як буквено-цифровий логін або електронна адреса, недостатні.

Різні галузі економіки, такі як банківська справа, електронна комерція та пов'язані з ними Інтернет-послуги, стикаються з проблемами шахрайства та неправдивих даних.

Впровадження запропонованої системи може зменшити та мінімізувати ризики реальних секторів економіки, що працюють з анонімними користувачами послуг, що також матиме сприятливий вплив на інформаційну безпеку та державу в цілому.

Обробка великих обсягів даних, що стосуються користувача, дозволить ідентифікувати користувача якомога точніше на основі непрямих даних, отриманих під час аналізу поведінки в мережі, трафіку та інших дій користувача.

ВИСНОВКИ

1. Доведена актуальність теми швидкості роботи пошукових систем з базами даних та проблемами якості повернених результатів.
2. Розроблене програмне забезпечення для роботи з пошуковим двигуном Elasticsearch та пошуковою системою написаною на мові PHP.
3. Використано фреймворк Laravel, бібліотеку jQuery, методологію DDD та архітектуру типу MVC для покращення якості коду програмного забезпечення.
4. Проведено дослідження з порівняння швидкості роботи з даними між пошуковою системою на мові PHP та пошуковим двигуном Elasticsearch.
5. Доведені переваги у використанні пошукового двигуна Elasticsearch порівняно зі звичайною мовою PHP та зроблені відповідні рекомендації.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Ваганов Л. В., Безверхий А.І. Дослідження пошукових систем на прикладі Elasticsearch: Матеріали І Всеукраїнської науково-практичної конференції здобувачів вищої освіти, аспірантів та молодих вчених «Актуальні питання сталого науково-технічного та соціально-економічного розвитку регіонів України» 19-21 жовтня 2021 р. Запоріжжя : ЗНУ, 2021. С. 321.
2. Ваганов Л. В., магістрант, Безверхий А. І., доцент — науковий керівник. Дослідження пошукових систем на прикладі Elasticsearch. *Молода наука-2021* : зб. наук. праць студентів, аспірантів і молодих вчених. Запоріжжя : ЗНУ, 2021. Т. 5. С. 82.
3. Shmueli G. Research Dilemmas with Behavioral Big Data. Big Data, 2017, P. 98–119.
4. Jank W, Shmueli G. Modeling online auctions. Hoboken, NJ: John Wiley and Sons, 2010.
5. Bender S, Jarmin R, Kreuter F, Lane J. Privacy and confidentiality. In: Big Data and Social Science Research: Theory and Practical Approaches. CRC Press, 2016.
6. Narayan A, Shmatikov V. Robust de-anonymization of large sparse datasets. In: Proceedings of 29th IEEE Symposium on Security and Privacy, 2008.
7. Kurt Rohloff and Richard E. Schantz. High-performance, massively scalable distributed systems using the MapReduce software framework: the SHARD triple-store. In Programming Support Innovations for Emerging Distributed Applications (PSI EtA '10). ACM, New York, NY, USA, Article 4, 2010.
8. Narayanan Venkateswaran, Suvamoy Changder Simplified Data Partitioning in a Consistent Hashing Based Sharding Implementation. Proc. of the 2017 IEEE Region 10 Conference (TENCON), Malaysia, November 5-8, 2017.
9. Costa, Caio H. et al. “Sharding by Hash Partitioning - A Database Scalability Pattern to Achieve Evenly Sharded Database Clusters.”, 2015.

10. E. Kokoris-Kogias, P. Jovanovic, L. Gasser, N. Gailly, E. Syta and B. Ford, "OmniLedger: A Secure, Scale-Out, Decentralized Ledger via Sharding," 2018 IEEE Symposium on Security and Privacy (SP), San Francisco, CA, 2018, P. 583-598.
11. Shmueli G, Koppius O. Predictive analytics in information systems research. MIS Q. 2011, P. 553–572.
12. Cleverdon C., The Cranfield tests on index language devices. In: Jones KS and Willett P, Eds. Readings in Information Retrieval, Morgan Kauffman, San Francisco, 1997, P. 47–59.
13. Salton G and Lesk ME, Computer evaluation of indexing and text processing. In: Jones KS and Willett P, Eds. Readings in Information Retrieval, Morgan Kauffman, San Francisco, 1997, P. 60–84.
14. Jain, Raj: The Art of Computer Systems Performance Analysis, John Wiley and Sons Inc, 1991.
15. Krause, Jörg: PHP, Webserver Programmierung unter Windows und Linux, Carl Hanser Verlag, Munich, 2000.
16. Joarder Kamal, Manzur Murshed, Rajkumar Buyya, Workload-aware incremental repartitioning of shared-nothing distributed databases for scalable OLTP applications, Future Generation Computer Systems, Volume 56, 2016, P. 421-435.
17. Hafeezul Rahman Mohammad, Keyang Xu, Jamie Callan, and J. Shane Culpepper. 2018. Dynamic Shard Cut-off Prediction for Selective Search. In The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval (SIGIR '18). ACM, New York, NY, USA, 85-94.
18. Zhuyun Dai, Chenyan Xiong, and Jamie Callan. 2016. Query-Biased Partitioning for Selective Search. In Proceedings of the 25th ACM International on Conference on Information and Knowledge Management (CIKM '16). ACM, New York, NY, USA, 1119-1128.
19. Anagha Kulkarni and Jamie Callan. 2015. Selective Search: Efficient and Effective Search of Large Textual Collections, 2015, 33 pages.
20. Anagha Kulkarni, Almer S. Tigelaar, Djoerd Hiemstra, and Jamie Callan. Shard ranking and cutoff estimation for topically partitioned collections. In Proceedings

of the 21st ACM international conference on Information and knowledge management, 2012, P. 555-564.

21. Dhulavvagol P.M., Totad S.G., Sourabh S. Performance Analysis of Job Scheduling Algorithms on Hadoop Multi-cluster Environment. In: Sridhar V., Padma M., Rao K. (eds) Emerging Research in Electronics, Computer Science and Technology. Lecture Notes in Electrical Engineering, vol 545, 2019.
22. Niranjan C Kundur1, Praveen M Dhulavvagol, Prasad." Recommendation System Based on Content Filtering for Specific Commodity" International Journal of Latest Technology in Engineering, Management & Applied Science (IJLTEMAS)Volume V, Issue VII, 2016, P. 2278-2540.
23. Menasce, D., Almeide, V.: Capacity Planning for Web Services, Prentice Hall PTR, 2002.
24. Muthukaruppan Annamalai, Kaushik Ravichandran, Harish Srinivas, Igor Zinkovsky, Luning Pan, Tony Savor, David Nagle, and Michael Stumm. Sharding the shards: managing datastore locality at scale with Akkio. In Proceedings of the 12th USENIX conference on Operating Systems Design and Implementation (OSDI'18). USENIX Association, Berkeley, CA, USA, 2018, P. 445-460.
25. S G Totad, R B Geeta, CR Prasanna, NK Santhosh, PV Reddy." Scaling data mining algorithms to large and distributed datasets" International Journal of Database Management Systems (IJDMS), Vol.2, No.4, 2010.
26. Y. Liu, Y. Wang and Y. Jin, "Research on the improvement of MongoDB Auto-Sharding in cloud environment," 2012 7th International Conference on Computer Science & Education (ICCSE), Melbourne, VIC, 2012, P. 851-854.
27. Yashaswini Joshi, Geeta R.B, Shashikumar G. Totad, "Mobile Agent based Frequent Pattern Mining for Distributed Databases", International Conference on Intelligent Computing and Communication on 2 to 4th August 2-4, 2017 at MAEER's MIT College of Engineering, Pune; Intelligent Computing and Information and Communication, 2018, P. 77-85.

Декларація

академічної доброчесності

здобувача ступеня вищої освіти ЗНУ

Я, Ваганов Леонід Володимирович, студент 2 курсу, форми навчання денної, Інженерного навчально-наукового інституту, спеціальність 121 Інженерія програмного забезпечення, адреса електронної пошти se216-01@stu.zsea.edu.ua, — підтверджую, що написана мною кваліфікаційна робота на тему «**Дослідження пошукових систем на прикладі Elasticsearch**» відповідає вимогам академічної доброчесності та не містить порушень, що визначені у ст. 42 Закону України «Про освіту», зі змістом яких ознайомлений.

- заявляю, що надана мною для перевірки електронна версія роботи є ідентичною її друкованій версії;

згоден на перевірку моєї роботи на відповідальність критеріям академічної доброчесності у будь-який спосіб, у тому числі за допомогою інтернет-систем, а також на архівування моєї роботи в базі даних цієї системи.

Дата 30.11.2021 Підпис _____ Ваганов Леонід Володимирович

(студент)

Дата 30.11.2021 Підпис _____ Безверхий Анатолій Ігорович

(науковий керівник)