МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ХАРЧОВИХ ТЕХНОЛОГІЙ

Інститут (факультет) __Автоматизації і комп'ютерних систем_____
Кафедра _Інформаційних технологій, штучного інтелекту і кібербезпеки_

«До захисту в ЕК»	**«До захисту допущено»**
Директор інституту(декан факультету)	Завідувач кафедри
(підпис) __Андрій ФОРСЮК___ (ім'я та прізвище)	(підпис) __Сергій ГРИБКОВ__ (ім'я та прізвище)
«_13_» ____грудня_____ 2024 р.	«_13_» ____грудня_____ 2024 р.

КВАЛІФІКАЦІЙНА РОБОТА
НА ЗДОБУТТЯ ОСВІТНЬОГО СТУПЕНЯ МАГІСТРА

зі спеціальності_____122 Комп'ютерні науки_____
(код і назва спеціальності)
освітньо-професійної програми _Управління інформацією та аналітика даних____
_____
на тему: Дослідження та створення аналітичної системи результатів вступної кампанії у вищі навчальні заклади України на основі відкритих даних ЄДБО____

Виконав: здобувач __2__ курсу, групи _КН-2-4М_

_____Карпишинець Вадим Васильович_____ _____
(прізвище, ім'я, по батькові повністю) (підпис)

Керівник ___Струзік Владислав Анатолійович__ _____
(прізвище , ім'я та по батькові повністю) (підпис)

Консультанти __Сергій ГРИБКОВ_ _____
(ім'я та прізвище) (підпис)

_____ _____
(ім'я та прізвище) (підпис)

_____ _____
(ім'я та прізвище) (підпис)

Рецензент Володимир ВРУЖИНУН _____
(ім'я та прізвище) (підпис)

Я як здобувач(ка) Національного університету харчових технологій розумію і підтримую політику університету з академічної доброчесності. Я не надавав(-ла) і не одержував(-ла) недозволеної допомоги під час підготовки цієї роботи. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело.

Здобувач_____
(підпис)

Київ — 2024 р.

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ХАРЧОВИХ ТЕХНОЛОГІЙ

Інститут (факультет) _Автоматизації і комп'ютерних систем_____
Кафедра інформаційних технологій, штучного інтелекту і кібербезпеки_____

Освітній ступінь _____Магістр_____

Спеціальність _____122 «Комп'ютерні науки»_____
(код і назва)

Освітньо-професійна програма _Управління інформацією та аналітика_____
__даних_____
(назва)

ЗАТВЕРДЖУЮ
Завідувач
кафедри Інформаційних технологій,
штучного інтелекту і кібербезпеки_

_____Грибков С.В
«_07_» _____жовтня_____ 2024 року

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ РОБОТУ ЗДОБУВАЧА

Карпишинця Вадима Васильовича
(прізвище, ім'я, по батькові)

1. Тема роботи __Дослідження та створення аналітичної системи результатів вступної кампанії у вищі навчальні заклади України на основі відкритих даних ЄДБО_____
керівник роботи _Струзік Владислав Анатолійович, канд. техн. наук_____,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом закладу вищої освіти від «07» _жовтня_ 2024 р. № _884-кс__
2. Строк подання здобувачем роботи: _____
3. Вихідні дані до роботи:
1. Відкриті дані Єдиної державної електронної бази з питань освіти
2. Відкриті інформаційні джерела пометодам парсингу та скрапінгу вебсайтів
3. Відкриті інформаційні джерела по методам ефективного збереження даних в базах даних
4. Відкриті інформаційні джерела по видам програмного забезпечення для аналізу великих об'ємів даних та побудова дашбордів
4. Зміст пояснювальної записки (перелік питань, які потрібно розробити):
1. Аналіз сучасного стану вищої освіти в Україні.
2.Розробка комплексного підходу для аналізу даних з урахуванням специфіки вступу до ВНЗ

3. Розробка методики збору даних, обробки та їх аналізу з різних джерел ЄДБО та "АбітПошук".
4. Реалізація аналітичної системи.
5. Оцінка ефективності системи.

6. Перелік графічного матеріалу:

1. Результати перевірки повноти даних
2. Порівняння даних з різних джерел
3. Дашборд

6. Консультанти розділів роботи:

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1.	Струзік В. А., асистент, кандидат технічних наук	22.09.2024	22.09.2024
2.	Струзік В. А., асистент, кандидат технічних наук	25.09.2024	25.09.2024
3.	Струзік В. А., асистент, кандидат технічних наук	10.10.2024	10.10.2024
4.	Струзік В. А., асистент, кандидат технічних наук	25.10.2024	25.10.2024
5.	Струзік В. А., асистент, кандидат технічних наук	15.11.2024	15.11.2024

7. Дата видачі завдання: 7 жовтня 2024 року

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів виконання кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Видача завдання	22.09.2024	Виконав
2	Виконання пошуку матеріалів	09.10.2024	Виконав
3	Оформлення розділу 1	12.10.2024	Виконав
4	Оформлення розділу 2	18.10.2024	Виконав
5	Оформлення розділу 3	25.10.2024	Виконав
6	Розробка системи	01.11.2024	Виконав
7	Оформлення розділу 4	25.11.2024	Виконав
8	Оформлення автореферату	01.12.2024	Виконав
9	Оформлення презентації	05.12.2024	Виконав

Здобувач _____ (підпис) Карпишинець В.В. (прізвище та ініціали)

Керівник роботи _____ (підпис) Струзік В. А (прізвище та ініціали)

АНОТАЦІЯ

Карпишинець В.В.. Кваліфікаційна робота на тему «Дослідження та створення аналітичної системи результатів вступної кампанії у вищі навчальні заклади України на основі відкритих даних ЄДБО» складається із 101 сторінок, включає 7 таблиць, 8 ілюстрацій, список із 36 використаних джерел, 6 додатків.

У роботі розглянуто проблему відсутності ефективних систем збору, обробки та аналізу даних вступних кампаній, орієнтованих на потреби закладів вищої освіти України. Обґрунтовано методологію розробки аналітичної системи, яка включає автоматизований збір даних, структурування та аналіз за допомогою сучасних інструментів, таких як HTML-скрапінг, база даних ClickHouse та інтерактивні дашборди Grafana. Визначено основні етапи створення системи: від розробки інфраструктури для збору даних до візуалізації та інтерпретації результатів.

Ключові слова: ВСТУПНІ КАМПАНІЇ, АНАЛІТИЧНА СИСТЕМА, ЄДБО, ВІЗУАЛІЗАЦІЯ ДАНИХ, КОНКУРЕНТОСПРОМОЖНІСТЬ УНІВЕРСИТЕТІВ, РОЗПОДІЛ ЗАЯВ, БАЗА ДАНИХ CLICKHOUSE, DASHBOARDS, GRAFANA, МАРКЕТИНГОВІ СТРАТЕГІЇ.

SUMMARY

Karpyshynets V.V..The qualification work titled "Research and Development of an Analytical System for University Admission Campaign Results in Ukraine Based on Open EDBO Data" consists of 101 pages, includes 7 tables, 8 illustrations, a list of references with 36 entries, and 6 appendices.

The work addresses the issue of the lack of effective systems for collecting, processing, and analyzing data from admission campaigns, tailored to the needs of higher education institutions in Ukraine. The methodology for developing an analytical system is substantiated, which includes automated data collection, structuring, and analysis using modern tools such as HTML scraping, ClickHouse database, and Grafana interactive dashboards. The main stages of system development are outlined, ranging from infrastructure creation for data collection to visualization and interpretation of results.

Keywords: ADMISSION CAMPAIGNS, ANALYTICAL SYSTEM, EDBO, DATA VISUALIZATION, UNIVERSITY COMPETITIVENESS, APPLICATION DISTRIBUTION, CLICKHOUSE DATABASE, DASHBOARDS, GRAFANA, MARKETING STRATEGIES

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

Скорочення	**Розшифрування**
ЄДБО	Єдина державна електронна база з питань освіти
API	Application Programming Interface
DOM	Document Object Model
СУБД	Система управління базами даних
ВНЗ	Вищі навчальні заклади

ВСТУП

Актуальність теми. У сучасних умовах глобалізації та цифрової трансформації освіти університети стикаються з жорсткою конкуренцією як на національному, так і на міжнародному рівнях. Інтеграція України до Європейського Союзу та реформи освітньої системи, які передбачають скорочення кількості закладів вищої освіти, підкреслюють необхідність підвищення конкурентоспроможності університетів. Для цього навчальні заклади повинні ефективно використовувати дані вступних кампаній, аналізувати тенденції попиту на спеціальності та приймати обґрунтовані стратегічні рішення. Однак існуючі інформаційні системи, такі як ЄДБО та "АбітПошук", орієнтовані переважно на потреби абітурієнтів і не забезпечують університетам необхідних аналітичних інструментів для глибокого аналізу. Це визначає актуальність розробки системи, яка б дозволила університетам використовувати дані вступних кампаній для вдосконалення освітніх програм, маркетингових стратегій та управління ресурсами.

Зв’язок роботи з науковими програмами, планами, темами. Дослідження виконане в межах напряму наукової роботи кафедри Інформаційних технологій, штучного інтелекту та кібербезпеки. Тема роботи відповідає пріоритетам університету щодо підвищення якості освітніх послуг та інтеграції в європейський освітній простір.

Мета дослідження. Метою роботи є розробка та впровадження аналітичної системи для збору, обробки та аналізу даних вступних кампаній, яка забезпечить університети ефективними інструментами для прийняття стратегічних рішень.

Завдання дослідження:

- провести аналіз існуючих платформ для збору та обробки даних вступних кампаній;
- дослідити та визначити підходи аналізу та інтерпретації результатів вступної кампанії у ВНЗ;
- розробити проєкт системи збору, обробки та аналізу даних;

- реалізувати проєкт для автоматизованого збору даних із відкритих джерел, що буде підтримувати зберігання та обробки великих обсягів освітніх даних;
- створити інтерактивні дашборди для візуалізації результатів аналізу;
- оцінити ефективність розробленої системи у реальних умовах.

Об’єктом дослідження є процес вступу в заклади вищої освіти України.

Предметом дослідження є методи та засоби аналізу статистичних даних результатів вступу підчас вступних кампаній у ВНЗ.

Методи дослідження.

У процесі дослідження методів аналізу використано:

- загальнонаукові методи дослідження (аналіз та синтез, індукція та дедукція, абстрагування);
- емпіричні методи дослідження (спостереження, порівняння).

Наукова новизна одержаних результатів. Запропоновано комплексний підхід для аналізу результатів вступної кампанії у ВНЗ, який забезпечує сегментування даних за пріоритетами вступників і здійснює прогнозування тенденцій попиту на спеціальності.

Практичне значення одержаних результатів. Система може бути використана університетами для аналізу результатів вступних кампаній. Аналіз може бути використаний для вдосконалення освітніх програм, планування маркетингових стратегій та підвищення їх конкурентоспроможності. Застосування системи сприятиме покращенню якості управління освітніми процесами та відповідності сучасним вимогам ринку праці.

РОЗДІЛ 1. АНАЛІТИЧНИЙ ОГЛЯД ЛІТЕРАТУРИ

В сьогоднішніх економічних умовах університети можуть успішно працювати та зберігати конкурентоспроможність лише за умови постійного розвитку інтелектуального капіталу в межах глобальної інституційної екосистеми.

У сучасному глобалізованому світі університети стали ключовими центрами для створення, збереження та передачі знань, а також формування і концентрації інтелектуального потенціалу [1].

У розвинених країнах вони, зокрема дослідницькі університети, відіграють провідну роль не лише у створенні нових знань, але й у розробці інноваційних проривів. Світова практика демонструє, що вища освіта разом із інноваціями є основними двигунами глобальної конкурентоспроможності, про що свідчить тісний зв’язок між місцем країн у рейтингах глобальної конкурентоспроможності та їх позиціями у рейтингах інноваційності та якості систем вищої освіти.

Як важливі складові національних інноваційних систем університети перебувають у центрі уваги наукових досліджень, оцінок, моніторингів та рейтингування.

Для розвитку, модернізації країни та впровадження економічних реформ, у яких освіті відводиться ключова роль як рушійній силі змін, необхідність відновлення конкурентоспроможності університетів України на міжнародній арені відіграє ключову роль.

Україна була відомою своїм потужним інженерно-технічним потенціалом і спеціалізацією у високотехнологічному машинобудуванні. Освічені та кваліфіковані спеціалісти є головним ресурсом і майбутнім нашої країни.

В умовах реформування освітньої системи України, зумовленого інтеграцією до Європейського Союзу, постає завдання скорочення кількості закладів вищої освіти до стандартів, прийнятих у країнах ЄС.

На сьогодні в Україні функціонує 151 державний і 90 приватних університетів, але очікується, що до 2030 року їхня кількість буде зменшена до приблизно до сотні [2]. Основним критерієм для збереження університету стане

якість надання освітніх послуг і привабливість для абітурієнтів, зокрема кількість здобувачів, які вступають на бюджетну форму навчання. Заклади, де менше третини вступників вступають на бюджет, розглядатимуться як кандидати на оптимізацію або закриття. Така ситуація ставить усі університети в умови жорсткої конкуренції, адже залишаться існувати лише ті, які не тільки залучають найбільшу кількість талановитих здобувачів, а й забезпечують високу якість освіти. Як наслідок потрібно активніше працювати над своїми маркетинговими стратегіями, модернізацією освітніх програм і впровадженням інновацій, щоб відповідати очікуванням держави, абітурієнтів та європейських стандартів. На основі цього виникає потреба в аналізі чинників, що впливають на конкурентну спроможність вищих навчальних закладів та виявлення тенденцій під час вступу.

1.1. Вступна кампанія

Основним джерелом для дослідження факторів, що впливають на розвиток певних галузей знань та спеціальностей є вступна кампанія. Вступна кампанія — це процес набору абітурієнтів до закладів вищої освіти, який проводиться щороку у визначені терміни. Процес регулюється Міністерством освіти і науки України (МОН) і має на меті забезпечити прозорий, справедливий та ефективний відбір здобувачів на різні освітні програми.

Етапи вступної кампанії:

- реєстрація електронних кабінетів - абітурієнти створюють електронні кабінети через спеціальну платформу, куди завантажують документи, необхідні для вступу;
- подання заяв - абітурієнти можуть подавати обмежену кількість заяв до обраних закладів вищої освіти на різні спеціальності, визначаючи пріоритетність кожної заяви;
- конкурсний відбір - відбір здійснюється на основі конкурсного бала, який складається з результатів зовнішнього незалежного оцінювання (ЗНО) або національного мультипредметного тесту (НМТ), опціонально

творчого іспиту, співбесіди, середнього бала атестата та додаткових коефіцієнтів (регіонального, галузевого, сільського тощо);

- зарахування абітурієнтів - результати вступу публікуються в електронних кабінетах, і абітурієнти повинні підтвердити своє бажання навчатися шляхом подання оригіналів документів та підтвердженням вибору місця навчання до відповідного закладу.

У цьому контексті значну роль відіграє Єдина державна електронна база з питань освіти, яка забезпечує централізований збір та зберігання даних про освітні заклади та абітурієнтів в Україні [3]. Початковий етап розвитку ЄДБО був спрямований на стандартизацію та уніфікацію освітніх даних, що дозволило створити єдине інформаційне середовище для всіх учасників освітнього процесу. Це сприяло підвищенню прозорості вступних кампаній, забезпеченню відкритого доступу до інформації про освітні програми та спеціальності.

В процесі розвитку технологій та забезпеченні більш зручного пошуку інформації про заяви вступників з'явилися платформи, які використовували дані з ЄДБО для надання інформаційних послуг абітурієнтам та освітнім установам. Одним з найпопулярніших став веб-портал “АбітПошук” [4]. Ця платформа отримала широке визнання серед користувачів завдяки доступності та простоті використання.

Однак обидві платформи орієнтовані на представлення інформації абітурієнтам, а не університетам та не надають інструментів для глибокого аналізу тенденцій вступу.

1.2. Аналіз ключових чинників впливу на розвиток університетів

Ключовою ознакою успішного розвитку будь-якого навчального закладу є приріст абітурієнтів. Для цього освітні заклади працюють над розробкою маркетингових стратегій, активно працюючи над тим, щоб зрозуміти поведінку вступників ще на етапі, коли ті навчаються у школі. Для цього проводять численні заходи, орієнтовані на школярів: профорієнтаційні зустрічі, дні відкритих дверей,

презентації спеціальностей. Усе це покликано створити позитивний образ університету, щоб він став пріоритетним у виборі абітурієнта.

Водночас, для ефективності таких заходів, потрібно мати глибоке розуміння стану ринку освітніх послуг. Конкуренція між закладами освіти є значною, і кожен із них прагне випередити інших у своїй ніші. Для цього важливо аналізувати, як розподіляються заяви вступників між різними університетами, які спеціальності обирають майбутні здобувачі, які напрями є найбільш популярними. Особливу увагу слід приділяти тому, як абітурієнти визначають пріоритети в своїх заявах. Інформація про те, куди вступники подають свої заяви першим пріоритетом, а куди лише як резервний варіант, може дати університету розуміння, наскільки привабливою є його пропозиція.

Окремим важливим аспектом є аналіз альтернатив, які розглядають вступники. Для вибору напрямку розвитку спеціальностей в університеті необхідно мати розуміння, які спеціальності конкурують між собою за одного й того ж вступника. Наприклад, чи вибирає абітурієнт між інженерією та інформаційними технологіями, чи, можливо, між економікою та правом. Дослідження конкретних конкурентних конкурсних пропозицій і визначення до якої з них був зарахований абітурієнт надає інформацію про фактори, які впливають на вибір вступника тієї чи іншої конкурсної пропозиції. Така інформація дозволяє університетам не лише оцінити попит на окремі спеціальності, але й виявити слабкі місця у своїх освітніх програмах або маркетинговій стратегії.

Важливу роль відіграє розуміння ринку праці. Абітурієнти та їхні батьки часто орієнтуються на те, які галузі знань мають найбільші перспективи з точки зору працевлаштування. Університети, у свою чергу, повинні враховувати ці тренди під час планування своїх освітніх програм. Наприклад, галузі, які краще фінансуються і на які виділяється більше бюджетних місць, є важливими індикаторами для аналізу. Освітні заклади, які зможуть адаптувати свої пропозиції до потреб ринку праці, матимуть конкурентну перевагу у залученні абітурієнтів.

Вчасне коригування освітніх програм до актуальних вимог вакансій на ринку праці є чи не головним фактором, який впливає на якість надання освітніх

послуг університетом. А це, як наслідок, відзначається на підготовці висококваліфікованих спеціалістів в певній галузі, що є потужним показником якості надання освітніх послуг та безпосередньо підвищує рейтинг закладу.

Для того, щоб ефективно виконувати такі дослідження, необхідно володіти потужними інструментами аналізу. Створення аналітичної системи, яка дозволяє інтегрувати дані про вступні кампанії, аналізувати популярність спеціальностей, оцінювати конкуренцію з іншими закладами освіти і вивчати ринок праці, є логічним і необхідним кроком.

Так як, існуючі платформи здебільшого орієнтовані на потреби абітурієнтів та надають інформацію в узагальненому вигляді, вони не забезпечують можливості гнучкого налаштування параметрів аналізу, фільтрації даних за специфічними критеріями, що є критично важливим для університетів при прийнятті управлінських рішень. Відсутність візуалізації даних ускладнює процес інтерпретації інформації, не дозволяючи побачити приховані закономірності та тенденції, що в свою чергу знижує ефективність використання таких систем для стратегічного планування та прийняття рішень.

1.3. Формулювання наукової проблеми

Виходячи з вищезазначеного, стає очевидним, що існує наукова проблема, пов'язана з відсутністю підходів до аналізу даних результатів вступних кампаній, орієнтованих на потреби університетів в Україні.

Розробка таких підходів передбачає аналіз джерел відкритих даних для подальшого їх збору та обробки, що при прийнятті управлінських рішень дозволить врахувати особливості ринку освітніх послуг та поведінки абітурієнтів.

Впровадження таких підходів вимагає наявності якісних даних та відповідної інфраструктури для їх збору та збереження. Важливим є питання інтеграції різних джерел даних, таких як відкриті дані ЄДБО про вступну кампанію і веб-портал “АбітПошук”, і забезпечення їх актуальності та точності.

Тому виникає необхідність в подальшому створенні ефективної системи, в основу якої будуть покладені розроблені підходи, що дозволить університетам

використовувати дані з ЄДБО для прийняття обґрунтованих рішень, коригування маркетингових кампаній та підвищення своєї конкурентоспроможності, коригувати освітні програми спеціальностей відповідно до інтересів вступників. Наприклад, на основі аналізу даних можна прогнозувати попит на ті чи інші спеціальності.

Проведення такого дослідження матиме значний практичний та науковий внесок. На його основі можна заповнити існуючі прогалини у знаннях, що сприятиме розвитку теорії та практики управління вищими навчальними закладами. Результати дослідження можуть бути використані не лише конкретним університетом, але й іншими освітніми установами, що підвищить загальний рівень ефективності освітньої системи в Україні.

1.4. Висновок по першому розділу

На основі проведеного аналізу літератури можна зробити висновок, що проведення досліджень у сфері розробки підходів до аналізу даних з ЄДБО, орієнтованої на потреби університетів, є актуальною та своєчасною. Відповідає сучасним тенденціям розвитку інформаційних технологій, потребам ринку освітніх послуг та стратегічним завданням розвитку вищої освіти в Україні.

РОЗДІЛ 2. ОБГРУНТУВАННЯ ВИБОРУ НАПРЯМУ ДОСЛІДЖЕННЯ

Вступна кампанія є одним із ключових процесів, який визначає майбутній контингент здобувачів, а отже, і подальший розвиток закладу вищої освіти. Попри наявність значних обсягів даних представлених на різних платформах, університети обмежені у своїх можливостях щодо їхнього аналізу через відсутність інструментів, що в основі мають спеціалізовані підходи.

Для аналізу вступної кампанії університети можуть використовувати загальнодоступні ресурси, такі як платформа ЄДБО та веб-портал "АбітПошук". Проте ці платформи орієнтовані переважно на потреби абітурієнтів і надають інформацію у вигляді, що не підходить для глибокого стратегічного аналізу. Вони не забезпечують можливості детальної фільтрації, порівняння даних або візуалізації тенденцій вступу та попиту на різні галузі знань, що ускладнює процес прийняття обґрунтованих управлінських рішень.

Відсутність аналітичної системи, що ґрунтується на спеціалізованих підходах, змушує університети проводити аналіз даних вручну, що забирає багато часу та людських ресурсів, водночас підвищуючи ризик помилок і робить процес неефективним. Наприклад, в середньому, кожного року у вищі навчальні заклади подають заяви близько 400 тис. вступників, а враховуючи, що один вступник може подати до п'яти заяв на бюджетну форму навчання та до десяти заяв на контрактну, то загальна кількість заяв може перевищувати 5 млн. Для оцінки популярності певної спеціальності чи конкуренції з іншими закладами освіти необхідно вручну збирати всі ці дані з різних джерел, обробляти їх у таблицях та намагатися виявити закономірності без належних інструментів візуалізації. Така ситуація майже унеможливлює оперативно реагувати на зміни ринку освітніх послуг. І, як наслідок, університети не можуть швидко адаптувати свої маркетингові стратегії, коригувати освітні програми або прогнозувати попит на спеціальності, що негативно впливає на їхню конкурентоспроможність та здатність залучати талановитих абітурієнтів та покращувати якість надання освітніх послуг.

Оскільки кінцевою метою дослідження є створення інструменту, що базується на розроблених підходах до аналізу, який дозволить автоматизувати процес збору та інтерпретації результатів вступної кампанії, розробка такої системи вирішить низку актуальних проблем:

- економія ресурсів – автоматизація процесів збору та обробки даних дозволить значно зменшити використання людських ресурсів, що надасть час для детального аналізу та прийняття рішень;
- підвищення точності - використання спеціалізованих інструментів зменшить ймовірність помилок, пов'язаних з людським фактором при збору та обробці даних;
- оперативність - система дозволить відносно швидко збирати актуальну інформацію з джерел;
- глибокий аналіз - можливість деталізованого аналізу даних за різними параметрами (галузь знань, спеціальність, регіон, пріоритети абітурієнтів тощо) сприятиме більш обґрунтованому прийняттю рішень;
- візуалізація даних - інтерактивні графіки та діаграми допоможуть краще зрозуміти тенденції та закономірності, зроблять інформацію наочною та доступною для різних рівнів управління;
- можливість кастомізації - використання потужних інструментів візуалізації надають можливості самостійно створювати дашборди для різних розрізів аналізу через користувацький інтерфейс.

Основні завдання, покладені на систему:

- збір даних з різних джерел, таких як відкриті дані ЄДБО та веб-порталу “АбітПошук”, що також забезпечує можливість порівняння точності зібраних даних;
- валідація та очищення даних;
- збереження даних в базі даних для ефективного доступу до інформації;
- візуалізація даних на інтерактивних дашбордах для їх аналізу.

Новизна та практична значущість дослідження підкреслюється відсутністю підходів аналізу, що допоможуть вирішити поставленні вище наукові проблеми..

Університети потребують індивідуальних рішень, адаптованих до їхніх специфічних потреб та особливостей.

Такий підхід сприятиме розвитку культури даних у закладі вищої освіти. Використання даних для прийняття рішень стане невід'ємною частиною управлінських процесів, що відповідатиме сучасним тенденціям цифрової трансформації освіти. Інтеграція різних джерел даних та створення єдиної інформаційної платформи відкриває перспективи для подальшого розвитку системи, додавання нових функцій, таких як дослідження змін в роках та прогнозування тенденцій вступу.

2.1. Висновок по другому розділу

Таким чином, вибір напряму дослідження є виправданим з точки зору актуальності проблеми, її практичної значущості та перспектив розвитку. Розробка підходів аналізу результатів вступної кампанії у ВНЗ і подальша їх реалізація в спеціалізованій аналітичній системі для університетів відповідає сучасним викликам та сприятиме підвищенню ефективності управління.

РОЗДІЛ 3. ДОСЛІДЖЕННЯ МЕТОДІВ АНАЛІЗУ ДАНИХ ТА РОЗРОБКА КОМПЛЕКСНОГО ПІДХОДУ З УРАХУВАННЯМ СПЕЦИФІКИ ВСТУПУ ДО ВНЗ

Вступна кампанія у ВНЗ України є складним процесом, який характеризується великими обсягами різнорідних даних, динамічними змінами та залежністю від численних факторів, таких як регіональні особливості, попит на спеціальності та державні квоти. Для визначення специфіки вступної кампанії та розробки комплексного підходу до аналізу даних необхідно дослідити існуючі методи аналізу даних та вибрати ті, які найкраще підходять для розв'язання наукової проблеми, що полягає у відсутності ефективних підходів до аналізу даних результатів вступних кампаній, орієнтованих на потреби університетів в Україні.

3.1. Специфікація даних вступної кампанії

У процесі аналізу цих даних були виділені основні критерії, які є ключовими для оцінки та інтерпретації результатів, а саме: галузі знань, спеціальності, форма фінансування (бюджет/контракт), пріоритетність вибору вступника та регіональний розподіл.

Галузі знань та спеціальності відіграють ключову роль у розумінні структурних особливостей попиту на вищу освіту. Класифікація заяв за галузями знань дозволяє виявити найбільш популярні та затребувані напрями підготовки серед абітурієнтів. Форма фінансування навчання, тобто розподіл місць на бюджетну та контрактну форми, є важливим параметром, що впливає на доступність вищої освіти для різних верств населення. Аналіз кількості зарахованих на бюджет та контракт у розрізі галузей знань дозволяє оцінити пріоритетність фінансування тих чи інших спеціальностей з боку держави. Пріоритетність вибору вступника відображає його переваги та мотивацію при виборі спеціальності та університету. Аналіз розподілу заяв за першим пріоритетом та кількості зарахованих за цим пріоритетом дозволяє виявити найбільш бажані галузі знань та спеціальності серед абітурієнтів. Пріоритетність є важливим

індикатором реального попиту на освітні програми та допомагає університетам коригувати свою пропозицію, а також розробляти ефективні стратегії залучення абітурієнтів. Регіональний розподіл заяв та зарахувань є ключовим для розуміння географічних особливостей попиту на вищу освіту. Аналіз даних у розрізі регіонів дозволяє виявити нерівномірності у доступі до освіти, міграційні тенденції серед абітурієнтів та визначити регіони, які потребують додаткової підтримки.

3.2. Опис методів аналізу даних

Відповідно до специфіки вступної кампанії розглянуто методи аналізу даних, що можуть бути застосовані для розв’язання наукової проблеми [5].

Описовий аналіз - використовується для узагальнення даних та опису поточної ситуації за допомогою показників, таких як середнє значення, медіана, мода та стандартне відхилення. Дає змогу побачити основні тенденції в наборі даних, полегшуючи їх подальшу інтерпретацію.

Діагностичний аналіз - аналіз, спрямований на розуміння причин подій. Даний аналіз дозволяє вивчати дані глибше, щоб визначити фактори, які вплинули на певну ситуацію. Наприклад, діагностичний аналіз може пояснити, чому виникли зміни в поведінці користувачів чи показниках.

Прогностичний аналіз - даний метод аналізу використовує історичні дані та статистичні методи для передбачення майбутніх результатів. Він допомагає оцінити ймовірність подій, таких як майбутній попит на продукти чи зміни ринку, з високою точністю.

Наказовий аналіз - метод, спрямований на визначення найоптимальніших дій на основі отриманих даних. Він поєднує висновки з описового, діагностичного та прогнозного аналізів, пропонуючи оптимальні рішення для досягнення цілей.

Кількісний аналіз - метод, що передбачає використання математичних і статистичних методів для роботи з числовими даними. Цей метод дозволяє обчислювати важливі показники, проводити статистичні перевірки та отримувати об'єктивні висновки.

Метод якісних досліджень - зосереджується на аналізі концепцій, думок або досвіду, отриманих із текстових чи невербальних джерел та допомагає виявляти теми та закономірності, які не відображаються в числових даних.

Метод аналізу часових рядів - передбачає розгляд даних, що були зібрані протягом певних інтервалів часу, для визначення закономірностей, циклів та сезонних змін. Використовується для виявлення довгострокових тенденцій.

Регресійний аналіз - використовується для оцінки залежності між однією залежною змінною та кількома незалежними змінними, що дозволяє виявляти, як певні фактори впливають на досліджуваний результат.

Кластерний аналіз - застосовується для групування даних на основі їх подібності. Це в свою чергу дозволяє сегментувати великі набори даних, виявляючи групи із схожими характеристиками.

Факторний аналіз - виявляє основні фактори, які пояснюють взаємозв'язки між великою кількістю змінних, зменшуючи їх кількість і спрощуючи інтерпретацію даних.

Статистичний метод - передбачає збір, обробку, аналіз і інтерпретацію даних. Використовується для перевірки гіпотез і отримання висновків із даних у широкому контексті.

Метод Монте-Карло - використовує випадкову вибірку для моделювання складних систем і оцінки ризиків. Ефективний для аналізу невизначеностей і сценарного моделювання.

Когортний аналіз - метод, що допомагає вивчити поведінку груп людей із подібними характеристиками протягом певного часу. Використовується для виявлення тенденцій, наприклад, утримання клієнтів.

Обґрунтована теорія - використовується для створення теорій на основі систематично зібраних даних та допомагає розкрити приховані закономірності і побудувати моделі поведінки.

Інтелектуальний аналіз даних - досліджує великі набори даних для виявлення закономірностей і зв'язків. Використовується для прогнозів, сегментації ринку та рекомендаційних систем.

Інтерпретація даних - суть методу полягає в наданні сенсу отриманій інформації через аналіз. Цей метод допомагає формулювати висновки та розробляти стратегії на основі даних.

3.3. Обґрунтування комплексу методів для аналізу вступної кампанії

Враховуючи специфіку даних вступної кампанії та наукову проблему, що полягає у відсутності ефективних підходів до аналізу результатів вступних кампаній, орієнтованих на потреби університетів в Україні, було обрано комплекс методів аналізу даних. Цей комплекс включає описовий аналіз, діагностичний аналіз, прогностичний аналіз, кластерний аналіз та аналіз часових рядів.

Описовий аналіз є фундаментальним методом, який використовується для узагальнення даних та опису поточної ситуації за допомогою статистичних показників, таких як середнє значення, медіана, мода та стандартне відхилення. У контексті вступної кампанії цей метод дозволяє виявити основні тенденції, такі як популярність галузей знань та спеціальностей, розподіл заяв за формою фінансування (бюджет/контракт), пріоритетність вибору вступників та регіональний розподіл заяв.

Діагностичний аналіз спрямований на розуміння причин подій та виявлення факторів, які вплинули на певні результати. У випадку вступної кампанії цей метод дозволяє глибше вивчити, чому певні галузі знань або спеціальності є більш популярними, як регіональні особливості впливають на вибір вступників, та які чинники впливають на успішність зарахування.

Прогностичний аналіз використовує історичні дані та статистичні методи для передбачення майбутніх результатів. У контексті вступної кампанії він допомагає оцінити майбутній попит на спеціальності, що є критично важливим для планування ресурсів університетів та державної освітньої політики.

Сегментація даних є невід’ємною частиною аналізу даних про вступну кампанію. Реалізувати це можна за допомогою кластерного аналізу. Цей метод дозволяє групувати спеціальності або університети на основі схожих характеристик, таких як популярність чи конкуренція. Наприклад, кластери

можуть виявити групи університетів або спеціальностей, які є найбільш привабливими для абітурієнтів із високими академічними показниками.

Аналіз часових рядів передбачає розгляд даних, зібраних протягом певних інтервалів часу, для визначення закономірностей, трендів та сезонних змін. У випадку вступної кампанії це дозволяє відстежувати зміни в попиті на спеціальності та освітні програми протягом років.

Порівняння ключових переваг і недоліків кожного з обраних методів аналізу наведено в таблиці 3.1.

Таблиця 3.1. Порівняння ключових переваг і недоліків методів аналізу

Метод	**Переваги**	**Недоліки**	**Застосування в контексті вступної кампанії**
Описовий аналіз	Простота, загальний огляд даних	Відсутність глибинного розуміння взаємозв’язків	Узагальнення даних, виявлення загальних тенденцій
Діагностичний аналіз	Виявлення причинно-наслідкових зв’язків	Складність, вимога якісних даних	Розуміння факторів популярності спеціальностей
Прогностичний аналіз	Передбачення майбутніх тенденцій	Залежність від якості даних, непередбачуваність	Прогнозування попиту на спеціальності, планування ресурсів
Кластерний аналіз	Виявлення структур в даних, сегментація	Суб’єктивність, обчислювальна складність	Групування університетів та спеціальностей за характеристиками
Аналіз часових рядів	Виявлення трендів та сезонності	Чутливість до аномалій, складність методів	Відстеження змін попиту, оцінка ефективності рішень

3.4. Висновок по третьому розділу

Використання комплексу методів, що включає описовий, діагностичний, прогностичний, кластерний аналізи та аналіз часових рядів, є обґрунтованим підходом до аналізу даних вступної кампанії. Поєднання зазначених методів забезпечує комплексний підхід до аналізу даних вступної кампанії, що дозволяє вирішити наукову проблему щодо відсутності ефективних підходів до аналізу результатів вступних кампаній, орієнтованих на потреби університетів в Україні. Кожен метод доповнює інші, забезпечуючи всебічне дослідження різних аспектів вступної кампанії.

РОЗДІЛ 4. РОЗРОБКА СПЕЦІАЛІЗОВАНОЇ АНАЛІТИЧНОЇ СИСТЕМИ РЕЗУЛЬТАТІВ ВСТУПНОЇ КАМПАНІЇ У ВНЗ

4.1. Методи збору інформації з веб-джерел

Для створення системи аналізу вступної кампанії університету необхідно інтегрувати дані з відкритих джерел, таких як ЄДБО. Основними методами збору даних із веб-джерел є використання API та HTML-скрапінгу.

API як метод збору інформації

API — інструмент, що представляє собою набір інструкцій і стандартів, що забезпечують взаємодію між різними додатками та системами. Завдяки цьому розробники можуть створювати нові програми та сервіси, використовуючи готові функціональні можливості, надані іншими програмами чи платформами [6].

За допомогою API можна інтегрувати різні компоненти програми тим самим підвищуючи її ефективність та гнучкість. Основними принципами під час роботи з API є простота та надійність. Для можливості легкої та швидкої інтеграції цього інструменту у свої програми, API повинен бути легким для розуміння та простим для використання. Щоб різні програми, що використовують API, працювали без збоїв, він повинен гарантувати стабільність і безперебійну роботу без помилок.

Використання API має такі переваги:

- швидкість розробки - значно прискорюють процес розробки систем, оскільки надають готові сценарії для виконання певних функцій із поверненням результату;
- економія ресурсів - дозволяє розробникам уникнути створення власних функцій і сервісів;
- розширення функціональності - можна збільшувати можливості додатків і сервісів, інтегруючи нові функції та розширюючи їхню функціональність.

Недоліки використання API:

- обмеженість використання - API мають певні правила використання та обмеження, визначені розробниками, що звужує їхню функціональність або доступність;
- залежність від системи сторонніх сервісів - робота програми або сервісу, що використовує API, залежить від його стабільності, тому зміни в роботі API можуть безпосередньо впливати на функціональність системи, що використовує його;
- безпека - використання API може створити ризики, що підвищує вразливість системи до кібератак, якщо не забезпечити відповідних заходів захисту.

Схема роботи API наведено на рисунку 4.1.

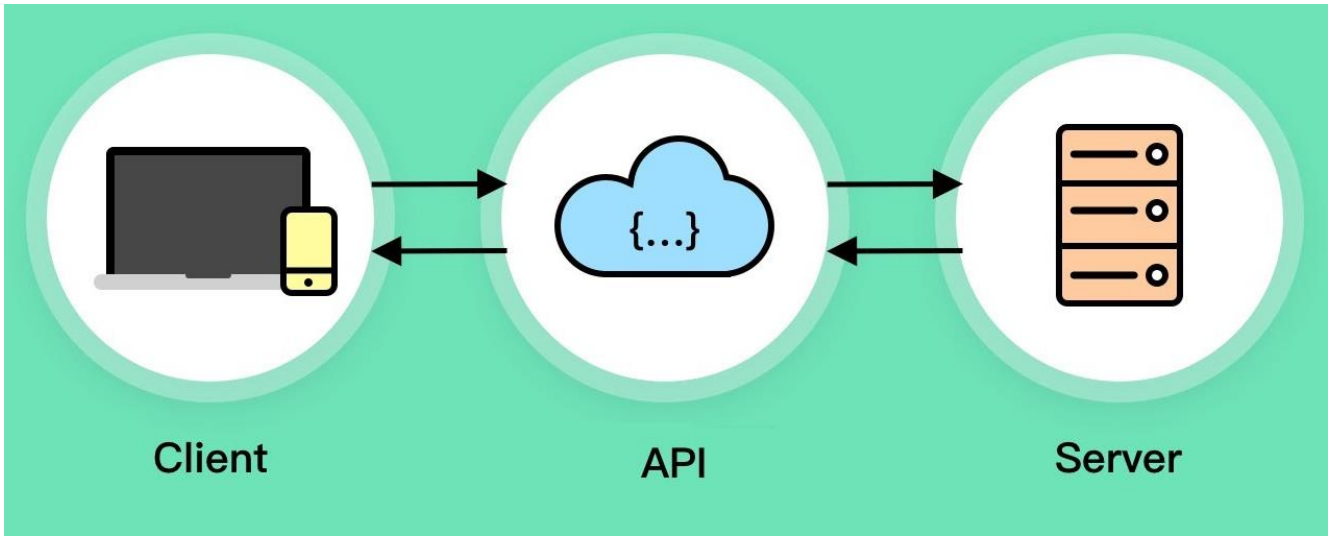

Рисунок 4.1 - Схема роботи API

Оскільки платформа ЄДБО не надає публічного API для отримання даних про конкурсні пропозиції та результати вступних кампаній, це виключає можливість використання цього методу для інтеграції даних.

HTML-скрапінг

Парсинг даних є процесом автоматизованого вилучення структурованої інформації з неструктурованих джерел, таких як веб-сторінки, текстові файли, тощо. Парсинг дозволяє програмно отримувати дані з різних форматів для подальшої обробки. Залежно від джерела інформації та типу даних, парсинг реалізується різними методами [7].

Одним із найпоширеніших методом парсингу є HTML-парсинг, тобто парсинг даних, які представлені у вигляді структурованих HTML-сторінок і є основним форматом для представлення даних у веб-середовищі. HTML-документи структуровані як деревоподібні моделі, де кожен елемент (тег) має певне місце в ієрархії. Така структура елементів дозволяє програмно отримувати доступ до будь-якого з них. Підтипом HTML-парсингу є DOM-парсинг, який дозволяє інтерпретувати HTML як структуру, представлену у вигляді дерева елементів.

DOM-парсинг працює за принципом побудови дерева об’єктів, де кожен елемент сторінки стає вузлом цього дерева. Наприклад, програмно можна знайти елементи за тегами, класами, атрибутами чи іншими селекторами та отримати їхній вміст.

До переваг DOM-парсингу можна віднести [8]:

- автоматизацію збору інформації з веб-джерел;
- ефективність;
- легке впровадження;
- точність, у випадку добре структурованого джерела;

До недоліків DOM-парсингу відносяться:

- аналіз даних - у порівнянні з API парсингом, аналізувати отримані дані з HTML сторінок важче;
- залежність від стабільності джерела - оскільки HTML-структура веб-сайтів може змінюватись, то це може впливати на роботу парсеру, особливо для веб джерел де дані не підвантажуються динамічно. Це вимагає постійної перевірки і адаптації парсера під веб-джерело;

У випадку роботи з платформою ЄДБО HTML-парсинг є найефективнішим методом для збору даних, оскільки структура веб-сторінок ЄДБО дозволяє легко інтегрувати DOM-парсинг для доступу до чітко структурованих елементів.

4.2. Мови програмування для збору даних з веб-джерел

Для розробки системи збору даних із веб-джерел, таких як ЄДБО, необхідно вибрати мову програмування, яка найкраще відповідає завданням парсингу та обробки даних. Для розгляду обрано такі мови програмування Python, JavaScript та PHP.

PHP — одна з найпопулярніших мов програмування, особливо серед веб-розробників. Її створив Расмус Лердорф, який спочатку розробив набір інструментів для спрощення створення динамічних веб-сторінок [9]. Хоча дана мова найчастіше використовується для серверної обробки HTML-коду, що згодом відображається у веб-браузері, сучасний PHP став мовою загального призначення.

Основні напрямки використання PHP [10]:

- серверна обробка - PHP найчастіше використовується для створення скриптів, які виконуються на серверній стороні. Для цього потрібні лише PHP-парсер (у вигляді CGI-додатка чи модуля сервера), веб-сервер і веб-браузер;
- скрипти для командного рядка - PHP також дозволяє створювати скрипти, які виконуються безпосередньо через командний рядок, без необхідності у веб сервері чи браузері. Такий підхід підходить для завдань, які виконуються автоматично через cron (на Unix або macOS) чи планувальник завдань (на Windows), а також для обробки текстових даних.

Однією з ключових переваг PHP є його підтримка широкого спектра баз даних. PHP дозволяє легко підключатися до будь-якої бази даних, яка підтримує стандарт Open Database Connectivity (ODBC), через відповідне розширення. Для роботи з іншими базами даних, такими як CouchDB, можна використовувати cURL або сокети.

PHP пропонує потужні можливості для обробки тексту, включаючи підтримку регулярних виразів, сумісних із Perl (PCRE). Мова має багато розширень та інструментів для роботи з XML-документами. Усі XML-розширення PHP базуються на бібліотеці libxml2, що забезпечує стандартизовану основу. Серед додаткових можливостей — підтримка SimpleXML, XMLReader та XMLWriter, які надають можливості для обробки й створення XML-документів.

Для веб-скрапінгу PHP надає такі інструменти як Guzzle та DomCrawler [11].

Guzzle — це клієнтська бібліотека HTTP, яка дозволяє зручно отримувати вміст веб-сторінок, відправляти запити та обробляти відповіді.

DomCrawler — це інструмент для синтаксичного аналізу HTML, який спрощує вилучення конкретних даних з HTML-документів.

Недоліки PHP [12]:

- оскільки PHP є мовою з відкритим вихідним кодом, його безпека може бути під загрозою, адже код програми легкодоступний;
- слабка типізація в PHP може призводити до помилок у відображенні даних і неточної інформації для користувачів;
- надмірне використання функцій фреймворків та інструментів PHP може негативно впливати на швидкодію веб-додатків;
- різноманітність фреймворків PHP спричиняє відмінності в їхній роботі, що впливає як на продуктивність, так і на функціональні можливості.

JavaScript (JS) — це динамічна, об’єктно-орієнтована прототипна мова програмування, що реалізує стандарт ECMAScript. Найчастіше її використовують для створення сценаріїв веб сторінок, які дозволяють виконувати інтерактивні дії на стороні клієнта (пристрою користувача), взаємодіяти з користувачем, керувати функціями браузера, здійснювати асинхронний обмін даними із сервером, а також змінювати структуру і зовнішній вигляд веб-сторінки [13]. JavaScript є мультипарадигмальною мовою з динамічною типізацією, яка підтримує об’єктно-орієнтований, імперативний і декларативний стилі програмування. Застосовується в таких програмних продуктах як Node.js або Apache CouchDB [14].

Для веб-скрапінгу, тестування додатків та роботи з динамічним контентом JavaScript надає такі інструменти як Playwright, Selenium та Puppeteer [15]. Вони дозволяють програмно взаємодіяти з браузерами, виконуючи дії, схожі на дії користувача, наприклад, кліки, введення тексту, скролінг чи заповнення форм.

Усі ці інструменти інтегруються з JavaScript через асинхронні функції, використовуючи async/await, що у поєднанні з promise-ланцюгами дозволяє виконувати кілька асинхронних операцій одночасно та чекати завершення всіх.

Переваги JavaScript [16]:

- ефективність та економія ресурсів - JavaScript виконується безпосередньо на клієнтській стороні, що зменшує навантаження на сервер і прискорює обробку запитів;
- завдяки XMLHttpRequest-об’єктам можна здійснювати асинхронний обмін даними між клієнтом і сервером без необхідності перезавантаження сторінки;
- JavaScript працює у всіх сучасних браузерах, забезпечуючи однакову функціональність і результат незалежно від платформи;
- мова легко взаємодіє з іншими технологіями та використовується у створенні різноманітних програм і систем;
- JavaScript є інтуїтивно зрозумілою мовою, що робить його ідеальним вибором для новачків. Він також дозволяє створювати інтерактивні інтерфейси для користувачі.

Недоліки JavaScript [16]:

- JavaScript-код завжди відкритий для перегляду, що може створювати потенційні загрози для безпеки;
- незважаючи на високу швидкість виконання самого JavaScript, робота з DOM залишається порівняно повільною, що збільшує час рендеренгу HTML;
- виникнення помилка під час виконання JavaScript-коду може зупинити рендеринг усієї веб сторінки;

- JavaScript інтерпретується по-різному в різних браузерах, що ускладнює написання та читання кросбраузерного коду;
- неефективне налагодження сценаріїв відносно таких редакторів, як для C/C++, що ускладнює пошук та усунення помилок.

Python — це інтерпретована мова програмування високого рівня, яка поєднує об’єктно-орієнтований підхід із динамічною типізацією [17]. Створена у 1990 році Гвідо ван Россумом і широко використовується для створення веб сайтів, аналізу даних, машинного навчання, інженерії даних та для багатьох інших областей. Завдяки простим у використанні структурам даних, динамічній семантиці та гнучким можливостям зв’язування, Python ідеально підходить для швидкої розробки програм та інтеграції різних компонентів.

Мова підтримує модулі та пакети, що дозволяє розробникам створювати модульний код і багаторазово його використовувати. Інтерпретатор Python і стандартна бібліотека доступні на всіх основних платформах як у вихідному коді, так і у вигляді скомпільованих пакетів. Python підтримує кілька стилів програмування, включно з об’єктно-орієнтованим, процедурним, функціональним та аспектно-орієнтованим підходами, що робить його універсальним для розв’язання різноманітних завдань.

Для парсингу даних із веб сторінок Python надає інструменти, такі як Scrapy, Selenium, BeautifulSoup та Playwright, які дозволяють ефективно працювати як із статичним, так і з динамічним контентом.

Ключовою перевагою Python для веб-скрапінгу є його економічність [18]. Порівняно з мовами, такими як Java або C++, Python потребує менше коду та часу на розробку, що значно знижує витрати для компаній. Завдяки безкоштовним і відкритим бібліотекам можна уникнути додаткових витрат на ліцензії чи використання пропрієтарного програмного забезпечення.

До переваг використання Python також відноситься його гнучкість і можливість налаштування. Розробники можуть легко адаптувати скрипти до конкретних вимог або потреб, а також інтегрувати в них наявні бібліотеки. Це дозволяє створювати інструменти веб-скрапінгу, які точно відповідають

поставленим завданням, мінімізуючи обробку зайвої інформації, що може уповільнювати процес або викликати помилки.

Порівняння мов та обґрунтування вибору

У таблиці 4.1 наведено основні характеристики мов Python, JavaScript і PHP у контексті веб-парсингу.

Таблиця 4.1. Порівняння критеріїв мов програмування для веб-скрапінгу

Критерій	**Python**	**JavaScript**	**PHP**
Популярні бібліотеки для веб-скрапінгу	Scrapy, BeautifulSoup, urllib3, lxml, Selenium, Playwright, Requests	Cheerio, Puppeteer, Selenium, jsdom, Axios	Guzzle, DomCrawler
Динамічна обробка вмісту	так	так	Обмежені можливості обробки динамічного контенту
Обробка даних	Бібліотеки, такі як NumPy, Pandas для ефективної обробки даних	Не оптимальний для обробки та аналізу даних	Обмежені можливості для обробки даних
Продуктивність	Висока продуктивність, особливо для масштабного парсингу	Висока продуктивність завдяки рідній підтримці JavaScript	Висока продуктивність
Масштабованість	Добре масштабована	Добре масштабована	Добре масштабована

Продовження таблиці 4.1. Порівняння критеріїв мов програмування для веб-скрапінгу

Критерій	**Python**	**JavaScript**	**PHP**
Асинхронність	Не має нативної підтримки, але підтримується через додаткові бібліотеки	Нативна підтримка асинхронності (Promises, async/await)	Не має нативної підтримки, але підтримується через додаткові бібліотеки
Складність	Низька	Середня	Низька

Платформа ЄДБО завантажує дані динамічно через JavaScript, що ускладнює використання стандартних методів парсингу. Для успішного збору такої інформації необхідно застосовувати інструменти, здатні рендерити JavaScript і працювати з динамічним контентом. Playwright та Selenium є інструментами, які інтегрується з Python і дозволяють отримати дані динамічного контенту. Тому, на основі проведеного аналізу було обрано Python як основну мову програмування для збору даних із веб-ресурсів.

Додатковою перевагою Python є простота у використанні та швидкість розробки. Його лаконічний і зрозумілий синтаксис дозволяє швидко створювати функціональний код, що особливо важливо для задач, які потребують оперативних змін. Це також важливо для майбутньої підтримки системи.

Хоча Python не має нативної підтримки асинхронності, використання бібліотек, таких як asyncio, у поєднанні з Playwright дозволяє працювати з багатопоточними задачами. Таким чином можна забезпечити одночасну обробку кількох запитів, а це значно підвищує продуктивність системи збору даних.

Таким чином, вибір Python як основної мови програмування для збору даних є оптимальним рішенням для роботи з динамічним контентом на платформі ЄДБО.

4.3. Обґрунтування вибору фреймворку для збору даних з веб-джерел

Оскільки в якості мови програмування було обрано Python, і зважаючи на потребу роботи з динамічним контентом на сторінках платформи ЄДБО, найбільш оптимальними інструментами, які підтримуються обраною мовою, є Playwright та Selenium.

Selenium — це один із найпопулярніших і найнадійніших інструментів для веб-автоматизації та тестування, який вже понад десять років займає провідну позицію у цій сфері. Завдяки тривалому періоду існування Selenium сформував велику спільноту розробників, яка забезпечує численні ресурси, документацію та підтримку. Це робить його перевіреним вибором для реалізації автоматизації веб-завдань.

Фреймворк підтримує сумісність із багатьма браузерами, такими як Chrome, Firefox, Safari та Edge і ця сумісність дозволяє проводити тестування та автоматизацію збору даних із веб-джерел у різних середовищах, забезпечує універсальність і широкі можливості для адаптації до конкретних потреб проєкту [19].

Playwright — це сучасний інструмент для веб-автоматизації, створений компанією Microsoft, як засіб, що зможе розширити можливості Selenium. Завдяки своєму інноваційному підходу Playwright пропонує зручні функції для роботи з динамічними веб-сторінками [19].

Playwright переходить на використання сучасних технологій браузерів, забезпечуючи підтримку власних протоколів, таких як Chrome DevTools Protocol. Він сумісний із основними двигунами браузерів, включаючи Chromium, WebKit і Gecko (Firefox), але не підтримує застарілі браузери, такі як Internet Explorer.

Порівняння фреймворків

Обидва фреймворки надають Playwright підтримку роботи в режимі headless (без відображення інтерфейсу браузера) і headful (з інтерфейсом браузера), виконання JavaScript-коду та автоматизацію взаємодії користувача з елементами веб-сторінок.

Playwright забезпечує більш простий та уніфікований API для роботи з різними браузерами. Його методи дозволяють створювати коротші та більш зрозумілі для читання сценарії, спрощуючи автоматизацію. А підтримка сучасних функцій браузерів за замовчуванням, дає змогу легко виконувати складні завдання автоматизації [20].

На відміну від Playwright, Selenium часто потребує додаткових етапів кодування або складних конфігурацій для виконання завдань, таких як блокування запитів, емуляція пристроїв чи налаштування автоматичного очікування [21]. У Playwright ці функції інтегровані в API сторінки, що дозволяє виконувати подібні задачі простіше та швидше.

Selenium працює з браузером зовні, імітуючи дії, які виконує тестувальник вручну. Наприклад, коли тестувальник натискає кнопку, Selenium посилає команду браузеру «клік», а для введення тексту — відповідну команду «ввести». Такий підхід забезпечує базову автоматизацію, але створює додаткову затримку через послідовну передачу команд.

На відміну від цього, Playwright працює безпосередньо всередині браузера, що значно підвищує його швидкість і розширює можливості. Замість імітації дій користувача, Playwright інтегрується з внутрішніми механізмами браузера, отримуючи та обробляючи запити безпосередньо з додатку. Це забезпечує не лише швидшу взаємодію, але й доступ до додаткових функцій.

Наприклад, Playwright може інтегруватися з інструментами розробника DevTools, які дозволяють відстежувати використання ресурсів, аналізувати запити, виявляти помилки та проводити налагодження (debugging). Selenium не має прямого доступу до таких інструментів. Таким чином, завдання, пов’язані з детальним аналізом або посиланням запитів, виконати через Selenium значно складніше, оскільки його функціонал обмежений взаємодією лише на рівні зовнішніх команд[22].

Playwright є швидшим у порівнянні із Selenium завдяки сучасній архітектурі, яка підтримує асинхронні операції. Використовуючи асинхронність, стає можливим створювати декілька вкладок браузера, і поки одна вкладка очікує на

відповідь від ресурсу, інша може взяти на себе потік обробки. Selenium, хоч і залишається потужним і широко використовуваним інструментом, але залежить від зовнішніх драйверів для взаємодії з браузером, а це додає затримки під час виконання операцій. Такий підхід робить Selenium менш оптимальним для сучасних потреб автоматизації, порівняно з Playwright.

Для оцінки продуктивності інструментів автоматизації Selenium та Playwright було проведено тестування, яке складалося зі 100 ітерацій на однаковому цільовому веб-сайті. Тест виконувався на машині з характеристиками: 16 ГБ оперативної пам’яті та процесором із частотою 2,6 ГГц. Час виконання кожної ітерації вимірювався за допомогою модуля Python time, а середнє значення обчислювалось за допомогою модуля statistics.

Результати тесту показали (рисунок 4.2), що Playwright був швидшим із середнім часом виконання 290,37 мілісекунди, у той час як Selenium мав середній час виконання 536,34 мілісекунди.

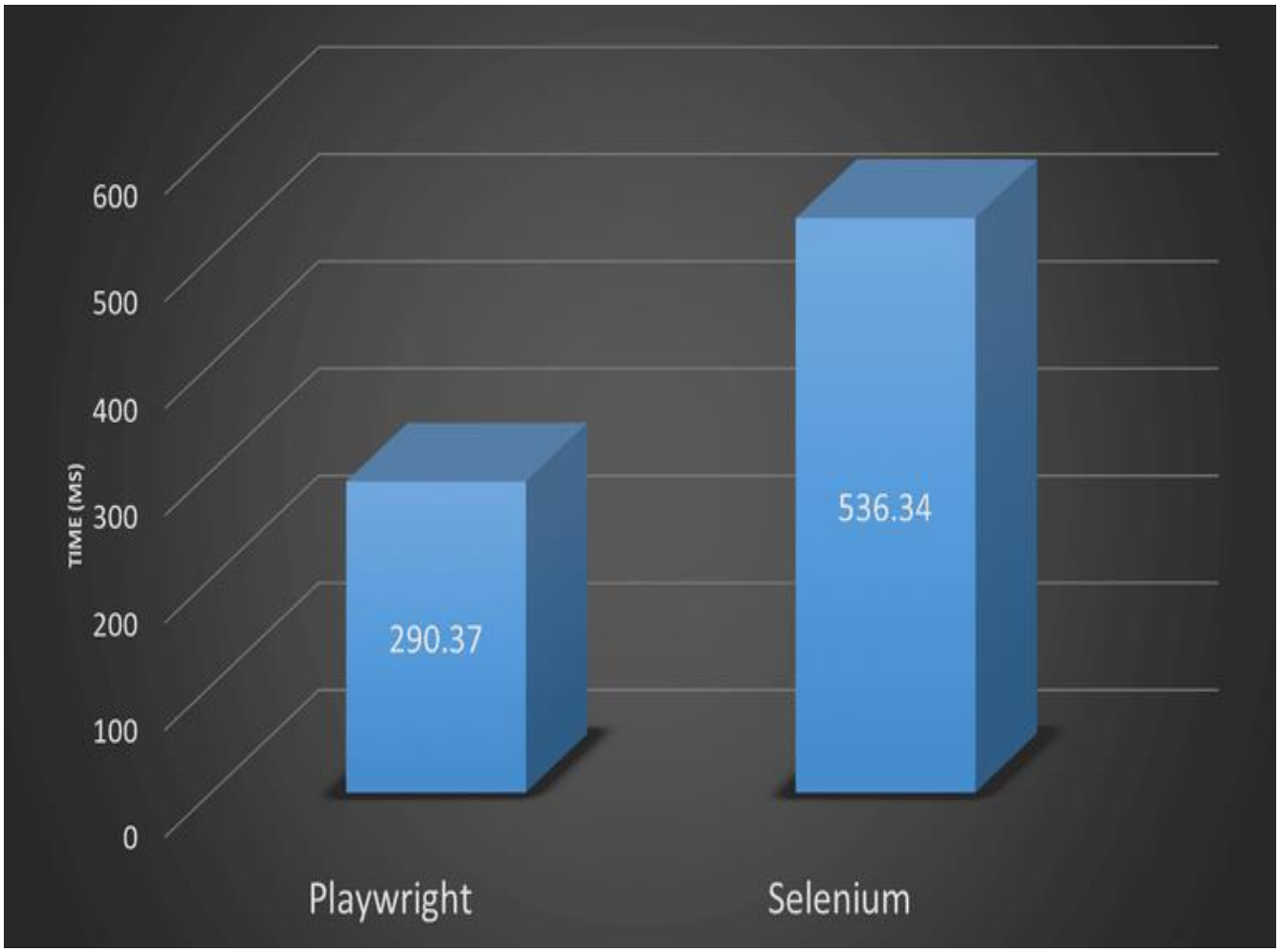

Рисунок 4.2 - Результати тестування швидкості обробки сторінок за допомогою інструментів Playwright та Selenium

Порівняння ключових параметрів кожного з фреймворків наведено в таблиці 4.2.

Таблиця 4.2. Порівняння ключових параметрів фреймворків

Параметр	**Playwright**	**Selenium**
Сумісні мови програмування	Python, JavaScript, Java, .NET, C#, TypeScript,	Python, JavaScript, PHP, .NET, Java, Kotlin, C#, Perl, Ruby.
Підтримка браузерів	Chrome, Safari, Firefox, Edge	Chrome, Firefox, Safari, Edge, Internet Explorer

Продовження таблиці 4.2. Порівняння ключових параметрів фреймворків

Параметр	Playwright	Selenium
Підтримка операційних систем	Windows, Linux, macOS	Windows, Linux, macOS, Solaris
Архітектура	Chrome DevTools Protocol (CDP) і нативні протоколи для Firefox і WebKit	W3C WebDriver Protocol
Простота у використанні	Відносно проста для освоєння	Вимагає більше зусиль для навчання
Швидкість роботи	Висока	Низька
Підтримка реальних пристроїв	Емуляція мобільних пристроїв. Підтримки реальних пристроїв немає	Доступна через хмарні сервіси та віддалені сервери
Інструменти для дебагінгу	Playwright Codegen	Selenium IDE

На основі проведеного аналізу та порівняння інструментів автоматизації Playwright і Selenium було обрано Playwright як оптимальний фреймворк для збору даних із веб-джерел. Рішення ґрунтується наступними перевагами:

- підтримує асинхронні операції та забезпечує вищу продуктивність;
- вбудована підтримка браузерів таких як Chromium, WebKit, Gecko) без необхідності завантаження окремих драйверів, що спрощує налаштування;
- інтегрується безпосередньо з механізмами браузера, що дозволяє швидше обробляти запити, отримувати доступ до інструментів розробника (DevTools) і виконувати складні завдання автоматизації, включаючи дебагінг і аналіз ресурсів;
- оскільки написання скриптів за допомогою Playwright є простішим, то в подальшому систему буде легше підтримувати та розширювати.

4.4. Обгрунтування вибору браузеру для збору даних з веб-джерел

Так як в якості фреймворку для автоматизації було обрано Playwright, вибір браузера здійснювався з наступних підтримуваних варіантів: Chromium, WebKit та Firefox.

Chromium — це браузер із відкритим кодом, який служить базою для багатьох сучасних браузерів, таких як Google Chrome, Microsoft Edge та інші. Він побудований на механізмі візуалізації Blink і має високу підтримку JavaScript, HTML5 та інших сучасних стандартів [23]. Це надає в свою чергу високу продуктивність при роботі з динамічними сторінками.

Firefox є браузером із відкритим кодом, який використовує механізм візуалізації Gecko [24]. Основною його перевагою є незалежність від Google , що забезпечує додатковий рівень конфіденційності та приватності. Підтримує складні політики безпеки, такі як Content Security Policy, що дозволяє працювати з веб-сторінками, де обмежено виконання скриптів або завантаження ресурсів.

WebKit є механізмом візуалізації веб-сторінок, що використовується переважно в браузері Safari, а також в інших додатках, орієнтованих на пристрої Apple [25]. Оптимізація для роботи на пристроях Apple забезпечує сумісність із веб-сторінками, розробленими спеціально для цієї екосистеми. Серед основних сфер використання WebKit для парсингу можна виділити збір даних зі сторінок, орієнтованих на пристрої Apple, а також тестування та автоматизацію веб-додатків, які функціонують у межах екосистеми macOS та iOS.

Оскільки Chromium поєднує в собі високу продуктивність при роботі з динамічним контентом та є оптимізованою версією браузеру Google Chrome, який, є найпоширенішим браузером у світі, його використання гарантує високу сумісність із більшістю сучасних веб-ресурсів, оскільки веб-розробники орієнтуються саме на Chrome під час тестування своїх сайтів. Це робить Chromium особливо зручним для роботи з платформами, такими як ЄДБО.

Таким чином, вибір Chromium як браузера для автоматизації даних із веб-джерел базується на його продуктивності, сумісності та інтеграції з Playwright.

4.5. Обґрунтування вибору СУБД для збереження зібраних даних з веб-джерел

Як вже згадувалось раніше, система повинна забезпечувати збереження великого обсягу даних та швидкий доступ до них. Оскільки кількість заяв вступників може перевищувати 1 мільйон, необхідно використовувати СУБД, яка здатна ефективно обробляти великі обсяги інформації. Це особливо важливо для забезпечення зберігання даних за кілька років вступних кампаній, а також для швидкого та зручного представлення інформації користувачам. Обрана СУБД повинна мати високий рівень продуктивності, масштабованості і здатності оптимізації запитів для роботи з великими даними.

Система управління базами даних — це програмне забезпечення, яке дозволяє працювати з базами даних та забезпечує створення, збереження, оновлення та пошук інформації, а також управляє доступом до цих даних, забезпечуючи їхню безпеку та цілісність.

Великі дані (Big Data) в інформаційних технологіях — це обсяги інформації, які можуть включати як структуровані, так і неструктуровані дані, і які настільки масштабні, що традиційні методи обробки, такі як інструменти бізнес-аналітики чи системи управління базами даних, виявляються неефективними для їх аналізу.

Найкращим варіантом для роботи з великими даними є колонкові СУБД, оскільки вони оптимізовані саме для аналітичних завдань.

Колонкова СУБД — це тип системи керування базами даних, у якій дані організовані за стовпцями замість рядків. Такий підхід дозволяє оптимізувати продуктивність при виконанні запитів, що працюють із великими обсягами даних, оскільки зчитуються лише необхідні стовпці, а не весь рядок [26].

Як і звичайна СУБД, що орієнтована на порядкове оброблення записів, так і колонкова працюють із таблицями, що містять рядки із записами, і стовпцями, для опису поля. Однак, їх різниця полягає в підходах до збереження даних. Рядково-орієнтовані СУБД зберігають дані кожного рядку разом, тоді як колонкові СУБД зберігають дані кожного стовпця разом (рисунок 4.3).

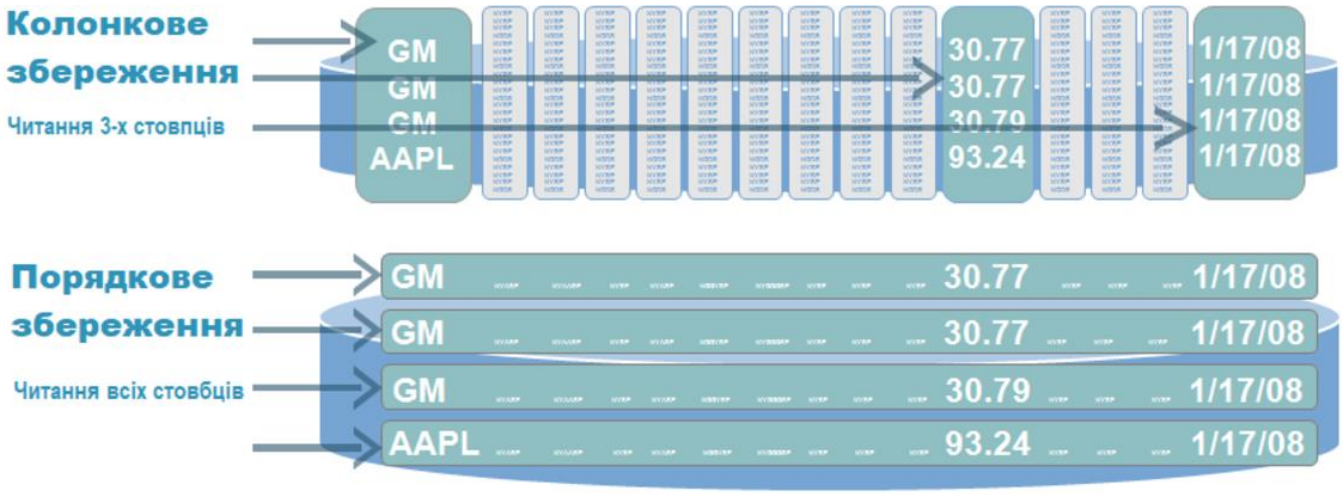

Рисунок 4.3 - Принцип читання записів колонкової і рядкової бази даних

Основним фактором, що обмежує продуктивність сучасних великих баз даних, є не швидкість процесора, а швидкість доступу до даних, які зберігаються на диску. Оскільки обсяг даних у таких базах часто сягає терабайтів і навіть петабайтів, зберігати всю інформацію в оперативній пам'яті (RAM) неможливо. Тому її доводиться зчитувати безпосередньо з диска. Навіть найшвидші твердотільні накопичувачі NVMe не можуть працювати на рівні продуктивності сучасних процесорів, що робить швидкість читання з диска критичним обмеженням для баз даних [27].

Послідовне зчитування даних із дисків відбувається набагато швидше, ніж випадкове або часткове. Це особливо помітно для традиційних жорстких дисків (HDD), але також є актуальним для твердотільних накопичувачів (SSD). Таким чином, спосіб організації та зберігання даних на диску суттєво впливає на загальну продуктивність бази даних.

Основною перевагою стовпчастих баз даних є їхня висока продуктивність під час виконання операцій читання. Це зумовлено тим, що більшість аналітичних запитів зазвичай звертаються до даних у певних стовпцях, а не у всьому рядку. А завдяки такій структурі зчитується менше даних з диска, а це значно підвищує швидкість виконання запитів. Колонковий підхід дозволяє вибірково завантажувати лише ті стовпці, які необхідні для запиту, уникаючи додаткового навантаження на систему, що сприяє масштабуванню колонкових баз даних.

Колонкові бази даних дозволяють значно стискати дані, а це є важливим для зменшення обсягу збереженої інформації. Порожні комірки просто пропускаються, а подібні дані, наприклад числові або текстові, об’єднуються для застосування складних методів стиснення, таких як стиснення довжини циклу, словникове стиснення або LZW. Завдяки цьому виконання операцій зі стовпцями, таких як MIN, MAX, SUM, COUNT і AVG, відбувається значно швидше. Денормалізовані бази даних у стовпчастій архітектурі можуть займати такий самий обсяг пам’яті, як нормалізовані реляційні бази даних, але забезпечують вищу продуктивність під час читання завдяки денормалізації. Також колонкові бази даних підтримують самоіндексацію, яка потребує значно менше дискового простору, ніж реляційна база даних із кількома індексами.

З урахуванням того, що колонкові СУБД створені для таких задач, як обробка даних у системах онлайн-аналітичної обробки (OLAP), де використовуються багатовимірні “куби” для аналізу інформації, вибір саме цього типу СУБД є найкращим для зберігання та аналізу великої кількості даних.

Однією з найпопулярніших колонкових СУБД є ClickHouse [28]. Завдяки колонковій архітектурі, ClickHouse забезпечує ефективне стиснення даних та прискорює виконання запитів, оперуючи стовпцями замість рядків, це забезпечує високу швидкість виконання запитів для агрегації, фільтрації та сортування даних. Система має архітектуру, яка дозволяє розподіляти дані та запити між декількома вузлами. Завдяки чому вона є доступною і має високу продуктивність навіть при значному збільшенні навантаження. ClickHouse підтримує формати даних, такі як csv, tsv, json та інші. ClickHouse є програмним забезпеченням із відкритим кодом, яке дозволяє модифікувати систему та налаштовувати її відповідно до конкретних потреб.

Таким чином ClickHouse є оптимальною СУБД для збереження та аналізу даних про вступні кампанії.

4.6. Обґрунтування вибору системи для візуалізації даних

Grafana та Metabase є популярними інструментами для візуалізації та аналізу даних, проте вони мають різні цілі, функціональність і підходи до роботи з даними.

Grafana — це веб-додаток з відкритим кодом, призначений для аналітики та інтерактивної візуалізації даних, широко використовуваний для моніторингу продуктивності систем. Інструмент дозволяє отримувати дані з різних джерел, виконувати запити, налаштовувати та відображати інформацію у вигляді графіків і діаграм, створювати інформаційні панелі для візуалізації та встановлювати сповіщення про виявлення аномалій. Водночас, через відкритий вихідний код, користувачам необхідно самостійно забезпечувати підтримку власного екземпляра Grafana, що може бути спрощено завдяки використанню Hosted Grafana від MetricFire, яка надає всі функції без додаткових витрат на адміністрування [29].

Так як Grafana часто використовується для моніторингу лог-файлів серверів, інструмент надає потужні можливості для налаштування сповіщень. Користувачі можуть створювати правила сповіщень на основі визначених умов, а Grafana надсилає повідомлення через різні канали, такі як електронна пошта, Slack чи PagerDuty. Таким чином надає можливість оперативно реагувати на критичні ситуації [30].

Grafana оснащена потужним редактором запитів, який дозволяє отримувати, фільтрувати та обробляти дані з різних джерел, включаючи Graphite, InfluxDB, Prometheus, Elasticsearch, AWS CloudWatch та багато інших. Ця гнучкість забезпечує можливість отримання даних з різних систем і відображення їх на єдиній інформаційній панелі.одним із найпоширеніших методом парсингу.

Підтримка багатокористувацького доступу із різними ролями (глядач, редактор, адміністратор), дозволяє ефективно організувати співпрацю на інформаційних панелях. Система забезпечує створення окремих робочих просторів для різних команд чи проєктів.

Екосистема Grafana включає широкий набір плагінів і розширень, які розширюють функціональність інструмента. Це охоплює плагіни для нових типів

візуалізацій, інтеграції з додатковими джерелами даних, а також плагіни додатків для об'єднання інформаційних панелей, сповіщень і налаштувань у межах єдиного середовища.

Інтерфейс користувача Grafana реалізовано як веб-інтерфейс, створений за допомогою JavaScript і React. Цей інтерфейс відповідає за рендеринг інформаційних панелей, окремих панелей і візуалізацій. Через інтерфейс користувачі можуть виконувати основні функції системи: налаштовувати інформаційні панелі, створювати запити до джерел даних, встановлювати правила сповіщень та керувати конфігурацією системи. Завдяки інтуїтивному дизайну інтерфейс Grafana забезпечує зручність у роботі як для технічних, так і не технічних користувачів.

Інтерфейс користувача Grafana взаємодіє з сервером через RESTful API. Цей рівень API дозволяє автоматизувати роботу з системою, здійснювати віддалене керування та інтегрувати Grafana з іншими інструментами. Завдяки API можна виконувати операції, такі як створення інформаційних панелей, управління джерелами даних, налаштування сповіщень і багато іншого, що робить систему гнучкою для інтеграції у складні програмні екосистеми.

Основний сервер Grafana, написаний мовою програмування Go, відповідає за управління загальним станом системи, обробку запитів API, автентифікацію користувачів і потоком даних між інтерфейсом користувача та джерелами даних. Цей сервер виконує ключову роль у забезпеченні стабільності та функціональності всієї системи.

Серверна частина Grafana включає адаптери для підключення до різних джерел даних, таких як Prometheus, InfluxDB, MySQL, Elasticsearch тощо. Кожне джерело даних реалізоване у вигляді плагіна, що дозволяє Grafana надсилати запити, отримувати та обробляти дані з різних серверних програм. Ця модульна структура забезпечує гнучкість у роботі з багатьма типами джерел даних.

Серверний механізм обробляє запити, створені користувачами через інтерфейс. Він трансформує запити у формат, сумісний із конкретним джерелом

даних, отримує результати та обробляє їх перед передачею в інтерфейс для візуалізації.

Обробка запитів здійснюється наступним чином: коли користувач створює або змінює інформаційну панель, інтерфейс Grafana надсилає запит на основний сервер. Сервер, у свою чергу, звертається до вказаних джерел даних для отримання необхідної інформації. Обробка запиту здійснюється плагіном відповідного джерела даних, який формує запит у форматі, зрозумілому для цього джерела, і отримує потрібні дані.

Після отримання даних від джерела серверна частина Grafana може виконувати додаткову обробку, наприклад, агрегування або фільтрацію даних часових рядів. Це дозволяє підготувати дані у зручному для користувача вигляді. Оброблені дані повертаються до інтерфейсу користувача, де вони візуалізуються на панелях у вигляді графіків, таблиць чи інших елементів.

Для підвищення продуктивності та зменшення навантаження на джерела даних Grafana використовує механізм кешування. Часто запитувані дані зберігаються в кеші, що дозволяє значно пришвидшити доступ до них і оптимізувати роботу системи, особливо під час виконання однотипних запитів.

Дані часових рядів, які Grafana використовує для візуалізації, отримуються та відображаються безпосередньо із зовнішніх баз даних. Водночас Grafana зберігає конфігураційні дані, зокрема визначення інформаційних панелей, профілі користувачів і системні налаштування, у реляційній базі даних. Для цього можуть застосовуватися СУБД, такі як SQLite, MySQL або PostgreSQL.

Grafana підтримує стан користувацьких сеансів, ролей і іншої метаінформації, яка необхідна для організації безпечного багатокористувацького доступу. Ці дані зберігаються на сервері Grafana, що дозволяє забезпечити стабільну роботу системи, враховуючи права доступу, стан роботи сеансів та інші параметри.

Grafana підтримує кілька механізмів автентифікації для забезпечення безпечного доступу до системи, включаючи LDAP, OAuth та базову автентифікацію. Ці механізми дозволяють інтегрувати Grafana з існуючими

системами управління доступом. Авторизація користувачів здійснюється через налаштування ролей і дозволів, що регулюють доступ до інформаційних панелей і джерел даних. Контроль доступу реалізований на основі ролей (RBAC), що дозволяє адміністраторам встановлювати деталізовані дозволи для інформаційних панелей, папок і джерел даних. Ця система гарантує, що кожен користувач має доступ лише до тих ресурсів, які дозволені відповідно до його ролі, забезпечуючи тим самим безпеку та контроль у багатокористувацькому середовищі.

За допомогою встроєних інструментів Grafana надає можливість експорту та імпорту вже налаштованих дашбордів у json форматі.

Metabase — це інструмент з відкритим вихідним кодом, призначений для візуалізації та дослідження даних, який дозволяє користувачам підключатися до своїх джерел даних, створювати інтерактивні інформаційні панелі та легко отримувати статистичні дані. Інструмент орієнтований на зручність використання, що робить його доступним для користувачів із різним рівнем технічної підготовки.

Система підтримує створення динамічних і інтерактивних інформаційних панелей, що включають діаграми, графіки, таблиці та фільтри. Такий підхід дозволяє гнучко досліджувати дані та аналізувати їх із різних аспектів.

Metabase інтегрується з багатьма джерелами даних, серед яких MySQL, PostgreSQL, MongoDB, ClickHouse і Google BigQuery [31].

Система дозволяє виконувати деталізоване дослідження даних із використанням фільтрації, групування та агрегації, що сприяє швидкому отриманню аналітичної інформації та виявленню трендів.

Інтерфейс користувача спроєктований для забезпечення простоти у використанні та мінімізації часу навчання нових користувачів. Елементи інтерфейсу є зрозумілими.

Для забезпечення ефективної взаємодії та колективний аналіз даних, Metabase дозволяє ділитися створеними інформаційними панелями та звітами між членами команди чи іншими зацікавленими сторонами.

Порівняння Metabase та Grafana

Grafana є самостійним додатком, орієнтованим на візуалізацію даних часових рядів із різноманітних джерел. Система забезпечує широкий спектр функцій для візуалізації, відома своєю гнучкістю та можливістю розширення. Metabase, навпаки, являє собою легкий додаток, розроблений для спрощення дослідження та аналізу даних, зокрема для користувачів без технічної підготовки. Вона пропонує інтуїтивно зрозумілий інтерфейс із обмеженими можливостями налаштування, що робить її придатною для швидкого аналізу [32].

Grafana підтримує широкий спектр джерел даних, включаючи бази даних, хмарні платформи та інструменти моніторингу. Система пропонує плагіни та API для інтеграції з різними джерелами, що дозволяє створювати складні візуалізації та інформаційні панелі. Metabase підтримує обмежену кількість джерел даних "з коробки" і переважно орієнтована на підключення до реляційних баз даних та виконання простих запитів, часто без потреби писати SQL-код.

Grafana пропонує гнучкий і настроюваний інтерфейс користувача з функцією перетягування елементів, що дозволяє створювати інтерактивні панелі з графіками, таблицями та сповіщеннями. Система включає багатий набір функцій для аналізу, фільтрації даних та деталізації. Metabase забезпечує простіший інтерфейс, орієнтований на аналітику самообслуговування, пропонуючи попередньо визначені шаблони візуалізацій для швидкого отримання даних без необхідності технічних навичок.

З точки зору безпеки, Grafana надає розширені функції контролю доступу, включаючи автентифікацію кількох користувачів, деталізований контроль доступу та інтеграцію із зовнішніми постачальниками ідентифікаційних даних. Таким чином можна точно визначати, хто може отримувати доступ до певних панелей чи джерел даних. Metabase має обмежені функції безпеки й більше підходить для невеликих команд або організацій, де немає потреби в строгому контролі доступу.

Grafana має обмежені можливості для розширеної аналітики та машинного навчання, зосереджуючись переважно на візуалізації даних часових рядів.

Metabase, у свою чергу, не підтримує розширену аналітику або машинне навчання, оскільки орієнтована на базовий аналіз і візуалізацію даних.

Grafana є гнучким та розширюваним інструментом візуалізації даних, який підтримує широкий спектр джерел даних і пропонує розширені можливості аналітики, що робить його придатним для складних бізнес-завдань. Metabase, навпаки, є простим і легким у використанні інструментом, орієнтованим на нетехнічних користувачів із обмеженими потребами в налаштуванні. Оскільки Grafana пропонує більшу гнучкість і набагато більші можливості, то цей інструмент є кращим для глибокого аналізу і подальшої підтримки або розширення проєкту.

4.7. Обґрунтування вибору використання брокера повідомлень

Для покращення продуктивності системи збору, зберігання та обробки даних доцільно впровадити RabbitMQ - популярна система черг повідомлень, яка забезпечує асинхронну комунікацію між різними компонентами проєкту [33]. RabbitMQ дозволяє налаштувати обмін повідомленнями між сервісами без потреби чекати на відповідь одного сервісу перед обробкою іншого. Це дозволить підвищити ефективність і швидкість роботи системи, адже дані можуть збиратися, зберігатися та оброблятися одночасно, що особливо важливо при роботі з великими обсягами інформації. Також RabbitMQ сприяє масштабуванню системи. Якщо потрібно обробляти велику кількість запитів, RabbitMQ дозволяє розподілити навантаження між кількома сервісами, що робить систему більш надійною. Важливим аспектом застосування брокера повідомлень є необхідність контролювати кількість запитів, відправлених до ЄДБО, щоб не перевищити ліміти та не бути заблокованими. З використанням RabbitMQ можна створити архітектуру, яка легко адаптується до змін у вимогах до продуктивності, забезпечуючи гнучкість у налаштуваннях.

4.8. Загальна структура проєкту

Загалом, проєкт має багаторівневу структуру, яка забезпечує ефективне розгортання та функціонування системи аналізу даних вступної кампанії з ЄДБО. Вся система розгортається на віртуальній машині за допомогою Vagrant та VirtualBox.

Oracle VirtualBox — це програмний інструмент для створення та управління віртуальними машинами. Він працює як гіпервізор, що дозволяє запускати кілька операційних систем на одному фізичному комп’ютері. VirtualBox підтримує архітектуру x86 і може віртуалізувати різні операційні системи, включаючи Windows, Linux, macOS та інші [34].

Vagrant — це інструмент із відкритим кодом, який спрощує створення, налаштування та управління віртуальними машинами для розробки й тестування програмних проєктів. У стандартній версії Vagrant працює з VirtualBox, але за допомогою плагінів його можна інтегрувати з іншими системами віртуалізації. Основною перевагою Vagrant є можливість створення повністю готового до роботи середовища розробки за допомогою єдиного конфігураційного файлу [35].

Основою проєкту є директорія, яка інтегрується у віртуальну машину. У середовищі віртуальної машини налаштовані скрипти для автоматизованого розгортання Docker-контейнерів, таких як Clickhouse, RabbitMQ та Grafana.

Docker — це програмне забезпечення з відкритим кодом, яке є найпопулярнішою платформою для управління контейнерами. Воно забезпечує ізольоване середовище для запуску додатків, дозволяючи розробникам створювати, розгортати та масштабувати програмне забезпечення незалежно від операційної системи. Контейнери Docker містять усі необхідні компоненти для роботи додатка, такі як бібліотеки та залежності, що гарантує стабільність роботи програмного забезпечення в будь-якому середовищі.

Docker спрощує використання системних ресурсів, дозволяє швидко розгортати та тестувати додатки, а також забезпечує можливість безпечного одночасного запуску кількох контейнерів на одному хості [36].

У окремій директорії проєкту зберігаються Python-скрипти, які виконують функції продюсера та консюмера. Продюсер відповідає за парсинг посилань на конкурсні пропозиції з ЄДБО та відправку їх у чергу RabbitMQ. Консюмер, у свою чергу, отримує ці дані з черги, збирає всі відомості про конкурсну пропозицію та заяви вступників, здійснює їх обробку та зберігає у базі даних Clickhouse. Такий підхід дозволяє розподілити навантаження та забезпечити масштабованість системи при обробці великої кількості даних.

4.9. Висновок по четвертому розділу

У розділі обґрунтовано вибір інструментів, використаних у системі. Clickhouse використовується як високопродуктивна колонкова база даних для зберігання та швидкого запиту великих обсягів даних. RabbitMQ слугує системою обміну повідомленнями. Це забезпечує асинхронну взаємодію між компонентами системи, такими як продюсер та консюмер. Grafana використовується для візуалізації даних та моніторингу метрик, даючи можливість аналізувати результати в режимі реального часу.

5. АНАЛІЗ І УЗАГАЛЬНЕННЯ РЕЗУЛЬТАТІВ ДОСЛІДЖЕННЯ

У даному розділі здійснюється оцінка роботи створеної системи, яка автоматизує збір і аналіз даних про вступні кампанії до закладів вищої освіти України. Акцент зроблено на аналізі зібраних даних, точності отриманої інформації, продуктивності системи та відповідності функціоналу системи заданим вимогам.

5.1. Аналіз результатів роботи системи

5.1.1. Оцінка ефективності автоматизованого збору даних

Застосування автоматизації дозволило обробити всі конкурсні пропозиції, доступні на платформі ЄДБО за один рік, приблизно за 20 годин. Швидкість обробки досягає від 4 до 6 тисяч посилань на конкурсні пропозиції щогодини. Середній обсяг конкурсних пропозицій за один рік становить близько 95 тисяч, що вимагає значного обсягу обчислювальних ресурсів для забезпечення стабільної роботи системи. Зокрема, з цих конкурсних пропозицій витягується близько 1.4 мільйона заяв на рік. Такі показники свідчать про високу масштабованість системи та її здатність працювати з великими обсягами інформації у короткі терміни. При одночасній роботі декількох консюмерів система демонструє високу стабільність. Наприклад, під час роботи 10 консюмерів, кількість посилань, шо обробляється, коливається від 1.4 до 1.8 посилань щосекунди. Це забезпечує рівномірний розподіл навантаження між компонентами системи та дозволяє досягти оптимальної швидкості обробки даних навіть при значному збільшенні обсягу інформації.

Завдяки вбудованому графічному інтерфейсу RabbitMQ користувачі мають змогу в реальному часі відстежувати кількість оброблених посилань на конкурсні пропозиції. Ця функціональність є корисною для моніторингу роботи системи та оперативного реагування на можливі проблеми. На рисунку 5.1 представлено графічний інтерфейс RabbitMQ, що дозволяє контролювати роботу парсера в процесі обробки даних.

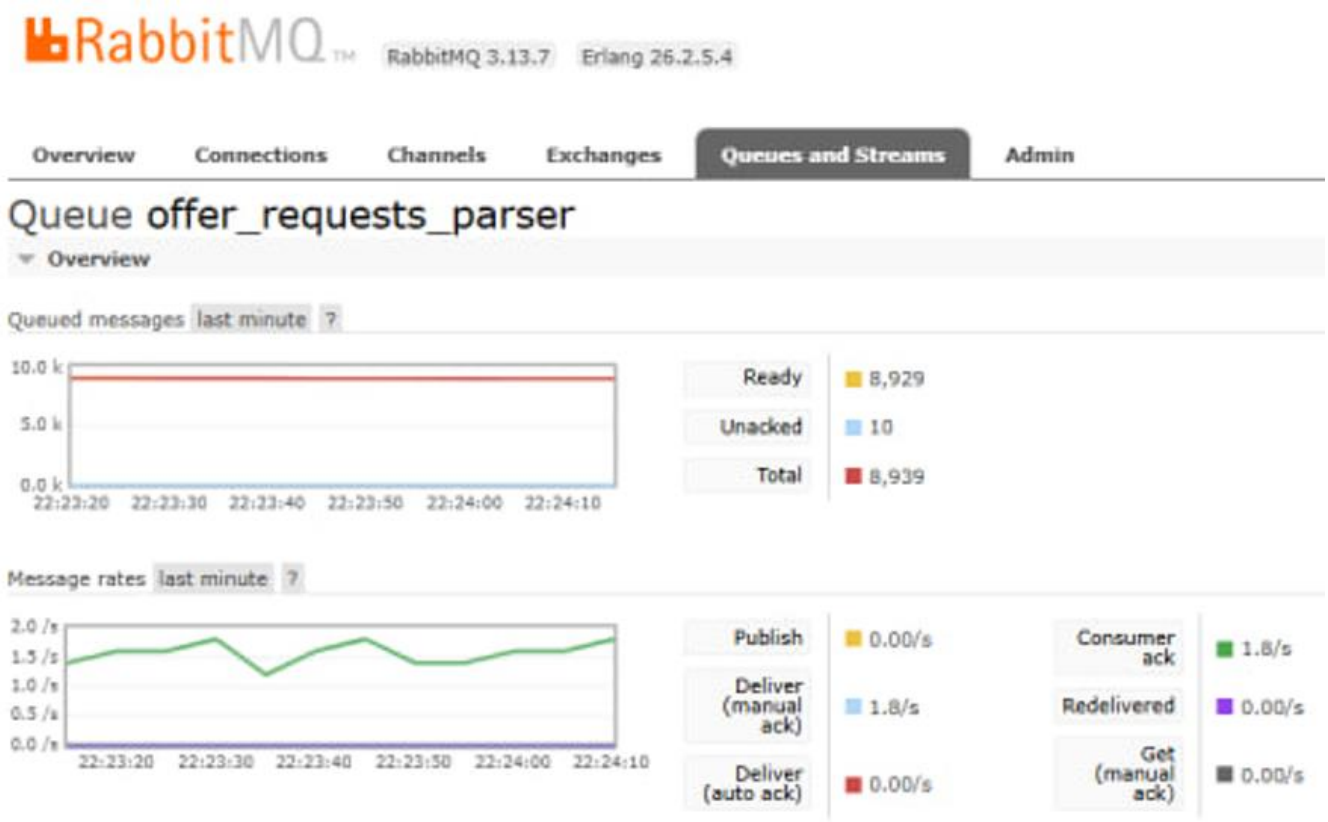

Рисунок 5.1 – Графічний інтерфейс RabbitMQ для відслідковування роботи парсеру

5.1.2. Оцінка продуктивності бази даних

Оцінка часу виконання запитів

Для оцінки продуктивності бази даних ClickHouse проведено аналіз виконання типового запиту. Поданий запит виконує групування та агрегацію даних за кількома полями з використанням умов фільтрації.

SQL запит:

```
SELECT count(1), speciality_name, university_code, degree, year
FROM offer_request
WHERE degree = 'Бакалавр'
AND admission_basis = 'Повна загальна середня освіта'
GROUP BY speciality_name, university_code, degree, year;
```

Результати виконання:

- кількість рядків у результаті: 18,092;
- оброблено рядків: 5.91 мільйона;
- обсяг оброблених даних: 863.38 МБ;
- час виконання: 0.498 секунди;
- швидкість обробки: 11.86 млн рядків/сек, 1.73 ГБ/сек;
- пікове використання пам’яті: 8.19 МіБ.

Час обробки запиту склав менше 0.5 секунди для ~6 млн рядків, що свідчить про можливість виконання запитів у реальному часі навіть за значних обсягів даних, а швидкість у 1.73 ГБ/сек підтверджує високу продуктивність системи. Мінімальне використання пам’яті забезпечує стабільну роботу навіть за обмежених ресурсів, що дозволяє застосовувати ClickHouse для аналітичних запитів із подібною структурою без потреби у складній оптимізації.

Оцінка обсягу збереження даних

Для аналізу обсягу збережених даних у таблиці offer_request в базі даних edu_campaign виконано запит, який обчислює загальний розмір таблиці на диску.

SQL - запит:

```
SELECT
`table`,
formatReadableSize(sum(bytes_on_disk)) AS size_on_disk
FROM system.parts
WHERE (database = 'edu_campaign') AND (`table` = 'offer_request')
GROUP BY `table`
```

Результати виконання:

- таблиця: offer_request;
- розмір на диску: 211.54 MiB;

- час виконання: 0.006 секунди;
- оброблено рядків: 4 (метадані частин таблиці);
- обсяг даних, оброблених запитом: 204 байти;
- пікове використання пам’яті: 203.34 КіБ.

Можна зробити висновок, що таблиця offer_request має компактний розмір (211.54 MiB) завдяки ефективному стисненню даних, що дозволяє зберігати мільйони записів із мінімальним використанням дискового простору.

Оцінка рівня стисненості

Для оцінки ефективності стиснення даних у таблиці offer_request бази даних edu_campaign виконано запит, що аналізує розміри стиснутих і нестиснутих даних для кожного стовпця, а також коефіцієнт стиснення.

SQL запит:

```
SELECT
   name,
   formatReadableSize(sum(data_compressed_bytes)) AS compressed_size,
   formatReadableSize(sum(data_uncompressed_bytes)) AS uncompressed_size,
   sum(data_uncompressed_bytes) / sum(data_compressed_bytes) AS compression_ratio
FROM system.columns
WHERE (database = 'edu_campaign') AND (`table` = 'offer_request')
GROUP BY name ORDER BY compressed_size DESC;
```

Результати виконання:

- час виконання: 0.005 секунд;
- оброблено рядків: 36 (стовпці таблиці);
- розмір оброблених даних: 2.83 КіБ;
- пікове використання пам’яті: 217.96 КіБ.

Порівння стиснення окремих колонок наведено в таблиці 5.1.

Таблиця 5.1. Порівняння стиснення окремих колонок

Стовпець	**Стиснений розмір**	**Нестиснений розмір**	**Коефіцієнт стиснення**
education_price	965.80 КіБ	22.53 МіБ	23.89x
speciality_code	946.07 КіБ	22.53 МіБ	24.39x
village_coefficient	912.03 КіБ	22.53 МіБ	25.30x
contract_capacity	892.33 КіБ	22.53 МіБ	25.86x
degree	886.73 КіБ	127.49 МіБ	147.23x
university_region	822.47 КіБ	173.00 МіБ	215.39x
study_form	819.24 КіБ	63.58 МіБ	79.47x
offer_name	7.95 МіБ	478.97 МіБ	60.24x
scores	69.06 МіБ	651.21 МіБ	9.43x
status	6.56 МіБ	185.87 МіБ	28.31x
industry_coefficient	580.91 КіБ	22.53 МіБ	39.72x
speciality_sub_name	568.94 КіБ	50.48 МіБ	90.86x
education_program	5.18 МіБ	347.99 МіБ	67.22x
course	436.56 КіБ	61.96 МіБ	145.33x
student_scores_hash	32.55 МіБ	45.06 МіБ	1.38x
max_state_order	317.96 КіБ	22.53 МіБ	72.56x
quota	315.23 КіБ	3.64 МіБ	11.83x
university_name	3.54 МіБ	722.15 МіБ	203.71x
priority	3.51 МіБ	14.78 МіБ	4.22x

Продовження таблиці 5.1. Порівняння стиснення окремих колонок

Стовпець	**Стиснений розмір**	**Нестиснений розмір**	**Коефіцієнт стиснення**
faculty	3.26 МіБ	399.91 МіБ	122.76x
speciality_name	3.09 МіБ	229.21 МіБ	74.15x
regional_coefficient	247.64 КіБ	22.53 МіБ	93.16x
subjects	23.52 МіБ	3.21 ГіБ	139.78x
request_number	2.42 МіБ	17.19 МіБ	7.11x
documents	2.22 МіБ	18.86 МіБ	8.50x
study_term	2.13 МіБ	207.04 МіБ	97.15x
offer_code	2.08 МіБ	42.66 МіБ	20.49x
avg_score	14.55 МіБ	22.53 МіБ	1.55x
student_name	14.33 МіБ	35.72 МіБ	2.49x
university_code	123.39 КіБ	21.65 МіБ	179.64x
year	101.81 КіБ	22.53 МіБ	226.61x
admission_basis	1.80 МіБ	257.33 МіБ	143.19x
application_term	1.41 МіБ	146.45 МіБ	104.08x
confirmation	1.30 МіБ	11.66 МіБ	8.99x
offer_type	1.18 МіБ	181.44 МіБ	153.86x
license_capacity	1.01 МіБ	22.53 МіБ	22.42x

На основі наведеної таблиці, можна зробити висновок, що ClickHouse демонструє високу ефективність стиснення для стовпців із низькою кардинальністю (наприклад, year, university_code), досягаючи співвідношення

стиснення понад 200x, тоді як стовпці з високою кардинальністю, як-от student_scores_hash і student_name, мають нижчу ефективність через унікальність даних.

5.1.3. Обсяг зібраної інформації

Під час виконання роботи було зібрано інформацію про вступні кампанії з 2021 по 2024 роки. Всього зібрано 5.9 мільйонів заяв вступників. Розподіл кількості заяв і конкурсних пропозицій по роках наведено в таблиці 5.2.

Таблиця 5.2. Розподіл кількості заяв та конкурсних пропозицій

Рік	**Кількість заяв**	**Кількість конкурсних пропозицій**
2021	1 735 345	92 064
2022	1 323 275	109 992
2023	1 444 182	117 608
2024	1 403 517	97 585

Кількість зібраних заяв була перевірена шляхом порівняння отриманих результатів із даними, представленими на веб-порталі “АбітПошук”. У ході перевірки була оцінена точність даних, зібраних автоматизованою системою, шляхом розрахунку відносної похибки для кожного року, що аналізувався.

Результати аналізу показали: відносна похибка між даними, отриманими системою, та даними “АбітПошуку” є мінімальною. Похибка розбіжностей по роках наведена у таблиці 5.3.

Таблиця 5.3. Відносна похибка розбіжності зібраних даних

Рік	**Відносна похибка**
2020	-
2021	0.08%
2022	0.16%
2023	6.00%
2024	4.82%

Згідно з отриманими даними, найбільша відносна похибка була зафіксована у 2023 році і становила 6%, тоді як найменша похибка спостерігалася у 2021 році — лише 0.08%. У 2024 році похибка склала 4.82%, а у 2022 році похибка склала лише 0.16%.

Водночас варто зауважити, що інформація про прізвища та ініціали вступників на ЄДБО доступна тільки за останній 2024 рік, це безпосередньо впливає на ідентифікацію кожного вступник, і в подальшому впливає на аналіз. На веб порталі “АбітПошук” інформація про прізвище та ініціали вступника доступна за всі роки, тому дослідження тенденцій в роках за даними, зібраними з цієї платформи є більш оптимальною. Дослідження та аналіз тенденцій в поточному році є більш оптимальним на основі даних зібраних з платформи ЄДБО, оскільки інформація на ній частіше оновлюється і є більш повною.

Дані про заяви зібрані в одну таблицю, яка відповідає вимогам першої нормальної форми: усі поля містять атомарні значення, без множинних або повторюваних груп. Наприклад, інформація про предмети і результати вступних випробувань структурована у форматі JSON у полях subjects і scores, що дозволяє уникнути дублювання колонок. Кожен запис ідентифікується унікальними комбінаціями ключових полів (year, university_code, offer_code, request_number),

що забезпечує однозначність даних. Структура інформації про заяву вступника описана в таблиці Додатку В.

Оцінка точності отриманих даних з платформ ЄДБО та “АбітПошук”

Так як, виявлено деяку розбіжність в отриманих даних за 2023 та 2024 роки, проведено дослідження для виявлення причин цієї розбіжності.

Дослідження полягало у групування даних за різними параметрами (такими як регіон, спеціальність, освітній ступінь та ін.) та визначенні, де саме спостерігається критична розбіжність. За основу взято вступну кампанію за 2023 рік.

Першим етапом порівняння було виявлення чи не пропущено заяви якогось регіону, тому створено запит для групування по регіонам і порівняно результати які наведено на рисунку 5.2.

Назва регіону	Кількість заяв в ЄДБО	Кількість заяв в Абіт Пошук
Івано-Франківська область	49811	47942
Волинська область	39487	37559
Вінницька область	52986	51082
Дніпропетровська область	102089	95900
Донецька область	14490	13083
Житомирська область	30658	29321
Закарпатська область	23484	21927
Запорізька область	42908	36795
Київська область	24326	21918
Кіровоградська область	13220	12396
Луганська область	11738	10000
Львівська область	169349	162265
Миколаївська область	20679	17787
Одеська область	82183	75820

Рисунок 5.2. - Порівняння розподілу кількості заяв по регіонах

Оскільки розбіжність спостерігається по всіх регіонах, то фактор регіону на неточність даних не впливає. Наступним фактором для порівняння обрано форму навчання. Результат порівняння наведено на рисунку 5.3.

Форма навчання	Кількість в ЄДБО	Кількість в Абіт Пошук
Вечірня	967	790
Денна	1252510	1196817
Дистанційна	3145	2743
Заочна	187560	157362

Рисунок 5.3. - Порівняння розподілу кількості заяв за формою навчання

З цього порівняння також видно, що кількість заяв суттєво відрізняється по всіх формах навчання. Тому цей фактор теж не впливає.

Наступним параметром було порівняння кількості заяв за освітніми ступенями. Результат цього порівняння наведено на рисунку 5.4.

Освітній ступінь	Кількість заяв з АбітПошук	Кількість заяв з ЄДЕБО
Бакалавр	809007	857315
Магістр	299806	329833
Фаховий молодший бакалавр	247547	257034
Молодший бакалавр	1352	

Рисунок 5.4. - Порівняння кількості заяв за освітніми ступенями

Оскільки спостерігається суттєва розбіжність по кількості заяв, то цей фактор також не можна брати як основу впливу на різницю кількості заяв.

Далі було проведено дослідження по кількості зарахованих абітурієнтів на контракт та бюджет. Результат наведено на рисунку 4.5.

Зараховані	Кількість зарахованих в Абіт Пошук	Кількість зарахованих в ЄДБО
До наказу (б)	176121	175745
До наказу (к)	249910	381115

Рисунок 5.5. - Порівняння кількості зарахованих абітурієнтів на контракт та бюджет

З цього порівняння видно, що показники кількості абітурієнтів зарахованих на бюджет майже збігаються, і відносна похибка лежить в межах 0.2%, що є в межах норми і вона спричинена відмовою вступників від бюджетних місць. А розбіжність по кількості зарахованих на контракт складає 34.4%. Це спричинено тим, що інформація, яка представлена у веб-порталі “АбітПошук” останній раз оновлена 01.09.2023р, а в ЄДБО - 01.01.2024р. Зважаючи на те, що на контракт абітурієнтів зараховують до кінця вересня, то певна кількість заяв не врахована, і відповідно статус певної кількості заяв був змінений.

5.1.4. Оцінка якості візуалізації даних

В результаті виконання роботи, було побудовано 2 дашборди що представляють загальну та більш деталізовану інформацію про розподіл заяв за різними параметрами. Візуалізації на цих дашбордах створені з акцентом на структуроване подання великих обсягів даних щодо розподілу заяв та зарахувань за галузями знань. Для легкого сприйняття ключової інформації, на візуалізаціях виведено числові та відсоткові показники категорій. Кольорова сегментація кругових діаграм допомагає візуально розрізняти категорії, такі як освітні галузі чи форми навчання, а відсоткове співвідношення додає контекст до числових значень.

Дашборди є інтерактивними завдяки впровадженню функціональних фільтрів, які забезпечують динамічний і адаптивний аналіз даних. Фільтри дозволяють здійснювати вибірку за ключовими параметрами, такими як рік, галузь знань, освітній ступінь, форма навчання та основа вступу та ін. Це забезпечує можливість швидкого налаштування відображення даних відповідно до дослідницьких завдань користувача. Інтерфейс фільтрів органічно інтегрований у

дашборд, підтримуючи баланс між інформативністю та зручністю використання. Функціонал фільтрів підвищує ефективність роботи з великими масивами даних, дозволяючи зосередитися на конкретних аспектах аналізу без необхідності переглядати зайву інформацію. Такий підхід сприяє структурованому та наочному поданню результатів, що відповідає сучасним вимогам до аналітичних інструментів.

Кругові діаграми і гістограми наочно демонструють популярність спеціальностей, розподіл зарахувань на бюджетні чи контрактні місця, а також конкуруючі галузі знань, що допомагає визначати найбільш привабливі напрямки для абітурієнтів. Дашборди дають змогу оцінити, як вступники розставляють пріоритети в заявах, що дозволяє університетам зрозуміти, які галузі знань і спеціальності конкурують між собою за одного й того ж вступника. Це особливо важливо для адаптації освітніх програм до ринкових трендів і вдосконалення маркетингових стратегій. А можливість аналізу трендів на ринку праці через розподіл бюджетних місць і популярність галузей знань дозволяє університетам вчасно коригувати свої освітні програми для підвищення конкурентоспроможності.

5.2. Опис роботи скрипта

В директорії проєкту parser/ представлені 4 основні скрипта для парсингу:

Скрипт main_form_parser_producer.py - генерує всі можливі кобінації із можливих освітніх рівнів, основ вступу та спеціальностей та передає їх в чергу RabbitMQ offer_links_parser. Для запуску цього скрипту необхідно в директорії використати команду: python3 main_form_parser_producer.py 'domain_name', де в якості domain_name треба передати доменне ім'я ресурсу, з якого здійснюватиметься парсинг, наприклад vstup2023.edbo.gov.ua.

Скрипт main_form_parser_consumer.py (Додаток Г) - запускає консюмера, який здійснює обробку повідомлень із черги offer_links_parser. Даний скрипт безпосередньо витягує повідомлення з черги із комбінаціями параметрів для запиту, і здійснює запит на сторінку із вказаними параметрами. На цій сторінці

скрипт збирає всі пропозиції від всіх університетів і зберігає посилання на них в базу даних в таблицю offer_link. Скрипт можна запустити безпосередньо командою python3 main_form_parser_consumer.py, що запустить 1 консюмера. Для запуску декількох консюмерів паралельно, можна скористатись командою ./run_main_form_consumers.sh що за замовч. запустить 10 консюмерів паралельно у фоновому режимі (кількість консюмерів можна редагувати, змінюючи параметр в циклі скрипту). Якщо було запущено консюмерів скриптом ./run_main_form_consumers.sh, то для зупинки консюмерів треба скористатись командою ./stop_main_form_consumers.sh, оскільки вони працюють у фоновому режимі. Якщо під час обробки повідомлення з черги виникла помилка, то консюмер перемістить це повідомлення в dead_letter_queue_offer_links. В подальшому можна перемісти ці повідомлення в чергу offer_links_parser для повторної обробки через інтерфейс RabbitMQ.

Скрипт offer_page_parser_producer.py - витягує всі посилання на пропозиці з БД за вказаним роком та передає їх в чергу RabbitMQ offer_request_parser. Для запуску цього скрипту необхідно в директорії використати команду: python3 offer_page_parser_producer.py 'year', де в якості year треба передати рік, наприклад 2024.

Скрипт offer_page_parser_consumer.py (Додаток Д) - запускає консюмера, який здійснює обробку повідомлень із черги offer_requests_parser. Даний скрипт безпосередньо витягує повідомлення з черги і здійснює запит на сторінку пропозиції. На цій сторінці скрипт збирає всі заявки та відомості про пропозицію та зберігає їх в БД в таблицю offer_request. Скрипт можна запустити безпосередньо командою python3 offer_page_parser_consumer.py, що запустить 1 консюмера. Для запуску декількох консюмерів паралельно, можна скористатись командою ./run_offer_page_consumers.sh що за замовч. запустить 10 консюмерів паралельно у фоновому режимі (кількість консюмерів можна редагувати, змінюючи параметр в циклі скрипту). Якщо було запущено консюмерів скриптом ./run_offer_page_consumers.sh, то для зупинки консюмерів треба скористатись командою ./stop_offer_page_consumers.sh, оскільки вони працюють у фоновому

режимі. Якщо під час обробки повідомлення з черги виникла помилка, то консюмер перемістить це повідомлення в dead_letter_queue_offer_requests. В подальшому можна перемісти ці повідомлення в чергу offer_requests_parser для повторної обробки через інтерфейс RabbitMQ.

5.3. Аналіз отриманих даних з дашборду

Проведено аналіз першого дашборду (Додаток А), що демонструє загальний розподіл кількості заяв по галузях знань, вступу на бакалаврат на основі повної загальної середньої освіти за 2024 рік. Загальна кількість заяв становить 649840, із яких 177055 подано на бюджетну форму навчання, а 130843 — на контрактну форму. Найбільшу кількість заяв отримали галузі “Соціальні науки, журналістика та інформація”, “Бізнес, адміністрування та право”, а також “Інформаційні технології”, що свідчить про їхню високу популярність серед абітурієнтів. Особливо помітний інтерес до спеціальностей, пов’язаних із сучасними напрямками, такими як інформаційні технології.

Галузь “Інформаційні технології” демонструє стабільно високий попит. Значна частина заяв із першим пріоритетом подана саме на спеціальності цього напряму (14% від всіх заяв), що вказує на те, що абітурієнти вважають інформаційні технології перспективним вибором для побудови кар’єри.

Інші популярні галузі, такі як “Бізнес, адміністрування та право” та “Соціальні науки”, також демонструють високий попит, але значна частина зарахувань припадає на контрактну форму навчання, що вказує на комерціалізацію цих напрямів. Галузі, пов’язані з інженерією, природничими науками та культурою, мають нижчу частку заяв із першим пріоритетом, але за рахунок більшої кількості бюджетних місць вони залишаються важливими для державної стратегії розвитку.

На основі аналізу другого дашборду (Додаток Б), що деталізує інформацію саме по конкретній галузі (в моєму випадку “Інформаційні технології”, вступ на бакалаврат на основі повної загальної середньої освіти), можна зробити наступні висновки.

Спеціальності в галузі “Інформаційні технології” є одними з найпопулярніших серед абітурієнтів. 35% заяв у цій галузі припадають на спеціальність “Комп'ютерні науки”, що свідчить про її найбільшу привабливість. Інші напрямки, такі як “Інженерія програмного забезпечення” (24%), “Кібербезпека та захист інформації” (19%) та “Комп'ютерна інженерія” (11%), також демонструють високий попит. Це вказує на те, що абітурієнти усвідомлюють перспективність цих спеціальностей для майбутнього працевлаштування.

На бюджетну форму навчання за спеціальностями “Інформаційних технологій” припадає 15% всіх бюджетних місць серед галузей знань. Це свідчить про важливість цієї галузі для держави, проте, порівняно з “Інженерією, виробництвом та будівництвом” (26%), кількість бюджетних місць для ІТ-спеціальностей залишається відносно меншою. Водночас, така пропорція створює жорстку конкуренцію серед абітурієнтів, які прагнуть отримати освіту за кошти державного бюджету.

Великий відсоток абітурієнтів, зарахованих на контрактну форму навчання, припадає на спеціальності ІТ. На цю галузь припадає 10% усіх контрактних місць, що свідчить про готовність абітурієнтів інвестувати в освіту у сфері інформаційних технологій.

Дані дашборда показують, що абітурієнти, які обирають спеціальності з галузі "Інформаційні технології", також розглядають альтернативи. Найбільше перетинаються вибори з галузями “Інженерія, виробництво та будівництво” (25%), “Бізнес, адміністрування та право” (22%), а також “Соціальні науки, журналістика та інформація” (17%). Це свідчить про те, що вступники бачать у цих напрямах схожі можливості для кар'єрного зростання та професійної реалізації.

5.4. Узагальнення отриманих результатів

Результати проведеного дослідження підтверджують, що поставлена наукова проблема, пов’язана з відсутністю ефективних підходів до аналізу даних вступних кампаній, орієнтованих на потреби університетів, була успішно вирішена. Створена система забезпечує університети інструментами для глибокого

аналізу вступної кампанії, дозволяючи інтегрувати дані з різних джерел, таких як ЄДБО та “АбітПошук”, що відкриває можливості для глибокого аналізу розподілу заяв по галузям знань, дослідження зміни розподілу бюджетних та контрактних місць по галузям знань та окремим спеціальностям та впливу інших показників на вибір закладу навчання. Це надасть змогу коригування освітніх програм відповідно до інтересів абітурієнтів та підвищення конкурентоспроможності закладу освіти.

У порівнянні з існуючими платформами, такими як ЄДБО та “АбітПошук”, розроблена система має значні переваги. Вона дозволяє здійснювати гнучке налаштування параметрів аналізу, включаючи сегментацію за галузями знань, пріоритетами вступників та іншими ключовими критеріями. Додатковою перевагою є візуалізація даних, яка сприяє швидкій інтерпретації результатів, виявленню прихованих закономірностей та прийняттю стратегічно обґрунтованих рішень. На відміну від узагальнених даних, які пропонуються існуючими платформами, система надає деталізовану інформацію, що дозволяє університетам адаптувати свої маркетингові стратегії та освітні програми.

Попри досягнуті результати, розроблена система має певні обмеження. Її ефективність залежить від доступності якісних і актуальних даних. Оскільки для збору даних використовується метод HTML-парсингу, отримання данних напряму залежить від незмінності структури веб сторінок ЄДБО. В разі змін, може виникати необхідність коригування парсеру.

Таким чином, створена система відповідає стратегічним завданням сучасних університетів і сприяє вирішенню ключових викликів, пов’язаних із аналізом ринку освітніх послуг. Вона може бути ефективно використана не лише окремими закладами, але й на державному рівні, забезпечуючи університети необхідними інструментами для прийняття обґрунтованих управлінських рішень і підвищення якості освітніх послуг.

5.5. Висновок по п’ятому розділу

У цьому розділі проведено оцінку зібраних даних і роботи системи. Система продемонструвала високу продуктивність і стабільність, забезпечуючи обробку великих обсягів інформації у короткі терміни. Використання бази даних ClickHouse

дозволило досягти швидкого виконання запитів і ефективного стиснення даних. Інтерактивні дашборди забезпечують зручний інструмент для аналізу даних. Отримані результати підтверджують відповідність системи поставленим вимогам, попри залежність від можливих змін у структурі джерел даних.

ВИСНОВКИ

У ході виконання кваліфікаційної роботи було досягнуто поставлену мету: розроблено аналітичну систему для аналізу результатів вступних кампаній вищих навчальних закладів України на основі відкритих даних ЄДБО. Розроблено комплексний підхід до аналізу даних результатів вступних кампаній вищих навчальних закладів України з урахуванням специфіки процесу вступу. Розроблені підходи базуються на поєднанні методів аналізу даних, таких як описовий, діагностичний, прогностичний аналізи, кластерний аналіз та аналіз часових рядів. Застосування комплексного підходу дозволяє університетам глибоко аналізувати ключові показники вступу, визначати тенденції попиту на галузі знань та спеціальності, виявляти фактори, що впливають на успішність зарахування, та прогнозувати майбутні зміни в освітньому середовищі.

На основі розроблених підходів було створено аналітичну систему для аналізу результатів вступних кампаній на основі відкритих даних ЄДБО. Система відповідає актуальним вимогам цифровізації управлінських процесів і дозволяє університетам оперативно аналізувати ключові показники вступу, визначати тенденції попиту на спеціальності та адаптувати освітні програми до потреб абітурієнтів та ринку праці.

У процесі роботи було розглянуто та обґрунтовано методологію розробки системи, яка охоплює всі ключові аспекти — від збору даних і їх структурування до побудови ефективної системи візуалізації. Основою для розробки стали сучасні інструменти роботи з великими обсягами даних, такі як HTML-скрапінг, високопродуктивні бази даних (ClickHouse) та інтерактивні дашборди (Grafana). Методологічний підхід передбачав інтеграцію збору, обробки та аналізу даних у єдину систему, що забезпечує стабільну роботу навіть за умов значного навантаження.

У процесі дослідження були реалізовані наступні завдання:

- використання методів HTML-скрапінгу для автоматизованого збору даних з відкритих джерел, що забезпечило ефективність збору великого обсягу інформації;
- розроблено базу даних на основі ClickHouse, яка дозволяє зберігати, обробляти та аналізувати великий обсяг структурованих даних з високою швидкістю;
- впроваджено інтерактивні дашборди на базі Grafana, які забезпечують зручну візуалізацію даних і гнучкі можливості аналізу для адміністрації університетів.

Система орієнтована на потреби університетів і допомагає розв’язувати ключові завдання, пов’язані з маркетинговими стратегіями, коригуванням освітніх програм і плануванням вступних кампаній. Її функціонал дозволяє університетам оперативно реагувати на зміни, підвищувати свою привабливість серед абітурієнтів і адаптуватися до глобальних освітніх стандартів.

У контексті сучасних економічних умов, коли університети змушені працювати в умовах жорсткої конкуренції, запропонована система може допомогти закладам вищої освіти зберігати конкурентоспроможність завдяки глибокому аналізу ключових показників. Умови реформування української освітньої системи, зокрема скорочення кількості університетів, ще більше підкреслюють важливість таких систем для оцінки попиту, якості освітніх послуг і ефективності стратегій залучення абітурієнтів.

Розроблена система є потужним інструментом для прийняття стратегічних рішень, сприяє підвищенню ефективності освітнього процесу та дозволяє університетам не лише залишатися конкурентоспроможними, але й стати провідними центрами розвитку інтелектуального капіталу в межах глобальної інституційної екосистеми. Її подальший розвиток та інтеграція з іншими інформаційними платформами дозволять ще більше розширити можливості аналізу та прогнозування освітніх процесів.

ВИКОРИСТАНА ЛІТЕРАТУРА

1. Вісник Національної академії педагогічних наук України. URL: https://visnyk.naps.gov.ua/index.php/journal/article/download/113/144/ (дата звернення: 01.10.2024).
2. ДонПатріот. URL: https://donpatriot.news/rezultaty-matchiv-13-turu-anglijskoyi-futbolnoyi-prem%ca%bcyer-ligy (дата звернення: 17.10.2024).
3. Реєстр суб'єктів освітньої діяльності. URL: https://registry.edbo.gov.ua/ (дата звернення: 12.10.2024).
4. АбітПошук. URL: https://abit-poisk.org.ua/ (дата звернення: 12.10.2024).
5. Simplilearn.com. What Is Data Analysis: Examples, Types, & Applications URL: https://www.simplilearn.com/data-analysis-methods-process-types-article (дата звернення: 14.10.2024).
6. Step Cloud. Що таке API: основи роботи та використання програмістами. URL: https://cloud.itstep.org/blog/what-is-an-api-why-is-it-used-by-programmers-and-the-basics-of-working-with-it (дата звернення: 20.10.2024).
7. Що таке парсинг даних: визначення, переваги та проблеми. URL: https://ltesocks.io/ua/blog-ua/shho-take-parsing-danih-viznachennya-perevagi-ta-problemi/ (дата звернення: 22.10.2024).
8. Medium. Переваги та недоліки веб-скрапінгу. URL: https://raluca-p.medium.com/why-web-scraping-a-full-list-of-advantages-and-disadvantages-fdbb9e8ed010 (дата звернення: 21.10.2024).
9. FreeHost. Що таке PHP. URL: https://freehost.com.ua/ukr/faq/wiki/chto-takoe-php/ (дата звернення: 26.10.2024).
10. PHP: Manual. URL: https://www.php.net/manual/uk/introduction.php (дата звернення: 28.10.2024).
11. ScrapFly. Веб-скрапінг на PHP: основи. URL: https://scrapfly.io/blog/web-scraping-with-php-101/ (дата звернення: 28.10.2024).

12. GeeksForGeeks. Переваги та недоліки PHP. URL: https://www.geeksforgeeks.org/advantages-and-disadvantages-of-php/ (дата звернення: 28.10.2024).
13. Wikipedia. JavaScript. URL: https://uk.wikipedia.org/wiki/JavaScript (дата звернення: 29.10.2024).
14. Mozilla Developer Network. JavaScript. URL: https://developer.mozilla.org/en-US/docs/Web/JavaScript (дата звернення: 29.10.2024).
15. Zyte. Методи веб-скрапінгу. URL: https://www.zyte.com/blog/web-scraping-methods-you-need-to-know/ (дата звернення: 30.10.2024).
16. GeeksForGeeks. Переваги та недоліки JavaScript. URL: https://www.geeksforgeeks.org/advantages-and-disadvantages-of-javascript/ (дата звернення: 29.10.2024).
17. Wikipedia. Python. URL: https://uk.wikipedia.org/wiki/Python (дата звернення: 30.10.2024).
18. Import.io. Переваги та недоліки Python для веб-скрапінгу. URL: https://www.import.io/post/python-web-scraping-what-are-the-pros-and-cons (дата звернення: 30.10.2024).
19. ScrapFly. Playwright vs Selenium. URL: https://scrapfly.io/blog/playwright-vs-selenium/ (дата звернення: 03.11.2024).
20. ZenRows. Порівняння Playwright та Selenium. URL: https://www.zenrows.com/blog/playwright-vs-selenium#ease-of-use (дата звернення: 05.11.2024).
21. BrowserStack. Playwright vs Selenium. URL: https://www.browserstack.com/guide/playwright-vs-selenium (дата звернення: 04.11.2024).
22. DOU. Форум про Playwright. URL: https://dou.ua/forums/topic/45780/ (дата звернення: 03.11.2024).
23. Chromium. Інформація про Blink. URL: https://www.chromium.org/blink/ (дата звернення: 05.11.2024).

24. Wikipedia. Mozilla Firefox. URL: https://uk.wikipedia.org/wiki/Mozilla_Firefox (дата звернення: 05.11.2024).
25. WebKit. Офіційний сайт WebKit. URL: https://webkit.org/ (дата звернення: 05.11.2024).
26. Atlan. Що таке колоночна база даних. URL: https://atlan.com/what-is/columnar-database/ (дата звернення: 07.11.2024).
27. TechTarget. Визначення колоночної бази даних. URL: https://www.techtarget.com/searchdatamanagement/definition/columnar-database (дата звернення: 07.11.2024).
28. Fast Open-Source OLAP DBMS - ClickHouse?. What Is ClickHouse? | ClickHouse Docs URL: https://clickhouse.com/docs/en/intro (дата звернення: 07.11.2024).
29. Medium. Вступ до Grafana. URL: https://medium.com/@MetricFire/what-is-grafana-8de44d241765 (дата звернення: 08.11.2024).
30. LinkedIn. Переваги та недоліки Grafana. URL: https://www.linkedin.com/pulse/pros-cons-different-tools-grafana-craig-risi-phlwf (дата звернення: 08.11.2024).
31. Medium. Метабейс: посібник для початківців. URL: https://medium.com/@techlatest.net/introduction-to-metabase-a-beginner-friendly-guide-to-metabase-an-open-source-data-visualization-a248ee46bcf6 (дата звернення: 09.11.2024).
32. StackShare. Порівняння Grafana і Metabase. URL: https://stackshare.io/stackups/grafana-vs-metabase (дата звернення: 10.11.2024).
33. DigitalOcean. Документація RabbitMQ. URL: https://docs.digitalocean.com/products/marketplace/catalog/rabbitmq/ (дата звернення: 05.11.2024).
34. Wikipedia. VirtualBox. URL: https://en.wikipedia.org/wiki/VirtualBox (дата звернення: 09.10.2024).

35. Wikipedia. Vagrant. URL: https://uk.wikipedia.org/wiki/Vagrant (дата звернення:07.10.2024).
36. QA Group. Що таке Docker і навіщо він потрібен. URL: https://qagroup.com.ua/publications/shcho-take-docker-i-navishcho-vin/ (дата звернення: 15.10.2024).

ДОДАТКИ

Додаток А. Дашборд розподілу заяв по галузях

Додаток Б. Дашборд деталізованої інформації про розподіл заяв конкретної галузі

Додаток В. Структура інформації про заяву вступника

Поле	Опис
year	Рік вступу до закладу вищої освіти.
university_code	Унікальний код університету в системі.
university_name	Назва закладу вищої освіти.
degree	Рівень освіти, наприклад, "Бакалавр", "Магістр".
admission_basis	Основа вступу, наприклад, "Повна загальна середня освіта".
speciality_code	Код спеціальності згідно з державним класифікатором.
speciality_name	Назва спеціальності, наприклад, "Маркетинг".
speciality_sub_name	Додаткова або уточнююча назва спеціальності (може бути порожнім).
offer_name	Назва освітньої пропозиції, наприклад, "Маркетинг".
offer_code	Унікальний код освітньої пропозиції.
education_program	Назва освітньої програми, наприклад, "Маркетинг".
faculty	Назва факультету або підрозділу, наприклад, "Економічний факультет".
study_form	Форма навчання, наприклад, "Денна" або "Заочна".
course	Курс зарахування на навчання, наприклад, "1 курс".
offer_type	Тип освітньої пропозиції, наприклад, "Відкрита, з пріоритетом".
study_term	Термін навчання, наприклад, "01.09.2024 – 30.06.2028".
application_term	Період подання заяв, наприклад, "19.07.2024 – 31.07.2024".
license_capacity	Ліцензійний обсяг спеціальності (загальна кількість місць).

Поле	Опис
max_state_order	Максимальна кількість місць державного замовлення.
contract_capacity	Кількість місць за контрактом (платна форма навчання).
education_price	Вартість навчання за рік у гривнях.
subjects	Список предметів із коефіцієнтами та мінімальними балами для вступу.
regional_coefficient	Регіональний коефіцієнт для обчислення конкурсного бала.
village_coefficient	Сільський коефіцієнт для обчислення конкурсного бала.
industry_coefficient	Галузевий коефіцієнт для обчислення конкурсного бала.
quota	Тип квоти, наприклад, "Квота 2".
request_number	Унікальний номер заяви вступника.
student_name	Код заяви та ПІБ вступника.
status	Статус заяви, наприклад, "деактивовано (зарах. на бюджет)".
priority	Пріоритетність заяви вступника (1 – найвищий пріоритет).
confirmation	Підтвердження вступником вибору цієї пропозиції (Yes/No).
documents	Наявність поданих документів (Yes/No).
avg_score	Середній конкурсний бал вступника.
scores	Результати ЗНО (або інших іспитів) за предметами, наприклад, "Українська мова: 145".
university_region	Регіон розташування закладу освіти, наприклад, "Івано-Франківська область".

Додаток Г. Код скрипту, що збирає та формує посилання конкурсних пропозицій

/main_form_parser_producer.py

```
import argparse
import pika
import json
import logging
from itertools import product
from bs4 import BeautifulSoup
import requests

logging.basicConfig(
    level=logging.INFO,
    format='%(asctime)s - %(levelname)s - %(message)s',
    handlers=[
        logging.FileHandler("logs/main_form_parser_producer.log", encoding='utf-8'),
    ]
)

def parse_args():
    parser = argparse.ArgumentParser(description="Парсер пропозицій університетів")
    parser.add_argument("base_url", type=str, help="Базовий URL для завантаження, наприклад, vstup2023.edbo.gov.ua")
    return parser.parse_args()

def fetch_initial_data(url):
    response = requests.get(url, verify=False)
    if response.status_code == 200:
        response.encoding = 'utf-8'
```

```
        return BeautifulSoup(response.text, 'html.parser')
    else:
        logging.error(f"Не вдалося завантажити сторінку, статус-код: {response.status_code}")
        return None

def extract_options_from_selector(soup, selector_id):
    selector = soup.find('select', id=selector_id)
    if not selector:
        logging.warning(f"Селектор з ID '{selector_id}' не знайдено на сторінці.")
        return {}
    options = selector.find_all('option')
    return {
        option.get('value'): option.text.strip()
        for option in options
        if option.get('value') and option.text.strip()
    }

def send_to_queue(data):
    try:
        connection = pika.BlockingConnection(pika.ConnectionParameters(host='localhost'))
        channel = connection.channel()

        channel.queue_declare(queue='dead_letter_queue_offer_links', durable=True)

        channel.queue_declare(
            queue='offer_links_parser',
            durable=True,
            arguments={
```

```
            'x-dead-letter-exchange': '',
            'x-dead-letter-routing-key': 'dead_letter_queue_offer_links'
        }
    )

    properties = pika.BasicProperties(
        delivery_mode=2,
        headers={'x-retry-count': 0}
    )

    channel.basic_publish(
        exchange='',
        routing_key='offer_links_parser',
        body=json.dumps(data),
        properties=properties
    )

    logging.info(f"Повідомлення успішно відправлено в чергу 'offer_links_parser': {data}")

  except Exception as e:
    logging.error(f"Помилка при відправленні даних у чергу: {e}")

  finally:
    if 'connection' in locals():
        connection.close()

def main():
  args = parse_args()
  base_url = f'https://{args.base_url}/offers/'
```

```
    soup = fetch_initial_data(base_url)
    if not soup:
        return

    education_base = extract_options_from_selector(soup, 'offers-search-education-base')
    search_qualification = extract_options_from_selector(soup, 'offers-search-qualification')
    offers_search_speciality = extract_options_from_selector(soup, 'offers-search-speciality')

    allowed_combinations = {
        '1': ['40', '530', '610', '520', '620'],
        '2': ['40', '530', '610', '520', '620', '640'],
        '9': ['30', '40', '540', '510', '520']
    }

    all_combinations = product(search_qualification.keys(), education_base.keys(), offers_search_speciality.keys())

    for qualification, education_base_key, speciality in all_combinations:
        if education_base_key not in allowed_combinations.get(qualification, []):
            logging.info(
                f"Пропущено невалідну комбінацію: Кваліфікація={qualification}, Освітня база={education_base_key}")
            continue

        data = {
            "qualification": qualification,
            "education_base_key": education_base_key,
            "speciality": speciality,
```

```
            "base_url": args.base_url
        }
        send_to_queue(data)
        logging.info(f"Завдання надіслано до черги: {data}")


if __name__ == "__main__":
    main()
```

Додаток Д. Код скрипту, що парсить інформацію про заяви вступників

/offer_page_parser_producer.py

```
import asyncio
import json
import logging
import random
import re
import signal
import sys
import aio_pika
from bs4 import BeautifulSoup
import clickhouse_connect
from playwright.async_api import async_playwright
from playwright_stealth import stealth_async

logging.basicConfig(
    level=logging.INFO,
    format='%(asctime)s - %(levelname)s - %(message)s',
    handlers=[
        logging.StreamHandler(),
    ]
)
success_logger = logging.getLogger("success_logger")
success_logger.setLevel(logging.INFO)
success_handler = logging.FileHandler("logs/success_links.log")
success_handler.setLevel(logging.INFO)
success_handler.setFormatter(logging.Formatter('%(asctime)s - %(message)s'))
success_logger.addHandler(success_handler)
```

```
client = clickhouse_connect.get_client(
    host='localhost',
    port='8123',
    username='vagrant',
    password='admin123',
    database='edu_campaign'
)

def signal_handler(sig, frame):
    logging.info('Отримано сигнал завершення, зупинка консюмера...')
    if 'client' in globals():
        client.close()
        logging.info('З\'єднання з базою даних закрито.')
    sys.exit(0)
signal.signal(signal.SIGINT, signal_handler)
signal.signal(signal.SIGTERM, signal_handler)

def get_application_count(soup):
    count_div = soup.find('div', class_='stats-field-t')
    if count_div:
        value_div = count_div.find('div', class_='value')
        if value_div and value_div.get_text(strip=True).isdigit():
            return int(value_div.get_text(strip=True))
    logging.warning("Не вдалося знайти або визначити кількість заяв.")
    return 0

async def setup_browser():
    async_playwright_instance = await async_playwright().start()
    browser = await async_playwright_instance.chromium.launch(headless=True)
```

```
    context = await browser.new_context(
        user_agent="Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/130.0.0.0 Safari/537.36",
        ignore_https_errors=True
    )
    return async_playwright_instance, browser, context

async def teardown_browser(playwright_instance, browser, context):
    await context.close()
    await browser.close()
    await playwright_instance.stop()

async def fetch_page_source(context, url):
    page = None
    try:
        page = await context.new_page()
        await stealth_async(page)
        await page.goto(url, timeout=30000)
        await asyncio.sleep(1)

        while True:
            try:
                load_more_button = await page.wait_for_selector("#requests-load", timeout=5000)
                if load_more_button:
                    logging.info("Натискання кнопки 'Завантажити додаткові заяви'...")
                    await load_more_button.click()
                    await asyncio.sleep(random.uniform(2, 3))
            except Exception:
                logging.warning("Кнопка 'Завантажити додаткові заяви' не доступна.")
```

```
                break

        content = await page.content()
        return content

    except Exception as e:
        logging.error(f"Помилка під час завантаження сторінки {url}: {e}")
        raise
    finally:
        if page:
            await page.close()

def parse_offers(soup):
    logging.info("Парсинг блоку 'offer'...")

    offer = soup.find('div', class_='offer')
    if not offer:
        logging.warning("Блок 'offer' не знайдено.")
        raise

    offer_code = offer.get('data-id', '')
    offer_data = {
        'year': '',
        'university_code': '',
        'university_name': '',
        'degree': '',
        'admission_basis': '',
        'speciality': '',
        'offer_name': '',
        'offer_code': '',
```

```
        'education_program': '',
        'faculty': '',
        'study_form': '',
        'course': '',
        'offer_type': '',
        'study_term': '',
        'application_term': '',
        'license_capacity': '',
        'max_state_order': '',
        'contract_capacity': '',
        'education_price': '',
        'subjects': '[]'
    }

    h2_tag = soup.find('h2', class_='banner-title')
    if h2_tag:
        year_span = h2_tag.find('span', string=re.compile(r'\d{4}'))
        if year_span:
            year_text = year_span.get_text(strip=True)
            if year_text.isdigit():
                offer_data['year'] = int(year_text)

    a_tag = soup.find('a', href=re.compile(r'https://registry\.edbo\.gov\.ua/university/\d+/'))
    if a_tag:
        href = a_tag['href']
        university_code_match = re.search(r'/university/(\d+)/', href)
        university_code = university_code_match.group(1) if university_code_match else ''
        university_name_tag = a_tag.find('h5')
```

```
        university_name = university_name_tag.get_text(strip=True) if
university_name_tag else ''
        offer_data['university_code'] = university_code
        offer_data['university_name'] = university_name

    h6_tag = soup.find('h6', class_='text-primary')
    if h6_tag:
        degree_span = h6_tag.find('span', class_='text-uppercase')
        basis_span = h6_tag.find('span', class_='text-lowercase')
        degree = degree_span.get_text(strip=True) if degree_span else ''
        basis_text = basis_span.get_text(strip=True) if basis_span else ''
        basis_match = re.search(r'основа вступу - (.+)\)', basis_text)
        admission_basis = basis_match.group(1) if basis_match else ''
        offer_data['degree'] = degree
        offer_data['admission_basis'] = admission_basis

    speciality_dt = offer.find('dt', string="Спеціальність")
    if speciality_dt:
        speciality_dd = speciality_dt.find_next_sibling('dd')
        if speciality_dd:
            speciality_code = speciality_dd.find('span', class_='badge badge-
primary').get_text(
                strip=True) if speciality_dd.find('span', class_='badge badge-primary') else ''
            speciality_name = speciality_dd.find('span', class_='text-uppercase text-
primary').get_text(
                strip=True) if speciality_dd.find('span', class_='text-uppercase text-primary')
else ''
            speciality_sub_name = speciality_dd.find('span', class_='badge badge-secondary
text-lowercase').get_text(
```

```
            strip=True) if speciality_dd.find('span', class_='badge badge-secondary text-lowercase') else ''
            offer_data['speciality_code'] = f"{speciality_code}".strip()
            offer_data['speciality_name'] = f"{speciality_name}".strip()
            offer_data['speciality_sub_name'] = f"{speciality_sub_name}".strip()

    offer_name_dt = offer.find('dt', string="Назва пропозиції")
    if offer_name_dt:
        offer_name_dd = offer_name_dt.find_next_sibling('dd')
        if offer_name_dd:
            offer_data['offer_name'] = offer_name_dd.get_text(strip=True)

    offer_data.update({
        'offer_code': offer.find('dl', class_='row offer-university-specialities').find('dd').get_text(
            strip=True) if offer.find('dl', class_='row offer-university-specialities') else offer_code,
        'education_program': offer.find('dl', class_='row offer-study-programs').find('dd').get_text(
            strip=True) if offer.find('dl', class_='row offer-study-programs') else '',
        'faculty': offer.find('dl', class_='row offer-university-facultet-name').find('dd').get_text(
            strip=True) if offer.find('dl', class_='row offer-university-facultet-name') else '',
        'study_form': offer.find('dl', class_='row offer-education-form-name').find('dd').get_text(
            strip=True) if offer.find('dl', class_='row offer-education-form-name') else '',
        'course': offer.find('dl', class_='row offer-course-name').find('dd').get_text(strip=True) if offer.find('dl', class_='row offer-course-name') else '',
        'offer_type': offer.find('dl', class_='row offer-offer-type-name').find('dd').get_text(
```

```
            strip=True) if offer.find('dl', class_='row offer-offer-type-name') else '',
        'study_term': offer.find('dl', class_='row offer-education-term').find('dd').get_text(strip=True) if offer.find(
            'dl', class_='row offer-education-term') else '',
        'application_term': offer.find('dl', class_='row offer-request-term').find('dd').get_text(
            strip=True) if offer.find('dl', class_='row offer-request-term') else '',
        'license_capacity': offer.find('dl', class_='row offer-order-license').find('dd').get_text(
            strip=True) if offer.find('dl', class_='row offer-order-license') else '',
        'max_state_order': offer.find('dl', class_='row offer-max-order').find('dd').get_text(strip=True) if offer.find(
            'dl', class_='row offer-max-order') else '',
        'contract_capacity': offer.find('dl', class_='row offer-order-contract').find('dd').get_text(
            strip=True) if offer.find('dl', class_='row offer-order-contract') else '',
        'education_price': offer.find('dl', class_='row offer-education-price').find('dd').get_text(
            strip=True) if offer.find('dl', class_='row offer-education-price') else ''
    })

    subjects = []
    subject_divs = offer.find_all('div', class_='offer-subject-wrapper')
    for subject in subject_divs:
        subject_data = {
            'subject_number': subject.find('div', class_='subject-number').get_text(strip=True) if subject.find('div', class_='subject-number') else '',
            'coefficient': subject.find('div', class_='coefficient').get_text(strip=True) if subject.find('div', class_='coefficient') else '',
```

```
            'subject_name': subject.find('div', class_='subject-name').get_text(strip=True) if subject.find('div', class_='subject-name') else '',
            'min_value': ''
        }

        min_value_div = subject.find('div', class_='min-value')
        if min_value_div:
            min_value_text = min_value_div.get_text(strip=True)
            match = re.search(r'\d+', min_value_text)
            subject_data['min_value'] = match.group() if match else ''

        subjects.append(subject_data)

    offer_data['subjects'] = json.dumps(subjects, ensure_ascii=False)

    logging.info("Парсинг блоку 'offer' завершено.")
    return offer_data

def parse_requests(soup):
    requests = []
    request_divs = soup.find_all('div', class_='offer-request')

    logging.info("Парсинг блоків 'request'...")
    for request in request_divs:
        request_data = {
            'request_number': '',
            'student_name': '',
            'status': '',
            'priority': '',
            'confirmation': 'No',
```

```
        'documents': 'No',
        'avg_score': '',
        'scores': '[]',
        'regional_coefficient': '1',
        'village_coefficient': '1',
        'industry_coefficient': '1',
        'quota': ''
    }

    request_data.update({
        'request_number': (request.find('div', class_='offer-request-n').find('div').get_text(strip=True)
                           if request.find('div', class_='offer-request-n') and request.find('div', class_='offer-request-n').find(
            'div') else ''),
        'student_name': (request.find('div', class_='offer-request-fio').find('div').get_text(strip=True)
                         if request.find('div', class_='offer-request-fio') and request.find('div', class_='offer-request-fio').find(
            'div') else ''),
        'status': (request.find('div', class_='offer-request-status').find('div').get_text(strip=True)
                   if request.find('div', class_='offer-request-status') and request.find('div', class_='offer-request-status').find(
            'div') else ''),
        'priority': (request.find('div', class_='offer-request-priority').find('div').get_text(strip=True)
```

```
                if request.find('div', class_='offer-request-priority') and request.find('div',
                                                                        class_='offer-request-
priority').find(
                    'div') else ''),
                'confirmation': 'Yes' if request.find('div', class_='offer-confirm') and (
                        request.find('div', class_='offer-confirm').find('div', class_='od-2') or
request.find('div',
                                                                                class_='offer-
confirm').find(
                            'div', class_='od-1')) else 'No',
                'documents': 'Yes' if request.find('div', class_='offer-documents') and
request.find('div',
                                                                        class_='offer-
documents').find(
                    'div', class_='od-1') else 'No',
                'avg_score': (
                    request.find('div',                                 class_='offer-request-
kv').find('div').get_text(strip=True).replace(',', '.')
                    if request.find('div', class_='offer-request-kv') and request.find('div',
                                                                    class_='offer-request-kv').find(
                        'div') else ''),
                'scores': json.dumps([], ensure_ascii=False)
            })

            subjects = {}
            subjects_div = request.find('div', class_='offer-subjects')
            if subjects_div:
                subject_divs = subjects_div.find_all('div', class_='offer-subject')
                for subject in subject_divs:
                    if 'indicator-rk' in subject.get('class', []):
```

```
            kv_div = subject.find('div', class_='kv')
            if kv_div:
                match = re.search(r"[\d.]+", kv_div.get_text(strip=True))
                request_data['regional_coefficient'] = match.group() if match else '1'
        if 'indicator-sk' in subject.get('class', []):
            kv_div = subject.find('div', class_='kv')
            if kv_div:
                match = re.search(r"[\d.]+", kv_div.get_text(strip=True))
                request_data['village_coefficient'] = match.group() if match else '1'
        if 'indicator-gk' in subject.get('class', []):
            kv_div = subject.find('div', class_='kv')
            if kv_div:
                match = re.search(r"[\d.]+", kv_div.get_text(strip=True))
                request_data['industry_coefficient'] = match.group() if match else '1'
        if 'indicator-q' in subject.get('class', []):
            sn_div = subject.find('div', class_='sn')
            request_data['quota'] = sn_div.get_text(strip=True) if sn_div else ''

        subject_name = subject.find('div', class_='sn').get_text(strip=True) if
subject.find('div',
                                                                  'sn') else ''
        formula = subject.find('div', class_='f').get_text(strip=True) if
subject.find('div', 'f') else ''

        if subject_name and formula:
            score = formula.split('=')[1].strip().split(' ')[0]
            subjects[subject_name] = score

    request_data['scores'] = json.dumps(subjects, ensure_ascii=False)
```

```
        if request_data['request_number']:
            requests.append(request_data)

    logging.info(f"Парсинг блоків 'request' завершено. Кількість заяв: {len(requests)}")
    return requests

def save_to_db(offer_data, requests, db_client):
    data_to_insert = []
    for request in requests:
        row = (
            offer_data['year'],
            offer_data['university_code'],
            offer_data['university_name'],
            offer_data['degree'],
            offer_data['admission_basis'],
            offer_data['speciality_code'],
            offer_data['speciality_name'],
            offer_data['speciality_sub_name'],
            offer_data['offer_name'],
            offer_data['offer_code'],
            offer_data['education_program'],
            offer_data['faculty'],
            offer_data['study_form'],
            offer_data['course'],
            offer_data['offer_type'],
            offer_data['study_term'],
            offer_data['application_term'],
            int(offer_data['license_capacity']) if offer_data['license_capacity'] else 0,
            int(offer_data['max_state_order']) if offer_data['max_state_order'] else 0,
            int(offer_data['contract_capacity']) if offer_data['contract_capacity'] else 0,
```

```
            int(offer_data['education_price']) if offer_data['education_price'] else 0,
            offer_data['subjects'],
            float(request['regional_coefficient']) if request['regional_coefficient'] else 1.0,
            float(request['village_coefficient']) if request['village_coefficient'] else 1.0,
            float(request['industry_coefficient']) if request['industry_coefficient'] else 1.0,
            request['quota'],
            request['request_number'],
            request['student_name'],
            request['status'],
            request['priority'],
            request['confirmation'],
            request['documents'],
            float(request['avg_score']) if request['avg_score'] else 0.0,
            request['scores'],
            '' #Регіон - він витягується окремо
        )
        data_to_insert.append(row)

    try:
        db_client.insert("edu_campaign.offer_request", data_to_insert)
        logging.info(f"{len(data_to_insert)} за'явок збережено в БД для {offer_data['year']} року")
    except Exception as e:
        logging.error(f"Помилка під час збереження записів в БД: {e}")
        raise e


async def process_link(context, link, client):
    logging.info(f"Обробка посилання: {link}")
    page_source = await fetch_page_source(context, link)
```

```
    soup = BeautifulSoup(page_source, 'html.parser')
    expected_application_count = get_application_count(soup)
    logging.info(f"Очікувана кількість заяв: {expected_application_count}")
    offer_data = parse_offers(soup)
    requests_data = parse_requests(soup)
    if len(requests_data) != expected_application_count:
        raise ValueError(
            f"Кількість витягнутих заяв ({len(requests_data)}) не відповідає очікуваній ({expected_application_count}).")
    if requests_data:
        save_to_db(offer_data, requests_data, client)
    success_logger.info(f"Success: {offer_data['offer_code']}")

async def callback(message: aio_pika.IncomingMessage, context):
    async with message.process(requeue=False):
        link = message.body.decode()
        await process_link(context, link, client)
        logging.info("Повідомлення успішно оброблено та підтверджено.")

async def consume():
    async_playwright_instance, browser, context = await setup_browser()
    try:
        connection = await aio_pika.connect_robust(
            host='localhost',
            port=5672,
            login='guest',
            password='guest'
        )

        async with connection:
```

```
        channel = await connection.channel()
        await channel.set_qos(prefetch_count=1)

        dead_letter_queue = await channel.declare_queue('dead_letter_queue_offer_requests', durable=True)
        main_queue = await channel.declare_queue(
            'offer_requests_parser',
            durable=True,
            arguments={
                'x-dead-letter-exchange': '',
                'x-dead-letter-routing-key': 'dead_letter_queue_offer_requests'
            }
        )

        async def callback_with_context(message: aio_pika.IncomingMessage):
            try:
                await callback(message, context)
            except Exception as e:
                logging.error(f"Unhandled exception in callback_with_context: {e}", exc_info=True)
        await main_queue.consume(callback_with_context, no_ack=False)
        logging.info("Консюмер запущено. Очікування повідомлень...")
        await asyncio.Future()
    finally:
        await teardown_browser(async_playwright_instance, browser, context)
def main():
    asyncio.run(consume())
if __name__ == "__main__":
    main()
```

Додаток Е. Налаштування Docker

/docker-compose.yml

```
services:
  clickhouse:
    image: clickhouse/clickhouse-server:latest
    container_name: clickhouse
    environment:
      - CLICKHOUSE_USER=vagrant
      - CLICKHOUSE_PASSWORD=admin123
      - CLICKHOUSE_DB=edu_campaign
    volumes:
      - ./create-database.sql:/docker-entrypoint-initdb.d/create-database.sql
      - ./clickhouse-backups:/backups
      - ./clickhouse-config/conf.d/backups_disk.xml:/etc/clickhouse-server/conf.d/backups_disk.xml:ro
      - ./clickhouse-config/config.d/storage.xml:/etc/clickhouse-server/config.d/storage.xml:ro
    ports:
      - '9000:9000'
      - '8123:8123'
    restart: unless-stopped

  rabbitmq:
    image: rabbitmq:3-management
    container_name: rabbitmq
    ports:
      - "5672:5672"
      - "15672:15672"
    environment:
```

```
      - RABBITMQ_DEFAULT_USER=guest
      - RABBITMQ_DEFAULT_PASS=guest
    restart: unless-stopped

  grafana:
    image: grafana/grafana:latest
    container_name: grafana
    ports:
      - '3000:3000'
    depends_on:
      - clickhouse
    restart: unless-stopped
    environment:
      - GF_INSTALL_PLUGINS=grafana-clickhouse-datasource
    volumes:
      - ./provisioning:/etc/grafana/provisioning
      - ./dashboards:/var/lib/grafana/dashboards
    user: '1000:1000'

volumes:
  clickhouse_data:
  grafana:
  rabbitmq:
```