МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ХАРЧОВИХ ТЕХНОЛОГІЙ

Інститут (факультет) Автоматизації і комп'ютерних систем
Кафедра Інформаційних технологій, штучного інтелекту та кібербезпеки

«До захисту в ЕК»	**«До захисту допущено»**
Директор інституту(декан факультету)	Завідувач кафедри
Андрій ФОРСЮК (ім'я та прізвище)	Сергій ГРИБКОВ (ім'я та прізвище)
«24» червня 2025 р.	«24» червня 2025 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 інформаційної системи для аналізу даних про вступників до закладів вищої освіти України

Виконав: здобувач 2 курсу, групи КН-1-4М

Дзюба Анастасія Олексіївна _____
(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 Кашперюк О.С _____
(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 — 2025 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
Сергій ГРИБКОВ
«15» квітня **2025 року**

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ РОБОТУ ЗДОБУВАЧА

Дзюби Анастасії Олексіївни
(прізвище, ім'я, по батькові)

1. Тема роботи Дослідження та створення інформаційної системи для аналізу даних про вступників до закладів вищої освіти України
керівник роботи Струзік Владислав Анатолійович, асистент,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом закладу вищої освіти від «15» квітня 2025 р. № 237-кс
2. Строк подання здобувачем роботи: 11.06.2025
3. Вихідні дані до роботи:
1. Інформація з відкритих джерел ЄДЕБО
2. Нормативно-правова база Міністерства освіти і науки України
3. Історичні дані вступних кампаній попередніх років
4. Технічні інструменти для реалізації аналітики
5. Вимоги кінцевих користувачів (приймальні комісії, адміністрація ЗВО)
4. Зміст пояснювальної записки (перелік питань, які потрібно розробити):
1. Обґрунтування актуальності теми дослідження
2. Огляд нормативно-правової бази
3. Аналіз джерел та типів даних вступної кампанії
4. Методи обробки та аналізу освітніх даних
5. Обґрунтування архітектури інформаційної системи
6. Опис метамоделі та структури бази даних

7. Реалізація інструментів візуалізації на базі Grafana
8. Оцінка ефективності розробленої системи
9. Пропозиції щодо подальшого розвитку системи

5. Перелік графічного матеріалу:
1. Скріншоти інтерфейсів систем візуалізації BAT, Metabase, Grafana, Power BI
2. Скріншоти результатів роботи інформаційної системи для аналізу даних про вступників до закладів вищої освіти України

6. Консультанти розділів роботи:

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1.	Струзік В. А., асистент, кандидат технічних наук	16.04.2025	29.04.2025
2.	Струзік В. А., асистент, кандидат технічних наук	25.04.2025	06.05.2025
3.	Струзік В. А., асистент, кандидат технічних наук	02.05.2025	13.05.2025
4.	Струзік В. А., асистент, кандидат технічних наук	09.05.2025	30.05.2025

7. Дата видачі завдання: 15.04.2025

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів виконання кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Видача завдання	15.04.2025	Виконала
2	Виконання пошуку матеріалів	22.04.2025	Виконала
3	Оформлення розділу 1	29.04.2025	Виконала
4	Оформлення розділу 2	06.05.2025	Виконала
5	Оформлення розділу 3	13.05.2025	Виконала
6	Розробка системи	23.05.2025	Виконала
7	Оформлення розділу 4	30.05.2025	Виконала
8	Оформлення автореферату	07.06.2025	Виконала
9	Оформлення презентації	10.06.2025	Виконала

Здобувач _____ (підпис) Дзюба А. О. (прізвище та ініціали)

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

АНОТАЦІЯ

Дзюба А.О.. Кваліфікаційна робота на тему «Дослідження та створення інформаційної системи для аналізу даних про вступників до закладів вищої освіти України» складається із 73 сторінок, включає 7 ілюстрацій, висновків, списку використаної літератури та 3 додатків.

Кваліфікаційна робота присвячена дослідженню процесів аналізу даних вступних кампаній до закладів вищої освіти України та розробці інформаційної системи, яка автоматизує збір, обробку й візуалізацію цих даних. У роботі розглянуто специфіку джерел інформації, методи аналізу даних, а також обґрунтовано вибір інструментів, зокрема використання платформи Grafana для створення інтерактивних дашбордів. Результатом дослідження є побудова комплексного підходу до аналітики вступної кампанії, який дозволяє покращити управлінські рішення в сфері вищої освіти.

Ключові слова: ВСТУПНА КАМПАНІЯ, ІНФОРМАЦІЙНА СИСТЕМА, GRAFANA, КВАЛІФІКАЦІЙНА РОБОТА, АНАЛІЗ, ДАШБОРД, ОСВІТНЯ ПРОГРАМА, УПРАВЛІНСЬКІ РІШЕННЯ.

SUMMARY

Dziuba A.O. The qualification thesis titled "Research and Development of an Information System for Analyzing Data on Applicants to Higher Education Institutions of Ukraine" consists of 73 pages, includes 7 illustrations, conclusions, a list of references, and 3 appendices.

Qualification thesis is devoted to the study of data analysis processes of the admission campaigns to higher education institutions in Ukraine and the development of an information system that automates the collection, processing, and visualization of this data. The work examines the specifics of data sources, methods of educational data analysis, and substantiates the choice of tools, in particular the use of the Grafana platform for creating interactive dashboards. The result of the research is the development of a comprehensive approach to admission campaign analytics, which enables improved management decisions in the field of higher education.

Keywords: ADMISSION CAMPAIGN, INFORMATION SYSTEM, GRAFANA, QUALIFICATION THESIS, ANALYSIS, DASHBOARD, EDUCATIONAL PROGRAM, MANAGERIAL DECISIONS.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

Скорочення	**Розшифрування**
ЄДЕБО	Єдина державна електронна база з питань освіти
API	Application Programming Interface
СУБД	Система управління базами даних
ВНЗ	Вищі навчальні заклади
ЗВО	Заклад вищої освіти
BAT	Business Analysis Tool
МОН	Міністерство освіти і науки України

ВСТУП

Актуальність теми. Актуальність теми обумовлена сучасними викликами в системі вищої освіти, пов’язаними зі змінами демографічної ситуації, цифровізацією та зростаючими вимогами до аналітичної звітності. Існуючі методи не завжди забезпечують ефективний аналіз великих обсягів даних вступної кампанії для прийняття стратегічних рішень.

Розроблений комплексний підхід до аналізу даних про вступників, що складається з методів аналізу та візуалізації, дозволить покращити планування освітніх програм, прогнозувати навантаження та підвищити конкурентоспроможність закладів вищої освіти. Це важливо в умовах цифрової трансформації та необхідності підвищення прозорості й ефективності управління у сфері освіти України.

Зв’язок роботи з науковими програмами, планами, темами. Кваліфікаційна робота виконується в межах наукового напряму кафедри інформаційних технологій, штучного інтелекту і кібербезпеки факультету автоматизації і комп’ютерних систем НУХТ. Тематика дослідження відповідає стратегічним завданням університету з поліпшення якості освіти, цифрової трансформації освіти, впровадження сучасних інформаційних технологій, а також узгоджується з державними ініціативами у сфері відкритих даних та прозорості вступної кампанії, визначених МОН.

Мета дослідження. Підвищити ефективність управлінських рішень у закладах вищої освіти за рахунок впровадження інформаційної системи, що дозволить автоматизовувати інформаційні процеси даних вступних кампаній попередніх років.

Завдання дослідження:

- проаналізувати особливості вступних кампаній до закладів вищої освіти України на основі відкритих даних МОН та ЄДЕБО;
- визначити ключові показники, що характеризують динаміку вступу, популярність спеціальностей та освітні тенденції;

- розробити підсистему збору даних з відкритих джерел МОН та ЄДЕБО;
- розробити структуру бази даних для збереження та обробки інформації про вступні кампанії;
- реалізувати інструменти аналітики статистичних показників із використанням сучасної системи візуалізації.

Об’єктом дослідження є процес вступної кампанії до ЗВО України.

Предметом дослідження є методи та засоби аналізу історичних даних вступних кампаній до ЗВО України.

Методи дослідження:

- аналіз – розклад системи на складові компоненти з метою їх детального вивчення;
- синтез – об’єднання результатів аналізу в цілісну логічну модель функціонування інформаційної системи;
- індукція – формування припущень і висновків на основі спостережень та первинних даних;
- дедукція – логічна перевірка сформованих гіпотез і узагальнень;
- абстрагування – виділення ключових характеристик системи, ігноруючи другорядні ознаки;
- порівняльний метод – оцінка ефективності запропонованих рішень шляхом співставлення різних варіантів аналітики;
- експертне оцінювання – перевірка доцільності та якості реалізованих підходів в інформаційній системі.

Наукова новизна одержаних результатів. Формалізовано комплексний підхід до збору та обробки історичних даних вступних кампаній до ЗВО України, що дозволило забезпечити оперативну візуалізацію ключових показників, які характеризують динаміку вступу, популярність спеціальностей та освітні тенденції.

Практичне значення одержаних результатів. Розроблена в межах кваліфікаційної роботи інформаційна система забезпечує автоматизований збір,

збереження та аналіз даних вступних кампаній, що дає змогу комплексно оцінювати динаміку вступу, визначати актуальні тенденції за спеціальностями та оцінювати ефективність освітніх стратегій. Використання отриманих результатів дозволяє приймати обґрунтовані управлінські рішення для своєчасного вдосконалення освітніх програм і коригування напрямів підготовки відповідно до потреб вступників.

Аналітичні дані можуть слугувати базою для розробки нових підходів до управління освітнім процесом та стратегічного розвитку закладу. Таким чином, впровадження системи сприяє підвищенню ефективності управління та якості освітніх послуг на внутрішньому рівні закладу вищої освіти.

Структура та обсяг роботи. Кваліфікаційна робота складається зі вступу, 4 розділи, висновків, списку використаної літератури та додатків. Загальний обсяг роботи становить 73 сторінок.

РОЗДІЛ 1. АНАЛІТИЧНИЙ ОГЛЯД ЛІТЕРАТУРИ ТА ОБҐРУНТУВАННЯ ВИБОРУ НАПРЯМУ ДОСЛІДЖЕННЯ

1.1 Вступна кампанія

Вступна кампанія є одним із ключових етапів функціонування системи вищої освіти. Вона передбачає збір, обробку та аналіз інформації не лише про абітурієнтів, які планують вступ до закладів вищої освіти, а й про всі дані, що стають відомими після вступу — зокрема про зарахованих студентів, їх розподіл за спеціальностями, регіонами та інші важливі показники [1]. Ефективне управління цим процесом вимагає системного підходу та використання сучасних інформаційних технологій для автоматизації збору даних і забезпечення їх аналітичної обробки. Особливо актуальним це стає в умовах зростання обсягів інформації та необхідності оперативного прийняття управлінських рішень.

1.2 Аналіз ключових чинників впливу на розвиток університетів

Розвиток університетів значною мірою залежить від ефективності проведення вступної кампанії та здатності закладу адаптуватися до змін у попиті на освітні послуги. Основними чинниками впливу є демографічні зміни, технологічна модернізація освітніх процесів, а також постійне оновлення вимог ринку праці. Висока конкуренція серед закладів вищої освіти вимагає використання аналітичних систем, що допомагають прогнозувати тенденції та коригувати освітні програми відповідно до потреб студентів і роботодавців [2].

1.3 Формулювання наукової проблеми

Керівники ЗВО України постійно зіштовхуються з проблемою доступу до консолідованої аналітики, яка б могла дозволити ухвалювати стратегічні рішення щодо відкриття нових спеціальностей, закриття неефективних програм або розширення/скорочення підготовки з певних напрямів. Можливість порівнювати показники власної вступної кампанії з показниками конкурентів є важливим елементом виявлення, як сильних, так і слабких сторін.

Без комплексного аналізу, прийняття ефективних рішень щодо майбутніх вступних кампаній, коригування освітньої політики чи оптимізації освітніх програм стає складним завданням.

Аби зручно та ефективно проводити аналітику, повною мірою оцінювати вплив попередніх рішень, прогнозувати майбутні виклики та можливості, або ж розробляти стратегії, що базуються на глибокому розумінні минулого досвіду, доцільно впровадити спеціалізовану інформаційну систему, що забезпечить автоматизований збір, обробку та аналіз даних за попередні роки.

Результати дослідження свідчать про відсутність систематизованих методів аналізу даних, орієнтованих на оцінку вступних кампаній до ЗВО України. Як наслідок, це унеможливлює створення вищезгаданої інформаційної системи.

Отже, основною науковою проблемою є відсутність комплексного підходу до аналізу історичних даних щодо вступних кампаній у ЗВО України, що у свою чергу ускладнює формування обґрунтованих висновків та ефективних рішень.

1.4 Актуальність та сучасний стан проблеми

Вибір напряму дослідження обумовлений гострою потребою в підвищенні ефективності управління ВНЗ України. Сучасні умови характеризуються значними змінами в демографічній ситуації, що впливає на кількість абітурієнтів, а також швидкою цифровізацією освітніх процесів. Ці зміни зумовлюють необхідність впровадження інноваційних інформаційних систем, які забезпечать не лише автоматизований збір та зберігання даних, але й їх оперативну обробку, фільтрацію, агрегацію та візуалізацію для прийняття обґрунтованих управлінських рішень.

Однією з ключових проблем, що спонукає до вибору теми, є великий обсяг та різнорідність інформації, яка надходить під час вступної кампанії — від даних про подані заяви та вибір спеціальностей до кінцевих результатів вступу.

Традиційні методи аналізу, які здебільшого базуються на статичних звітах та ручній обробці, не відповідають вимогам сучасного швидкого та якісного аналізу, що призводить до затримок у прийнятті рішень і зниженні їх ефективності.

Організаційні виклики пов’язані з відсутністю єдиної централізованої платформи, яка б інтегрувала інформацію з різних джерел — відкритих баз даних МОН, ЄДЕБО та інших офіційних ресурсів. Це ускладнює формування цілісного уявлення про тенденції вступної кампанії, регіональні особливості, популярність спеціальностей та потенційні ризики, пов’язані з демографічними змінами.

Технічні можливості сучасних аналітичних баз даних, інструментів автоматизованого збору та обробки даних, а також платформ для візуалізації відкривають нові перспективи для вирішення вказаних проблем. Впровадження комплексної інформаційної системи дозволить покращити прозорість і оперативність звітності, оптимізувати планування навантажень на заклади вищої освіти, а також підвищити якість управлінських рішень на всіх рівнях освітньої системи.

1.5 Очікувані результати та їх значення

Розробка інформаційної системи для аналізу даних вступної кампанії дозволить автоматизувати процеси збору, збереження та обробки інформації, що, як наслідок, дозволить керівництву ЗВО своєчасно реагувати на зміни в попиті на освітні послуги, коригувати стратегії набору студентів і вдосконалювати освітні програми.

Таким чином, обраний напрям дослідження має вагоме практичне значення і сприятиме підвищенню якості управління освітніми процесами, що є важливим кроком на шляху цифрової трансформації та розвитку вітчизняної системи вищої освіти.

1.6 Висновок

Ефективне функціонування вступної кампанії є не лише оперативним завданням для закладів вищої освіти, а й елементом стратегічного розвитку університету. Сучасні виклики, пов’язані зі зростанням обсягів даних, динамікою ринку освітніх послуг та необхідністю гнучкого управління, зумовлюють потребу в цифровій трансформації вступної кампанії до ЗВО. У цьому контексті саме

інформаційні системи стають інструментом, що забезпечує прозорість, обґрунтованість і гнучкість ухвалення управлінських рішень.

Виявлені у дослідженні проблеми — зокрема відсутність комплексного підходу до аналізу історичних даних вступних кампаній та низький рівень інтеграції між джерелами інформації — суттєво знижують ефективність прийняття управлінських рішень у ЗВО. Ручна обробка даних, фрагментарність аналітики та обмеженість у візуалізації тенденцій створюють перешкоди для вчасної реакції на зміни в попиті на освітні послуги. Це підтверджує необхідність розробки та впровадження централізованих цифрових рішень, здатних акумулювати, узгоджувати та інтерпретувати великі обсяги освітніх даних.

Побудова спеціалізованої аналітичної інформаційної системи дозволить не лише оптимізувати процеси збору та обробки інформації, але й сформувати нову якість управління в освітній галузі. Зокрема, йдеться про можливість створення адаптивних стратегій набору, вдосконалення змісту освітніх програм, підвищення конкурентоспроможності університетів і забезпечення відповідності освітніх послуг актуальним потребам суспільства та економіки. Такий системний підхід сприятиме підвищенню ефективності освітньої політики держави.

Отже, результати дослідження підтверджують актуальність створення інструментів для аналітичного супроводу вступної кампанії на базі комплексного підходу до збору та обробки історичних даних вступних кампаній до ЗВО України. Інтеграція аналітики у процеси управління вищою освітою має стати пріоритетом для забезпечення сталого розвитку ЗВО в умовах демографічних, соціально-економічних змін, сприяючи підвищенню якості освітніх послуг та зміцненню довіри з боку абітурієнтів і роботодавців.

РОЗДІЛ 2. ДОСЛІДЖЕННЯ МЕТОДІВ АНАЛІЗУ ДАНИХ ТА РОЗРОБКА КОМПЛЕКСНОГО ПІДХОДУ З УРАХУВАННЯМ СПЕЦИФІКИ ВСТУПУ ДО ВНЗ

2.1. Структура та особливості даних вступної кампанії

Дані вступної кампанії містять велику кількість змінних, що охоплюють демографічну інформацію абітурієнтів, освітні характеристики, обрані спеціальності, пріоритетність заяв, джерело фінансування, а також інформацію про зарахування.

Ці дані мають реляційну структуру і, як правило, зберігаються у вигляді таблиць у системах типу ЄДЕБО [3]. Складність аналізу полягає в неоднорідності джерел, різному форматі подання інформації, наявності пропусків або дублювань, а також потребі в узгодженні історичних даних із поточними. Сама ж реляційна структура даних, в якій зберігаються дані щодо вступних кампаній, є недоцільною у використанні при аналізі, тому необхідно також денормалізувати дані та привести їх до стовпчикової моделі зберігання. Крім того, велике значення має часовий фактор — дані надходять хвилями і потребують оперативної обробки.

Вихідна структура даних складається із таких сутностей: конкурсна пропозиція, конкурсні заяви, основа вступу, заклад освіти, освітня програма, регіон, форма навчання, освітній рівень, спеціальність.

Подана ER-діаграма (рис 2.1) репрезентує структуру вступної кампанії до закладів вищої освіти базуючись на системі ЄДЕБО. Центральним елементом моделі є сутність "Конкурсна пропозиція", яка акумулює в собі ключові характеристики навчальних пропозицій: назву, факультет, курс зарахування, термін навчання, конкурсні предмети, кількість поданих заяв, а також дані про ліцензійний обсяг, вартість навчання, граничні бали та інші параметри, необхідні для організації конкурсу. Кожна конкурсна пропозиція пов’язана з певною освітньою програмою, формою навчання та основою вступу, що реалізовано за допомогою зовнішніх ключів.

Сутність "Конкурсні заяви" зберігає інформацію про окремі заявки вступників, зокрема їхній статус, тип фінансування (бюджет або контракт), відповідність вимогам до зарахування, а також індивідуальні показники й конкурсний бал. Кожна заява однозначно пов’язана з відповідною конкурсною пропозицією, що вказує на наявність відношення типу "багато до одного" (багато заяв можуть належати до однієї конкурсної пропозиції).

Сутність "Освітня програма" містить деталізацію програм, які пропонуються в межах конкурсних пропозицій. Вона включає такі атрибути, як назва програми, освітній рівень, спеціальність, форма навчання та заклад освіти, в якому реалізується дана програма. Таким чином, освітня програма є точкою перетину декількох сутностей, виступаючи об’єднувальним елементом для освітніх, адміністративних і регіональних даних.

Заклад освіти, у свою чергу, репрезентується через окрему сутність, що зберігає його унікальний ідентифікатор, назву та зв’язок з регіоном. Регіон виступає окремою сутністю з власним первинним ключем і назвою, забезпечуючи географічну класифікацію освітніх закладів. Така структуризація дає змогу реалізувати регіональні коефіцієнти та інші просторові показники при аналізі даних вступної кампанії.

Форма навчання також є самостійною сутністю, що дозволяє систематизувати освітні пропозиції за очною, заочною, дистанційною чи іншими формами здобуття освіти. Вона є обов’язковим атрибутом як для конкурсної пропозиції, так і для освітньої програми, що забезпечує узгодженість даних.

Освітній рівень у моделі виступає узагальненою характеристикою програм і визначає відповідність програми певному рівню акредитації. Крім того, він пов’язаний з основою вступу — ще однією сутністю, що визначає правову або кваліфікаційну підставу для подачі заяви (наприклад, на основі повної загальної середньої освіти, диплому молодшого спеціаліста тощо).

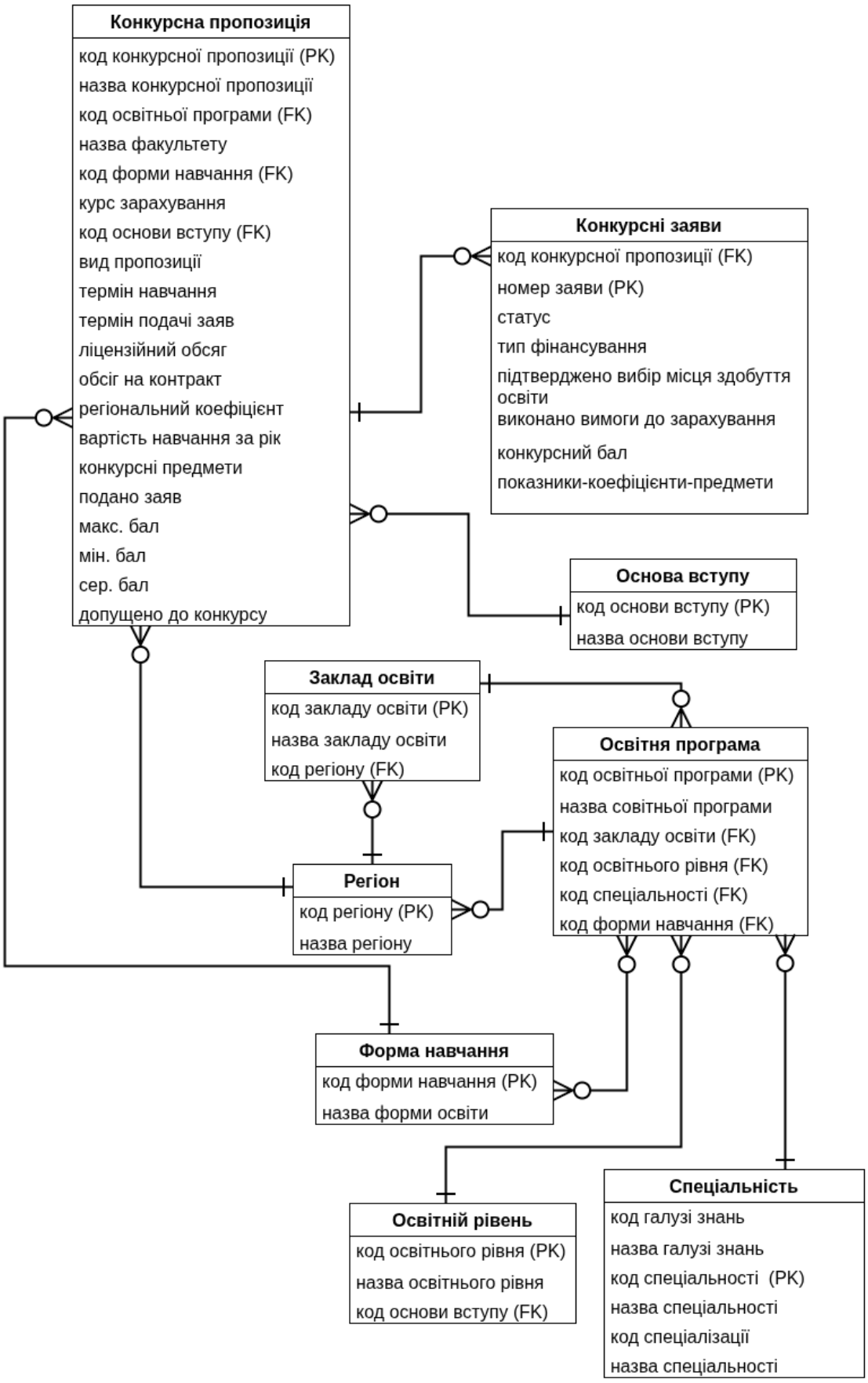

Рисунок 2.1 - ER-діаграма вихідної структури даних

Окрему увагу в моделі приділено класифікації спеціальностей. Сутність "Спеціальність" охоплює як коди й назви спеціальностей, так і галузі знань та спеціалізації, що дозволяє деталізувати напрями підготовки. Кожна освітня програма пов’язана з відповідною спеціальністю, що гарантує точне визначення профілю навчання та відповідає затвердженому переліку галузей знань і спеціальностей, за якими здійснюється підготовка здобувачів вищої та фахової передвищої освіти.

Загалом, модель бази даних є структурованою, логічно зв’язаною та підтримує нормалізовану організацію даних. Такий підхід дозволяє ефективно керувати інформацією про вступну кампанію, забезпечує цілісність і несуперечність даних, а також створює підґрунтя для подальшого аналізу, автоматизації процесів та звітності.

2.2. Методи обробки освітніх даних і побудова аналітичного підходу

Для аналізу вступної кампанії до закладів вищої освіти України використовуються як кількісні, так і якісні методи аналізу даних. Кількісні методи включають описову статистику, яка дозволяє оцінити загальні тенденції, розподіл вступників за спеціальностями, формами навчання та іншими параметрами.

До якісних методів належить контент-аналіз супровідної інформації, оцінювання освітніх тенденцій, а також інтерпретація поведінкових моделей вступників на основі сукупності ознак. Такі методи дозволяють доповнити кількісні висновки контекстуальним розумінням процесів.

У процесі аналізу часто використовуються інструменти для обробки та представлення даних, зокрема мови програмування Python і R, а також візуалізаційні платформи, що забезпечують побудову інтерактивних дашбордів і сприяють прийняттю рішень на основі достовірних і структурованих даних.

З огляду на специфіку даних вступної кампанії та нагальну наукову проблему – відсутність ефективних підходів до аналізу результатів вступних кампаній, які б повною мірою відповідали потребам університетів в Україні, – було обрано комплекс методів аналізу даних. Цей комплекс складається з описового,

діагностичного, прогностичного, кластерного аналізів, а також застосовується аналіз часових рядів.

Описовий аналіз становить основу аналітичного підходу, оскільки забезпечує можливість систематизації та узагальнення емпіричних даних шляхом обчислення ключових статистичних показників, зокрема середнього значення, медіани, моди та стандартного відхилення. У контексті вступної кампанії зазначений тип аналізу дозволяє виявити основні тенденції, такі як рівень популярності окремих галузей знань і спеціальностей, структуру поданих заяв за формами фінансування, переваги вступників у виборі спеціальностей, а також регіональні особливості динаміки подачі заяв.

Діагностичний аналіз орієнтований на виявлення причинно-наслідкових зв’язків, які зумовили спостережувані результати. У межах вступної кампанії цей підхід передбачає поглиблене дослідження чинників, що обумовлюють популярність окремих спеціальностей, вплив регіонального контексту на вибір вступників та ідентифікацію ключових детермінант успішного зарахування.

Прогностичний аналіз базується на використанні ретроспективних даних та відповідних статистичних моделей для формування оцінок майбутніх результатів. У сфері вступної кампанії його застосування дає змогу прогнозувати майбутній попит на спеціальності, що має вагоме значення для стратегічного планування ресурсного забезпечення закладів вищої освіти та розробки обґрунтованих напрямів державної освітньої політики.

Важливим інструментом у процесі аналізу даних вступної кампанії є сегментація, що реалізується, зокрема, за допомогою методів кластерного аналізу. Цей підхід дозволяє здійснювати групування освітніх програм або навчальних закладів за подібними характеристиками, такими як рівень популярності або інтенсивність конкуренції. Застосування кластерного аналізу сприяє виокремленню сегментів, які характеризуються підвищеною привабливістю для абітурієнтів з високими академічними досягненнями.

Аналіз часових рядів передбачає дослідження динаміки показників у часовій перспективі з метою виявлення закономірностей, трендів і сезонних коливань. У

контексті вступної кампанії такий підхід дозволяє здійснювати моніторинг змін у структурі попиту на спеціальності та освітні програми протягом тривалого періоду, що є важливою передумовою для формування довгострокових стратегій розвитку системи вищої освіти.

2.3. Висновок

У розділі було розглянуто специфіку даних вступної кампанії, основні підходи до їх обробки й аналізу, а також запропоновано комплексну методику, адаптовану до потреб вищих навчальних закладів. Такий підхід дозволяє ефективно працювати з великими обсягами інформації, підвищити точність прогнозування та підтримати прийняття обґрунтованих управлінських рішень.

РОЗДІЛ 3. РОЗРОБКА СПЕЦІАЛІЗОВАНОЇ АНАЛІТИЧНОЇ СИСТЕМИ РЕЗУЛЬТАТІВ ВСТУПНОЇ КАМПАНІЇ У ВНЗ

Сучасна система вищої освіти України потребує ефективних інструментів для аналізу великого обсягу даних. Необхідно створити аналітичну систему, здатну забезпечити оперативну та гнучку візуалізацію даних у зручному форматі. У межах цього розділу здійснено проєктування та реалізацію системи, що враховує специфіку джерел освітньої інформації, технічні обмеження та потреби кінцевих користувачів. Основною метою є автоматизація процесу збору, обробки та представлення даних про вступників для подальшого ухвалення управлінських рішень адміністрацією закладів вищої освіти.

Особливу увагу приділено визначенню вимог до системи, її архітектурній побудові та вибору інструментів реалізації. Також представлено логічну модель бази даних для зберігання освітньої інформації та методику попередньої обробки даних. Однією з ключових складових роботи є побудова інтерактивних дашбордів, що дозволяють здійснювати фільтрацію та порівняльний аналіз за різними параметрами.

Крім того, у розділі описано тестування системи, її ефективність за ключовими показниками, а також визначено напрями подальшого розвитку. Передбачено перспективи інтеграції з іншими державними платформами. Такий підхід дозволяє не лише виявити сильні сторони системи, а й визначити можливості для її масштабування.

3.1 Формування вимог до аналітичної системи

На етапі формування вимог до аналітичної системи основним завданням було визначення цілей, які має реалізовувати майбутнє програмне рішення. Система повинна забезпечувати ефективну обробку та візуалізацію освітніх даних, отриманих у ході вступної кампанії, а також бути зручною для кінцевих користувачів.

Основна мета — створення інформаційної системи, яка дає змогу оперативно виявляти тенденції змін при вступі, аналізувати конкурсні показники за різними

параметрами (спеціальностями, регіонами, формами навчання тощо) і підтримувати прийняття управлінських рішень на основі структурованої візуалізації даних.

Ключові вимоги:

- прозорість і доступність інформації в зручному форматі, придатному для публічного чи внутрішнього використання;
- фільтрація та групування даних за різними критеріями (рік, галузь знань, спеціальність, тип фінансування тощо);
- актуальність даних, що підтримується шляхом регулярного оновлення після кожної вступної кампанії;
- передбачити можливість розширення системи у майбутньому (додавання нових джерел даних, нових типів візуалізацій, показників);
- інтуїтивний інтерфейс для користувачів без досвіду роботи з подібними системами.

Під час формування вимог також було враховано досвід подібних реалізацій в освітній аналітиці. Наприклад, у статті аналітиків компанії SAS [4] підкреслюється, що візуалізація освітніх даних сприяє кращому розумінню показників успішності, дозволяє виявляти приховані тенденції та забезпечує основу для прийняття обґрунтованих управлінських рішень.

3.2 Аналіз веб-джерел для збору інформації про вступну кампанію

Для створення інформаційної системи аналізу даних про вступну кампанію до закладів вищої освіти України необхідно насамперед визначити релевантні джерела даних. Основними джерелами у цьому контексті є офіційні веб-сайти МОН, ЄДЕБО, інформаційні платформи закладів вищої освіти, а також агрегатори освітніх даних, такі як «Вступ.ОСВІТА.UA» [5].

ЄДЕБО є центральним джерелом структурованих даних про вступників, результати конкурсного відбору, розподіл бюджетних місць тощо. Веб-інтерфейс бази забезпечує публічний доступ до деяких даних, зокрема статистики поданих

заяв, кількості зарахованих вступників, рейтингових списків та результатів вступної кампанії в розрізі закладів, спеціальностей і регіонів.

Сайт МОН України надає нормативно-правові документи, календар вступної кампанії, інформацію про нововведення у правилах прийому до ВНЗ, що також є цінним джерелом для аналізу контексту даних.

Веб-платформи окремих закладів вищої освіти містять додаткову інформацію про конкурсні пропозиції, умови прийому, а також результати вступу в розрізі факультетів та спеціальностей. Ці дані можуть бути важливими для порівняльного аналізу між ЗВО.

Платформи-аналітики, як «Вступ.OCBITA.UA», агрегують дані з ЄДЕБО у зручному для користувача вигляді, дозволяючи швидко отримати узагальнені показники та рейтинги.

При виборі джерел враховувались такі критерії:

- відкритість доступу до даних (без авторизації);
- регулярне оновлення та актуальність інформації;
- стабільність структури HTML-документів для автоматизованого збору;
- наявність табличного представлення або чіткої візуальної ієрархії інформації.

Таким чином, для реалізації процесу збору даних були відібрані джерела, що поєднують нормативну достовірність, регулярність оновлення та технічну доступність для інтеграції з аналітичною системою.

3.3 Вибір бази даних для зберігання отриманих даних

Для ефективного зберігання та подальшої обробки великих обсягів даних, отриманих у результаті збору інформації про вступні кампанії, було розглянуто декілька сучасних систем керування базами даних, орієнтованих на аналітичні навантаження: ClickHouse, Vertica, Google BigQuery та Apache Cassandra.

Vertica — потужна аналітична СУБД, орієнтованою на обробку великих обсягів даних із високою швидкістю та підтримкою складних SQL-запитів. Вона

має вбудовану підтримку масштабованості, функції для глибокої аналітики, а також добре оптимізована для роботи в кластерному середовищі. Однак її використання передбачає ліцензійні обмеження, що не відповідає вимогам до гнучкості, відкритості та бюджетних обмежень освітніх або дослідницьких проєктів [6].

Google BigQuery — хмарна аналітична СУБД від Google, яка пропонує масштабованість, продуктивність і автоматичне управління інфраструктурою. Вона добре підходить для одноразового або епізодичного аналізу дуже великих обсягів даних. Проте залежність від хмарного середовища, платна модель використання, відсутність повного контролю над середовищем виконання та обмежені можливості локального розгортання роблять її менш привабливою для довготривалих дослідницьких ініціатив з потребою в автономному функціонуванні [7].

Apache Cassandra — високопродуктивна розподілена СУБД, яка найкраще підходить для обробки транзакцій і роботи з потоковими або часто змінюваними даними. Її сильними сторонами є стійкість до відмов, горизонтальне масштабування та висока доступність. Однак вона не є колоночною СУБД і не орієнтована на виконання складних аналітичних запитів або агрегацій, що знижує її ефективність у задачах аналітики історичних даних, якими є результати вступних кампаній [8].

ClickHouse — колоночна система керування базами даних з відкритим вихідним кодом, оптимізована для обробки аналітичних запитів у режимі реального часу. Основною перевагою ClickHouse є його здатність швидко опрацьовувати великі обсяги даних завдяки ефективному стисненню, індексації та паралельному виконанню запитів. Система підтримує складні аналітичні обчислення та масштабування, що дозволяє зберігати продуктивність навіть при зростанні обсягів даних. Завдяки повній сумісності з Grafana, ClickHouse дозволяє безперешкодно інтегрувати зібрані дані в систему візуалізації, забезпечуючи гнучкий і зручний інструмент для створення звітів та дашбордів. Відкритий код і

активна спільнота розробників також сприяють швидкому розвитку функціоналу та можливості адаптації системи під конкретні потреби дослідження [9].

З огляду на вищезазначене, для реалізації інформаційної системи було обрано СУБД ClickHouse як найбільш відповідну до вимог аналітичного навантаження, гнучкості, продуктивності, відкритості та сумісності з інструментами візуалізації.

3.4 Вибір мов програмування для збору та обробки даних з веб-джерел

Для реалізації процесу збору та обробки даних з веб-джерел важливо обрати мову програмування, яка забезпечує ефективну взаємодію з веб-ресурсами, підтримує бібліотеки для автоматизації браузера та парсингу HTML, а також дозволяє інтегруватися з системами зберігання та візуалізації даних.

Найбільш поширеною та зручною мовою для вирішення задач веб-скрапінгу та автоматизації є Python [10]. Її перевагами є:

- велика кількість бібліотек для роботи з HTML сторінками (requests, BeautifulSoup, lxml, Selenium, Playwright);
- простота синтаксису та читабельність коду;
- активна спільнота та широка документація;
- потужна підтримка роботи з JSON, XML, CSV, Excel та базами даних (через pandas, SQLAlchemy, sqlite3 тощо);
- легке розгортання скриптів на різних платформах.

Мова JavaScript (особливо в контексті Node.js) також активно використовується в задачах веб-скрапінгу, зокрема з такими фреймворками як puppeteer, cheerio, axios. Вона є ефективною при роботі з сайтами, що активно використовують JavaScript-рендерінг. Проте через складніший синтаксис та меншу поширеність у сфері академічних досліджень її застосування було визнано менш доцільним для даної системи [11].

Інші мови, як-от PHP, Java, C#, також мають засоби для доступу до веб-ресурсів, але через складність налаштування середовища та більшу вагу коду не були обрані як основні [12].

Зважаючи на необхідність автоматизації роботи з динамічними сторінками, обробки HTML-структур, обліку Cookies та можливості керування headless-браузерами, було обрано мову Python як основну для реалізації процесу збору та попередньої обробки даних.

3.5 Порівняльний аналіз систем візуалізації та аналітики даних

У цьому підрозділі розглянемо чотири популярні системи, що використовують для візуалізації даних та створення аналітичних звітів: Business Analysis Tool (BAT), Metabase, Grafana та Power BI.

Business Analysis Tool – це сучасний інструмент від BIT Impulse для аналізу бізнес-даних, який вже використовують понад 100 українських та міжнародних компаній (рис. 3.1). Цей програмний продукт допомагає одразу отримати єдиний звіт для управління даними компанії, працює з великим обсягом інформації, надає детальну фінансову звітність, допомагає оптимізувати процеси та збільшує продажі [13, 14].

Основні характеристики BAT [13]:

- підтримка підприємств різного масштабу (підходить для великих, середніх та малих організацій, включаючи державні установи);
- підключення до різних типів СУБД (забезпечує інтеграцію з різними джерелами даних, що дозволяє ефективно аналізувати інформацію);
- інтеграція з системами моніторингу (дозволяє відстежувати ключові показники в реальному часі);
- навчальні матеріали (наявність документації та підтримки від розробника);
- функціонуючий форум/спільнота (обмежена активність спільноти, але забезпечується технічна підтримка від виробника);
- локальна підтримка (розробка української компанії, що сприяє підтримці національного виробника та забезпечує адаптацію під специфічні потреби).

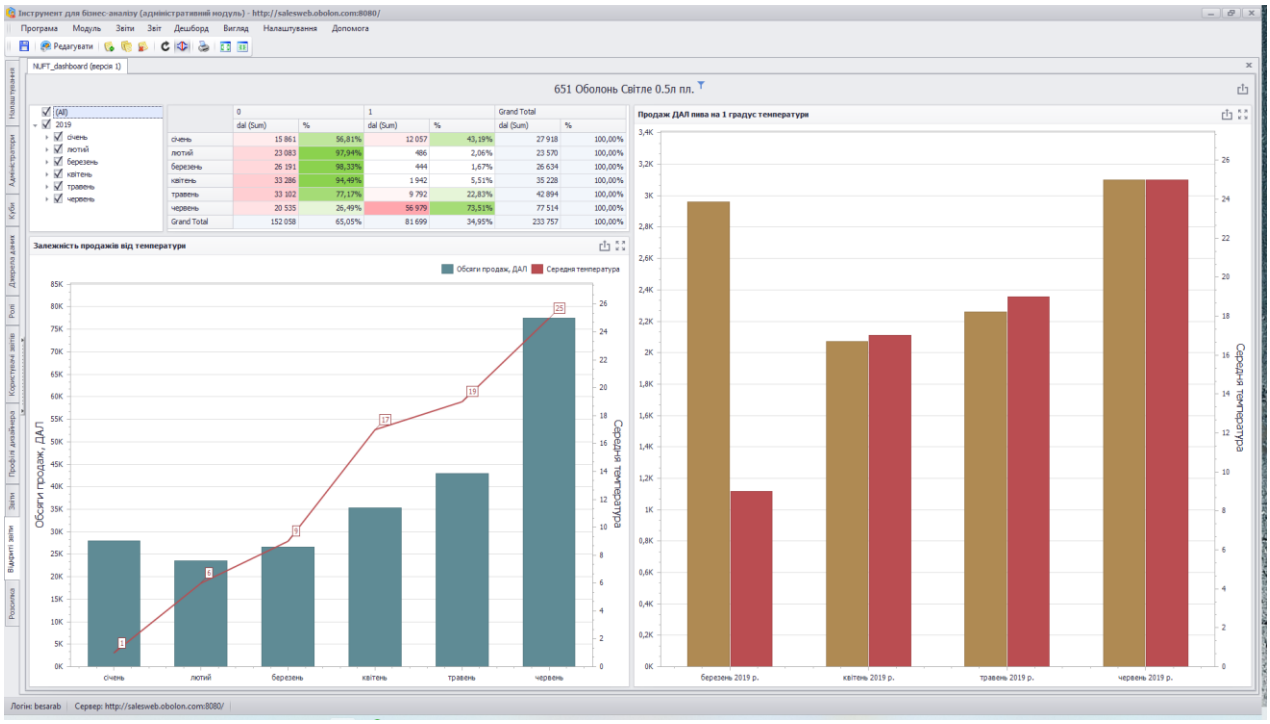

Рисунок 3.1 - Інтерфейс BAT, сформований дашборд

Metabase — це інструмент бізнес-аналітики з відкритим кодом, який можна підключити до багатьох популярних баз даних (рис 3.2). Metabase дозволяє ставити запитання щодо даних і відображає відповіді у зрозумілих форматах, будь то стовпчаста діаграма чи детальна таблиця [15].

Основні характеристики Metabase [14, 15]:

- підтримка підприємств різного масштабу (підходить для організацій будь-якого розміру);
- підключення до різних типів СУБД (підтримує різноманітні бази даних, включаючи PostgreSQL, MySQL та інші);
- інтеграція з системами моніторингу (можливість налаштування сповіщень при досягненні певних показників);
- навчальні матеріали (розвинена документація та навчальні ресурси);
- функціонуючий форум/спільнота (активна спільнота користувачів та розробників);
- додаткові можливості (підтримка завантаження результатів у різних форматах, налаштування сповіщень через електронну пошту або Slack).

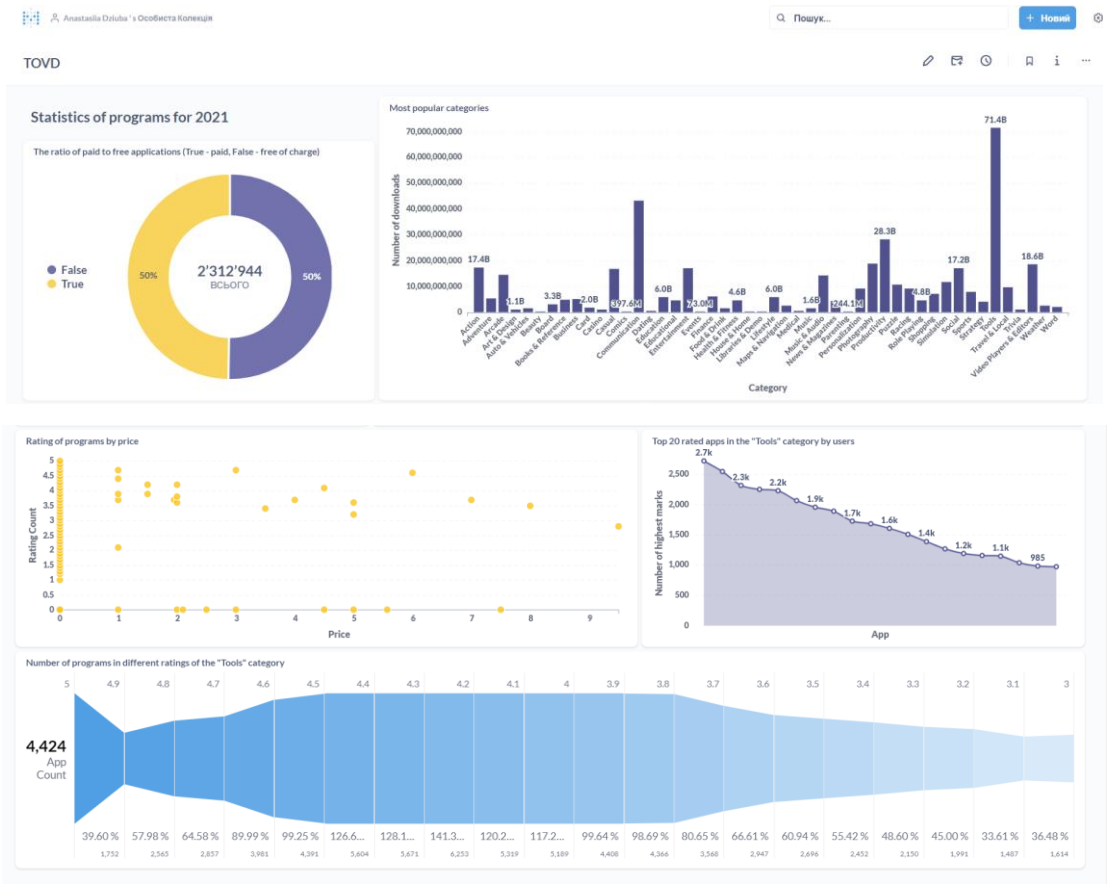

Рисунок 3.2 - Інтерфейс Metabase, сформований дашборд

Grafana — програмне забезпечення з відкритим кодом, яке дозволяє ставити запитання, візуалізувати дані, виводити попередження та досліджувати метрики, де б вони не зберігалися (рис 3.3). Grafana надає інструменти для перетворення даних бази даних часових рядів на зрозумілі графіки та візуалізації. Фреймворк плагінів Grafana також дозволяє підключати інші джерела даних, такі як бази даних NoSQL/SQL, інструменти для обробки заявок, такі як Jira або ServiceNow, та інструменти CI/CD, такі як GitLab [16].

Основні характеристики Grafana [16]:

- підтримка підприємств різного масштабу (підходить для організацій будь-якого розміру);
- підключення до різних типів СУБД (підтримує різноманітні джерела даних, включаючи SQL та NoSQL бази даних);
- інтеграція з системами моніторингу (спеціалізується на моніторингу в реальному часі, з можливістю налаштування сповіщень);
- навчальні матеріали (розвинена документація та навчальні ресурси);

- функціонуючий форум/спільнота (активна спільнота користувачів та розробників);
- додаткові можливості (підтримка розширень через плагіни, рольова модель доступу, можливість створення публічних дашбордів).

Рисунок 3.3 - Інтерфейс Grafana, сформований дашборд

Microsoft Power BI — це платформа візуалізації даних, яка використовується переважно для бізнес-аналітики (рис 3.4). Power BI розшифровується як Power Business Intelligence і стосується набору програмних сервісів, інструментів та конекторів, які допомагають перетворювати дані з різних джерел на практичні висновки [14, 17].

Розроблена для бізнес-фахівців з різним рівнем знань про дані, панель інструментів Power BI може звітувати та візуалізувати дані в широкому діапазоні стилів, включаючи графіки, карти, діаграми, діаграми розсіювання тощо. Тим часом функціональність Power BI «AI Insights» використовує штучний інтелект для пошуку висновків у наборах даних для користувачів [17].

Основні характеристики Power BI [17]:

- підтримка підприємств різного масштабу (підходить для організацій будь-якого розміру);
- підключення до різних типів СУБД (підтримує широкий спектр джерел даних, включаючи SQL Server, Excel та інші);

- інтеграція з системами моніторингу (можливість створення інтерактивних дашбордів з оновленням даних у реальному часі);
- навчальні матеріали (широкий спектр офіційної документації та навчальних курсів);
- функціонуючий форум/спільнота (велика спільнота користувачів та експертів);
- додаткові можливості (інтеграція з іншими продуктами Microsoft, такими як Excel та Teams, підтримка створення звітів з використанням мов R та Python).

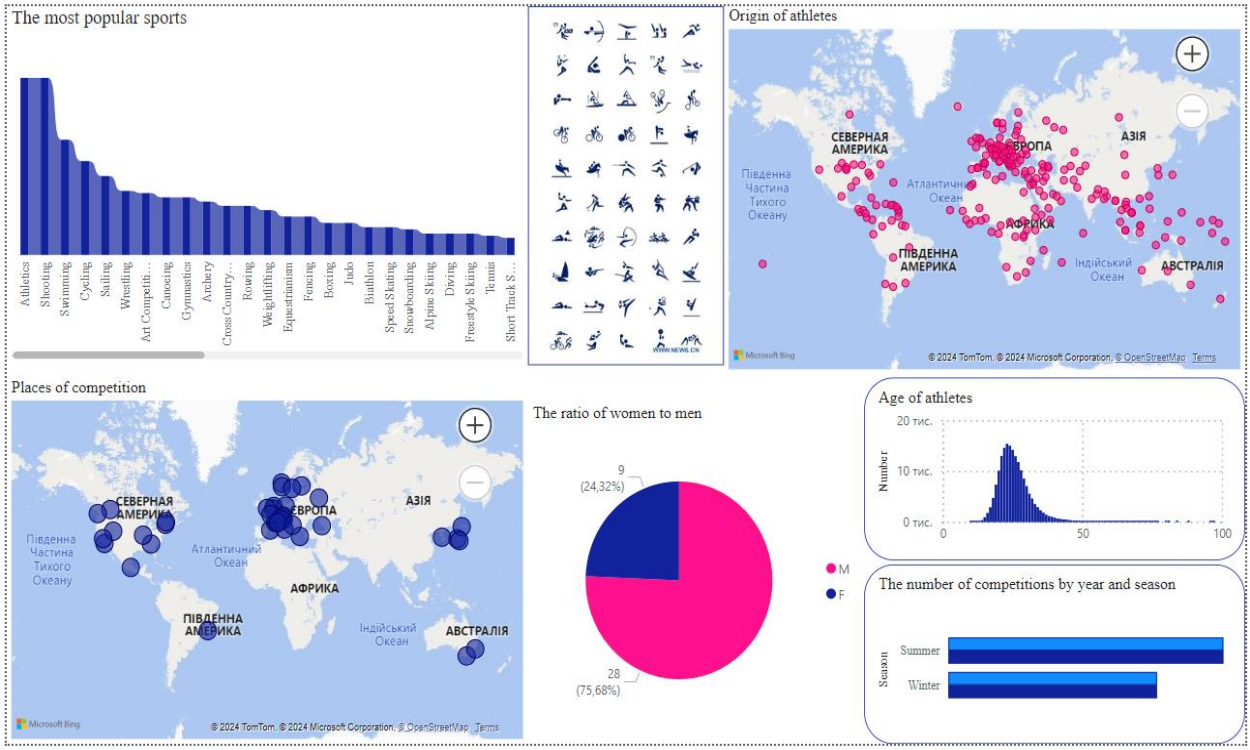

Рисунок 3.4 - Інтерфейс Power BI, сформований дашборд

Усі розглянуті інструменти мають свої переваги та можуть бути використані для аналізу та візуалізації даних. Вибір конкретного інструменту залежить від потреб, наявних ресурсів та технічних вимог.

Grafana, як відкрита платформа для моніторингу та аналізу даних, демонструє значні переваги над комерційними інструментами, такими як Metabase, PowerBI, та Business Analysis Tool, особливо у контексті операційної аналітики та моніторингу систем у реальному часі. Її архітектура, орієнтована на широкий спектр джерел даних (включаючи Prometheus, Graphite, Elasticsearch, InfluxDB

тощо), дозволяє агрегувати та візуалізувати метрики з різнорідних систем, що критично важливо для комплексного спостереження за розподіленими інфраструктурами. На відміну від PowerBI, який переважно орієнтований на структуровані бізнес-дані та інтеграцію з екосистемою Microsoft, або Metabase, що зосереджений на спрощенні запитів та звітів для бізнес-користувачів, Grafana вирізняється гнучкістю у візуалізації часових рядів та неструктурованих даних.

Ключова перевага Grafana полягає у її розширюваності та адаптивності, що забезпечується відкритою моделлю розробки та широкою екосистемою плагінів. Це дозволяє користувачам створювати високонастроювані інформаційні панелі, що відповідають специфічним потребам моніторингу, відстежуючи показники продуктивності у динамічному середовищі. Таким чином, серед представлених інструментів Grafana є доцільним вибором системи візуалізації для досягнення поставленої мети.

3.6 Архітектура аналітичної системи

Архітектура спеціалізованої аналітичної системи для візуалізації результатів вступної кампанії побудована за принципами модульності, масштабованості та ізоляції компонентів. В основі лежить використання віртуального середовища та контейнеризації, що забезпечує стабільну, контрольовану та легкорозгортувану інфраструктуру.

Уся система розгортається на віртуальній машині за допомогою Vagrant [18] та Oracle VirtualBox [19]. VirtualBox виступає як гіпервізор, що дозволяє запускати ізольоване середовище поверх основної операційної системи, а Vagrant відповідає за автоматизоване налаштування конфігурації цієї машини. Завдяки цьому досягається повна реплікація середовища розробки та розгортання без втручання у базову ОС.

Основна функціональність аналітичної системи реалізується через Docker-контейнери [20], які дозволяють запускати ізольовані сервіси, кожен з яких відповідає за окрему частину процесу обробки та аналізу даних.

Основні контейнери:

- Clickhouse (високопродуктивна колонкова СУБД, що зберігає великі обсяги аналітичних даних про вступників та конкурсні пропозиції);
- RabbitMQ [21] (брокер повідомлень, який забезпечує асинхронну передачу даних між компонентами системи);
- Grafana (платформа для побудови дашбордів, яка з'єднується з Clickhouse та відображає аналітичні звіти у зручному графічному форматі).

На рівні обробки даних використовуються два Python-скрипти, які реалізують логіку producer та consumer (Додаток А).

Скрипти та їх функції:

- producer здійснює парсинг відкритих даних з ЄДЕБО, зокрема, отримує посилання на конкурсні пропозиції та надсилає їх у чергу RabbitMQ.
- consumer отримує повідомлення з черги, завантажує всю пов'язану інформацію про конкурсну пропозицію та заяви вступників, обробляє їх і зберігає у базі даних.

Дана багаторівнева архітектура дозволяє забезпечити гнучке масштабування кожного компонента, надійне середовище обробки великих обсягів даних, просту підтримку та оновлення без впливу на інші модулі, зручну візуалізацію аналітичних звітів у режимі реального часу.

3.7 Висновок

У цьому розділі було сформульовано вимоги до інформаційної системи аналізу результатів вступної кампанії та визначено її архітектуру. Проведено порівняльний аналіз систем візуалізації, а саме Business Analysis Tool, Metabase, Grafana, Microsoft Power BI. В результаті проведеного аналізу обрано Grafana як оптимальне рішення для побудови інтерактивних дашбордів, адже вона є розширювана та адаптивна, що забезпечується відкритою моделлю розробки та широкою екосистемою плагінів.

Окрему увагу приділено методам збору даних з веб-джерел: розглянуто основні підходи до веб-скрапінг та API-інтеграції, охарактеризовано популярні мови програмування для цих завдань. При виборі мови програмування PHP, Java, C# були відкинуті відразу, адже складні у налаштування середовища та мають більшу вагу коду. Таким чином, вибір проводився між JavaScript та Python. Після проведення детального аналізу, визначено, що для виконання поставлених задач добре підходить і JavaScript, і Python. Проте JavaScript має більш складний синтаксиси та менше поширений у сфері академічних досліджень, таким чином для досягнення поставленої мети було обрано мову програмування Python.

У межах порівняння можливих рішень для системи управління базами даних розглядалися такі СУБД: Vertica, Google BigQuery, Apache Cassandra, ClickHouse. В результаті аналізу було обрано ClickHouse, як високопродуктивна аналітична СУБД, основною перевагою якої є забезпечення швидкого опрацювання великих даних завдяки стисненню, індексації та паралельній обробці. ClickHouse підтримує складну аналітику, масштабування та легко інтегрується з Grafana для візуалізації.

Таким чином, четвертий розділ закладає технічну основу для практичної реалізації інформаційної системи, забезпечуючи її цілісність, масштабованість і адаптивність до потреб освітньої аналітики.

РОЗДІЛ 4. АНАЛІЗ І УЗАГАЛЬНЕННЯ РЕЗУЛЬТАТІВ ДОСЛІДЖЕННЯ

4.1. Аналіз ефективності інформаційної системи

У ході дослідження було розроблено інформаційну систему для збору, обробки та візуалізації даних про вступників до закладів вищої освіти України. Основним критерієм ефективності системи є її здатність забезпечити оперативний доступ до актуальної інформації, зручність аналізу великих масивів даних та можливість виявлення ключових тенденцій вступної кампанії.

Архітектура системи реалізована з використанням сучасних компонентів: мова програмування Python для автоматизованого збору даних, фреймворк Playwright для інтеракції з веб-джерелами, база даних ClickHouse для збереження структурованих об’ємних даних, брокер повідомлень RabbitMQ для асинхронного обміну даними та платформа Grafana для інтерактивної візуалізації результатів аналізу.

Швидкодія системи забезпечується завдяки застосуванню ClickHouse — аналітичної колоночної СУБД, що дозволяє ефективно обробляти запити до мільйонів записів із низькою затримкою.

Гнучкість візуалізації досягнута завдяки використанню Grafana, де реалізовано низку дашбордів із параметрами фільтрації за роком вступу, університетом, формою навчання, підставою вступу, рівнем освіти тощо. Це дає змогу швидко переключатись між різними зрізами даних без необхідності повторного запуску запитів.

Стабільність збору даних забезпечується за рахунок використання Playwright, який дозволяє імітувати дії користувача у браузері та завантажувати динамічно згенеровані таблиці з веб-сайту vstup.edbo.gov.ua. Завдяки цьому зібрані дані є максимально актуальними, незалежно від структури HTML чи зміни DOM.

Таким чином, реалізована система відповідає вимогам до сучасного інструменту аналізу вступної кампанії: вона є масштабованою, продуктивною, придатною до повторного використання, гнучкою в налаштуванні та орієнтованою на кінцевого користувача.

4.2. Узагальнення висновків із візуалізації даних

Результати візуалізації даних, отриманих у процесі дослідження, дозволили сформувати систематизовані висновки щодо тенденцій вступу як на загальнодержавному рівні, так і в межах конкретного закладу вищої освіти — Національного університету харчових технологій. Використання інтерактивних панелей Grafana дозволило не лише оперативно дослідити структуру та динаміку вступної кампанії, але й виявити приховані закономірності та зміни в освітніх вподобаннях абітурієнтів.

Загальна статистика вступних кампанії ВЗО України. Аналіз статистики заяв до ЗВО України свідчить про стабільну популярність галузі знань «Інформаційні технології», що лишається однією з найконкурентніших протягом останніх років (Додаток Б). Зокрема, спеціальність «Комп’ютерні науки» займала лідируючі позиції за кількістю поданих заяв в межах цієї галузі впродовж 2021–2024 років, підтверджуючи зростаючий попит на цифрові професії.

Водночас, спостерігаються щорічні коливання у виборі пріоритетних галузей знань при поданні заяв:

- у 2021 році найбільшою популярністю користувався напрям «Соціальні науки, журналістика та інформація»;
- у 2022 та 2023 роках першість повернули «Інформаційні технології», що, ймовірно, пов’язано з розвитком ІТ-індустрії та появою великої кількості онлайн-курсів;
- у 2024 році пріоритетом абітурієнтів стала «Інженерія, виробництво та будівництво», що може свідчити про актуальність прикладних технічних спеціальностей у контексті відновлення інфраструктури та національної економіки.

Ці динамічні зміни в структурі заяв вказують на те, що вибір спеціальності абітурієнтом значною мірою визначається зовнішніми соціально-економічними факторами, змінами ринку праці та загальною політичною ситуацією в країні.

Статистика вступних кампаній у Національному університеті харчових технологій. У межах Національного університету харчових технологій —

тенденції дещо відрізняються від загальнодержавних (Додаток В). Зокрема, з 2022 року помітне поступове зменшення загальної кількості поданих заяв і кількості зарахованих студентів. Така тенденція, ймовірно, обумовлена як зовнішніми факторами (зменшення кількості випускників, війна, демографічна криза), так і внутрішніми — конкуренцією з боку інших ЗВО, маркетинговою кампанією та репутацією університету.

Окремо варто відзначити ситуацію зі спеціальністю «Комп'ютерні науки»: незважаючи на її популярність на національному рівні, в межах НУХТу кількість заяв та вступників на цю спеціальність поступово зменшується, починаючи з 2022 року. Це може свідчити про конкуренцію в межах цієї галузі або про недостатній рівень промоції освітньої програми на рівні університету.

Форма навчання у свою чергу також демонструє помітні зрушення:

- кількість студентів денної форми з 2021 року залишається стабільною, що свідчить про сталу зацікавленість в традиційному форматі навчання;
- натомість кількість студентів, які обрали заочну форму, у 2023 році зросла на 22 особи, що може свідчити про зростання попиту на гнучкі формати навчання серед працюючих або мобільних студентів.

Також виявлено зменшення кількості вступників, зарахованих на бюджетну форму, що може вказувати як на загальне скорочення бюджетного фінансування, так і на зниження конкурсного балу. Натомість кількість зарахованих на контракт має стабільну тенденцію до зростання. Це може свідчити про збереження попиту на освітні послуги навіть за умови власного фінансування.

Попри це, галузь знань «Інформаційні технології» залишається серед п'ятірки найпопулярніших у НУХТі, що підтверджує стабільний інтерес до ІТ-напрямів. Найбільш популярним напрямом упродовж останніх чотирьох років був «Інженерія, виробництво та будівництво», що свідчить про орієнтацію університету на інженерні професії. Для контрактної форми навчання особливий інтерес викликає також галузь «Бізнес, адміністрування та право», що може бути

пов’язано з її практичною спрямованістю та широкими можливостями працевлаштування.

Загалом, результати візуалізації дозволяють зробити висновок, що аналітична система не лише покращує розуміння поточних процесів у вступній кампанії, але й може бути використана як інструмент для прийняття стратегічних рішень на рівні університету — зокрема для коригування освітньої політики, підвищення конкурентоспроможності програм і покращення комунікації з абітурієнтами.

4.3. Обмеження дослідження та потенціал для подальшої роботи

У процесі реалізації дослідження були виявлені певні обмеження, які варто враховувати під час інтерпретації результатів і планування подальших етапів розвитку системи.

Аналітика ґрунтується на кількісних даних і не враховує чинники, що важко формалізуються — наприклад, репутацію університету, рекомендації знайомих, маркетингові кампанії, тощо, які впливають на вибір абітурієнтів.

Система створена на базі таких технологій, як ClickHouse, Grafana, Playwright, що забезпечують високу швидкодію й масштабованість, однак потребують налаштування та технічної підтримки для інтеграції з іншими державними чи освітніми інформаційними системами.

Оскільки вступна кампанія є щорічним процесом із постійними змінами нормативної бази, існує потреба в регулярному оновленні джерел, структур даних і запитів. Автоматизація цих процесів є важливою умовою для підтримки актуальності системи.

Потенціал для подальшої роботи:

- розширення джерел даних, зокрема шляхом інтеграції з платформами рейтингового оцінювання ЗВО, освітніми порталами, базами ринку праці;

- використання алгоритмів машинного навчання для виявлення прихованих патернів у виборі освітніх програм;
- поглиблений аналіз динаміки освітніх пріоритетів у розрізі регіонів та соціально-демографічних груп.

4.4 Висновок

Розділ 5 підсумовує результати впровадження й використання аналітичної системи для дослідження вступної кампанії до закладів вищої освіти України. Проведений аналіз підтверджує ефективність розробленого підходу до автоматизованого збору, обробки та візуалізації освітніх даних. Система виявилася дієвим інструментом для виявлення ключових тенденцій у динаміці заяв, пріоритетах вступників і популярності освітніх напрямів.

Завдяки візуалізації даних вдалося не лише окреслити загальнодержавні тенденції (наприклад, зростання інтересу до IT-галузі та інженерії), але й визначити локальні особливості вступу до окремих ЗВО — зокрема, Національного університету харчових технологій. Виявлено, як стабільні тренди (популярність певних напрямів, стабільність денної форми навчання), так і нові зрушення (зростання заочної форми, зменшення кількості вступників на бюджет).

У межах розділу також окреслено обмеження дослідження, пов’язані з неповнотою даних і потребою в подальшій автоматизації процесів. Натомість запропоновано низку перспективних напрямів розвитку системи, включаючи інтеграцію з додатковими джерелами, застосування методів машинного навчання та розширення функціональності.

Отже, результати розділу свідчать про практичну доцільність побудови подібних аналітичних систем у сфері вищої освіти, що можуть стати основою для обґрунтованого стратегічного планування.

ВИСНОВКИ

У межах даної кваліфікаційної роботи було досліджено та реалізовано інформаційну систему для збору, обробки й візуалізації даних про вступну кампанію до закладів вищої освіти України (Додаток Б, Додаток В). Актуальність теми обумовлена сучасними викликами, зокрема цифровізацією управлінських рішень у сфері освіти та потребою у швидкому доступі до структурованих аналітичних даних для ефективного планування освітніх програм.

Для досягнення поставленої мети запропоновано комплексний підхід, що сформований на базі аналітичних методів: описовий, діагностичний, прогностичний, кластерний аналізи та аналіз часових рядів. Кожен з цих методів має свою унікальну цінність, починаючи від базового узагальнення даних і виявлення тенденцій, до глибокого розуміння причинно-наслідкових зв'язків та прогнозування майбутнього попиту.

Можливість сегментувати дані за допомогою кластерного аналізу та відстежувати зміни в попиті протягом років через аналіз часових рядів забезпечує цілісну картину, необхідну для формування як стратегій розвитку окремих університетів, так і загальної державної освітньої політики. Це сприятиме підвищенню ефективності всієї системи вищої освіти в Україні.

У ході аналізу даних за галуззю «Інформаційні технології» було виявлено стабільно кількість поданих заяв з незначним падінням кількості зарахованих осіб протягом останніх років, що свідчить про актуальність спеціальності на ринку праці, але спаданням зацікавленості серед абітурієнтів, можливо, через зростання кількості приватних курсів у цій сфері. Використання кластерного аналізу дозволило сегментувати абітурієнтів за пріоритетами та формою навчання, виявляючи як загальні тенденції, так і специфічні вподобання різних груп. Аналіз часових рядів продемонстрував динаміку змін у попиті, що є важливим для прогнозування майбутніх освітніх потреб і стратегічного планування розвитку освітніх програм у галузі інформаційних технологій. Отримані результати підтверджують актуальність та практичну цінність розробленої аналітичної системи для підтримки управлінських рішень у сфері вищої освіти.

Запропонований підхід дозволяє виявити не лише загальні тенденції вступної кампанії, але й поглибити розуміння причин популярності окремих спеціальностей, передбачати зміни попиту на освітні програми, а також здійснювати сегментацію університетів і спеціальностей за релевантними критеріями. Окрему увагу приділено можливостям довгострокового стратегічного планування в системі вищої освіти на основі аналізу динаміки показників у часовому вимірі.

На основі аналізу наявних підходів до обробки освітніх даних було обґрунтовано вибір технологій і засобів реалізації системи, зокрема використання Python та Playwright для автоматизації збору даних, ClickHouse для їхнього збереження й обробки, RabbitMQ для передачі даних, а також Grafana для інтерактивної візуалізації. Реалізована архітектура забезпечила високу продуктивність, гнучкість у роботі з великими обсягами інформації та зручність у дослідженні різних аспектів вступної кампанії.

Завдяки впровадженню аналітичної системи стало можливим швидке виявлення загальнодержавних тенденцій щодо вибору спеціальностей, пріоритетів абітурієнтів, змін у формі та підставі навчання, а також дослідження динаміки вступу до окремого закладу вищої освіти. На прикладі Національного університету харчових технологій було продемонстровано прикладне значення системи в аналізі внутрішніх процесів вступної кампанії, виявленні факторів, що впливають на попит на освітні програми, і потенціалі до стратегічного планування.

Попри окремі обмеження, пов’язані з оновленням джерел даних, необхідністю інтеграції з іншими державними системами, система показала високу ефективність і практичну цінність. Результати дослідження засвідчують, що впровадження подібних аналітичних рішень є перспективним напрямом розвитку освітньої аналітики, що сприятиме підвищенню якості управлінських рішень та конкурентоспроможності вищої освіти в умовах цифрової трансформації.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Порядок прийому на навчання для здобуття вищої освіти в 2025 році. *mon.gov.ua*. 03.03.2025. URL: https://mon.gov.ua/static-objects/mon/sites/1/vishcha-osvita/vstup-2025/03/03/poryadok-pryyomu-na-navchannya-dlya-zdobuttya-vyshchoyi-osvity-v-2025-rotsi-03-03-2025.pdf. (дата звернення: 18.04.2025).
2. Консолідований рейтинг вишів України 2024 року. *OCBITA.UA*. URL: https://osvita.ua/vnz/rating/51741/. (дата звернення: 19.04.2025).
3. *Єдина державна електронна база з питань освіти*. URL: https://info.edbo.gov.ua/. (дата звернення: 19.04.2025).
4. Use of data visualization and analytics in education is growing. *SAS*. URL: https://www.sas.com/en_us/insights/articles/data-management/education-data-visualization.html. (дата звернення: 20.04.2025).
5. *Вступ.OCBITA.UA*. URL: https://vstup.osvita.ua/. (дата звернення: 20.04.2025).
6. Vertica Documentation. *Vertica*. URL: https://www.vertica.com/documentation/vertica/all/ (дата звернення: 22.04.2025).
7. From data warehouse to autonomous data and AI platform. *Google Cloud*. URL: https://cloud.google.com/bigquery?hl=uk (дата звернення: 22.04.2025).
8. Cassandra Documentation. *Apache Cassandra*. URL: https://cassandra.apache.org/doc/latest/ (дата звернення: 24.04.2025).
9. ClickHouseDocs. *ClickHouse*. URL: https://clickhouse.com/docs (дата звернення: 24.04.2025).
10. Amos D. A Practical Introduction to Web Scraping in Python. *Real Python*. 21.12.2024. URL: https://realpython.com/python-web-scraping-practical-introduction/. (дата звернення: 25.04.2025).
11. How to Scrape a Website Using Puppeteer in Node.js ?. *GeeksforGeeks*. 05.04.2023. URL: https://www.geeksforgeeks.org/how-to-scrape-a-website-using-puppeteer-in-node-js/?utm_source=chatgpt.com. (дата звернення: 25.04.2025).

12. The Best Web Scraping Tools & Software in 2025. *ScrapingBee*. 22.04.2025. URL: https://www.scrapingbee.com/blog/web-scraping-tools/. (дата звернення: 25.04.2025).
13. Updated Format of Business Analysis Tool From BIT Impulse. *LVIV IT CLUSTER*. 05.03.2024. URL: https://itcluster.lviv.ua/en/updated-format-of-business-analysis-tool-from-bit-impulse/. (дата звернення: 26.04.2025).
14. Дзюба Анастасія. Звіт з виробничої практики. Національний університет харчових технологій. 2024.(дата звернення: 26.04.2025)
15. A tour of Metabase. *Metabase*. URL: https://www.metabase.com/learn/metabase-basics/overview/tour-of-metabase. (дата звернення: 26.04.2025).
16. About Grafana. *Grafana Labs*. URL: https://grafana.com/docs/grafana/latest/introduction/. (дата звернення: 01.05.2025).
17. What is Power BI?. *Microsoft*. URL: https://learn.microsoft.com/en-us/power-bi/fundamentals/power-bi-overview. (дата звернення: 04.05.2025).
18. *HashiCorp*. URL: https://developer.hashicorp.com/vagrant. (дата звернення: 04.05.2025).
19. Powerful open source virtualization. VirtualBox. URL: https://www.virtualbox.org/. (дата звернення: 04.05.2025).
20. Гайд по Docker: концепція, устрій та принцип роботи. SPACELAB. URL: https://spacelab.ua/articles/Docker/ (дата звернення: 05.05.2025).
21. RabbitMQ 4.1 Documentation. RabbitMQ. URL: https://www.rabbitmq.com/docs (дата звернення: 05.05.2025).

ДОДАТКИ

Додаток А. Програмний код

Завантаження даних про конкурсні пропозиції /parser_01_init.py

```
import argparse
import json
import logging
from itertools import product
from typing import Dict, Any, Optional

import pika
import requests
from bs4 import BeautifulSoup

# --- Constants ---
LOG_FILE = "logs/main_form_parser_producer.log"
QUEUE_NAME = 'offer_links_parser'
DEAD_LETTER_QUEUE_NAME = 'dead_letter_queue_offer_links'
RABBITMQ_HOST = 'localhost'
ENCODING = 'utf-8'

# --- Configuration ---
class Config:
    """
    Manages application-wide configurations.
    """
    LOGGING_LEVEL = logging.INFO
    LOGGING_FORMAT = '%(asctime)s - %(levelname)s - %(message)s'
    REQUEST_TIMEOUT = 10  # seconds
```

```
    RABBITMQ_DELIVERY_MODE = 2 # Persistent messages

    # Defines allowed combinations of qualification and education base.
    # Key: qualification ID, Value: list of allowed education base IDs.
    ALLOWED_COMBINATIONS: Dict[str, list[str]] = {
        '1': ['40', '530', '610', '520', '620'],
        '2': ['40', '530', '610', '520', '620', '640'],
        '9': ['30', '40', '540', '510', '520']
    }

# --- Logging Setup ---
logging.basicConfig(
    level=Config.LOGGING_LEVEL,
    format=Config.LOGGING_FORMAT,
    handlers=[
        logging.FileHandler(LOG_FILE, encoding=ENCODING),
    ]
)

# --- Argument Parsing ---
def parse_args() -> argparse.Namespace:
    """
    Parses command-line arguments.

    Returns:
        argparse.Namespace: An object containing the parsed arguments.
    """
    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()

# --- Web Scraping Functions ---
def fetch_html_content(url: str) -> Optional[BeautifulSoup]:
    """
    Fetches HTML content from a given URL and parses it using BeautifulSoup.

    Args:
        url (str): The URL to fetch.

    Returns:
        Optional[BeautifulSoup]: A BeautifulSoup object if successful, None otherwise.
    """
    try:
        response = requests.get(url, verify=False, timeout=Config.REQUEST_TIMEOUT)
        response.raise_for_status()  # Raises HTTPError for bad responses (4xx or 5xx)
        response.encoding = ENCODING
        return BeautifulSoup(response.text, 'html.parser')
    except requests.exceptions.RequestException as e:
        logging.error(f"Помилка при завантаженні сторінки '{url}': {e}")
        return None

def extract_options_from_selector(soup: BeautifulSoup, selector_id: str) -> Dict[str, str]:
    """
    Extracts option values and text from a select HTML element.

    Args:
```

```
        soup (BeautifulSoup): The BeautifulSoup object containing the HTML.
        selector_id (str): The ID of the select element to target.

    Returns:
        Dict[str, str]: A dictionary where keys are option values and values are option texts.
    """
    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()
    }

# --- RabbitMQ Functions ---
def setup_rabbitmq_queues(channel: pika.channel.Channel):
    """
    Declares necessary RabbitMQ queues, including the dead-letter queue.

    Args:
        channel (pika.channel.Channel): The RabbitMQ channel.
    """
    channel.queue_declare(queue=DEAD_LETTER_QUEUE_NAME, durable=True)
    channel.queue_declare(
        queue=QUEUE_NAME,
        durable=True,
```

```
        arguments={
            'x-dead-letter-exchange': '',
            'x-dead-letter-routing-key': DEAD_LETTER_QUEUE_NAME
        }
    )

def send_to_queue(data: Dict[str, Any]):
    """
    Sends data as a JSON message to the RabbitMQ queue.

    Args:
        data (Dict[str, Any]): The dictionary data to send.
    """
    try:
        with
pika.BlockingConnection(pika.ConnectionParameters(host=RABBITMQ_HOST))     as
connection:
            channel = connection.channel()
            setup_rabbitmq_queues(channel)

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

            channel.basic_publish(
                exchange='',
                routing_key=QUEUE_NAME,
                body=json.dumps(data, ensure_ascii=False), # ensure_ascii=False for proper
Ukrainian character handling
```

```
            properties=properties
        )
        logging.info(f"Повідомлення успішно відправлено в чергу '{QUEUE_NAME}': {data}")
    except pika.exceptions.AMQPConnectionError as e:
        logging.error(f"Не вдалося підключитися до RabbitMQ за адресою '{RABBITMQ_HOST}': {e}")
    except json.JSONEncodeError as e:
        logging.error(f"Помилка серіалізації даних в JSON: {e}")
    except Exception as e:
        logging.error(f"Невідома помилка при відправленні даних у чергу: {e}")


# --- Main Logic ---
def main():
    """
    Main function to parse arguments, fetch data, generate combinations,
    and send tasks to the RabbitMQ queue.
    """
    args = parse_args()
    initial_url = f'https://{args.base_url}/offers/'
    soup = fetch_html_content(initial_url)
    if not soup:
        return
education_bases = extract_options_from_selector(soup, 'offers-search-education-base')
    qualifications = extract_options_from_selector(soup, 'offers-search-qualification')
    specialities = extract_options_from_selector(soup, 'offers-search-speciality')

    # Generate all possible combinations of qualification, education base, and speciality
    all_combinations = product(qualifications.keys(), education_bases.keys(), specialities.keys())
```

```
    for qualification_id, education_base_id, speciality_id in all_combinations:
        # Check if the combination of qualification and education base is allowed
        if education_base_id not in Config.ALLOWED_COMBINATIONS.get(qualification_id, []):
            logging.info(
                f"Пропущено невалідну комбінацію: "
                f"Кваліфікація='{qualifications.get(qualification_id, qualification_id)}' "
                f"({qualification_id}), "
                f"Освітня база='{education_bases.get(education_base_id, education_base_id)}' "
                f"({education_base_id})"
            )
            continue

        task_data = {
            "qualification_id": qualification_id,
            "education_base_id": education_base_id,
            "speciality_id": speciality_id,
            "base_url": args.base_url
        }
        send_to_queue(task_data)
        logging.info(f"Завдання надіслано до черги: {task_data}")
# --- Entry Point ---
if __name__ == "__main__":
    main()
```

Завантаження даних про вступників /parser_02_offers.py

```
import asyncio
import json
import logging
import random
import re
import signal
import sys
from typing import Any, Dict, List, Optional, Tuple

import aio_pika
import clickhouse_connect
from bs4 import BeautifulSoup
from playwright.async_api import Browser, BrowserContext, Page, Playwright, async_playwright
from playwright_stealth import stealth_async

# --- Configuration ---
class Config:
    """
    Manages application-wide configurations for the offer parser consumer.
    """
    # Logging
    LOG_LEVEL = logging.INFO
    LOG_FORMAT = '%(asctime)s - %(levelname)s - %(message)s'
    SUCCESS_LOG_FILE = "logs/success_links.log"

    # ClickHouse Database
    CH_HOST = 'localhost'
    CH_PORT = 8123
```

```
    CH_USERNAME = 'vagrant'
    CH_PASSWORD = 'admin123'
    CH_DATABASE = 'edu_campaign'
    CH_TABLE = 'edu_campaign.offer_request'

    # RabbitMQ
    RABBITMQ_HOST = 'localhost'
    RABBITMQ_PORT = 5672
    RABBITMQ_USERNAME = 'guest'
    RABBITMQ_PASSWORD = 'guest'
    MAIN_QUEUE = 'offer_links_parser'  # Changed from offer_requests_parser to match producer
    DEAD_LETTER_QUEUE = 'dead_letter_queue_offer_links' # Changed to match producer

    # Playwright
    BROWSER_HEADLESS = True
    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"
    REQUEST_TIMEOUT_MS = 30000  # 30 seconds
    LOAD_MORE_BUTTON_TIMEOUT_MS = 5000 # 5 seconds
    SLEEP_AFTER_GOTO_SEC = 1
    SLEEP_AFTER_CLICK_MIN_SEC = 2
    SLEEP_AFTER_CLICK_MAX_SEC = 3

    # Parsing Constants
    DEFAULT_COEFFICIENT = '1'
    DEFAULT_SCORE = 0.0
    DEFAULT_CAPACITY = 0
```

```
# --- Logging Setup ---
logging.basicConfig(
    level=Config.LOG_LEVEL,
    format=Config.LOG_FORMAT,
    handlers=[
        logging.StreamHandler(),
    ]
)

# A separate logger for successful processing
success_logger = logging.getLogger("success_logger")
success_logger.setLevel(logging.INFO)
success_handler = logging.FileHandler(Config.SUCCESS_LOG_FILE)
success_handler.setLevel(logging.INFO)
success_handler.setFormatter(logging.Formatter('%(asctime)s - %(message)s'))
success_logger.addHandler(success_handler)

# --- Global Database Client (initialized once) ---
# It's generally better to pass the client to functions or use a context manager
# if its lifetime is tied to specific operations. For a long-running consumer,
# a global client initialized once is acceptable, but ensure proper closing.
try:
    client = clickhouse_connect.get_client(
        host=Config.CH_HOST,
        port=Config.CH_PORT,
        username=Config.CH_USERNAME,
        password=Config.CH_PASSWORD,
        database=Config.CH_DATABASE
    )
    logging.info("Successfully connected to ClickHouse.")
```

```
except Exception as e:
    logging.critical(f"Failed to connect to ClickHouse: {e}")
    sys.exit(1) # Exit if database connection fails at startup

# --- Signal Handling ---
def signal_handler(sig: int, frame: Any):
    """
    Handles OS signals for graceful shutdown.
    Closes the ClickHouse connection before exiting.
    """
    logging.info('Received shutdown signal, stopping consumer...')
    if 'client' in globals() and client.is_connected():
        client.close()
        logging.info('Database connection closed.')
    sys.exit(0)

signal.signal(signal.SIGINT, signal_handler)
signal.signal(signal.SIGTERM, signal_handler)

# --- Helper for Safe HTML Element Extraction ---
def safe_get_text(element: Any, selector: str, default: str = '', attribute: str = None) -> str:
    """
    Safely extracts text or an attribute from a BeautifulSoup element.
    """
    found_element = element.find(selector) if isinstance(selector, str) else
element.select_one(selector)

    if found_element:
        if attribute:
            return found_element.get(attribute, default).strip()
```

```
        return found_element.get_text(strip=True)
    return default

# --- Playwright Browser Management ---
async def setup_browser() -> Tuple[Playwright, Browser, BrowserContext]:
    """
    Initializes Playwright, launches a browser, and creates a new context.
    """
    playwright_instance = await async_playwright().start()
    browser = await playwright_instance.chromium.launch(headless=Config.BROWSER_HEADLESS)
    context = await browser.new_context(
        user_agent=Config.USER_AGENT,
        ignore_https_errors=True
    )
    logging.info("Playwright browser and context set up.")
    return playwright_instance, browser, context

async def teardown_browser(playwright_instance: Playwright, browser: Browser, context: BrowserContext):
    """
    Closes Playwright browser and context.
    """
    await context.close()
    await browser.close()
    await playwright_instance.stop()
    logging.info("Playwright browser and context torn down.")

async def fetch_page_source(context: BrowserContext, url: str) -> str:
    """
```

```
    Fetches the full HTML content of a page, handling 'Load More' buttons.

    Args:
        context (BrowserContext): The Playwright browser context.
        url (str): The URL of the page to fetch.

    Returns:
        str: The HTML content of the page.

    Raises:
        Exception: If there's an error during page navigation or content fetching.
    """
    page: Optional[Page] = None
    try:
        page = await context.new_page()
        await stealth_async(page)  # Apply stealth techniques
        await page.goto(url, timeout=Config.REQUEST_TIMEOUT_MS)
        await asyncio.sleep(Config.SLEEP_AFTER_GOTO_SEC)

        # Continuously click "Load More" button until it disappears
        while True:
            try:
                # Wait for the button to appear within a shorter timeout
                load_more_button = await page.wait_for_selector(
                    "#requests-load", state='visible',
timeout=Config.LOAD_MORE_BUTTON_TIMEOUT_MS
                )
                logging.info("Clicking 'Load More Applications' button...")
                await load_more_button.click()
```

```
                await
asyncio.sleep(random.uniform(Config.SLEEP_AFTER_CLICK_MIN_SEC,
Config.SLEEP_AFTER_CLICK_MAX_SEC))
            except Exception:
                # Button not found or not visible, implies all data loaded
                logging.debug("No 'Load More Applications' button found or it's no longer
available.")
                break

        content = await page.content()
        return content

    except Exception as e:
        logging.error(f"Error while loading page {url}: {e}", exc_info=True)
        raise # Re-raise to be caught by the message callback for dead-lettering
    finally:
        if page:
            await page.close()

# --- HTML Parsing Functions ---
def parse_offers(soup: BeautifulSoup) -> Dict[str, Any]:
    """
    Parses the main offer details from the BeautifulSoup object.

    Args:
        soup (BeautifulSoup): The BeautifulSoup object of the page.

    Returns:
        Dict[str, Any]: A dictionary containing parsed offer data.
```

```
Raises:
    ValueError: If the main 'offer' block is not found.
"""
logging.info("Parsing 'offer' block...")
offer = soup.find('div', class_='offer')
if not offer:
    logging.error("Main 'offer' block not found on the page.")
    raise ValueError("Main 'offer' block not found.")

offer_data: Dict[str, Any] = {
    'year': 0,
    'university_code': '',
    'university_name': '',
    'degree': '',
    'admission_basis': '',
    'speciality_code': '',
    'speciality_name': '',
    'speciality_sub_name': '',
    'offer_name': '',
    'offer_code': offer.get('data-id', ''), # Default based on the main offer div
    'education_program': '',
    'faculty': '',
    'study_form': '',
    'course': '',
    'offer_type': '',
    'study_term': '',
    'application_term': '',
    'license_capacity': Config.DEFAULT_CAPACITY,
    'max_state_order': Config.DEFAULT_CAPACITY,
    'contract_capacity': Config.DEFAULT_CAPACITY,
```

```
        'education_price': Config.DEFAULT_CAPACITY,
        'subjects': '[]' # Will be JSON string
    }

    # Extract Year
    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)

    # Extract University Info
    a_tag = soup.find('a',
href=re.compile(r'https://registry\.edbo\.gov\.ua/university/\d+/'))
    if a_tag:
        href = a_tag.get('href', '')
        university_code_match = re.search(r'/university/(\d+)/', href)
        offer_data['university_code'] = university_code_match.group(1) if
university_code_match else ''
        offer_data['university_name'] = safe_get_text(a_tag, 'h5')

    # Extract Degree and Admission Basis
    h6_tag = soup.find('h6', class_='text-primary')
    if h6_tag:
        offer_data['degree'] = safe_get_text(h6_tag, 'span.text-uppercase')
        basis_text = safe_get_text(h6_tag, 'span.text-lowercase')
        basis_match = re.search(r'основа вступу - (.+)\)', basis_text)
        offer_data['admission_basis'] = basis_match.group(1) if basis_match else ''
```

```
    # Extract Speciality Details
    speciality_dt = offer.find('dt', string="Спеціальність")
    if speciality_dt:
        speciality_dd = speciality_dt.find_next_sibling('dd')
        if speciality_dd:
            offer_data['speciality_code'] = safe_get_text(speciality_dd, 'span.badge.badge-primary')
            offer_data['speciality_name'] = safe_get_text(speciality_dd, 'span.text-uppercase.text-primary')
            offer_data['speciality_sub_name'] = safe_get_text(speciality_dd, 'span.badge.badge-secondary.text-lowercase')

    # Extract other offer details using safe_get_text and direct find
    offer_data['offer_name'] = safe_get_text(offer, 'dt', default='Назва пропозиції').find_next_sibling('dd').get_text(strip=True) if offer.find('dt', string="Назва пропозиції") else ''
    offer_data['offer_code'] = safe_get_text(offer, 'dl.row.offer-university-specialities dd', offer_data['offer_code'])
    offer_data['education_program'] = safe_get_text(offer, 'dl.row.offer-study-programs dd')
    offer_data['faculty'] = safe_get_text(offer, 'dl.row.offer-university-facultet-name dd')
    offer_data['study_form'] = safe_get_text(offer, 'dl.row.offer-education-form-name dd')
    offer_data['course'] = safe_get_text(offer, 'dl.row.offer-course-name dd')
    offer_data['offer_type'] = safe_get_text(offer, 'dl.row.offer-offer-type-name dd')
    offer_data['study_term'] = safe_get_text(offer, 'dl.row.offer-education-term dd')
    offer_data['application_term'] = safe_get_text(offer, 'dl.row.offer-request-term dd')

    # Parse capacities and price, converting to int where possible
```

```
    offer_data['license_capacity'] = int(safe_get_text(offer, 'dl.row.offer-order-license dd', str(Config.DEFAULT_CAPACITY)))
    offer_data['max_state_order'] = int(safe_get_text(offer, 'dl.row.offer-max-order dd', str(Config.DEFAULT_CAPACITY)))
    offer_data['contract_capacity'] = int(safe_get_text(offer, 'dl.row.offer-order-contract dd', str(Config.DEFAULT_CAPACITY)))
    offer_data['education_price'] = int(re.sub(r'[^\d]', '', safe_get_text(offer, 'dl.row.offer-education-price dd', str(Config.DEFAULT_CAPACITY)))) # Remove non-digits

    # Parse Subjects
    subjects_list: List[Dict[str, str]] = []
    subject_divs = offer.find_all('div', class_='offer-subject-wrapper')
    for subject_div in subject_divs:
        subject_data = {
            'subject_number': safe_get_text(subject_div, 'div.subject-number'),
            'coefficient': safe_get_text(subject_div, 'div.coefficient'),
            'subject_name': safe_get_text(subject_div, 'div.subject-name'),
            'min_value': ''
        }
        min_value_text = safe_get_text(subject_div, 'div.min-value')
        match = re.search(r'\d+', min_value_text)
        subject_data['min_value'] = match.group() if match else ''
        subjects_list.append(subject_data)
    offer_data['subjects'] = json.dumps(subjects_list, ensure_ascii=False)

    logging.info("Parsing 'offer' block completed.")
    return offer_data

def parse_requests(soup: BeautifulSoup) -> List[Dict[str, Any]]:
    """
```

```
Parses individual application request details from the BeautifulSoup object.

Args:
    soup (BeautifulSoup): The BeautifulSoup object of the page.

Returns:
    List[Dict[str, Any]]: A list of dictionaries, each representing a parsed request.
"""
requests_list: List[Dict[str, Any]] = []
request_divs = soup.find_all('div', class_='offer-request')

logging.info(f"Parsing {len(request_divs)} 'request' blocks...")
for request_div in request_divs:
    request_data: Dict[str, Any] = {
        'request_number': safe_get_text(request_div, 'div.offer-request-n div'),
        'student_name': safe_get_text(request_div, 'div.offer-request-fio div'),
        'status': safe_get_text(request_div, 'div.offer-request-status div'),
        'priority': safe_get_text(request_div, 'div.offer-request-priority div'),
        'confirmation': 'No',
        'documents': 'No',
        'avg_score': Config.DEFAULT_SCORE,
        'scores': '[]', # Will be JSON string
        'regional_coefficient': Config.DEFAULT_COEFFICIENT,
        'village_coefficient': Config.DEFAULT_COEFFICIENT,
        'industry_coefficient': Config.DEFAULT_COEFFICIENT,
        'quota': ''
    }

    # Check for confirmation and documents
    if request_div.find('div', class_='offer-confirm'):
```

```
            if request_div.find('div', class_='offer-confirm').find('div', class_='od-2') or \
               request_div.find('div', class_='offer-confirm').find('div', class_='od-1'):
                request_data['confirmation'] = 'Yes'
        if request_div.find('div', class_='offer-documents') and \
           request_div.find('div', class_='offer-documents').find('div', class_='od-1'):
            request_data['documents'] = 'Yes'

        # Parse average score, converting string to float
        avg_score_text = safe_get_text(request_div, 'div.offer-request-kv div').replace(',', '.')
        try:
            request_data['avg_score'] = float(avg_score_text) if avg_score_text else
Config.DEFAULT_SCORE
        except ValueError:
            logging.warning(f"Could not convert avg_score '{avg_score_text}' to float for
request: {request_data['request_number']}")
            request_data['avg_score'] = Config.DEFAULT_SCORE

        # Parse subject scores and coefficients
        subject_scores: Dict[str, str] = {}
        subjects_div = request_div.find('div', class_='offer-subjects')
        if subjects_div:
            for subject_item in subjects_div.find_all('div', class_='offer-subject'):
                classes = subject_item.get('class', [])
                kv_text = safe_get_text(subject_item, 'div.kv')
                sn_text = safe_get_text(subject_item, 'div.sn')
                formula_text = safe_get_text(subject_item, 'div.f')

                if 'indicator-rk' in classes:
                    match = re.search(r"[\d.]+", kv_text)
```

```
                request_data['regional_coefficient'] = match.group() if match else
Config.DEFAULT_COEFFICIENT
            elif 'indicator-sk' in classes:
                match = re.search(r"[\d.]+", kv_text)
                request_data['village_coefficient'] = match.group() if match else
Config.DEFAULT_COEFFICIENT
            elif 'indicator-gk' in classes:
                match = re.search(r"[\d.]+", kv_text)
                request_data['industry_coefficient'] = match.group() if match else
Config.DEFAULT_COEFFICIENT
            elif 'indicator-q' in classes:
                request_data['quota'] = sn_text
            else: # Regular subject score
                if sn_text and formula_text:
                    score_match = re.search(r"=\s*([\d.]+)", formula_text)
                    if score_match:
                        subject_scores[sn_text] = score_match.group(1)
        request_data['scores'] = json.dumps(subject_scores, ensure_ascii=False)

        if request_data['request_number']: # Only add if request number exists
            requests_list.append(request_data)

    logging.info(f"Parsing 'request' blocks completed. Total applications found:
{len(requests_list)}")
    return requests_list

def get_application_count(soup: BeautifulSoup) -> int:
    """
    Extracts the total expected number of applications from the page.
    """
```

```
    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("Could not find or determine the total number of applications from the page.")
    return 0

# --- Database Interaction ---
def save_to_db(offer_data: Dict[str, Any], requests_data: List[Dict[str, Any]], db_client: clickhouse_connect.ClickHouseClient):
    """
    Prepares and inserts parsed offer and request data into the ClickHouse database.

    Args:
        offer_data (Dict[str, Any]): Dictionary containing parsed offer details.
        requests_data (List[Dict[str, Any]]): List of dictionaries, each with request details.
        db_client (clickhouse_connect.ClickHouseClient): The ClickHouse client instance.
    """
    data_to_insert = []
    for request in requests_data:
        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'],
        offer_data['license_capacity'],
        offer_data['max_state_order'],
        offer_data['contract_capacity'],
        offer_data['education_price'],
        offer_data['subjects'],
        float(request['regional_coefficient']),
        float(request['village_coefficient']),
        float(request['industry_coefficient']),
        request['quota'],
        request['request_number'],
        request['student_name'],
        request['status'],
        request['priority'],
        request['confirmation'],
        request['documents'],
        float(request['avg_score']),
        request['scores'],
        '' # Placeholder for Region, as specified
    )
    data_to_insert.append(row)
```

```
    if not data_to_insert:
        logging.info("No data to insert into ClickHouse.")
        return

    try:
        db_client.insert(Config.CH_TABLE, data_to_insert)
        logging.info(f"Successfully saved {len(data_to_insert)} applications to DB for offer code {offer_data['offer_code']}.")
    except Exception as e:
        logging.error(f"Error while saving records to ClickHouse DB for offer code {offer_data['offer_code']}: {e}", exc_info=True)
        raise # Re-raise to signal failure to the message callback

# --- Message Processing Logic ---
async def process_link(context: BrowserContext, link: str, db_client: clickhouse_connect.ClickHouseClient):
    """
    Orchestrates the processing of a single link: fetches, parses, and saves data.

    Args:
        context (BrowserContext): The Playwright browser context.
        link (str): The URL of the offer page to process.
        db_client (clickhouse_connect.ClickHouseClient): The ClickHouse client instance.

    Raises:
        ValueError: If data inconsistencies are found (e.g., application count mismatch).
        Exception: For any underlying errors during fetching, parsing, or saving.
    """
    logging.info(f"Processing link: {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"Expected number of applications: {expected_application_count}")

    offer_data = parse_offers(soup)
    requests_data = parse_requests(soup)

    if len(requests_data) != expected_application_count:
        error_msg = (f"Extracted application count ({len(requests_data)}) "
                     f"does not match expected ({expected_application_count}) for link: {link}.")
        logging.error(error_msg)
        raise ValueError(error_msg)

    if requests_data:
        save_to_db(offer_data, requests_data, db_client)

    success_logger.info(f"Success processing: Offer Code={offer_data.get('offer_code', 'N/A')}, URL={link}")

async def rabbitmq_callback(message: aio_pika.IncomingMessage, context: BrowserContext):
    """
    Callback function for RabbitMQ messages. Processes the message and acknowledges it.
    If an error occurs, the message is not re-queued to prevent infinite loops.
    """
    link = message.body.decode(errors='ignore') # Decode with error handling
```

```
    logging.info(f"Received message for link: {link}")
    try:
        # Pass the global client directly
        await process_link(context, link, client)
        await message.ack() # Acknowledge successful processing
        logging.info(f"Message for {link} successfully processed and acknowledged.")
    except Exception as e:
        logging.error(f"Failed to process message for link: {link}. Error: {e}", exc_info=True)
        # Nack the message without re-queueing; it will go to the dead-letter queue
        # if configured properly in the producer or if this consumer's queue has DLX.
        # Given the previous configuration, `requeue=False` is typically preferred here
        # for unrecoverable errors to prevent message loops.
        await message.nack(requeue=False)
        logging.info(f"Message for {link} negatively acknowledged (not re-queued).")


# --- Consumer Main Loop ---
async def consume_messages():
    """
    Sets up the RabbitMQ consumer and starts listening for messages.
    Manages Playwright browser lifecycle.
    """
    playwright_instance, browser, context = await setup_browser()
    connection: Optional[aio_pika.RobustConnection] = None
    try:
        connection = await aio_pika.connect_robust(
            host=Config.RABBITMQ_HOST,
            port=Config.RABBITMQ_PORT,
            login=Config.RABBITMQ_USERNAME,
            password=Config.RABBITMQ_PASSWORD
```

```
    )

    async with connection:
        channel = await connection.channel()
        await channel.set_qos(prefetch_count=1) # Process one message at a time

        # Declare dead-letter queue (consumer side)
        await channel.declare_queue(Config.DEAD_LETTER_QUEUE, durable=True)
        logging.info(f"Dead-letter queue '{Config.DEAD_LETTER_QUEUE}' declared.")

        # Declare main queue with dead-letter exchange configured
        main_queue = await channel.declare_queue(
            Config.MAIN_QUEUE,
            durable=True,
            arguments={
                'x-dead-letter-exchange': '',
                'x-dead-letter-routing-key': Config.DEAD_LETTER_QUEUE
            }
        )
        logging.info(f"Main queue '{Config.MAIN_QUEUE}' declared with DLX to '{Config.DEAD_LETTER_QUEUE}'.")

        # Start consuming messages
        # Use a lambda to pass the `context` to the callback
        await main_queue.consume(lambda message: rabbitmq_callback(message, context), no_ack=False)
        logging.info(f"Consumer started on queue '{Config.MAIN_QUEUE}'. Waiting for messages...")
```

```
            # Keep the consumer running indefinitely
            await asyncio.Future()

    except aio_pika.exceptions.AMQPConnectionError as e:
        logging.critical(f"Failed to connect to RabbitMQ: {e}")
    except Exception as e:
        logging.critical(f"An unhandled error occurred in the consumer: {e}", exc_info=True)
    finally:
        if connection and not connection.is_closed:
            await connection.close()
            logging.info("RabbitMQ connection closed.")
        await teardown_browser(playwright_instance, browser, context)

# --- Entry Point ---
def main():
    """Entry point for the script."""
    asyncio.run(consume_messages())

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

Додаток Б. Статистика вступної кампанії у галузі "Інформаційні технології" по ВНЗ України

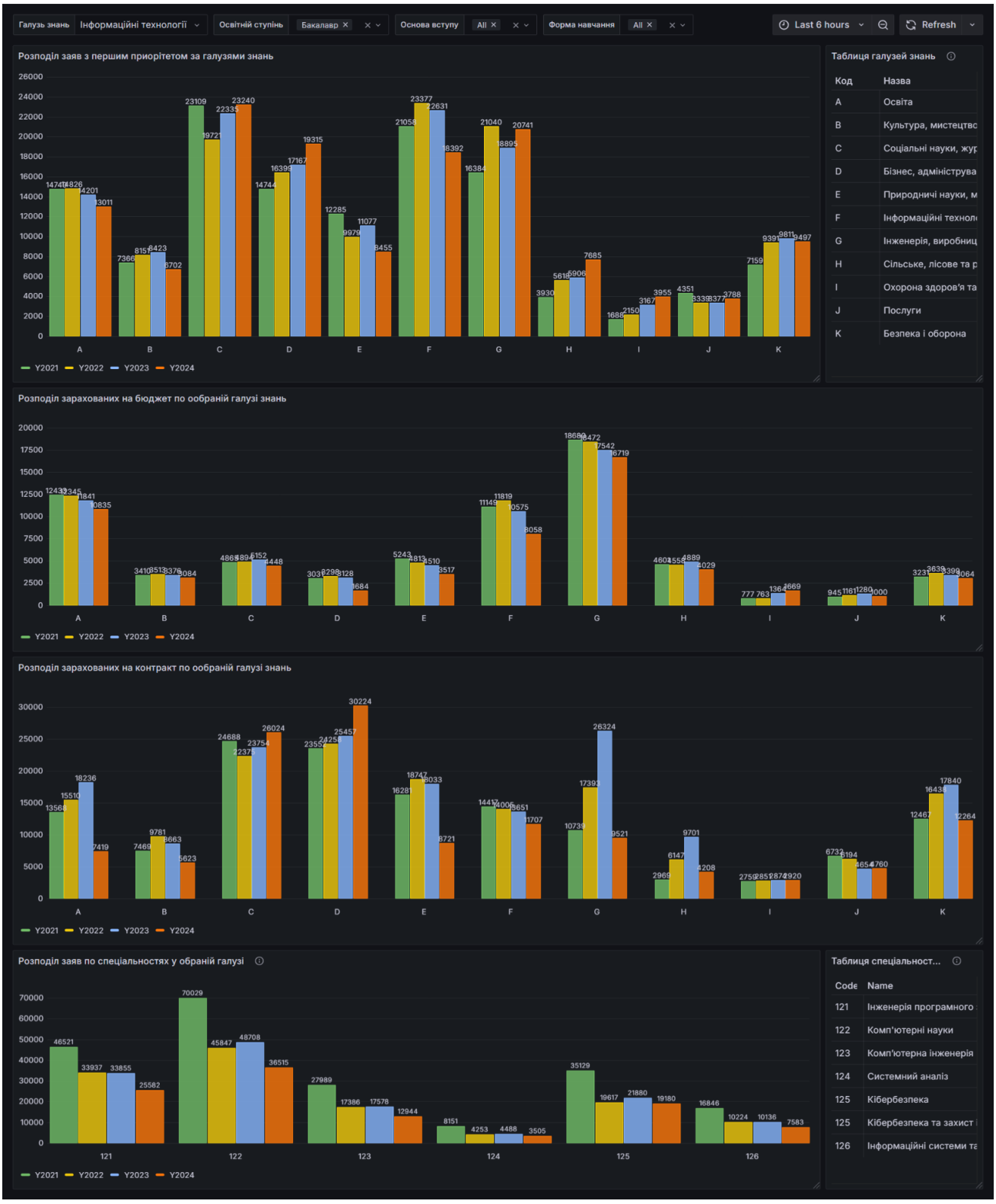

Рисунок Б.1. - Ретроспективний аналіз кількісного розподілу заяв в галузі знань “Інформаційні технології”

Додаток В. Статистика вступної кампанії у Національному університеті харчових технологій

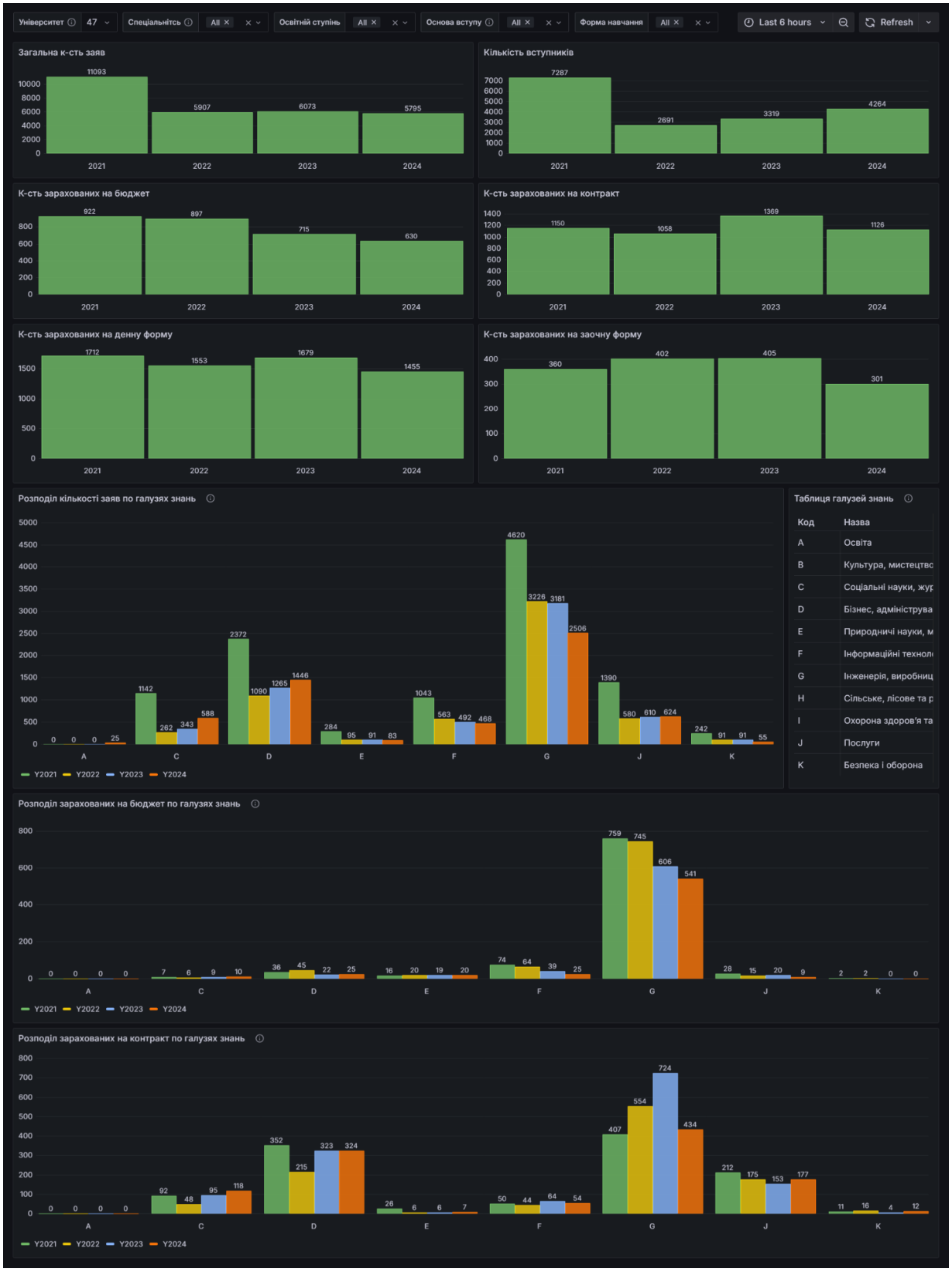

Рисунок В.1. - Ретроспективний аналіз розподілу заяв за галузями знань