Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Інформаційно-аналітичних технологій та менеджменту
(повна назва)
Кафедра Інформатики
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти другий (магістерський)

РОЗРОБКА ТА ДОСЛІДЖЕННЯ МЕТОДІВ ЗІСТАВЛЕННЯ ТОВАРНИХ ПРОПОЗИЦІЙ ДЛЯ СЕРВІСІВ-АГРЕГАТОРІВ НА ОСНОВІ АНАЛІЗУ МУЛЬТИМОДАЛЬНОЇ ІНФОРМАЦІЇ
(тема)

Виконав:
здобувач 2 року навчання,
групи ІНФм-24-1

Топчій М.А.
(прізвище, ініціали)

Спеціальності 122 Комп'ютерні науки
(код і повна назва спеціальності)

Тип програми освітньо-професійна

Освітня програма Інформатика
(повна назва освітньої програми)

Керівник доц. Яковлева О.В.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри _____ Кобилін О.А.
(підпис) (прізвище, ініціали)

2025 р.

Харківський національний університет радіоелектроніки

Факультет Інформаційно-аналітичних технологій та менеджменту
(повна назва)

Кафедра Інформатики
(повна назва)

Рівень вищої освіти другий (магістерський)

Спеціальність 122 Комп'ютерні науки
(код і повна назва)

Тип програми освітньо-професійна

Освітня програма Інформатика
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«____»_____ 2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові Топчію Микиті Андрійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Розробка та дослідження методів зіставлення товарних пропозицій для сервісів-агрегаторів на основі аналізу мультимодальної інформації

затверджена наказом університету від 14 листопада 2025 року № 1045 Ст

2. Термін подання студентом роботи до екзаменаційної комісії 12 грудня 2025 р.

3. Вихідні дані до роботи науково-методична та науково-технічна література, матеріали конференцій, дані інтернет-мережі, бібліотека комп'ютерного зору з відкритим кодом Tensorflow.js, програмна платформа Node.js, мова програмування TypeScript, фреймоврк puppeteer для Node.js, фреймворк для веброзробки SvelteKit, хмарний сервіс бази даних MongoDB Atlas, .середовище розробки JetBrains Webstorm, середовище розробки Microsoft Visual Studio Code, SBERT, CLIP.

4. Перелік питань, що потрібно опрацювати в роботі
1. Огляд всіх методів вебскрейпінгу.
2. Огляд існуючих методів вирішення проблеми зіставлення товарів.
3. Аналіз сучасних онлайн-крамниць музичних інструментів.
4. Проєктування архітектури вебплатформи для об'єднання товарних пропозицій.
5. Розробка алгоритму вирішення проблеми зіставлення товарів.
6. Розробка вебплатформи для об'єднання товарних пропозицій.

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри) Актуальність проблеми об'єднання даних, постановка задачі, опис архітектури, демонстрація роботи алгоритмів вебскрейпінгу, огляд сучасних онлайн-крамниць, демонстрація роботи алгоритмів вирішення проблеми зіставлення товарів.

6. Консультанти розділів роботи (п.6 включається до завдання за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Строк / терміни виконання етапів роботи	Примітка
1	Отримання завдання на кваліфікаційну роботу	29.09.2025	
2	Аналіз завдання, підбір літератури	30.09.25-07.10.25	
3	Аналіз літератури з досліджуваної проблеми	08.10.25-14.10.25	
4	Аналіз технічних засобів	15.10.25-20.10.25	
5	Розробка алгоритмів	21.10.25-27.10.25	
6	Програмна реалізація	28.10.25-05.11.25	
7	Обґрунтування отриманих результатів	06.11.25-11.11.25	
8	Оформлення пояснювальної записки	12.11.25-14.11.25	
9	Перевірка на нормоконтроль	25.11.25-27.11.25	
10	Перевірка на плагіат	27.11.25-29.11.25	
11	Рецензування	30.11.25	
12	Підготовка презентації та доповіді	01.12.25-02.12.25	
13	Занесення роботи в електронний архів	22.12.25	
14	Попередній захист кваліфікаційної роботи	22.12.25	

Дата видачі завдання 29 вересня 2025 р.

Студент _____
(підпис)

Керівник роботи _____ доц. Яковлева О.В.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка до кваліфікаційної роботи: 80 с., 2 табл., 43 рис., 41 джерела.

АГРЕГАТОР ТОВАРІВ, ВЕБПАРСЕР, ЕВРИСТИЧНІ МЕТОДИ, ЗІСТАВЛЕННЯ ЗОБРАЖЕНЬ, КЛАСИФІКАЦІЯ ТЕКСТУ, МУЗИЧНІ ІНСТРУМЕНТИ, МУЛЬТИМОДАЛЬНІ МОДЕЛІ, CLIP, PRODUCT MATCHING, PYTORCH, SBERT, VISION TRANSFORMER.

Об’єктом дослідження є товарні пропозиції музичних інструментів з онлайн-крамниць, представлені у вигляді текстових описів та зображень.

Предметом дослідження є методи автоматичного зіставлення таких пропозицій на основі мультимодального аналізу.

Метою дослідження є розробка ультимативного алгоритму зіставлення товарних пропозицій музикальних інструментів на основі мультимодальної інформації та створенням програмного застосунку-агрегатора, що забезпечує автоматичне виявлення однакових інструментів у різних магазинах.

У дослідженні проведено аналіз сучасних методів оброблення текстової та зорової інформації, зокрема евристичних моделей нормалізації назв, семантичних моделей на основі Sentence-BERT, а також візуальних моделей ResNet-50, Vision Transformer та CLIP, реалізованих у бібліотеках PyTorch та TensorFlow. Сформовано структурований датасет товарів із найбільших українських інтернет-магазинів музичних інструментів.

Наукова новизна дослідження полягає у створенні комплексного методу зіставлення товарних пропозицій музичних інструментів, який поєднує евристичні, семантичні та візуальні ознаки в єдиному мультимодальному просторі. Запропонований підхід дозволяє значно підвищити коректність зіставлення в умовах високої структурної подібності зображень та неоднорідності текстових описів.

ABSTRACT

Explanatory note to the qualification work: 80 pages, 2 table, 43 figures, 41 sources.

CLIP, COMPUTER VISION, DATA AGGREGATION, HEURISTIC SIMILARITY, MULTIMODAL MODELS, MUSICAL INSTRUMENTS, PRODUCT MATCHING, PYTORCH, SBERT, VISION TRANSFORMER, WEB SCRAPING.

The object of the study is the product offers of musical instruments from online stores, presented in the form of text descriptions and images.

The subject of the study is the methods of automatic comparison of such offers based on multimodal analysis.

The aim of the study is to develop and analyse modern methods of comparing products by name, description and photos with the subsequent creation of a software application-aggregator that provides automatic detection of identical instruments in different stores.

The study analyzes modern methods of processing text and visual information, in particular heuristic models of name normalization, semantic models based on Sentence-BERT, as well as visual models ResNet-50, Vision Transformer and CLIP, implemented in the PyTorch and TensorFlow libraries. A structured dataset of products from the largest Ukrainian online stores of musical instruments has been formed.

The scientific novelty of the study lies in the creation of a comprehensive method of comparing product offers of musical instruments, which combines heuristic, semantic and visual features in a single multimodal space. The proposed approach allows to significantly increase the correctness of comparison in conditions of high structural similarity of images and heterogeneity of text descriptions.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

ШІ – штучний інтелект

CI/CD – Continuous integration, contiuous delivery

CLIP – Contrastive Language-Image Pre-training

Fine-Tuning – «донавчання»

IDE – Integrated Development Environment (інтегроване середовище розробки)

PIL – Python Imaging Library (бібліотека зображень Python)

Product Matching – Зіставлення товарних пропозицій

ResNet – Residual Network («залишкова мережа»)

SBERT – Sentence Transformers

TF-IDF – Term Frequency-Inverse Document Frequency

ViT – Vision Transformer

Word2Vec – technique for creating word embeddings

ВСТУП

У сучасному світі електронної комерції агрегатори товарних пропозицій відіграють ключову роль у зручному порівнянні товарів з різних онлайн-магазинів. Однак, незважаючи на численні переваги, існує суттєва проблема: однакові товари можуть бути представлені з різними назвами, описами, артикулом або зображеннями. Це ускладнює процес ідентифікації та зіставлення товарів, що є критичним для ефективної роботи агрегаторів.

Вирішення проблеми зіставлення товарів у контексті продукції музичних інструментів полягає в тому, щоб розробити унікальний та найефективніший метод знаходження однакових товарів серед різнорідної та неоднозначної мультимодальної інформації у вузькій галузі музичного асортименту. У таких ситуаціях на допомогу приходять мультимодальні методи, які поєднують текстову та візуальну інформацію для покращення точності зіставлення товарів.

Дослідження показують, що інтеграція текстових та візуальних даних може значно покращити результати зіставлення товарів; демонструють, як поєднання текстових та візуальних ознак може підвищити точність ідентифікації однакових товарів у онлайн-торгівлі [1].

Також, деякі роботи показують, що використання мультимодальних векторних представлень дозволяє досягти високої точності виявлення дублікатів товарів, що є важливим аспектом для агрегаторів [2].

Метою цього дослідження є аналіз методів зіставлення товарних пропозицій для сервісів-агрегаторів на мультимодальної інформації. Особливу увагу буде приділено вирішенню проблеми ідентифікації однакових товарів з різними назвами та зображеннями, що є актуальним завданням у сфері електронної комерції. Аналіз основних методів вебскрейпінгу та огляд існуючих сервісів порівняння товарних пропозицій в галузі музикальних інструментів

1 ПОТОЧНИЙ СТАН ТА УСПІХИ В РОВ'ЯЗАННЯ ЗАДАЧІ ЗІСТАВЛЕННЯ ТОВАРІВ

1.1 Проблема зіставлення однакових товарів із різних магазинів

У сучасній електронній комерції зростає кількість онлайн-магазинів, які пропонують схожі або однакові товари. Незважаючи на те, що продукція може бути ідентичною, її подання на різних платформах значно відрізняється. В описах товарів застосовуються різні слова та фрази, що описують один і той самий продукт, а фотографії можуть бути зроблені під різними ракурсами або в іншому освітленні. Це створює складність для автоматизованих систем агрегаторів, які повинні визначати, чи є два лістинги одним товаром [3-4].

Відмінності у характеристиках товарів ще більше ускладнюють процес зіставлення. Продавці можуть вказувати різні атрибути, такі як розмір, колір, бренд або матеріал, або не вказувати їх зовсім. Динамічні зміни у товарах, оновлення цін, описів і фотографій призводять до появи нових лістингів для того самого товару, що створює додаткові перешкоди для точного зіставлення [5-7]. Внаслідок цього платформи ризикують відображати дублікати або неповну інформацію, що негативно впливає на досвід користувачів та ефективність пошуку.

Невірне зіставлення товарів може призвести до серйозних наслідків для онлайн-магазинів і агрегаторів. Користувачі можуть стикатися з суперечливою інформацією, що знижує довіру до платформи, а неправильне відображення наявності товарів впливає на управління запасами і логістику. Також це може зменшувати конверсію, оскільки користувачі можуть не знайти потрібний товар через дублювання або неповні лістинги [8-9]. Саме через це проблема зіставлення однакових товарів із різних магазинів залишається однією з ключових задач у сфері електронної комерції.

Для ефективного вирішення цієї проблеми необхідно поєднувати різні джерела інформації, включно з текстовими описами, зображеннями та атрибутами товарів, а також застосовувати сучасні алгоритми мультимодального аналізу, що здатні підвищити точність визначення однакових товарів і мінімізувати негативні наслідки для користувачів та бізнесу.

1.2 Огляд сервісів для порівняння товарів серед онлайн-магазинів

У сучасному електронному комерційному середовищі сервіси для порівняння цін та характеристик товарів стали важливим інструментом як для споживачів, так і для бізнесів. Вони дозволяють швидко знаходити найвигідніші пропозиції, оцінювати конкурентні ціни та приймати обґрунтовані рішення щодо покупки або ціноутворення. До загальних платформ порівняння належать Google Shopping, Price.com та PriceGrabber, які агрегують дані з численних онлайн-магазинів і надають користувачам можливість порівнювати ціни, переглядати рейтинги та відгуки, а також здійснювати покупки безпосередньо через платформу. Ці сервіси використовують різні методи збору даних, включно з веб-скрейпінгом, API інтеграціями та партнерськими програмами, щоб забезпечити актуальну інформацію про товари та їх доступність [10].

Попри широку популярність, існуючі платформи мають суттєві обмеження. По-перше, не всі продавці представлені у базах даних сервісів порівняння або не є частиною юридичного договору про партнерство, що одразу може виключити маловідомі або починаючі магазини, що обмежує повноту інформації для користувачів. По-друге, актуальність цін може не відповідати дійсності через затримки у оновленні даних або технічні збої, що призводить до відображення застарілої інформації. Крім того, сервіси часто не враховують індивідуальні потреби користувачів, що знижує ефективність

персоналізованого пошуку. Важливим аспектом є також питання доступності: деякі платформи не відповідають стандартам для людей з обмеженими можливостями, що ускладнює їх використання певними категоріями користувачів. Нарешті, у деяких юрисдикціях існують законодавчі обмеження на застосування сервісів порівняння цін, що може обмежувати їхню ефективність і доступність.

Спеціалізовані сервіси, такі як Price2Spy, NetRivals та Acelerar, пропонують більш точне порівняння товарів і автоматизоване зіставлення даних. Вони використовують алгоритми машинного навчання для визначення однакових товарів на різних платформах та стандартизації атрибутів продуктів, що зменшує кількість помилок та підвищує точність порівняння. Однак навіть ці сервіси не завжди повністю охоплюють специфічні категорії товарів, наприклад музичні інструменти. У сфері музичних інструментів існують відносно невеликі спеціалізовані платформи, такі як Sweetwater та Gear4music, які дозволяють порівнювати ціни та характеристики окремих моделей. Проте вони часто обмежені в інтеграції між собою, не забезпечують повної мультимодальної інформації про товари і переважно фокусуються на текстових описах, залишаючи поза увагою візуальні або аудіо характеристики продуктів [11].

Таким чином, існуючі сервіси порівняння товарів надають споживачам та бізнесам зручні інструменти для оцінки ринку, однак вони мають суттєві обмеження в охопленні продавців, точності даних, персоналізації, доступності та інтеграції мультимодальної інформації. Це обґрунтовує потребу у розробці нових інтегрованих платформ для порівняння товарів на основі принципово нового алгоритму із врахуванням мультимодальної інформації про товари, який зможе забезпечити більш точне та комплексне зіставлення пропозицій, включно з музичними інструментами та обладнанням.

1.3 Аналіз основних методів вебскрейпінгу

Вебскрейпінг є важливим інструментом для збору даних з вебсайтів, особливо коли офіційний API відсутній або обмежений. Існують різні методи вебскрейпінгу, кожен з яких має свої переваги та обмеження.

Мануальний вебскрейпінг полягає у ручному копіюванні даних з вебсторінок. Цей метод є найпростішим, але він надзвичайно трудомісткий і не підходить для обробки великих обсягів даних. Він може бути корисним для одноразового збору інформації або для перевірки даних, але не є ефективним для регулярного використання.

Використання API є більш ефективним методом, коли вебсайт надає офіційний інтерфейс для доступу до своїх даних. API зазвичай надають структуровані дані у форматах, таких як JSON або XML, що полегшує їх обробку та інтеграцію. Однак не всі вебсайти надають API, і навіть коли вони є, доступ може бути обмежений або платним.

DOM парсинг передбачає аналіз структури HTML документа для виділення необхідних даних. Це дозволяє автоматично витягувати інформацію з вебсторінок без необхідності взаємодії з API. DOM парсинг є потужним інструментом, але він вимагає детального розуміння структури вебсторінки та може бути чутливим до змін у її макеті [12].

Кожен з цих методів має свої переваги та недоліки, і вибір оптимального методу залежить від конкретних вимог та умов задачі. Вибір методу слід здійснювати з урахуванням доступності даних, необхідної точності, обсягу інформації та технічних можливостей.

1.3.1 Мануальний вебскрейпінг

Мануальний вебскрейпінг є найпростішим і найдавнішим методом збору даних з вебсайтів, який передбачає ручне копіювання інформації з

вебсторінок у локальні таблиці або документи. Цей підхід не вимагає спеціалізованого програмного забезпечення і може бути застосований у випадках, коли потрібно зібрати невелику кількість даних або коли сторінка має складну структуру, яку важко обробити автоматичними засобами [13]. Основною перевагою мануального вебскрейпінгу є точність даних, оскільки людина може відфільтрувати непотрібну інформацію та виправити помилки одразу під час збору.

Проте цей метод має суттєві обмеження, що роблять його малоефективним для великих обсягів даних або регулярного моніторингу змін на сайтах. Процес є надзвичайно трудомістким і потребує значного часу, особливо якщо необхідно зібрати інформацію з багатьох сторінок або різних платформ одночасно. Крім того, мануальний вебскрейпінг не забезпечує автоматичного оновлення даних, що ускладнює підтримку актуальної інформації. Через це цей метод використовується переважно для тестових або одноразових завдань, а для масштабних проектів з аналізу товарних пропозицій від онлайн-магазинів його ефективність є обмеженою.

1.3.2 Використання API

Використання API є одним із найбільш ефективних методів збору даних з вебсайтів, особливо коли мова йде про регулярне оновлення інформації та інтеграцію з іншими системами. API (Application Programming Interface) надає стандартизований інтерфейс для доступу до даних вебресурсу, дозволяючи отримувати інформацію у структурованому форматі, такому як JSON або XML. Це значно спрощує обробку даних та зменшує ризик помилок у порівнянні з мануальним збором або парсингом HTML-документів [14].

Застосування API дозволяє автоматизувати процес збору даних, що є критично важливим для проектів з великим обсягом інформації, таких як

платформи для порівняння товарів у різних онлайн-магазинах. Крім того, API часто надають додаткові функціональні можливості, наприклад, фільтрацію, сортування та пошук по атрибутах товарів, що дозволяє зменшити обсяг обробки на стороні клієнта та підвищити ефективність аналізу.

Проте використання API має свої обмеження. Доступ до нього може бути платним або обмеженим за кількістю запитів на день, що впливає на масштабованість проекту. Крім того, не всі вебсайти надають відкритий API, особливо це стосується невеликих або нішевих магазинів, що змушує розробників поєднувати API з іншими методами збору даних. Незважаючи на ці обмеження, API залишається одним із найбільш надійних та безпечних способів збору структурованих даних з вебресурсів.

1.3.3 Метод парсингу DOM

Метод парсингу DOM (Document Object Model) є методом збору даних з вебсторінок, який передбачає аналіз структури HTML або XML документа для виділення необхідної інформації. За допомогою DOM парсингу сторінка представлена у вигляді деревовидної структури, де кожен елемент HTML розглядається як вузол. Це дозволяє програмно навігувати сторінкою, вибирати потрібні елементи за тегами, класами або ідентифікаторами та витягати текст, атрибути або інші дані. Такий підхід дає змогу автоматизувати збір інформації навіть зі складних вебсайтів, де ручний збір даних був би неефективним.

Цей метод є потужним інструментом, оскільки дозволяє адаптуватися до різних структур вебсторінок та виділяти лише релевантні дані, ігноруючи зайву інформацію, наприклад рекламні блоки або меню навігації. Метод широко застосовується у проєктах з вебскрейпінгу для побудови баз даних товарів, порівняння цін або моніторингу контенту на сайтах конкурентів. Незважаючи на свої переваги, DOM парсинг має певні обмеження: він

чутливий до змін у структурі вебсторінки, що може призводити до помилок при витяганні даних, а також потребує більшої уваги до оптимізації коду, щоб забезпечити ефективну обробку великих обсягів інформації. Для масштабних проектів часто використовують поєднання DOM парсингу із іншими методами збору даних, наприклад API або автоматизованим вебскрейпінгом за допомогою фреймворків на кшталт Puppeteer або Selenium.

1.4 Сучасні досягнення в обробці текстової та зорової інформації

Сучасний розвиток технологій обробки тексту та зображень визначається переходом від ручного створення ознак до використання глибинних нейронних мереж та трансформерних архітектур. У сфері обробки природної мови це означає перехід від класичних статистичних методів, таких як TF-IDF [15] і латентно-семантичний аналіз, до контекстуалізованих мовних моделей, здатних враховувати значення слів залежно від контексту.

Подальший розвиток мовних моделей пов’язаний із впровадженням нейромереж на основі трансформерів, таких як BERT, RoBERTa, GPT та їх похідні, які дозволяють формувати семантичні векторні подання текстів різної довжини та структури. На відміну від класичних підходів, ці моделі здатні ефективно враховувати контекст, полісемію слів та синтаксичні залежності, що є критично важливим для задач зіставлення товарних назв, які часто містять скорочення, варіативні написи брендів і моделей, а також маркетингові вставки. У результаті застосування контекстуалізованих ембеддінгів значно підвищується точність обчислення семантичної подібності між текстовими описами товарів.

Паралельно з розвитком обробки текстової інформації відбувся суттєвий прогрес у галузі комп’ютерного зору. Класичні методи вилучення ознак із зображень, такі як SIFT, SURF, ORB та HOG, тривалий час залишалися основою для задач порівняння візуальних об’єктів, однак вони

мають обмежену стійкість до змін ракурсу, освітлення та фону. Сучасні підходи ґрунтуються на використанні згорткових нейронних мереж, зокрема архітектур ResNet, VGG, CLIP, які автоматично навчаються виокремлювати ієрархічні візуальні ознаки без ручного проєктування дескрипторів. Такі моделі продемонстрували високу ефективність у задачах класифікації, детекції та пошуку схожих зображень.

Схема класичного підходу TF-IDF зображена на рисунку 1.1.

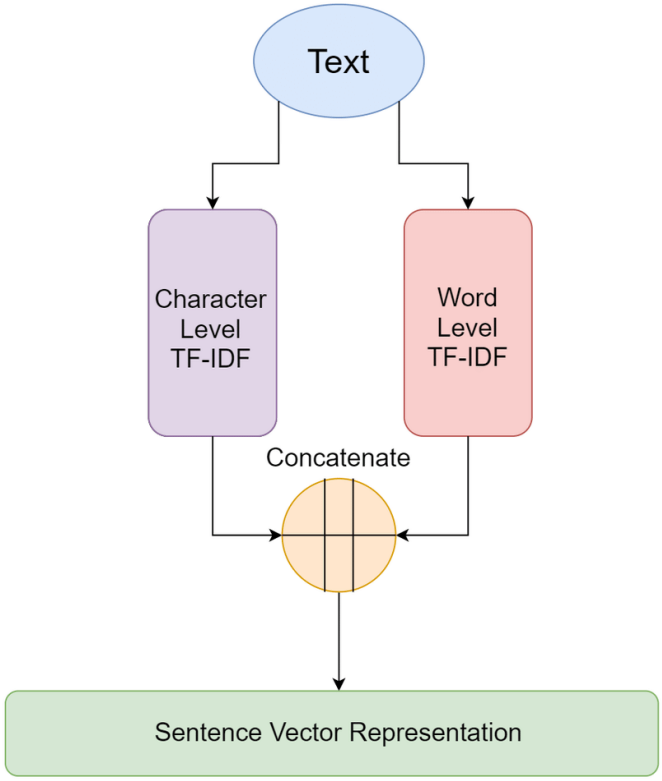

Рисунок 1.1 – Схема класичного підходу TF-IDF

У галузі обробки зображень ключовим проривом стало поширення згорткових нейронних мереж, що дали змогу навчати системи автоматично витягувати релевантні ознаки без ручного програмування дескрипторів. Подальший розвиток призвів до появи архітектур на основі трансформерів, таких як Vision Transformer [16] (ViT) (рис. 1.2), які забезпечують високу стійкість до змін ракурсу й фону. Такі моделі застосовуються для класифікації та пошуку схожих зображень, що є критично важливим для

завдань на кшталт зіставлення товарів, де потрібно порівнювати фотографії товарів, зроблені в різних умовах.

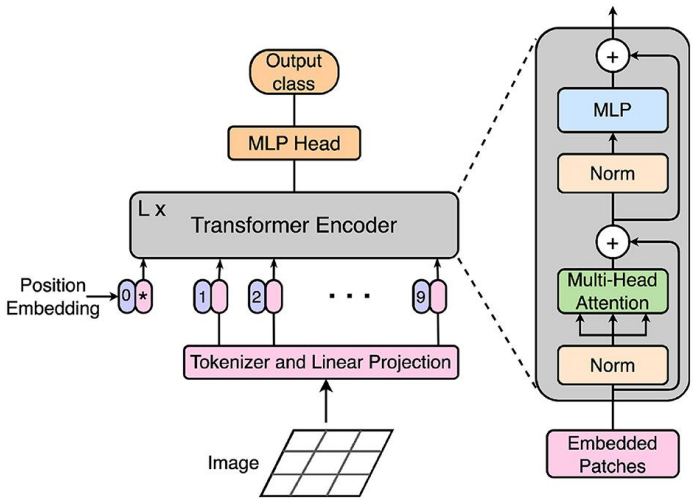

Рисунок 1.2 – Схема архітектури Vision Transformer

Важливим досягненням є також поява мультимодальних моделей, які здатні інтегрувати текстові та візуальні дані у спільному просторі ознак. Моделі на кшталт CLIP (рис. 1.3) навчаються таким чином, щоб описи й зображення, що відповідають одному об’єкту, мали схожі вектори представлення. Це дозволяє виконувати пошук за зображенням або текстом, ранжувати результати та здійснювати зіставлення товарних пропозицій навіть тоді, коли доступна лише одна модальність [17].

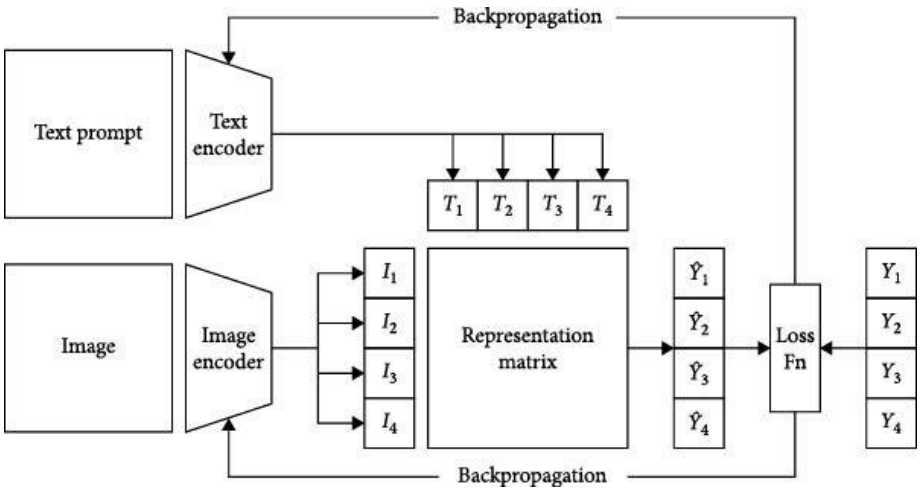

Рисунок 1.3 – Архітектура моделі CLIP

Архітектура мультимодального пайплайна зображена на рисунку 1.4.

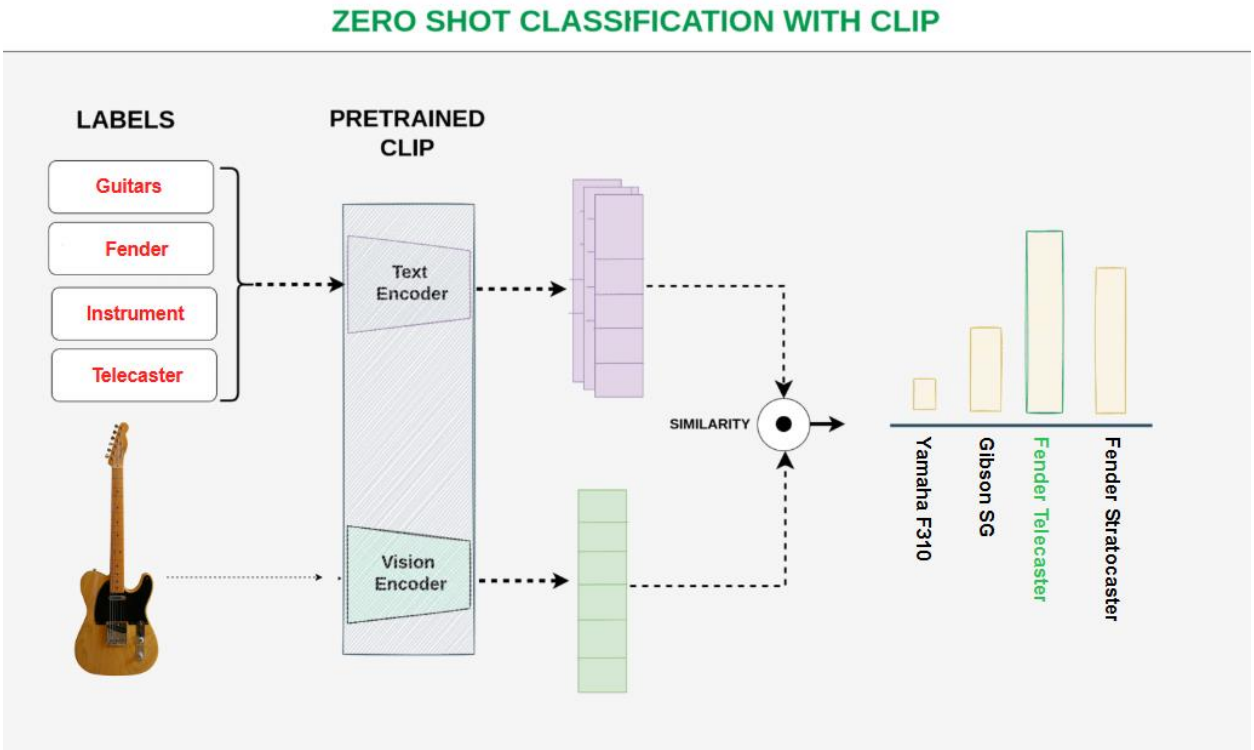

Рисунок 1.4 – Ілюстрація архітектури мультимодального пайплайна

Крім моделей, значну роль відіграють підходи до підготовки даних і оцінювання результатів. Стандартизовані протоколи передбачають формування збалансованих вибірок із прикладами «матч» і «не-матч», використання негативних прикладів і обчислення метрик точності, повноти та міри F1 для текстових і візуальних каналів. Це дає змогу коректно оцінювати продуктивність систем зіставлення й порівнювати різні алгоритми в єдиних умовах.

Таким чином, сучасні досягнення в обробці тексту та зображень створюють основу для побудови систем, здатних виконувати зіставлення товарних пропозицій із високою точністю. Інтеграція контекстуалізованих текстових моделей, глибинних візуальних енкодерів і мультимодальних архітектур відкриває шлях до створення комплексних рішень, які ефективно працюють у сценаріях із різнорідними даними й забезпечують користувачів релевантними результатами пошуку [18].

1.5 Постановка задачі дослідження

Таким чином, проблема автоматизованого зіставлення товарних пропозицій у сервісах-агрегаторах є одним із ключових аспектів сучасної електронної комерції, оскільки саме коректність ідентифікації однакових товарів визначає якість пошуку, точність формування каталогів та рівень користувацького досвіду. Різні онлайн-магазини подають інформацію у неоднорідних форматах, застосовують відмінні правила найменування, по-різному структурують технічні характеристики й використовують несумісні стилі фотографій, що ускладнює побудову єдиної, узгодженої моделі даних. Поєднання методів текстового аналізу, комп'ютерного зору та статистичного порівняння ознак є необхідною умовою для створення надійної системи визначення відповідностей між товарними лістингами.

Прийнято рішення щодо розроблення комплексного підходу до мультимодального зіставлення товарів, який поєднує обробку текстових описів, нормалізацію структурованих характеристик і аналіз візуальних зображень. Така система має виявляти відповідні пропозиції навіть за умов значного шуму в даних: спотворених назв, невпорядкованих описів, різних ракурсів фотографій та варіацій освітлення. Крім того, алгоритм повинен забезпечувати масштабованість для обробки великих каталогів у реальному часі, що є надзвичайно важливою вимогою для інтеграції в комерційні агрегатори [19].

Об'єктом дослідження є процес автоматизованого аналізу та порівняння товарних пропозицій онлайн-магазинів.

Предметом дослідження є методи текстового та візуального зіставлення даних, що забезпечують визначення ступеня подібності між товарними лістингами в умовах мультимодальної та неоднорідної інформації.

Метою дослідження є створення ефективного методу мультимодального зіставлення товарних пропозицій, який забезпечує

формування єдиних груп ідентичних товарів, підвищує точність класифікації пар лістингів і оптимізує роботу агрегаторів електронної комерції.

Для досягнення поставленої мети необхідно вирішити такі завдання:

– провести аналіз сучасних наукових підходів до текстового та візуального зіставлення даних у задачах зіставлення товарів;

– дослідити вплив неоднорідності форматування, шуму в текстах і різниці у фотознімках на точність алгоритмів класифікації пар товарів;

– розглянути можливості застосування векторних моделей представлення тексту та ознак зображень для формування єдиного простору мультимодальних даних;

– визначити функцію подібності, що забезпечує коректне оцінювання відповідності товарних пропозицій та максимізацію показників точності;

– розробити методику поєднання текстових і візуальних дескрипторів з урахуванням вимог масштабованості та обробки потокових оновлень каталогів;

– спроєктувати архітектуру системи зіставлення товарів, здатної працювати з великими онлайн-каталогами у режимі близькому до реального часу;

– провести експериментальне тестування запропонованого підходу на вибірці даних онлайн-магазинів, оцінити точність класифікації позитивних і негативних пар, а також стійкість алгоритму до варіацій текстових і візуальних шумів;

– сформувати висновки щодо ефективності розробленого методу та можливостей його інтеграції у практичні системи агрегування товарних пропозицій.

2 ДОСЛІДЖЕННЯ МЕТОДУ ЗІСТАВЛЕННЯ ТОВАРІВ ІЗ РІЗНИХ МУЗИЧНИХ МАГАЗИНІВ

2.1 Представлення товару в музичному онлайн магазині

У музичних онлайн-магазинах структура подання товарів має вирішальне значення для ефективної організації каталогу, навігації користувача та подальшої автоматизованої обробки даних. Кожен товарний запис у такому магазині, як правило, містить текстову, структуровану та візуальну інформацію, що у сукупності формує його унікальне цифрове представлення. До базових елементів належать назва товару, артикул або модель, опис, технічні характеристики, атрибути бренду та категорії, ціна, а також набір зображень, що демонструють зовнішній вигляд продукту (рис. 2.1).

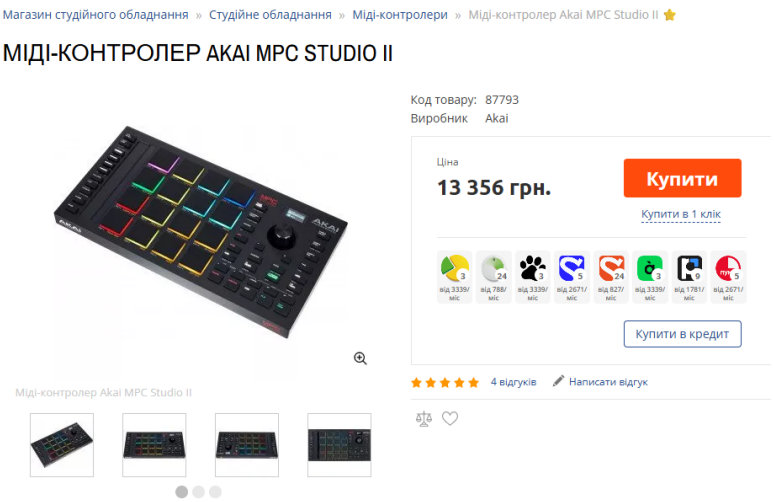

Рисунок 2.1 – Приклад базової інформації про товар

Важливу роль у представленні товару відіграє візуальний контент. Якість, роздільна здатність і композиція зображень значно варіюються залежно від джерела: у деяких випадках використовуються офіційні фотографії від виробника, в інших – власні знімки продавця (рис. 2.2).

Рисунок 2.2 –Приклад різних фото одного товару

У сучасних агрегаторах товарні записи доповнюються метаданими, що містять унікальні ідентифікатори (SKU, UPC, EAN), рейтинги, відгуки користувачів, інформацію про наявність та ціну. Ці поля відіграють роль ключових атрибутів під час побудови систем зіставлення товарів, однак не завжди присутні на сайтах. Багато українських і міжнародних магазинів музичних інструментів не надають артикули у відкритому вигляді або використовують внутрішні коди, що не мають прямої відповідності з кодами виробників. Саме тому для ідентифікації товарів доводиться поєднувати кілька джерел інформації – текстову, табличну та візуальну – створюючи мультимодальні моделі, здатні враховувати всю доступну інформацію.

2.2 Підходи щодо вирішення задачі зіставлення товарів

Задача зіставлення товарів належить до класу проблем ідентифікації сутностей, де метою є встановлення відповідності між записами, що описують один і той самий об’єкт, але походять із різних джерел. У контексті електронної комерції це означає визначення того, чи представляють два товарні лістинги один і той самий продукт, незважаючи на відмінності в текстових описах, назвах, атрибутах або зображеннях. Архітектура системи предсавлена на рисунку 2.3 [20].

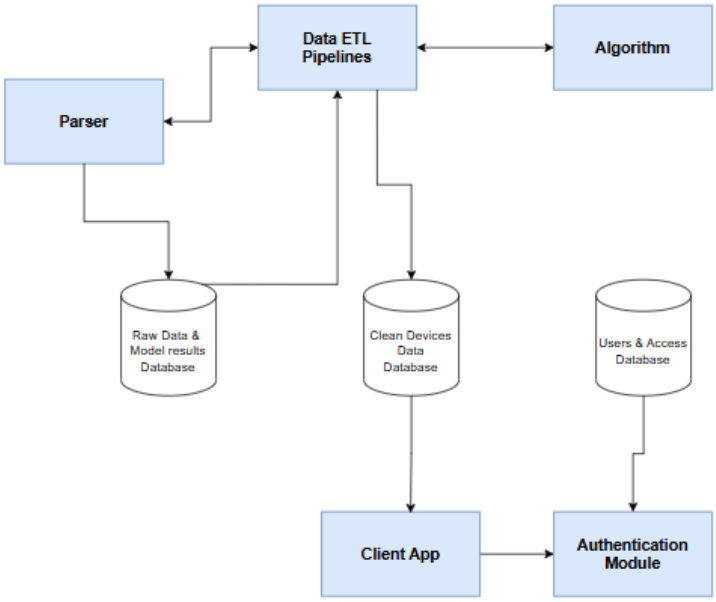

Рисунок 2.3 – Загальна архітектура системи

Підходи до вирішення цієї задачі умовно можна розділити на три великі групи: правила та евристики, методи машинного навчання та мультимодальні підходи, що поєднують кілька типів даних. Найпростішими є методи, засновані на правилах, які використовують регулярні вирази, символьні метрики схожості або атрибути, наприклад артикул, серійний номер чи штрих-код.

У текстовій модальності застосовуються методи на основі TF-IDF, Word2Vec, FastText або Sentence-BERT, які дозволяють вимірювати семантичну схожість між назвами й описами товарів. Для зображень використовуються згорткові нейронні мережі (ResNet, EfficientNet) або трансформери (ViT, Swin Transformer), що навчаються на великих наборах зображень і здатні розпізнавати товари незалежно від ракурсу й освітлення.

Третій напрям – мультимодальні підходи (рис. 2.4), що поєднують текстову та візуальну інформацію в єдиному просторі ознак. Такі системи дозволяють підвищити точність зіставлення, особливо коли один із каналів містить неповні або неоднозначні дані. Сучасні архітектури, такі як CLIP, BLIP-2 або Florence-2, навчаються одночасно на текстових описах і відповідних зображеннях, створюючи спільні векторні представлення, у яких

схожі товари наближаються один до одного. У таких моделях використовуються два основні типи інтеграції раннє злиття (early fusion), коли ознаки з обох модальностей об'єднуються до класифікації, та пізнє злиття (late fusion), де результати зіставлення за кожною модальністю комбінуються з певними ваговими коефіцієнтами.

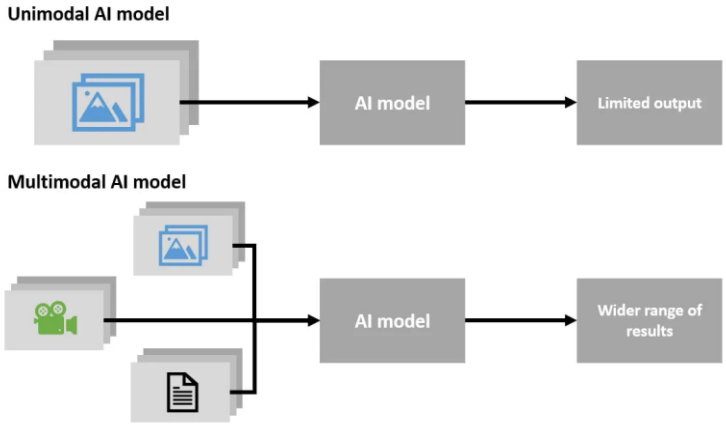

Рисунок 2.4 – Архітектура мультимодального підходу

Отже, ефективне вирішення задачі зіставлення товарів потребує поєднання кількох підходів: евристичних правил для попередньої нормалізації, машинного навчання для семантичного зіставлення та мультимодальних методів для узгодження текстових і візуальних даних [21]. Саме така комбінація дозволяє створити гнучку та стійку систему, придатну для масштабного порівняння товарних пропозицій у різних інтернет-магазинах музичних інструментів, де відмінності у форматі подання, стилі описів і фотографіях є нормою.

2.2.1 Підходи комп'ютерного зору

У візуальній модальності класичні підходи комп'ютерного зору базуються на використанні дескрипторів зображень, які витягують ключові точки та перетворюють їх у числові вектори. До найвідоміших належать SIFT

(Scale-Invariant Feature Transform), SURF (Speeded-Up Robust Features), ORB (Oriented FAST and Rotated BRIEF), гістограми напрямків градієнтів (HOG) та колірні гістограми.

Приклад обчислення схожості двох зображень за дескрипторами SIFT та HOG зображені на рисунку 2.5.

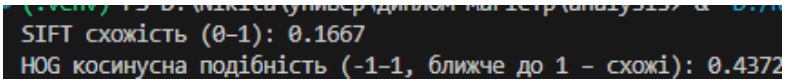

Рисунок 2.5 – Результат порівняння зображень однієї гітари за дескрипторами SIFT і HOG

2.2.2 Машинне навчання

Використання методів машинного навчання стало ключовим етапом розвитку систем зіставлення товарів, оскільки вони дозволяють автоматично виявляти закономірності у великих наборах даних і враховувати складні зв’язки між різними ознаками товарів. На відміну від евристичних підходів, що базуються на фіксованих правилах, машинне навчання дає змогу моделі навчатися на прикладах пар «однакових» та «різних» товарів, узагальнюючи цю інформацію для нових випадків.

У текстовій модальності базові підходи ґрунтуються на статистичних методах, таких як TF-IDF (Term Frequency – Inverse Document Frequency), який перетворює текст у числові вектори, що відображають важливість кожного слова у межах документа. Подібність між назвами або описами вимірюється за допомогою косинусної схожості, яка визначає кут між векторами текстів. Цей метод є простим у реалізації та добре працює для коротких текстів, але не враховує семантичного контексту слів, тому його ефективність знижується, коли різні слова описують одне й те саме поняття [22].

Подальшим кроком розвитку стали моделі Word2Vec та FastText, які навчаються на великих корпусах текстів і відображають слова у багатовимірному векторному просторі. Модель FastText, на відміну від транскриптора Word2Vec, враховує підслова (n-грамні структури), тому краще справляється з морфологічними варіаціями та орфографічними відмінностями.

Сучасним стандартом у текстовій обробці є Sentence-BERT (SBERT) (рис. 2.6) – модифікація моделі BERT, спеціально навчена для обчислення семантичної подібності між реченнями. Модель SBERT формує контекстуалізовані векторні представлення текстів, що дозволяє зіставляти назви й описи товарів із урахуванням значення, а не лише лексичної схожості.

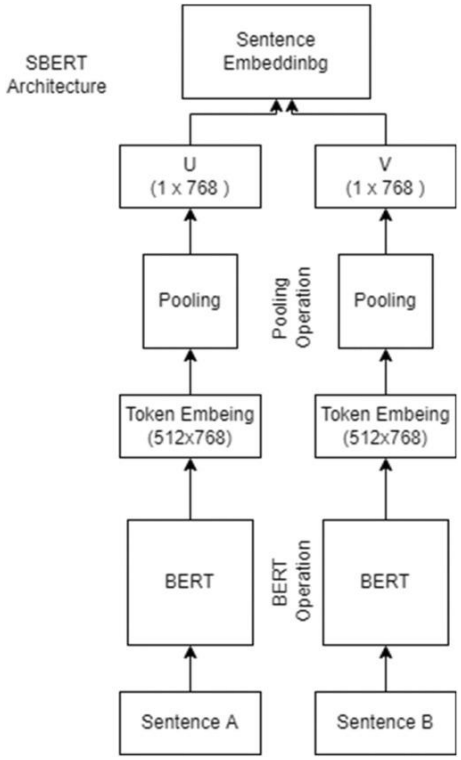

Рисунок 2.6 – Архітектура моделі SBERT

У візуальній модальності машинне навчання використовується для порівняння зображень товарів, що мають схожі форми, кольори або елементи дизайну. Найпоширенішими архітектурами є згорткові нейронні мережі

(CNN) – зокрема, ResNet, EfficientNet та MobileNet, які навчаються на великих наборах зображень і здатні витягати інваріантні до ракурсу й освітлення ознаки. Далі у розвитку комп'ютерного зору з'явилися Vision Transformer (ViT) та Swin Transformer, які використовують механізм самоуваги (self-attention).

2.2.3 Мультимодальні підходи

Мультимодальні підходи становлять один із найперспективніших напрямів розвитку систем зіставлення товарів, оскільки вони дають змогу інтегрувати різні типи інформації – текстову, візуальну, а іноді й структуровані атрибути – у спільному просторі ознак. Такі моделі враховують не лише семантичний зміст опису товару, але й його візуальні характеристики, що дозволяє досягати високої точності навіть у складних випадках, коли текстові або графічні дані поодинці не дають достатньої інформації. Для задачі зіставлення музичних інструментів це особливо важливо, оскільки фотографії, наприклад, гітар чи клавішних, можуть мати подібну форму, але належати до різних моделей, тоді як текстовий опис містить уточнюючі деталі, які компенсують неоднозначність зображень. На рисунку 2.7 зображена високорівнева схема мульимодальної архітектури.

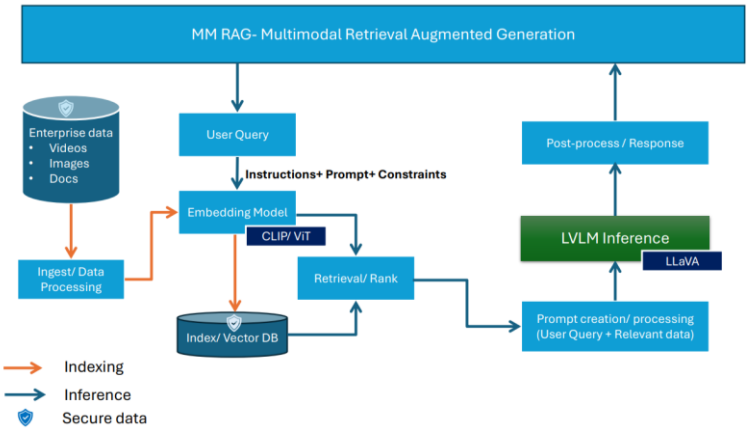

Рисунок 2.7 – Приклад високорівневої мультимодальної архітектури

Основна ідея мультимодального підходу полягає у тому, що кожна модальність (текст і зображення) спочатку обробляється окремим енкодером, який перетворює дані у векторні представлення. Далі ці вектори поєднуються у спільному просторовому представленні за допомогою механізмів злиття (fusion). Розрізняють два основні типи злиття – раннє (early fusion) та пізнє (late fusion) (рис. 2.8).

Провідним прикладом мультимодальних моделей є CLIP (Contrastive Language–Image Pre-training), створена компанією OpenAI. Вона навчається на великих наборах пар «зображення – текстовий опис» та формує спільний векторний простір, у якому відповідні об’єкти мають близькі представлення. Завдяки контрастивному навчанню CLIP здатна розпізнавати та зіставляти об’єкти навіть без додаткового донавчання на конкретних доменах [23].

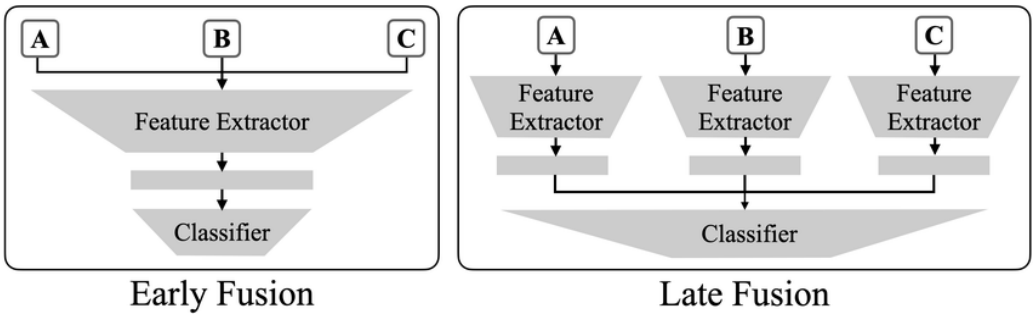

Рисунок 2.8 – Схема порівняння early та late fusion

Перевагою мультимодальних моделей є їхня здатність компенсувати відсутність або низьку якість даних в одній із модальностей. Якщо, наприклад, у товару відсутній опис або текст є надто коротким, система може використати візуальні ознаки для визначення відповідності. Приклад роботи моделі CLIP на рисунку 2.9.

```
Comparing 2 images of the same guitar with CLIP.
image 1: pictures/same/cort-ad810-open-pore-guitarhousejpg.jpg
image 2: pictures/same/cort-ad810-open-pore-muzline.jpg
0.932811975479126
```

Рисунок 2.9 – Приклад зіставлення двох зображень і текстів у CLIP-просторі

Застосування таких моделей у сфері музичних інструментів відкриває можливість точного зіставлення навіть тих пропозицій, які візуально відрізняються, але належать до одного продукту. Наприклад, модель CLIP може коректно зіставити дві фотографії однієї електрогітари, навіть якщо одна з них зроблена на білому фоні, а інша – у студійному інтер'єрі. У поєднанні з текстовим описом, який уточнює модель, бренд і тип звукознімачів, така система забезпечує високу точність розпізнавання.

2.2.4 Спеціальні підходи для системи зіставлення товарів

У багатоплатформеному середовищі електронної комерції завдання зіставлення товарів ускладнюється тим, що одна й та сама продукція може бути представлена на різних сайтах із різною структурою даних, термінологією та стандартами каталогізації. Наприклад, на одному майданчику зазначається код виробника, а на іншому – лише внутрішній артикул; на одному використовується метрична система вимірювання, на іншому – дюймова; або навіть назви брендів можуть бути записані різними мовами. Такі відмінності створюють необхідність у спеціальних методах системи зіставлення товарів, спрямованих на уніфікацію даних і побудову спільного простору атрибутів, у якому товари з різних джерел можуть бути коректно порівняні.

Одним із ключових рішень у цьому напрямі – звести товари з різних джерел до уніфікованої форми опису. У межах такого підходу формується набір базових атрибутів (бренд, модель, колір, матеріал, розміри, тип інструмента, рік випуску тощо), що використовуються як канонічна структура для подальшого зіставлення. Кожен новий товар із певного джерела проходить через процедуру нормалізації: назви очищаються від зайвих символів, параметри переводяться до спільних одиниць виміру, атрибути зіставляються з контрольованими словниками. Наприклад, «Fender

Strat MN 3TSB» та «Fender Stratocaster Maple Neck Sunburst» після нормалізації зводяться до одного канонічного запису. Це дозволяє системі розглядати товари з різних джерел як ідентичні, навіть якщо їхні первинні описи суттєво відрізняються.

Схема архітектури системи зіставлення товарів зображена на рисунку 2.10.

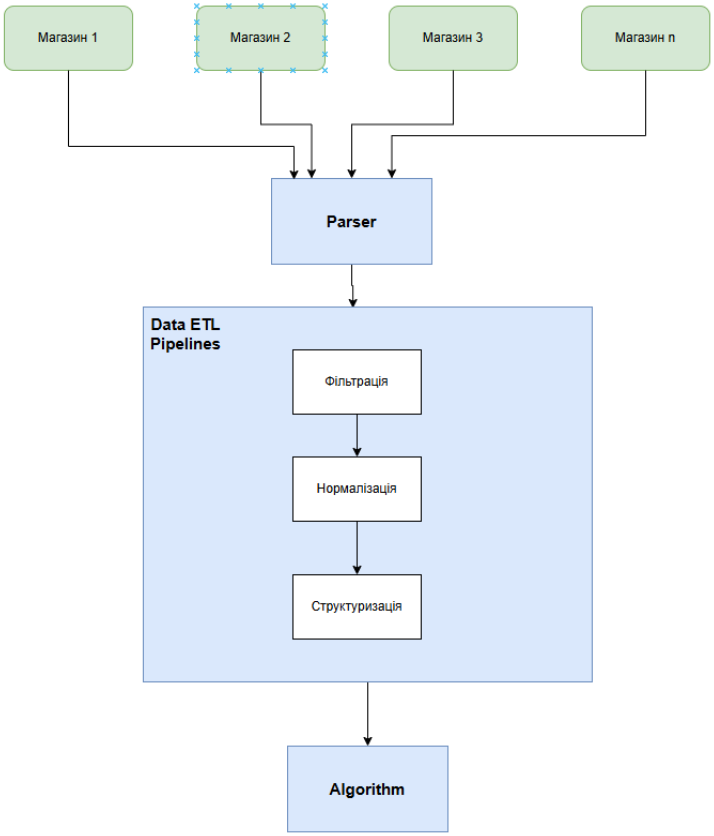

Рисунок 2.10 – Архітектура системи зіставлення товарів

Другим важливим напрямом є так званий метод Entity Resolution (ER) або Record Linkage – процес виявлення записів, що описують одну й ту саму сутність у різних базах даних. Такий підхід дозволяє автоматично визначати, які атрибути мають найбільшу інформативність для зіставлення, та адаптувати модель до нових доменів без ручного налаштування правил.

Для покращення якості моделей у випадках, коли даних для навчання недостатньо або розмітка неповна, застосовуються методи Active Learning (рис. 2.11) та Semi-supervised Matching. У першому випадку система

ідентифікує пари товарів, щодо яких має найменшу впевненість, і передає їх на перевірку експерту. Таким чином, кожна нова ітерація розмітки робить модель точнішою, мінімізуючи кількість необхідних вручну підтверджених прикладів. У другому випадку, при напівконтрольованому навчанні, модель використовує вже наявні «позитивні» пари (наприклад, товари з однаковим UPC-кодом) як основу для самонавчання, поступово розширюючи вибірку за рахунок псевдопозитивних прикладів. Це дає можливість ефективно масштабувати системи навіть за відсутності повного набору розмічених даних.

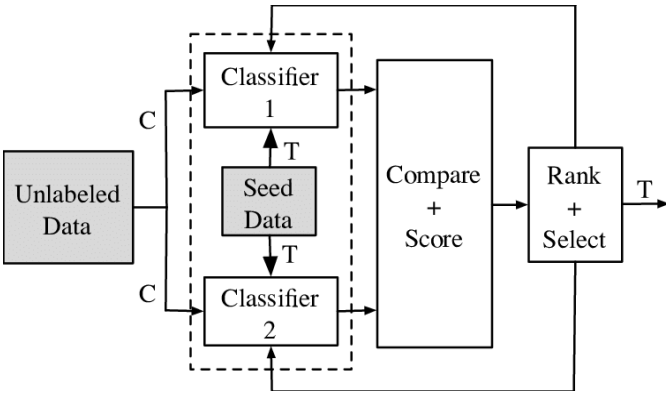

Рисунок 2.11 – Архітектура навчання Active Learning

У межах агрегаторів музичних інструментів такі підходи є особливо актуальними. Різні магазини застосовують власні структури каталогів і системи кодування, тому створення уніфікованого представлення є необхідною умовою для ефективного зіставлення.

2.3 Структуризація інформації про музичний інструмент

Для побудови ефективної системи зіставлення товарних пропозицій необхідно мати повні, структуровані та достовірні дані про товари з різних онлайн-магазинів. Одним із ключових етапів цього процесу є вебскрейпінг (рис. 2.12) – автоматизоване збирання інформації з вебсторінок за допомогою спеціалізованих програм або скриптів.

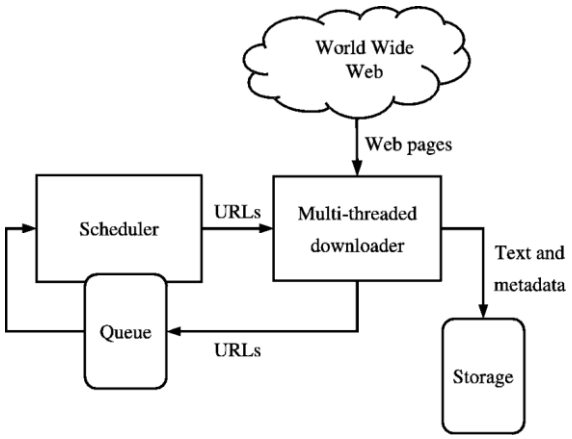

Рисунок 2.12 – Схема процесу вебскрейпінгу

Для ефективної подальшої роботи з даними формується єдина схема бази даних, що включає основні атрибути: id, brand, model, type, price, color, features, image_url, source.

Таким чином, вебскрейпінг і структуризація даних (рис. 2.13) є базовими компонентами процесу створення агрегатора товарів. Вони забезпечують отримання актуальної, узгодженої та придатної для аналізу інформації, на основі якої далі виконуються етапи зіставлення текстових і візуальних ознак.


```
_id: ObjectId('665b99b6ac30c3fe49ae6de6')
▸ type : Object
code : "89837"
▸ provider : Object
base : true
▸ brand : Object
▸ deviceClass : Object
▸ filters : Object
img : "https://img.muzline.ua/image/cache/catalog/product_pictures/2023/02/mu…"
inStock : true
▸ meta : Object
name : "Alfabeto Solid-RT 3TS + bag"
price : 5095
▸ priceHistory : Array (1)
▸ stockHistory : Array (1)
url : "https://muzline.ua/uk/akustichna-gitara-alfabeto-solid-rt-3ts---bag/"
```


Рисунок 2.13 – Приклад структури даних після вебскрейпінгу

2.4 Формування датасету товарів із різних платформ

Після етапу вебскрейпінгу й попередньої структуризації даних формується навчальний датасет, який є центральним елементом для побудови та тестування системи зіставлення товарів. Основна мета цього етапу – об’єднати інформацію з різних онлайн-магазинів музичних інструментів у єдину структуру, яка містить приклади пар товарів з мітками «match» або «non-match». Це дозволяє створити базу даних, придатну як для класичних алгоритмів машинного навчання, так і для глибинних мультимодальних моделей.

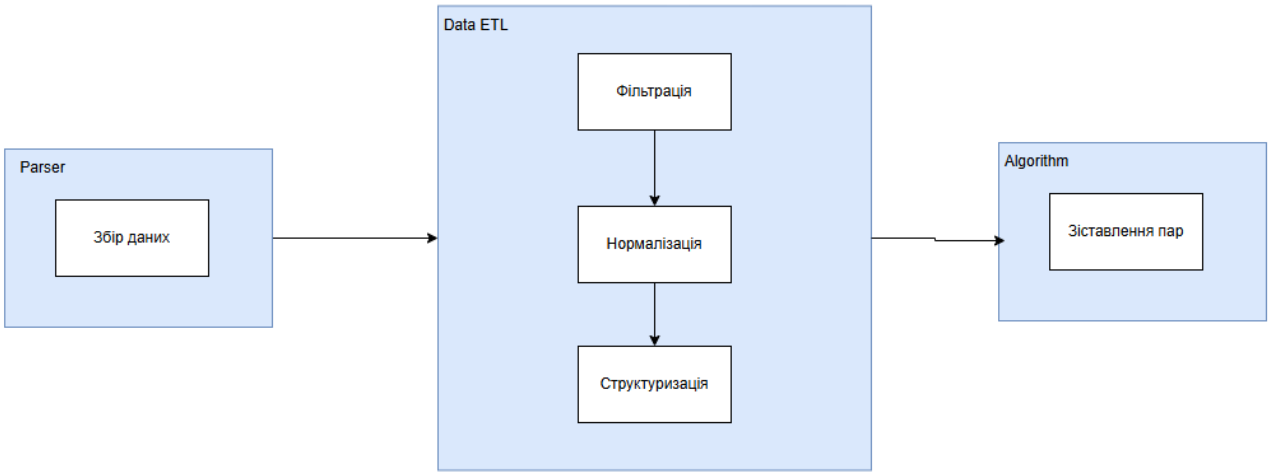

Рисунок 2.14 – Схема процесу формування датасету

Джерелами даних у даній роботі виступають три основні українські онлайн-платформи – Muzline, GuitarHouse і Muztorg, які пропонують схожий асортимент музичних інструментів, проте мають різні структури описів і формати подання товарів.

Важливим етапом є попередня нормалізація текстових і числових полів. Назви товарів очищаються від зайвих слів і спеціальних символів, бренди приводяться до єдиного написання, а ціни конвертуються до спільної валюти – гривні. Колір, матеріал та інші атрибути уніфікуються за

допомогою попередньо визначених словників. Для текстових полів застосовується токенізація та лематизація, що дозволяє зменшити варіативність у написанні. Візуальні дані (зображення) приводяться до стандартної роздільної здатності 224×224 пікселі, зберігаючись у форматі JPEG для подальшого оброблення моделями комп’ютерного зору.

2.5 Зіставлення текстової інформації

Розглянемо декілька текстових моделей машиного навчання та протестуємо їхню ефективність на тестовій вибірці даних, визначимо, яка з моделей найкраще вирішує проблему зіставлення товарних пропозицій музичних інструментів.

2.5.1 Опис проблеми однакових товарів в магазинах

Попри те, що музичні інструменти мають чітко визначені фізичні та технічні характеристики, у середовищі онлайн-торгівлі один і той самий товар дуже рідко подається однаково. Кожен магазин має власний стиль написання назв, довжину опису, формат зазначення характеристик, а інколи навіть замінює офіційну модель на комерційну назву. Саме тому перше зіткнення з даними виглядає як хаотична комбінація текстів, у якому одна і та сама електрогітара може бути представлена трьома чи п’ятьма різними способами – і жоден з них не збігається на 100 %.

Найочевидніша проблема полягає у відсутності будь-яких єдиних правил щодо того, що вважати назвою товару. Більше того, треба враховувати до цього ще різні мови, різні кодування кольорів («3TS», «3TSB», «3-Tone Sunburst»), випадкові скорочення («MN» = Maple Neck) і перефразування, на кшталт гриф з клену та кленовий гриф.

На цих прикладах можна помітити формулювання, що відрізняються, у назві одного й того ж товару серед різних магазинів. Відмінності полягають. У змінах порядку слів, скороченнях певних атрибутів, наявності додаткової інформації в назві, такої як кількість струн, звукові ефекти, форм-фактор, потужність, тощо.

В обох прикладах видно, що відмінності не випадкові – вони структурні. Магазини мають власну політику написання назв, додають або прибирають частину ознак, а іноді ще й перекодовують важливі технічні характеристики. Для системи зіставлення товарів усе це означає одне: жодне з полів, взяте у сирому вигляді, не може виступати надійним показником відповідності.

2.5.2 Методи зіставлення текстової інформації

Зіставлення текстових назв і описів є першим і обов’язковим етапом зіставлення товарів, оскільки саме текст найчастіше містить базові ідентифікаційні ознаки: бренд, серію, модель, конфігурацію, колір, тип інструмента. Однак через те, що різні продавці формують назви у своєму стилі, пряме порівняння ключових слів не працює – необхідні методи, здатні вловити подібність навіть тоді, коли формулювання суттєво відрізняються.

Для зіставлення тексту застосовують дві основні групи підходів: TF-IDF та семантичні векторні моделі (Word2Vec / FastText / SBERT).

Метод TF-IDF передбачає перетворення тексту у вектор, де кожне слово стає ознакою, а його вага визначається тим, наскільки воно унікальне саме для цього рядка, а не для всього корпусу.

Семантичні моделі (Word2Vec / FastText / SBERT) - це сучасний підхід, який не просто аналізує слова, а будує векторний змістовий образ кожного тексту.

Через те, що задача знаходження двох однакових товарних сутностей відрізняється від завдання пошуку схожих слів за контекстуальним значенням тим, що назви брендів та характеристик часто не залежать від синонімічної подібності, найточнішим методом у вирішенні проблеми зіставлення товарів буде використовувати комбінований підхід зіставлення тексту. Високорівневий приклад алгоритму проілюстровано на рисунку 2.15.

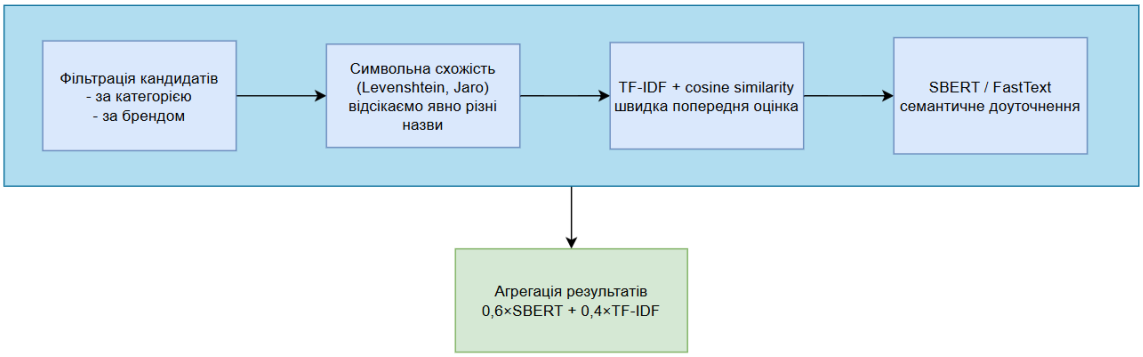

Рисунок 2.15 – Комбінований конвеєр зіставлення тексту

Такий підхід дозволяє знизити кількість помилкових зіставлень навіть у випадках, коли назви відрізняються на 60–70 % візуально, але описують той самий товар та точково підлаштувати систему під предметну область.

Важливий показник якості методу зіставлення товарних пропозицій – це розуміння, наскільки ефективно цей метод розрізняє неоднакові товари. Відберемо 5 випадкових назв різних товарів та порівняємо їх за допомоги.

Метод TF-IDF доволі гарно розрізняє як вочевидь різні товари, так і музичне обладнання одного бренду, виробника чи однієї лінійки. Однак, слід врахувати те, що метод показав себе найменш ефективно серед впізнання однакових товарів, тому поки що важко визначити його фаворитним. Перейдемо до огляду та тесування наступного методу текстового аналізу.

Модель Word2Vec навчається на великому корпусі текстів, вивчаючи, які слова часто зустрічаються поруч. Наприклад, у домені музичних інструментів слова «гитара», «електрогітара», «бас» або «струни» матимуть близькі вектори, оскільки часто з’являються в подібних контекстах.

Під час зіставлення назв модель обчислює середній вектор для кожної назви (усереднюючи вектори окремих слів), після чого використовується косинусна міра схожості між цими векторами.

В цілому, модель Word2Vec показує себе трохи краще у порівнянні з методом TF-IDF, однак при таких же самих умовах теж поводиться себе невпевнено оперуючи з назвами, в яких розбігається формулювання, регіст, скорочення або абревіатури

Також, модель Word2Vec, показує вищу семантичну стійкість, ніж метод TF-IDF, але все одно не може повністю подолати вплив скорочень і технічних кодів.

Модель SBERT є модифікацією архітектури BERT (Bidirectional Encoder Representations from Transformers), оптимізованою для порівняння текстів. Модель перетворює речення на щільні вектори (sentence embeddings) у багатовимірному просторі, де семантично подібні тексти мають близькі координати.

Модель SBERT чудово розпізнає семантичну близькість навіть при різних скороченнях, великих літерах або наявності спецсимволів. Важливе покращення це те, що модель розуміє ключові слова, такі як назва бренду, тип товару, модель як вагомі та не знижує радикально остаточну оцінку при відсутності менш значущих слів.

Загалом, модель SBERT доволі ефективно розпізнає різні номенклатури брендів та навіть деяких моделей, проте порівняння різних товарів одного бренду чи однієї лінійки може дати помилково високий показник схожості.

2.5.3 Висновки аналізу текстової інформації

Після проведення серії експериментів з трьома методами – TF-IDF, Word2Vec, та Sentence-BERT (SBERT) – можна зробити висновок, що ступінь

точності значно залежить від рівня семантичного розуміння, який закладено в обрану модель.

Для систем зіставлення товарів, де завдання полягає у відокремленні справжніх збігів від помилкових, найбільш показовою є метрика ROC-AUC (Area Under the Receiver Operating Characteristic Curve).

Вона відображає (рис. 2.16), наскільки добре модель відрізняє пари «однакових товарів» від «різних», незалежно від порогу прийняття рішення, де AUC = 1,0 – ідеальне розділення, AUC = 0,5 – випадкове вгадування (ефект монети), AUC < 0,5 – модель плутає класи (таблиця 2.1).

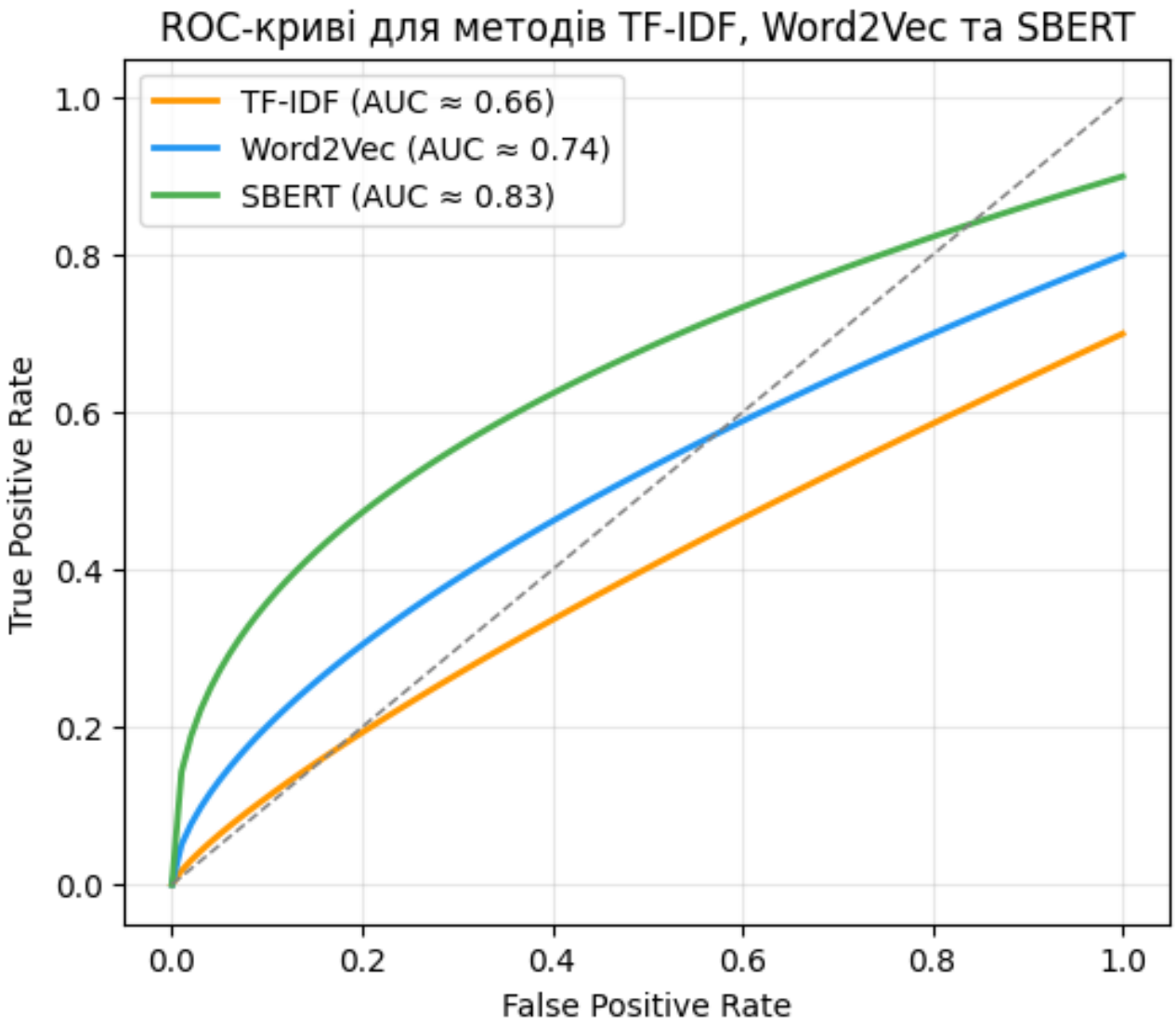

Рисунок 2.16 – Графік результату метрики ROC-AU

Таблиця 2.1 – Порівняння методів за метрикою ROC-AUC

Метод	ROC-AUC	Характеристика
TF-IDF + Cosine Similarity	0,66	Добре працює для назв, з точним збігом; втрачає ефективність при скороченнях
Word2Vec	0,74	Добре впізнає семантичну схожість, але погано бачить назви одного бренду.
SBERT	0,83	Кращий результат, але модель має проблеми з товарами одного типу та бренду.

Метод TF-IDF показав себе як швидкий та базовий підхід, який ефективно працює в умовах добре структурованих текстів.

Модель Word2Vec, яка оперує векторними представленнями слів, демонструє вищу гнучкість. Завдяки тому, що подібні слова мають близькі вектори, модель краще справляється зі скороченнями та частковими синонімами.

Модель Sentence-BERT (SBERT) продемонструвала найвищу точність, оскільки враховує не лише окремі слова, а й їхній контекст у межах фрази.

2.6 Зіставлення зображень товарів із різних магазинів

Завдання зіставлення зображень є другою ключовою складовою системи зіставлення товарів, оскільки візуальна інформація нерідко точніше передає ідентичність товару, ніж текстова.

Складнощі цієї задачі передбачають загальну схожість інструментів або приладів одного типу через близьку семантичну відстань. Отже, ключовою проблему, що потрібно вирішити, є пошук детальних відмінностей.

2.6.1 Опис проблеми однакових товарів

Різні торговельні платформи використовують різні стандарти подачі фотографій. Один і той самий інструмент може бути знятий під різним кутом, у різному освітленні або на різному фоні.

Наприклад, гітара Yamaha F-310 в одному магазині (рис. 2.17) може бути зображена на білому фоні, повернута ліворуч та з підвищеним контрастом, а у іншому та ж сама модель виглядатиме більш сіро та під прямим кутом (рис. 2.18).

Рисунок 2.17 – Приклад зображення гітари Yamaha F-310 у магазині Muzline

Рисунок 2.18 – Приклад зображення гітари Yamaha F-310 у магазині GuiarHouse

Такі детальні відмінності можуть створити виклик, оскільки два зображення одного товару мають різну піксельну структуру, але однакову семантику.

2.6.2 Методи зіставлення зображень (методи, метрики)

Загалом можна виділити три основні покоління методів зіставлення зображень: згорткові нейронні мережі (CNN), Vision Transformer, CLIP [26].

Поява згорткових нейронних мереж (CNN) (рис. 2.19) дозволила перейти від ручного опису ознак до автоматичного вилучення векторних представлень зображень [27].

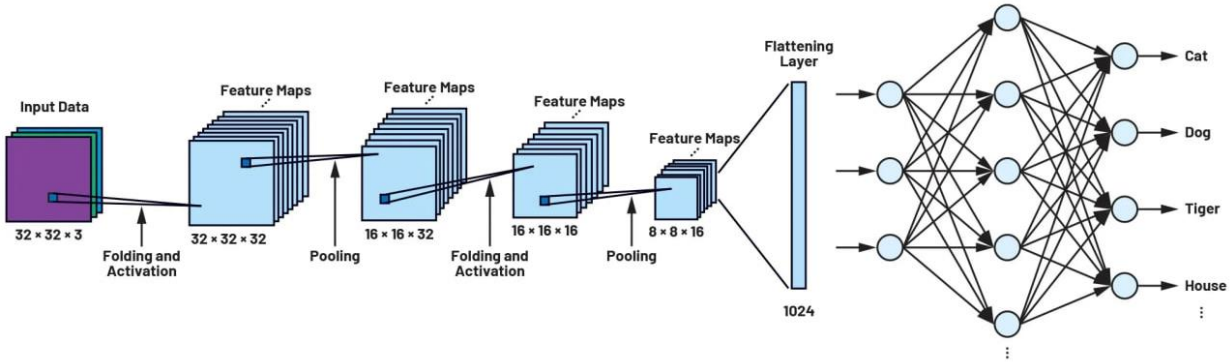

Рисунок 2.19 – Архітектура роботи згорткової нейронної мережі

Перетворювач Vision Transformer (ViT) (рис. 2.20) та його покращена версія Swin Transformer здатні ефективно вилучати семантичні ознаки зображень.

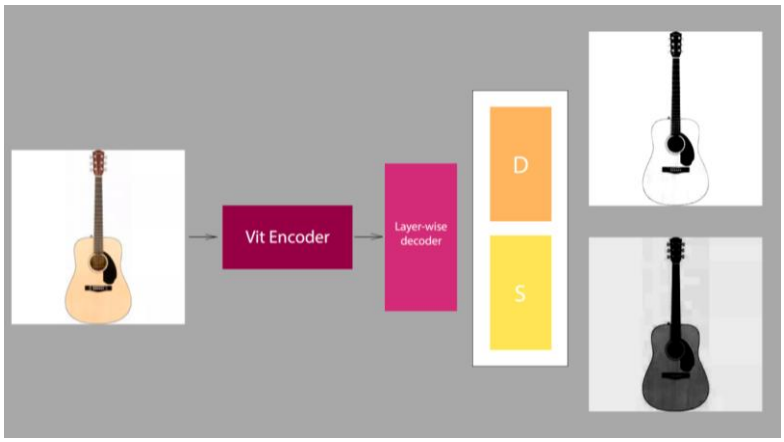

Рисунок 2.20 – Розбиття зобаження на патчі за архітектурою Vision Transformer

Перетворювач Swin Transformer доповнює підхід ViT ієрархічною структурою та ковзними вікнами (shifted windows), що підвищує здатність моделі розпізнавати деталі на зображеннях із високою роздільністю, не збільшуючи обчислювальні витрати [28]..

Найновіші архітектури, такі як CLIP (рис. 2.21), навчаються у спільному просторі зображень і текстів, враховуючи контекст зображення.

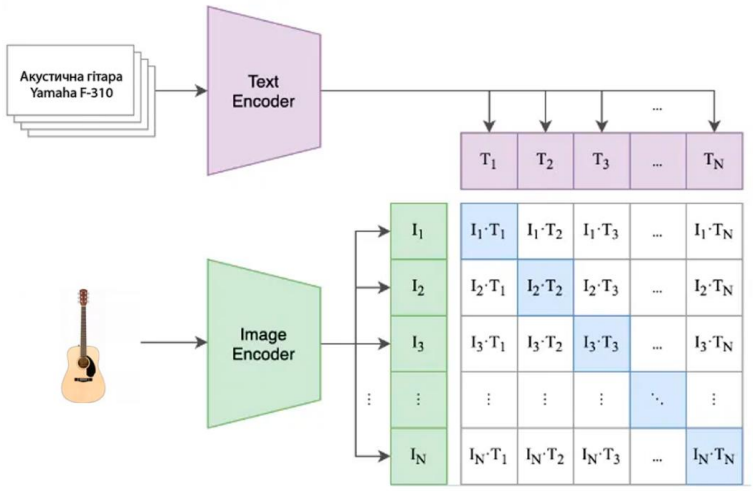

Рисунок 2.21 – Архітектура CLIP

Метрика ROC-AUC задовільняє перелічені умови, оскільки вона обчислює якість класифікації для всіх можливих порогів одночасно та відображає здатність моделі послідовно ставити вищу оцінку позитивним зразкам, ніж негативним.

Архітектура згорткових нейронних мереж є базовим підходом у задачах комп’ютерного зору, де метою є навчити модель виявляти візуальні закономірності в зображеннях. Однією з найуспішніших архітектур згорткових нейронних мереж у цій сфері є ResNet-50, що належить до родини «residual networks», або залишкових мереж [29].

Модель ResNet-50 складається з 50 шарів і впроваджує так звані skip connections – механізм, що дозволяє передавати інформацію між шарами безпосередньо, оминаючи проміжні перетворення.

На рисунку 2.22 продемонстровано архітектуру моделі ResNet-50.

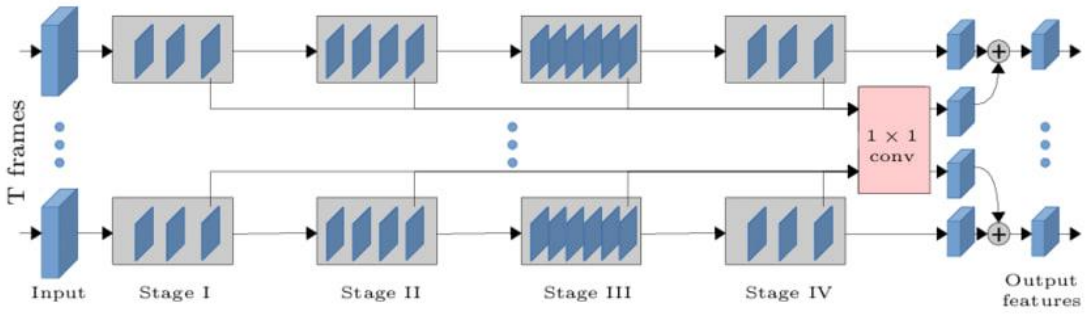

Рисунок 2.22 – Схема архітектури моделі ResNet-50

Загалом, підхід з використанням моделі ResNet-50 демонструє доволі високу спроможність ідентифікувати товари одного типу, проте нестабільно покузає себе при визначенні різниці при більш детальному порівнянні, наприклад товарів однієї категорії.

Перетворювач Vision Transformer (ViT) стали одним із ключових проривів у комп'ютерному зорі, перенісши ідеї трансформерів, раніше застосовуваних у мовних задачах, у сферу аналізу зображень. На відміну від згорткових мереж, що покладаються на локальні фільтри для виявлення ознак, Vision Transformer розглядає зображення як послідовність патчів, кожен із яких перетворюється на вектор і обробляється механізмом self-attention. Це дозволяє моделі глобально враховувати взаємозв'язки між частинами зображення, що є критично важливим для точного порівняння товарів, сфотографованих під різними кутами, з різними відблисками або в неоднакових масштабах [30].

У задачі зіставлення товарних пропозицій музичних інструментів модель CLIP є природним вибором, оскільки дозволяє інтегрувати як візуальні, так і текстові ознаки товару. Модель кодує фотографію та текст у вектори однакової розмірності, після чого схожість між ними обчислюється за косинусною мірою. Якщо обидва об'єкти – фото з різних магазинів – мають близькі вектори, це означає, що вони описують той самий товар або надзвичайно схожі моделі.

2.6.3 Висновки аналізу зіставлення зображень

Для кількісної оцінки ефективності методів використано метрику ROC-AUC – стандартну метрику для задач бінарної класифікації, що дозволяє оцінити, наскільки модель правильно відокремлює класи «однаковий товар» і «різні товари» незалежно від обраного порогу подібності.

Результати тестування наведені у таблиці 2.2.

Таблиця 2.2 – Оцінка ефективності візуальних моделей

Метод	ROC-AUC	Характеристика
ResNet-50	0,78	Знаходить схожість за загальним патерном
Vision Transformer	0,82	Покращене розділення, контекст зображення
CLIP	0,83	Кращі показники, проте плутає відмінності

На графіку (рис. 2.23) показаний результат тестування трьох моделей.

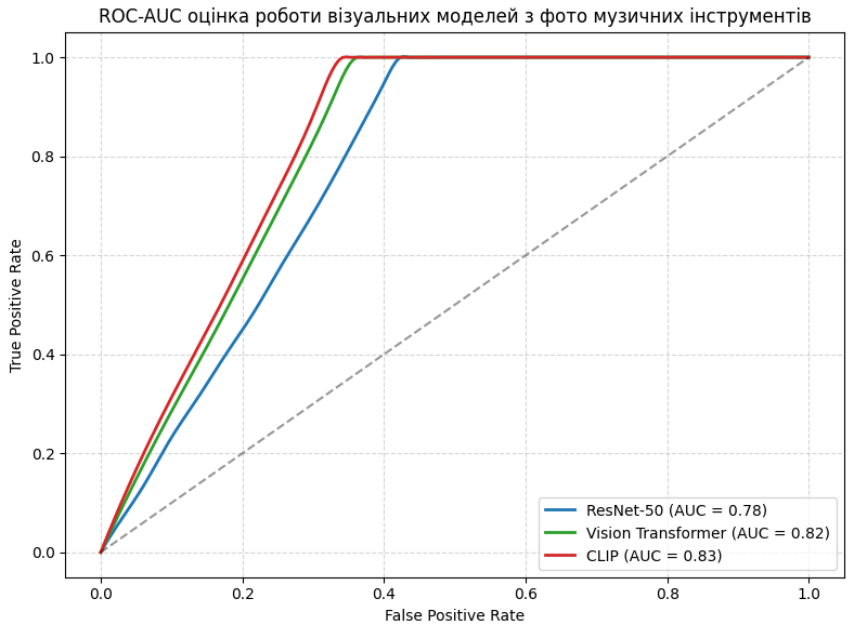

Рисунок 2.23 – Аналіз роботи візуальних моделей на вибірці даних за метрикою ROC-AUC

Незважаючи на загальну тенденцію, можна виділити характерні риси кожної архітектури. ResNet-50 часто помилково класифікував різні акустичні гітари як однакові, оскільки базується на локальних патернах текстур і не розпізнає тонких контекстуальних відмінностей (форма голови грифа, наявність вирізу тощо). Модель Vision Transformer завдяки механізму уваги краще сприймав глобальну структуру зображення, що підвищило показник AUC на кілька відсотків, але він теж схильний до плутанини при надмірній схожості корпусів гітар.

Модель CLIP, хоч і зберігла лідерство, проте також не впоралася із задачею по розпізнаванню схожих, але різних інструментів.

2.7 Розробка фінального алгорітму для зіставлення товарів

Фінальний алгоритм зіставлення товарних пропозицій музичних інструментів має усунути обмеження окремих візуальних моделей і текстових методів та забезпечити стійке розрізнення двох класів пар: «однаковий товар» і «різні товари». На попередніх етапах дослідження було встановлено, що жоден окремий метод, ані текстовий, ані візуальний, не забезпечує достатньої точності зіставлення товарних пропозицій музичних інструментів у випадках, коли товари мають високу внутрішньокласову схожість або неповні текстові описи [31-32].

Показник евристичної текстової схожості $S_{heuristic}$ базується на порівнянні нормалізованих назв і моделі інструмента. Тут використовуються правила нормалізації: видалення стоп-слів, уніфікація брендів, виявлення артикулів і номерів серій.

Показник семантичної схожості S_{sbert} обчислюється на основі моделі Sentence-BERT, який перетворює назви й описи у векторний простір, де подібні за змістом фрази розташовані близько одна до одної. На відміну від

евристики, цей метод дозволяє правильно зіставляти навіть ті назви, які відрізняються формально, але містять однакову семантичну інформацію.

Показник візуальної схожості S_{image}, отриманий за допомоги CLIP, дозволяє оцінити відповідність зображень незалежно від ракурсу, освітлення чи невеликих відмінностей у кольорі. CLIP навчається на парах «зображення – текст», тому модель може розпізнавати композиційні елементи інструмента й зберігати стабільність навіть у випадках, коли фото зроблені в різних умовах.

Унаслідок обчислення цих трьох показників формується інтегральна оцінка S_{final}, яка є зваженою сумою текстових і візуальних характеристик. Розрахунок остаточної оцінки схожості товарів представлений у формулі 2.1

$$S_{final} = w_{heuristic} \cdot S_{heuristic} + w_{image} \cdot S_{image} + w_{sbert} \cdot S_{sbert}, \quad (2.1)$$

де $w_{heuristic}$, w_{image}, w_{sbert} – коефіцієнт вагомості.

Реалізація алгоритму (лістинг 2.10) здійснюється за допомоги окремих модулів для текстового та візуального аналізу. Евристичні показники та результати роботи моделі Sentence-BERT розраховуються на основі нормалізованих назв і описів, тоді як модель CLIP використовується для вилучення ембеддингів зображень [33]. Усе це об’єднується в єдину функцію порівняння товарів, яка повертає як числову оцінку S_{final}, так і булевий результат збігу.

Семантика назви вирішує випадки надмірної подібності зображень, модель CLIP допомагає у ситуаціях неповних або шумних текстів, а евристика коригує помилки, спричинені неправильною сегментацією або неоднозначним формулюванням назв. Завдяки цьому остаточний алгоритм забезпечує високий рівень точності й стійкість до помилкових співставлень, що робить його придатним як основу для розробки системи агрегування товарних пропозицій музичних інструментів у реальних умовах.

3 ВЕБЗАСТОСУНОК ДЛЯ ПЕРЕГЛЯДУ ТОВАРІВ

3.1 Специфікація вимог до застосунку

Усі компоненти застосунку реалізовано з використанням платформи Node.js, у зв’язку з чим для розробки програмного забезпечення та написання вихідного коду необхідне середовище веброзробки, зокрема JetBrains WebStorm. Для створення вебзастосунку й керування залежностями на персональному комп’ютері також потрібно встановити платформу Node.js разом із менеджером пакетів npm.

Під час розробки сервісу парсингу вебданих використовуються бібліотеки Puppeteer, що забезпечує виконання процедур вебскрейпінгу, mongodb для взаємодії з базою даних, а також dotenv для роботи зі змінними середовища. Розробка клієнтської частини вебзастосунку передбачає використання таких залежностей, як SvelteKit, Vite, TypeScript, Tailwind та Svelte. Для реалізації API-модуля застосовуються бібліотеки SvelteKit, TypeScript, MongoDB і Vite.

Як розподілену систему керування версіями обрано сервіс GitHub. Розгортання застосунку здійснюється за допомогою хмарної платформи Vercel, яка підтримує широкий спектр сучасних вебтехнологій, зокрема Next.js, React, Angular, Vue та SvelteKit.

3.2 Середовища застосунку

У процесі розробки програмного забезпечення під середовищами розуміють сукупність окремих налаштувань і конфігурацій, з якими застосунок взаємодіє протягом усього життєвого циклу свого функціонування [34]. Кожне середовище відповідає певному етапу та є ізольованим від інших, що дає змогу підтримувати різні фази створення,

перевірки та впровадження програмного продукту. Використання таких середовищ забезпечує коректну роботу коду, конфігураційних параметрів і даних відповідно до поставлених цілей.

Організація процесу розробки через декілька середовищ дозволяє розробникам підтримувати високий рівень якості програмного коду, здійснювати перевірку правильності функціонування, гарантувати безпеку та досягати оптимальної продуктивності на кожному етапі життєвого циклу застосунку. Чітке розмежування середовищ створює умови для детального тестування, контрольованого внесення змін і стабільного розгортання програмного забезпечення для кінцевих користувачів.

З метою забезпечення стабільності системи, якості та захищеності даних, а також тестування нових функціональних можливостей, для кожного модуля застосунку було організовано два середовища розробки – «test» та «prod» для тестування та релізу відповідно. Такий підхід до побудови багатосередовищної архітектури дає змогу суттєво знизити ризик порушення працездатності застосунку під час впровадження нових функцій або пошкодження даних унаслідок можливих помилок.

Для кожного модуля передбачено створення окремих середовищ. У контексті хмарної платформи MongoDB Atlas роль середовищ виконують проєкти, які дозволяють логічно розділити конфігурації та дані. Приклад організації різних середовищ у модулі MongoDB Atlas наведено на рисунку 3.1.

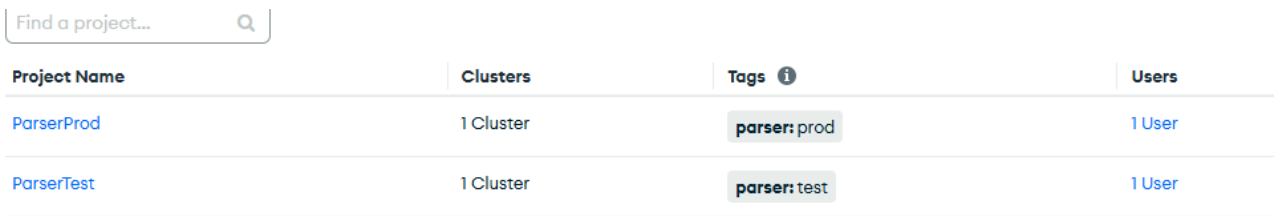

Рисунок 3.1 – Приклад різних середовищ всередині хмарної платформи MongoDB Atlas

Вебзастосунок та API зберігаються в одному git репозиторії на платформі GitHub. Створимо Приклад різних середовищ на платформі GitHub для вебзастосунку та API показано на рисунку 3.2.

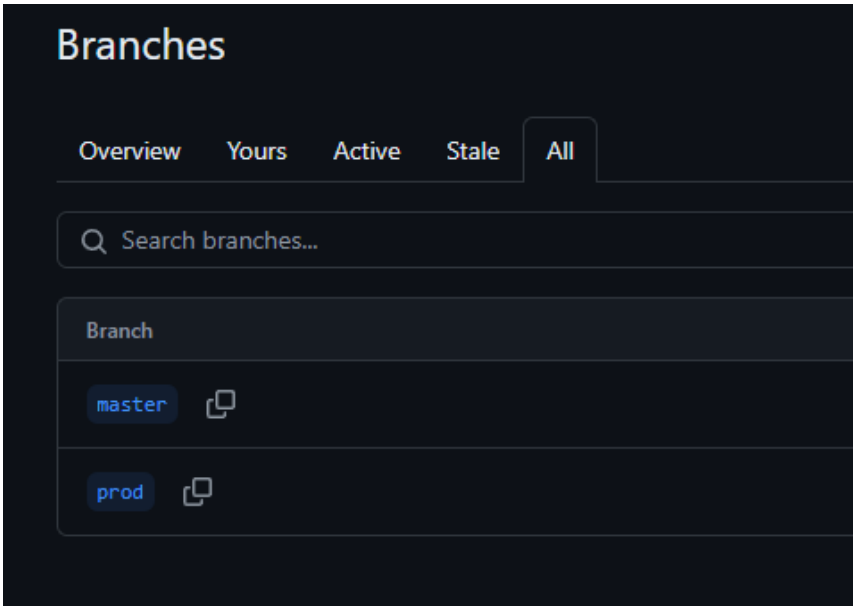

Рисунок 3.2 – Приклад різних середовищ усередині платформи GitHub

3.3 Розробка парсеру вебданих товарів

Для забезпечення коректного та точного вилучення релевантних даних з онлайн-магазинів необхідно попередньо сформувати перелік джерел та розробити окремі конфігураційні файли для кожного з них.

Структура проєкту передбачає наявність спільних файлів, до яких належать допоміжні функції, конфігураційний файл для підключення до бази даних, константи, а також модулі парсингу та обробки даних для кожного онлайн-магазину. У межах директорії, що відповідає конкретному магазину, розміщені піддиректорії з категоріями музичних товарів. Кожна така директорія містить два основні файли: конфігураційний файл і файл, що реалізує процедуру вебскрейпінгу (рис. 3.3). Загальна логіка виконання

вебскрейпінгу для кожного магазину винесена до файлу допоміжних функцій utils.js.

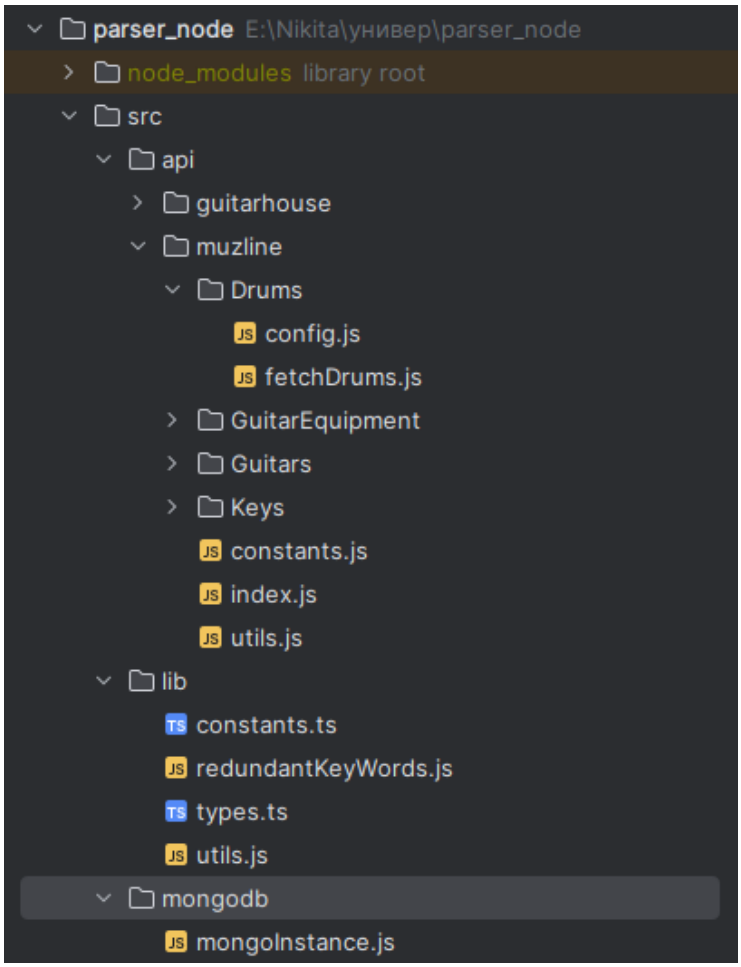

Рисунок 3.3 – Структура проєкту вебпарсера

Скрипти, що забезпечують автоматизований процес вебскрейпінгу, були реалізовані в окремому проєкті на платформі Node.js із використанням безкоштовної бібліотеки Chromium Puppeteer. Puppeteer є високорівневим програмним інтерфейсом для керування браузерами Chrome або Chromium, який розробляється командою Google Chrome. Дана бібліотека надає можливість автоматизувати браузерні процеси за допомогою платформи Node.js та забезпечує повний контроль над роботою браузера через протокол DevTools. Puppeteer широко застосовується для виконання таких завдань, як тестування вебзастосунків, збирання даних шляхом вебскрейпінгу,

формування PDF-документів і знімків екрана, а також автоматизації повторюваних браузерних операцій [35].

Однією з визначальних переваг бібліотеки Puppeteer є здатність точно відтворювати дії реального користувача в браузері. Це дає змогу виконувати складні інтерактивні сценарії, зокрема заповнення форм, натискання елементів керування, навігацію між сторінками та роботу з динамічним вмістом. Завдяки цьому розробники можуть створювати сценарії, що взаємодіють із вебсторінками аналогічно до поведінки користувача, що є особливо важливим під час перевірки функціональності вебзастосунків у реальних умовах. Такий підхід лежить в основі наскрізного тестування.

Бібліотека Puppeteer також характеризується високою точністю під час збору інформації та генерації контенту. Бібліотека дозволяє створювати знімки екранів із високою роздільною здатністю та формувати PDF-документи на основі вебсторінок із повним збереженням їхнього форматування та стилістики. Це робить бібліотеку Puppeteer ефективним інструментом для автоматизованого формування звітів, документації та презентацій, створених на основі вебконтенту. Додатковою перевагою є можливість виконання JavaScript-коду безпосередньо в середовищі браузера, що забезпечує доступ до динамічних даних і взаємодію зі складними вебзастосунками.

Важливою характеристикою бібліотеки Puppeteer є підтримка інтеграції з популярними фреймворками тестування, такими як Jest і Mocha, що спрощує його включення до наявних процесів автоматизації та тестування. Крім того, бібліотека підтримує виконання сценаріїв у безголовому режимі (headless mode), що дає змогу запускати автоматизовані процеси на серверних середовищах без використання графічного інтерфейсу.

Отже, бібліотека Puppeteer є потужним і гнучким інструментом для автоматизації браузерів, який надає широкий набір можливостей для тестування, збору даних і генерації контенту. Завдяки простоті використання та тісній інтеграції з платформою Node.js, дана бібліотека набула широкої

популярності серед розробників, які потребують надійних і ефективних засобів роботи з вебзастосунками. Команда для встановлення бібліотеки за допомогою менеджера пакунків npm наведена в лістингу 3.1.

Лістинг 3.1 Завантаження бібліотеки Puppeteer:

```
npm install puppeteer
```

Для отримання даних, перш за все треба ініціалізувати віртуальний браузер, потім створити так званий об’єкт вебсторінки, та визначити посилання, дані з якого нас цікавлять. Розглянемо цей процес на прикладі на лістингу 3.2.

Лістинг 3.2 Приклад навігації до вебресурсу та вилучення певних вебданих:

```
const browser = await puppeteer.launch({headless: true});
const page = await browser.newPage();
await page.goto(url, {waitUntil: 'domcontentloaded', timeout: 0});
await page.setViewport({width: 1920, height: 1080});
const data = await page.evaluate(retrieveYourDataFromPage())
```

3.4 Розробка БД

Оскільки відсутня можливість контролювати інформацію, що надається власниками інтернет-магазинів, а структура таких даних є неуніфікованою та динамічною, зокрема кількість і склад окремих категорій можуть змінюватися, доцільним є використання нереляційної бази даних MongoDB. Як первинний ключ, окрім внутрішнього ідентифікатора бази даних MongoDB _id, застосовується складений ідентифікатор, що формується на основі артикулу товару та назви магазину, який його пропонує.

До переліку ключових атрибутів, необхідних для подальшого відображення товарних пропозицій, належать ціна, назва, зображення, статус наявності, бренд і характеристики товару. При цьому набір характеристик не може бути визначений наперед, що зумовлює потребу у гнучкій структурі зберігання даних. З цією метою передбачено створення об’єкта filters, який акумулює всі можливі характеристики товару. Усередині даного об’єкта зберігаються поля filterName, що містить назву фільтра для подальшого відображення, та масив значень values, представлених у вигляді рядків, які відповідають конкретним властивостям товару. Такий підхід, зображений на лістингу 3.3, дозволяє коректно обробляти випадки, коли один товар має одночасно декілька значень певної характеристики.

Лістинг 3.3 Структура даних девайсу:

```
interface IDevice {
   "_id": ObjectId,
   "code": string,
   "provider": {
      "name": string,
      "url": string
   },
   "type": {
      "key": string,
      "name": string
   },
   "brand": {
      "name": string,
      "key": string
   },
   "deviceClass": {
      "key": string,
      "name": string
   },
   "filters": {
      "filterKey": {
         "filterName": string,
         "values": string[]
      },
   },
```

```
    "img": string,
    "inStock": false,
    "meta": {
       "updatedAt": string,
       "decomposed": string[]
    },
    "name": string,
    "price": number,
    "priceHistory": [
       {
          "updatedAt": string,
          "value": number
       }
    ],
    "stockHistory": {
       "updatedAt": string,
       "value": boolean
    }[],
    "url": string,
    "base": boolean
}
```

БД розташовано в хмарі сервісу MongoDB Cloud Atlas (рис. 3.4). Доступ до БД було здійснено через строку підключення (рис. 3.5).

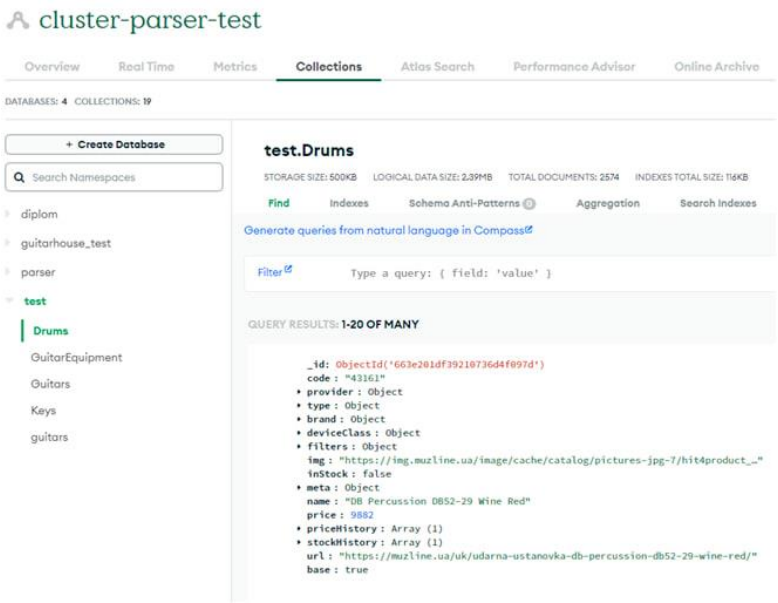

Рисунок 3.4 – Середовище MongoDB cloud, огляд тестової бази даних та чотирьох основних колекцій: Guitars, GuitarEquipment, Keys, Drums

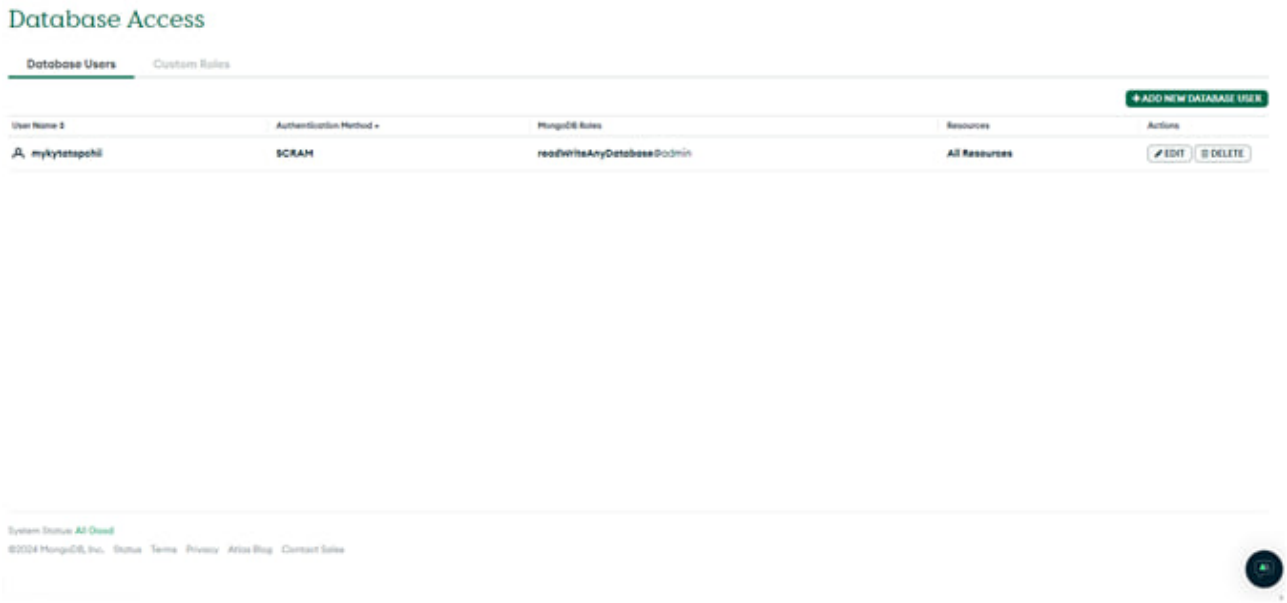

Рисунок 3.5 – Меню управління користувачами БД

3.5 Розробка вебзастосунку

Для реалізації клієнтської частини застосунку було обрано фронтенд-фреймворк Svelte у поєднанні з SvelteKit. Для оформлення візуального інтерфейсу вебсайту застосовується CSS-фреймворк Tailwind CSS. Навігація на сайті реалізована з використанням технології динамічної маршрутизації через slug URL.

3.5.1 Огляд фреймворків Svelte та SvelteKit

Фреймворк Svelte є сучасним фреймворком для розробки інтерфейсів користувача, створеним Річем Харрісом. На відміну від інших популярних фреймворків, таких як React чи Vue, Svelte виконує більшу частину обробки під час компіляції, а не під час виконання у браузері. Це дозволяє отримати менший розмір бандла та прискорити завантаження сторінок, оскільки відпадає потреба у великому runtime для рендерингу компонентів [36].

Однією з ключових переваг Svelte є його простота та зручність використання. Компоненти фреймворку Svelte розробляються у файлах із розширенням .svelte, де можна поєднувати мову розмітки HTML, стилів CSS і мову програмування JavaScript в одному файлі. Такий підхід забезпечує більш інтуїтивну організацію коду, коли логіка й стилі компонента зібрані разом. Реактивність реалізується за допомогою спеціальних синтаксичних конструкцій, що дозволяє автоматично оновлювати інтерфейс користувача при зміні стану без складних механізмів, які використовуються в інших фреймворках.

Фреймворк Svelte також підтримує стилізацію компонентів безпосередньо всередині файлів, ізольовані CSS-правила гарантують, що стилі застосовуються лише до конкретного компонента, що запобігає конфліктам. Крім того, є можливість використовувати зовнішні CSS-файли та препроцесори, такі як Sass або Less, для реалізації складніших стилізацій.

Ще однією важливою перевагою фреймворку Svelte є висока продуктивність. Оскільки більшість обчислень виконується під час компіляції, фінальний код у браузері є легким і оптимізованим, що особливо корисно для мобільних пристроїв або слабших комп’ютерів із обмеженими ресурсами. Svelte підтримує код-сплітінг і інші механізми оптимізації, що підвищують продуктивність великих вебзастосунків.

Загалом, фреймворк Svelte пропонує унікальний підхід до створення інтерфейсів користувача, поєднуючи простоту розробки та високу продуктивність, що робить його популярним серед розробників сучасних вебзастосунків.

Фреймворк SvelteKit є повнофункціональним фреймворком для створення вебзастосунків на основі фреймворку Svelte, що розширює його можливості. Він забезпечує інструменти для розробки клієнтських та серверних застосунків, підтримує статичний рендеринг, файловий роутінг та динамічну маршрутизацію.

Приклад файлового роутінгу з використанням технології файлового роутінгу «slug» у фреймворку SvelteKit наведено в лістингу 3.4.

Лістинг 3.4 Приклад файлового роутінгу у фреймворку SvelteKit:

```
src/routes/
  index.svelte       // головна сторінка
  about.svelte       // сторінка "Про нас"
  categories/
    index.svelte     // список категорій
    [slug].svelte    // сторінка конкретної категорії, де [slug] - динамічний параметр
```

Лістинг 3.5 Приклад створення сторінки у фреймворку Svelte:

```
<script>
  export let data;
</script>

<h1>About Us</h1>
<p>This is the about page of our website.</p>
```

Лістинг 3.6 Встановлення та ініціалізація проєкту фреймворку SvelteKit:

```
npm create svelte@latest my-app
cd my-app
npm install
npm run dev -- --open
```

3.5.2 Технологія файлового роутінгу

Для забезпечення ефективної та зручної навігації на вебсайті застосовано метод роутінгу з використання технології slug. Slug – це технологія файлового роутінгу, що являє собою частину URL, яка

ідентифікує конкретну сторінку у зрозумілій і читабельній формі. Зазвичай slug складається з букв, цифр та дефісів, які замінюють пробіли та спеціальні символи, роблячи URL більш зручним для користувачів і оптимізованим для пошукових систем.

Динамічні маршрути у фреймворку SvelteKit формуються шляхом використання квадратних дужок у назвах файлів у папці src/routes. Наприклад, файл [slug].svelte створює маршрут, здатний приймати різні значення slug.

Інформація про категорії завантажується з конфігураційного файлу, тоді як дані про конкретний товар отримуються з бази даних. Запит до бази даних виконується через API-модуль (рис. 3.6).

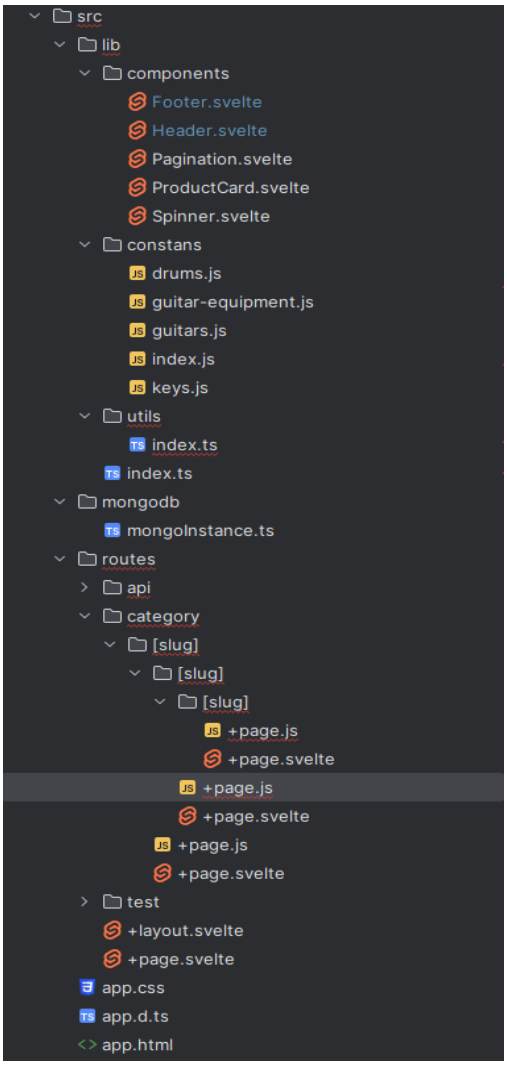

Рисунок 3.6 – Структура проєкту вебзастосунку

3.5.3 Tailwind CSS

Візуальна частина вебзастосунку реалізована з використанням CSS-фреймворку Tailwind CSS. Tailwind CSS є утилітарно-орієнтованою бібліотекою для створення інтерфейсів користувача, яка надає набір готових класів для стилізації елементів без необхідності написання власного CSS-коду. Класи можна застосовувати безпосередньо у HTML, що значно прискорює процес розробки та робить його більш інтуїтивним [37].

Для інтеграції Tailwind CSS у проєкт на фреймворку SvelteKit спершу необхідно встановити бібліотеку за допомогою відповідної команди, наведеної в лістингу 3.7.

Лістинг 3.7 Встановлення бібліотеки Tailwind:

```
npm install -D tailwindcss postcss autoprefixer
npx tailwindcss init -p
```

Лістинг 3.8 Приклад використання Tailwind-класів:

```
<div class=”flex justify-center items-center”>
```

Конфігураційний файл tailwind.config.js може містити налаштування кольорів, шрифтів та інших параметрів.

Лістинг 3.9 Приклад конфігураційного файлу Tailwind CSS:

```
module.exports = {
  theme: {
    extend: {
      colors: {
        primary: '#1D4ED8',
        secondary: '#9333EA',
      },
      spacing: {
```

```
      '128': '32rem',
    },
  },
 },
 variants: {},
 plugins: [],
}
```

Серед переваг Tailwind CSS можна виділити чотири ключові аспекти, які роблять цей фреймворк пріоритетним вибором серед інших CSS-бібліотек для швидкої та ефективної розробки сучасних вебзастосунків із привабливим дизайном:

– прискорення розробки завдяки використанню утилітарних класів, що усуває необхідність постійного перемикання між HTML і CSS-файлами;

– мінімальне навантаження стилями, забезпечене режимом JIT (Just-In-Time), який генерує лише потрібні класи, значно зменшуючи розмір кінцевого CSS-файлу;

– легкість підтримки коду, оскільки стилізація через утилітарні класи робить його зрозумілим і доступним для інших розробників;

– гнучкість і масштабованість, що дозволяє створювати унікальні дизайни без обмежень, накладених готовими шаблонами або компонентами інших фреймворків.

Поєднання сучасних фреймворків Svelte та Tailwind CSS забезпечує ефективну реалізацію дизайну та відображення даних із підтримкою реактивної фільтрації.

3.6 Розробка API

У проєкті на фреймворку SvelteKit важливою складовою є API, яке забезпечує взаємодію між клієнтською частиною та базою даних MongoDB.

API відповідає за отримання необхідних даних для відображення товарів на вебплатформі, а також за реалізацію функцій фільтрації та пошуку товарів за різними критеріями. У цьому розділі описано розробку API, що включає чотири основні GET-методи: getAllDevices, getDevicesByName, getAllBrands та getAllFilters.

API реалізовано з використанням фреймворку SvelteKit і написано на мові програмування TypeScript. Основною метою є забезпечення швидкої та ефективної взаємодії з базою даних MongoDB, включаючи отримання, обробку та передачу даних для подальшого відображення на клієнтській частині.

Для роботи з БД MongoDB використовується офіційний драйвер MongoDB для платформи Node.js. Нижче наведено приклад підключення до бази даних.

Лістинг 3.10 Підключення БД:

```
import { MongoClient } from 'mongodb';
import { MONGO_CONNECTION_STRING, DB_NAME}  from 
'$env/static/private';

class MongoInstance {
   static _db: any;
   static _client: any;

   constructor(){}

   static async client() {
      if(!this._client) {
         this._client = await 
MongoClient.connect(MONGO_CONNECTION_STRING);
      }
      return this._client;
   }

   static async db() {
```

```
        if(!this._db){
            this._db = (await MongoInstance.client()).db(DB_NAME);
        }
        return this._db;
    }
}
export default MongoInstance;
```

Після підключення до бази та написання коду, маємо готовий API для застосунку, який можна використовувати на стороні клієнта (лістинг 3.11).

Лістинг 3.11 Приклад запиту:

```
function fetchDevices() {
        const response = await (await
fetch(`/api/getDevice?category=${category}&type=${type}&name=${name
}`, {
            method: 'GET',
            headers: {
                'content-type': 'application/json',
            },
        }))?.json();
        isLoading = false;

        return response;
}
```

3.7 Модуль клієнтського вебзастосунку

Модуль клієнтського вебзастосунку відповідає за відображення даних, отриманих у процесі вебскрейпінгу, із використанням бази даних Devices. Вимоги до вебзастосунку такі, що остаточний клієнстький модуль має виглядати інтуїтивно зручно та легко; давати змогу користувачу шукати

потрібний товар за .фільтрами класифікації інструментів та швидко отримувати бажаний результат.

Архітектурні особливості модуля передбачають:

– інтерфейс користувача, що абезпечує зручну та інтуїтивно зрозумілу взаємодію для перегляду категорій музичних інструментів і їхніх характеристик;

– класифікація та пошук товарів, яка реалізована із можливістю фільтрації за різними параметрами, що спрощує пошук необхідних продуктів користувачами;

– відображення пропозицій реалізовано так, щоб кожен товар демонструвався з інформацією про ціни та наявність у різних онлайн-магазинах, що дозволяє обирати найбільш вигідні пропозиції;

– пагінація передбачена для зручного перегляду великих обсягів товарів.

3.8 Модуль API

Модуль API написаний на мові програрмування TypeScript у середовищі SvelteKit з підтримкою технології динамічної файлової маршрутизації slug та відповідає за отримання та обробку даних із бази даних MongoDB. Архітектурні особливості модуля включають:

– API, що забезпечує доступ до інформації, збереженої у MongoDB, дозволяючи зовнішнім застосункам отримувати та обробляти ці дані;

– запити для отримання списків товарів, фільтрів, брендів та конкретних продуктів, що забезпечує гнучке використання даних;

– підтримка фільтрації на рівні API, що оптимізує процеси пошуку та відбору товарів.

3.9 Проєктування архітектури застосунку

Вебзастосунок є системою, що складається з кількох компонентів і побудована за клієнт-серверною архітектурою. Клієнтська частина реалізована за допомогою сучасного фронтенд-фреймворку Svelte, а стилізація інтерфейсу виконана із застосуванням CSS-фреймворку Tailwind CSS. Серверна частина побудована на базі SvelteKit. Для зберігання даних обрана хмарна нереляційна база даних MongoDB Cloud. Парсер створено на основі Node.js та бібліотеки Puppeteer, який забезпечує завантаження первинної неструктурованої інформації з вебсайтів музичних магазинів, а також її обробку, включно з фільтрацією, нормалізацією та структуризацією.

Архітектура системи поділяється на три основні компоненти, кожен з яких виконує специфічні функції та взаємодіє з іншими через базу даних MongoDB та API. Кожен компонент реалізовано із застосуванням певного набору технологій, що забезпечує його ефективну роботу та інтеграцію в загальну систему.

Технології ParserService:

– платформа Node.js є серверною платформою для виконання JavaScript-коду на сервері, що забезпечує розробку високопродуктивних мережевих застосунків;

– база даних MongoDB це NoSQL база даних для зберігання зібраних даних у вигляді документів, обрана за гнучкість і масштабованість;

– бібліотека Puppeteer являється інструментов для автоматизації роботи з вебсторінками, що використовується для збору даних із онлайн-магазинів.

Технології Client:

– фреймворк SvelteKit це сучасний фреймворк для створення високопродуктивних вебзастосунків, що забезпечує швидку розробку та оптимізовану роботу інтерфейсу;

– база даних MongoDB виконує задачу зберігання інформації про товари та швидкого доступу до неї;

– фреймворк Tailwind CSS використаний для створення адаптивного та привабливого інтерфейсу;

– мова програмування TypeScript є надбудовою над мовою JavaScript із статичною типізацією, що підвищує якість коду та зменшує кількість помилок.

Технології API:

– фреймворк SvelteKit використовується для розробки API, забезпечуючи високу продуктивність і зручність створення серверної логіки;

– база даних MongoDB використана для зберігання та доступу до інформації про товари;

– мова програмування TypeScript забезпечує надійність і високу якість коду API.

Всі компоненти системи взаємодіють через базу даних MongoDB та API: парсер збирає й зберігає дані, вебзастосунок відображає їх користувачам, а API надає доступ до інформації зовнішнім сервісам. Така архітектура забезпечує високу гнучкість, масштабованість і ефективність системи.

3.10 Модуль парсингу товарних пропозицій

Модуль парсингу товарних пропозицій відповідає за навігацію по заданих вебсайтах, обробку та збереження отриманих даних у базі Devices. Крім того, цей модуль реалізує рішення проблеми зіставлення товарів, шляхом розбиття назви кожного товару на окремі слова та приведення їх до уніфікованої структури, після чого відбувається порівняння наборів слів для виявлення однакових товарів.

Архітектурні особливості модуля передбачають збір даних через парсер, який спочатку отримує інформацію з різних онлайн-магазинів, використовуючи бібліотеку Puppeteer для автоматизації браузерних дій. Пісдя чого зібрані дані обробляються для ідентифікації однакових товарів (зіставлення товарів), що дозволяє уникнути дублювання інформації. Наприкцінці дані зберігаються у базі даних MongoDB, що забезпечує їхню доступність для інших компонентів системи.

3.11 Розробка дизайну вебзастосунку

Дизайн вебзастосунку передбачає простоту та зрозумілість інтерфейсу. Функції пошуку товарів спрощені за рахунок категоризації продуктів, відображення основних категорій на головній сторінці та використання наочних елементів, таких як іконки. Для інтерфейсу обрано мінімалістичний стиль із домінуванням чорно-білих тонів та зелених акцентів. Головна сторінка застосунку демонструє основні категорії товарів, а безпосередньо під ними відображається список більш детальних підкатегорій (рис. 3.7).

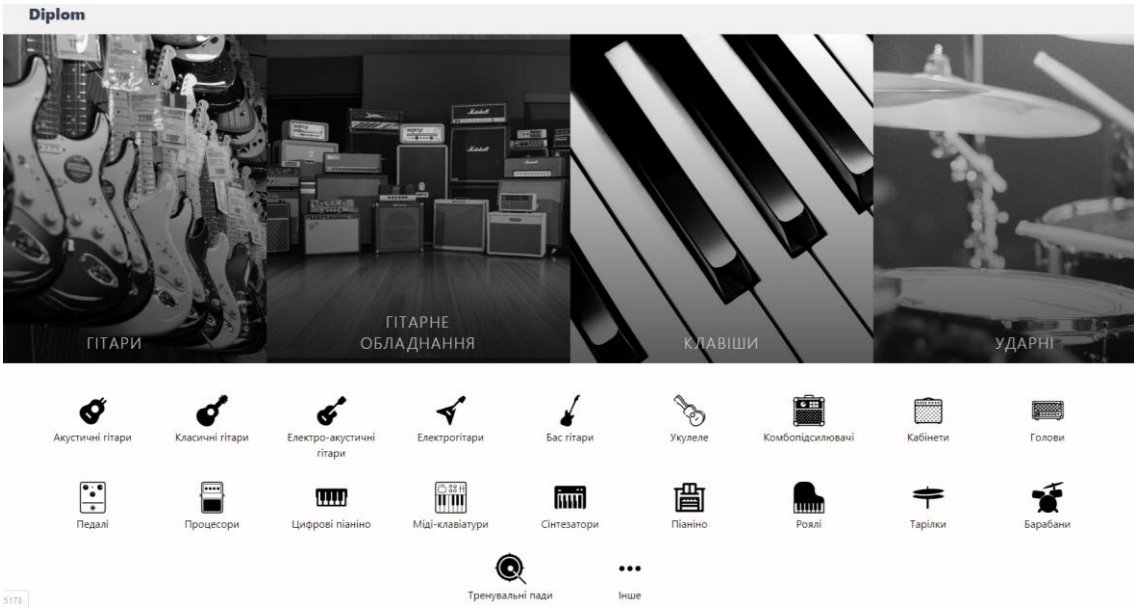

Рисунок 3.7 – Головна сторінка застосунку

Кожна категорія зі списку основних містить посилання на сторінку з переліком підгруп музичних інструментів (рис. 3.8). Для зручного сприйняття інформації підгрупи відображаються за допомогою SVG-іконок.

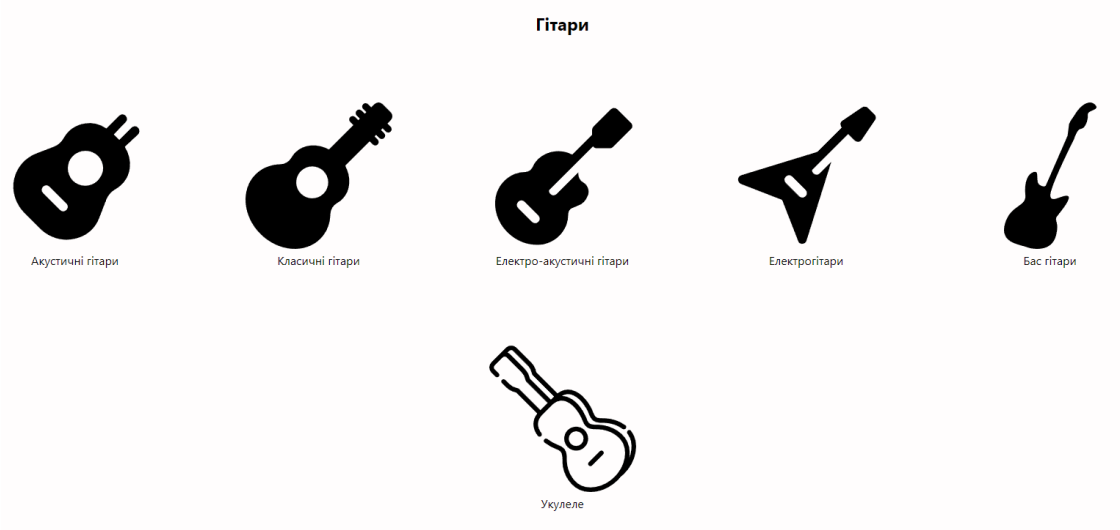

Рисунок 3.8 – Сторінка підгруп категорії

На сторінці підгрупи представлено список усіх музичних інструментів відповідної категорії, а також реалізовані фільтри і пагінація (рис. 3.9).

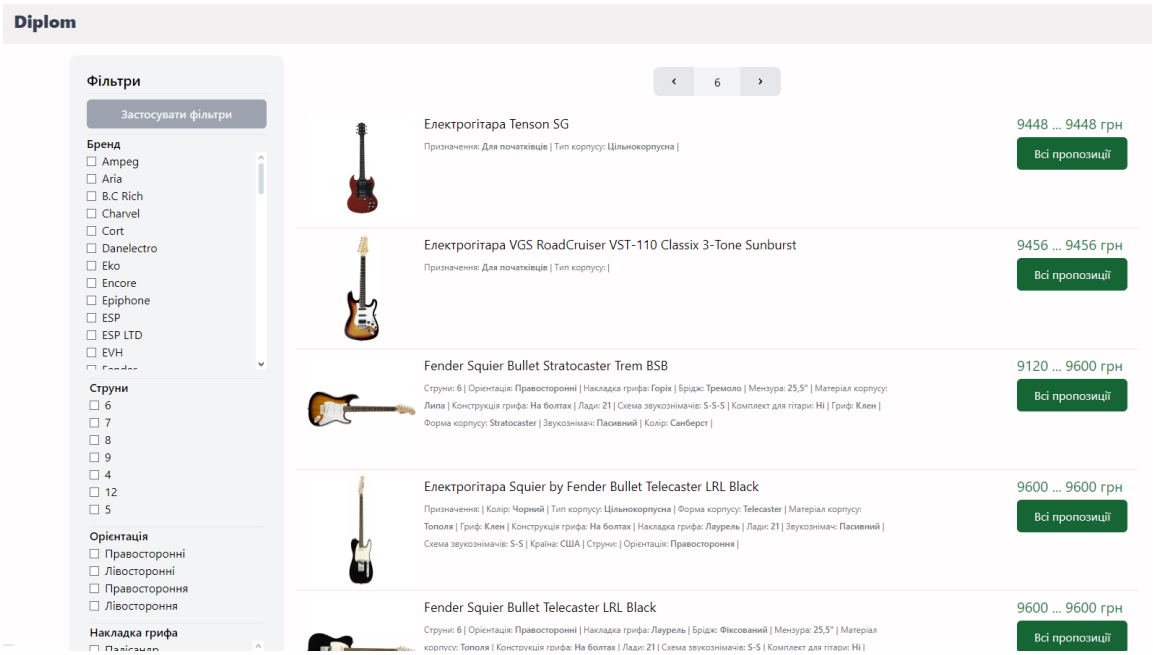

Рисунок 3.9 – Приклад списку товарів в категорії Гітари/Електрогітари

У кожному блоці товару в списку чітко та помітно відображається основна інформація про інструмент – назва, зображення та ціна. Характеристики товару подані дрібним шрифтом, щоб уникнути перевантаження візуальної інформації (рис. 3.10).

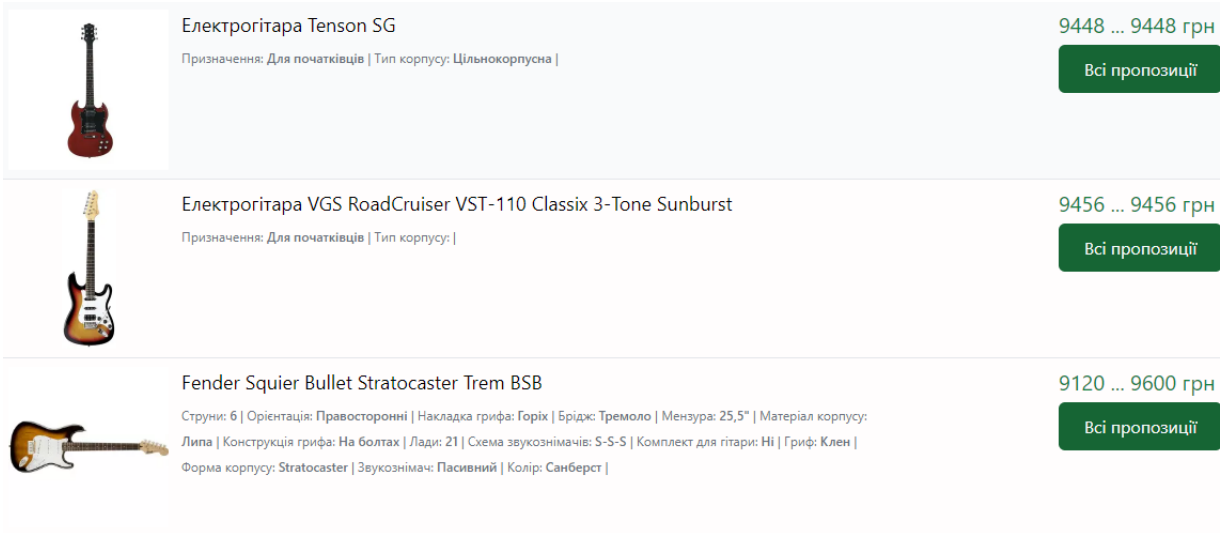

Рисунок 3.10 – Список товарів

Для спрощення пошуку передбачено блок фільтрації, розташований ліворуч від списку товарів. Щоб зробити процес фільтрації зручнішим, передбачено кнопку застосування обраних фільтрів (рис. 3.11).

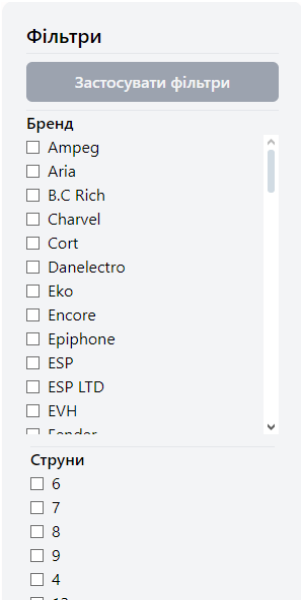

Рисунок 3.11 – Блок з фільтрами

Для комфорту перегляду та оптимізації завантаження даних, було розроблено блок пагінації (рис. 3.12).

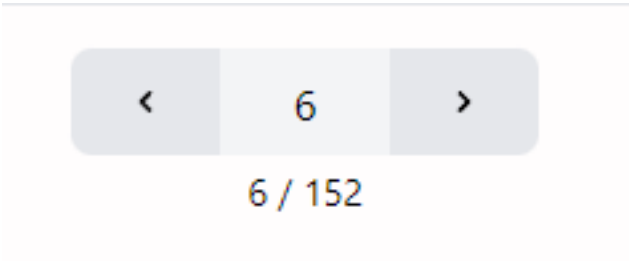

Рисунок 3.12 – Блок пагінації

Зі сторінки списку товарів має бути можливість перейти до сторінки кожного товару окремо (рис. 3.13).

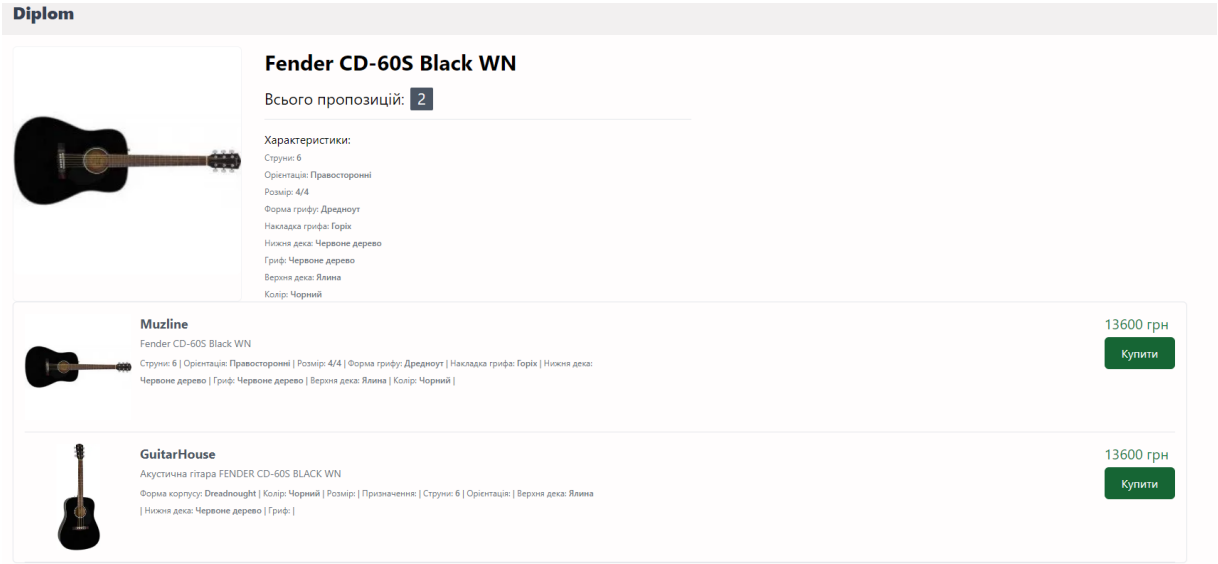

Рисунок 3.13 – Сторінка товару

На сторінці конкретного товару користувач має можливість ознайомитися з детальними характеристиками інструменту, переглянути кількість доступних пропозицій (рис. 3.14), а також ознайомитися зі списком магазинів, де представлений цей товар, з можливістю перейти безпосередньо до обраної онлайн-крамниці (рис. 3.15).

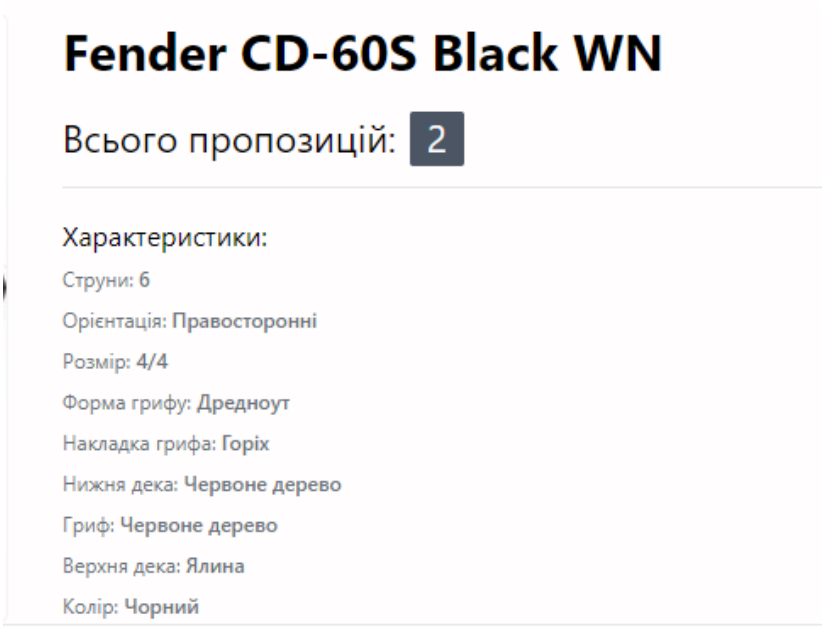

Рисунок 3.14 – Характеристика товару, кількість пропозицій

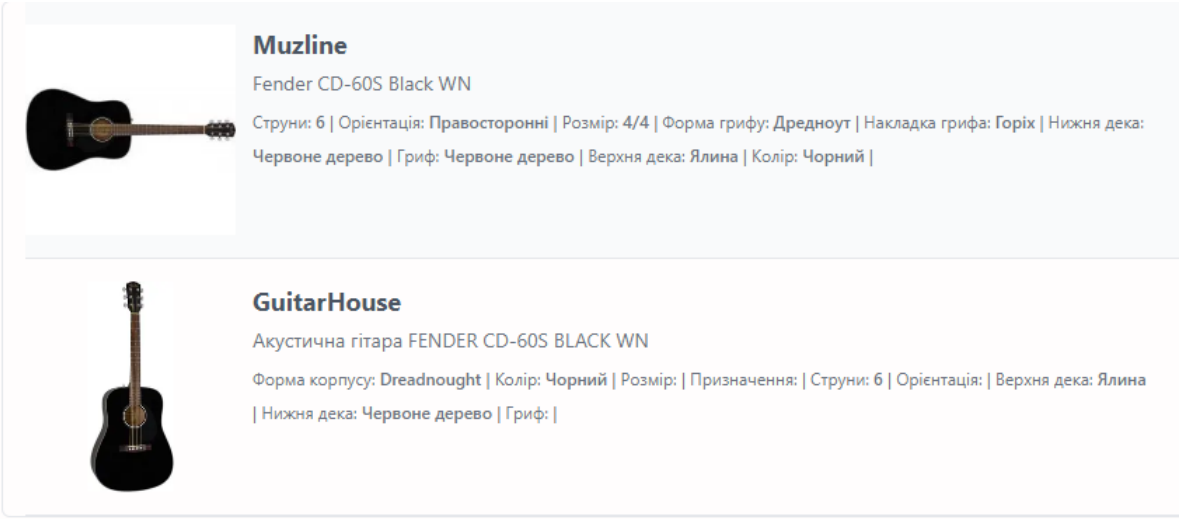

Рисунок 3.15 – Список магазинів з певним товаром

Футер вебплатформи містить узагальнену інформацію про сайт, включаючи основні категорії з можливістю переходу за посиланням на відповідну сторінку, а також контактні дані (рис. 3.16).

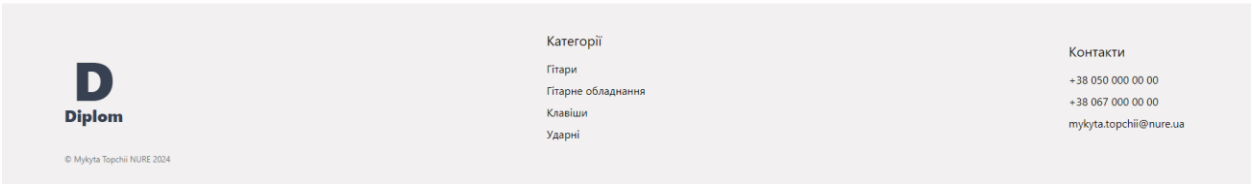

Рисунок 3.16 – Футер

3.12 Аналіз якості роботи розробленного застосунку

У цьому підрозділі представлено аналіз ефективності роботи розробленого застосунку–агрегатора музичних інструментів, який включає модуль вебпарсера та клієнтський вебінтерфейс. Тестування проводилося на спеціальному стенді, що складався з локального сервера з базою даних MongoDB і бекендом на платформі Node.js, а також клієнтської частини на SvelteKit. Для всіх експериментів використовувався однаковий набір джерел, який складається з онлайн-магазинів музичних інструментів із категоріями гітар, клавішних, ударних інструментів та гітарного обладнання, загальна кількість товарних позицій становила кілька тисяч. Умови тестування максимально наближені до реального сценарію, коли застосунок одночасно виконує періодичне оновлення даних і обробляє запити користувачів [38].

Аналіз швидкості оновлення показав, що модуль вебпарсера на платформі Node.js із Puppeteer забезпечує повне оновлення однієї категорії товарів за кілька хвилин при помірному навантаженні системи. Це досягається завдяки паралельному завантаженню сторінок, повторному використанню браузерних сесій і попередній фільтрації непотрібних запитів до сторонніх сервісів (скриптів, аналітики, рекламних блоків). Середній час повного циклу оновлення даних із усіх підтримуваних магазинів виявився прийнятним для щоденної синхронізації, а інкрементальне оновлення (тільки змінених позицій) виконується значно швидше, що дозволяє налаштувати частіший графік запуску без суттєвого впливу на продуктивність сервера. Важливо, що вебпарсер працює у фоновому режимі та не блокує роботу клієнтського застосунку, оскільки запис нових даних у базі даних MongoDB здійснюється пакетами, а запити читання використовують окремі індексовані колекції [39].

Швидкість пошуку товарів у клієнтському застосунку оцінювалася як за часом відповіді, так і за суб’єктивним сприйняттям користувача. Завдяки індексуванню основних полів (нормалізованої назви, бренду, категорії,

цінового діапазону) і використанню пагінації, середній час відповіді на типові запити, такі як «бренд + категорія» або «частина назви моделі», становить частки секунди, забезпечуючи майже миттєве відображення результатів у браузері. Навіть для складніших запитів із додатковими фільтрами та сортуванням за ціною затримка залишається комфортною для користувача, що свідчить про правильний вибір структури даних і коректну інтеграцію бекенду з клієнтською частиною: основна логіка відбору товарів виконується на сервері, а браузеру передається лише готова до відображення вибірка.

Особливу увагу приділено якості пошуку, тобто здатності системи знаходити саме ті товарні пропозиції, які очікує побачити користувач. Для цього сформовано контрольний набір запитів різних типів: точні назви моделей, запити з орфографічними помилками, комбіновані запити з брендом і характеристиками, пошук за частиною назви або псевдонімами інструментів. Практичний аналіз показав, що для більшості контрольних запитів цільові товари потрапляють на перші позиції списку результатів, а у випадках неоднозначних назв система пропонує всі релевантні варіанти, надаючи користувачеві можливість уточнити вибір.

Отже, розроблений застосунок забезпечує прийнятний баланс між швидкістю оновлення даних, часом відповіді на пошукові запити та точністю результатів. Вебпарсер швидко адаптується до змін у структурах цільових сайтів, а інкрементальна схема оновлення підтримує базу даних в актуальному стані без надлишкового навантаження. Пошуковий модуль завдяки поєднанню індексованої структури MongoDB та мультимодальних алгоритмів зіставлення забезпечує високу продуктивність і точність підбору товарів. Це підтверджує доцільність обраної архітектури та технологій і створює підґрунтя для подальшого розвитку функціональності, зокрема інтеграції складніших методів рекомендацій або персоналізації результатів пошуку.

ВИСНОВКИ

Таким чином, у кваліфікаційній роботі досліджено проблему автоматизованого зіставлення товарних пропозицій музичних інструментів на основі мультимодальної інформації та вирішено комплекс взаємопов’язаних завдань, спрямованих на створення ефективного підходу до ідентифікації однакових товарів у сервісах-агрегаторах:

– проведено систематичний аналіз літературних джерел, наукових публікацій та прикладних досліджень, що дозволило визначити сучасний стан розвитку методів зіставлення товарів, їхні переваги, обмеження та релевантні аспекти для домену музичних інструментів;

– досліджено текстові методи зіставлення товарів, включно з евристичною нормалізацією назв, виділенням ключових маркерів та застосуванням сучасної семантичної моделі SBERT, що дозволило підвищити точність інтерпретації назв із різними формулюваннями та зберегти ефективність при скорочених бренд-модельних позначеннях;

– реалізовано методи оцінювання візуальної подібності товарів за допомогою моделі CLIP, що дало змогу враховувати значення візуального каналу при зіставленні позицій з подібними назвами, але різними варіантами корпусу, кольору чи комплектації;

– побудовано та протестовано комбіновану метрику зіставлення товарів, яка інтегрує евристичний текстовий показник, семантичну близькість SBERT та візуальну схожість CLIP, визначено оптимальні вагові коефіцієнти та перевірено якість на реальних і тестових вибірках українських музичних магазинів;

– розроблено покроковий фінальний алгоритм зіставлення товарів, що включає модулі попередньої обробки даних, нормалізації назв, генерації пар товарів та ранжування на основі інтегрального показника, забезпечуючи масштабованість та стійкість алгоритму у реальних умовах агрегатора;

– побудовано архітектуру програмної системи, яка дозволяє автоматизувати процес зіставлення товарів, інтегрувати результати в базу даних агрегатора та розширювати модель новими джерелами даних;

– проведено експериментальне тестування окремих текстових та візуальних метрик, порівняння комбінацій вагових коефіцієнтів та формування фінального скору, що підтвердило практичну ефективність розробленого підходу.

Наукова новизна роботи полягає у розробленні та дослідженні комплексного мультимодального методу зіставлення товарних пропозицій музичних інструментів, який інтегрує евристичні техніки основані на текстових даних з сучасними семантичними та візуальними моделями. Запропонована інтегральна метрика дозволила досягти високої точності зіставлення товарів у сценаріях із неповними, неоднорідними та варіативними даними, що характерно для українського ринку музичних інструментів. Практична цінність полягає у можливості впровадження методики у реальні агрегатори товарних пропозицій, що підвищує якість зведених каталогів, зменшує дублювання позицій та покращує користувацький досвід.

Отже, виконано повний цикл дослідження, включно з аналітичною, методичною, експериментальною та прикладною частинами. Результати підтверджують доцільність використання мультимодального підходу до зіставлення товарних пропозицій та відкривають перспективи подальших досліджень щодо адаптації моделі до нових категорій товарів, удосконалення інтегральної метрики та використання генеративних моделей для підвищення точності порівняння текстів і зображень.

Результати дослідження апробовано у вигляді тез доповідей під час Міжнародного молодіжного форуму «Радіоелектроніка і молодь у XXI столітті» [40]; Матеріали XV-ої міжнародної науково-практичної конференції «Free And Open Source Software» [41].

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1.Web Scraping Basics. URL: https://www.datacamp.com/tutorial/web-scraping-using-python (дата звернення 28.09.2025).

2.Russell, M. (2019). Mining the Social Web. 3rd edition. O'Reilly Media, pp. 45–68.

3.API Basics. URL: https://www.redhat.com/en/topics/api/what-are-apis (дата звернення 28.09.2025).

4.Muzline – Інтернет-магазин музичних інструментів. URL: https://muzline.ua (дата звернення 10.10.2025).

5.GuitarHouse – Музичні інструменти та обладнання. URL: https://guitarhouse.com.ua (дата звернення 10.10.2025).

6.Muztorg – Музичне обладнання. URL: https://muztorg.ua (дата звернення 10.10.2025).

7.Yakovleva, O., Kovtunenko, A., Liubchenko, V., Honcharenko, V., & Kobylin, O. (2023). Face Detection for Video Surveillance-based Security System. *CEUR Workshop Proceedings* Vol. 3403. pp. 69-86. ISSN 1613-0073.

8.Yakovleva, O., Kovač, M., Ardasov, V. & Yeremenko, I. (2023). Study on adding functionality to the Zoom online conference system for monitoring the participant activities. *Public Administration and Regional Development*, 19(1), pp. 161–186.

9.Yakovleva O., Matúšová S., Tvoroshenko I., Isaiev Y. (2024). Visitor counting based on video stream analysis from surveillance cameras. Scientific Journal of Bratislava University of Economics and Management «Public Administration and Regional Development, Economics, Management and Marketing», vol. 20, no. 1, pp. 67–87.

10.Yanholenko, O., Grinchenko, M., Rohovyi, M., Yakovleva, O., & Rogovyi, A. (2025). The model and method of intelligent planning of IT project team work. PhD Workshop on Artificial Intelligence in Computer Science at 9th

International Conference on Computational Linguistics and Intelligent Systems (CoLInS-2025). CEUR Workshop Proceedings Vol. 3403. pp. 134-149.

11.Yakovleva, O., Matúšová, S., & Táncošová, J. (2024, December 16-18). Investigation of LLMs for generating answers based on user-provided content to support educational and organizational processes. Abstracts of XVI International Scientific and Practical Conference «Modern and new technical trends that help humanity». Thessaloniki, Greece, Pp. 289-295.

12.Yakovleva, O., Nebeský, L., & Kirichenko, A. (2023). Using the GPT models for responses based on custom content to develop neural consultant for university applicants. Abstracts of V International Scientific and Practical Conference «The world of modern technologies and inventions» Madrid, Spain. Pp. 172-178.

13.Cherednichenko, O., Ivashchenko, O., Cibák, Ľ. & Lincenyi, Marcel. (2023) "Item Matching Model in E-Commerce: How Users Benefit" Economics and Culture, vol.20, no.1, pp.77-90.

14.Cherednichenko, O., Ivashchenko, O., Lincényi, M., & Kováč, M. (2023). Information technology for intellectual analysis of item descriptions in e-commerce, Entrepreneurship and Sustainability Issues 11(1): 178-190.

15.Gorokhovatskyi V., Tvoroshenko I., Yakovleva O., & Hudáková M. (2025) Image description compression in classification structural methods, IEEE Access, vol. 13, pp. 43631-43641.

16.Yakovleva, O., & Nikolaieva, K. (2020). Research Of Descriptor Based Image Normalization And Comparative Analysis Of SURF, SIFT, BRISK, ORB, KAZE, AKAZE Descriptors. *Advanced Information Systems, 4*(4), 89-101.

17.Gorokhovatskyi, V., Tvoroshenko, I., & Yakovleva O. (2024) Transforming image descriptions as a set of descriptors to construct classification features, Indonesian Journal of Electrical Engineering and Computer Science, vol. 33, no. 1, pp. 113-125.

18.Gorokhovatskyi , O., & Yakovleva , O. (2024). Medoids as a Packing Of ORB Image Descriptors. Advanced Information Systems, 8(2), pp. 5–11.

19.Gorokhovatskyi, V., Tvoroshenko, I., Yakovleva, O., & Hudáková M., and Gorokhovatskyi O. (2024) Application a committee of Kohonen neural networks to training of image classifier based on description of descriptors set, IEEE Access, vol. 12, pp. 73376-73385.

20.Yakovleva, O., Nebeský, L, & Liakhov, P. (2023). Research methods of texture image analysis to solve the texture search problem. Proceedings of the IV International Scientific and Practical Conference «The world of modern technologies and inventions». Vienna, Austria. 2023. pp. 252-261.

21.Yakovleva, O., Matúšová, S., Liubchenko, V., & Maksimov, H. (2025). Innovative solutions based on AI in education, on the example of generating multimodal lecture notes. Scientific Journal of Bratislava University of Economics and Management «Public Administration and Regional Development, Economics, Management and Marketing», vol. 21, no. 1, pp.140–163.

22.Yakovleva, O., Matúšová, S., & Koshel, V. (2025, February 21). Implementation of AI approaches in current tools for managing image collections to improve the search capabilities. Proceedings of the IV Correspondence International Scientific and Practical Conference «Science in motion: classic and modern tools and methods in scientific investigations» in Periodical International scientific journal «Grail of science». Vinnytsia, Ukraine - Vienna, Austria. Vol. 49. pp. 752–755.

23.Cherednichenko, O., Kanishcheva, O., Yakovleva, O., & Arkatov, D. (2020). Collection and Processing of a Medical Corpus in Ukrainian. corpus, 2(4), 7-14.

24.Cherednichenko, O., Vovk, M., Yanholenko, O., & Yakovleva, O. (2020). Towards the Technology of Employers' Requirements Collection Development. In *Integrated Computer Technologies in Mechanical Engineering* (pp. 228-239). Springer, Cham.

25.Яковлева, О. В., & Кускова, І. В. (2006). Дослідження результатів сегментації зображень методом матриць збігів. Вісник Національного технічного університету "ХПІ". №39 - С.164 -171.

26.Яковлева, О. В., & Панченко, І. А. (2007). Застосування енергетичних характеристик Лавса для сегментації зображень. Біоніка інтелекту : науково-технічний журнал. №2(67). - С.94-98.

27.Яковлева О.В., & Нестерова О.П. (2009) Порівняльний аналіз методів характеристик Лавса і матриць збігів у задачах сегментації текстурних зображень. Прикладна радіо-електроніка: науч.-техн. журнал, Том 8, №2. - С.181 - 187.

28.Вечірська, І., Іщенко, О., & Яковлева, О. (2025). Предикатний аналіз даних для побудови системи вибору оптимального місця відпочинку у вигляді логічної мережі. Information Technology: Computer Science, Software Engineering and Cyber Security, (1), 26–32.

29.Соколова, А., Вечірська, І., & Яковлева, О. (2025). Побудова та аналіз ієрархічної моделі вибору локації для школи. Міжнародний науковий журнал «Грааль науки», (51), 299–308.

30.The Future Of Product Matching In E-Commerce. URL: https://medium.com/%40amandubey_6607/the-future-of-product-matching-in-e-commerce-trends-and-innovations-ae36a67c69cf (дата звернення: 28.10.2025).

31.Solving Ecommerce Challenges Via Product Matching. URL: https://anakin.company/blogs/solving-ecommerce-challenges-product-matching (дата звернення: 28.10.2025).

32.Smarter Product Matching in Ecommerce. URL: https://forbytes.com/blog/product-matching-in-ecommerce/ (дата звернення: 29.10.2025).

33.What is Product Matching in eCommerce? - Awesome Web. URL: https://www.moreawesomeweb.com/what-is-product-matching-in-ecommerce/ (дата звернення: 30.10.2025).

34.Price2Spy – Product Matching Service. Price2Spy. URL: https://www.price2spy.com/product-matching-service.html (дата звернення 01.11.2025).

35.NetRivals – Product Matching Software Powered by AI. Lengow. URL: https://www.lengow.com/solutions/netrivals/product-matching-software/ (дата звернення 01.11.2025).

36.Acelerar – eCommerce Product Data Matching Services. Acelerar. URL: https://www.acelerartech.com/ecommerce/ecommerce-product-data-matching-services/ (дата звернення 02.11.2025).

37.FasterCapital. Are there any disadvantages or limitations to using price comparison websites? URL: https://fastercapital.com/topics/are-there-any-disadvantages-or-limitations-to-using-price-comparison-websites.html/1 (дата звернення 04.11.2025).

38.Levenshtein, V. 1966. Binary Codes Capable of Correcting Deletions, Insertions, and Reversals. Soviet Physics Doklady, 10(8), pp. 707–710.

39.Dalal, N., Triggs, B. 2005. Histograms of Oriented Gradients for Human Detection. IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 886–893.

40.Топчій М.А. (2025). Розгляд питання об’єднання товарів з різних платформ на прикладі онлайн магазинів музичних інструментів. Радіоелектроніка і молодь у XXI столітті: Тези доповідей 29-го Міжнародного молодіжного форуму (Харків, 16–19 квітня 2025 р.). Харків: ХНУРЕ. Т. 7, с. 145–147.

41.Топчій М.А., & Яковлева О.В. (2024). Огляд бібліотеки Puppeteer для вирішення задачі Data Scraping з ціллю збору цінних даних. Матеріали XV-ої міжнародної науково-практичної конференції «Free And Open Source Software», Україна, Харків, 13-14 лютого 2024. С.101.