-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 перший (бакалаврський)

РОЗРОБКА ВЕБПЛАТФОРМИ ДЛЯ ОБ'ЄДНАННЯ ТОВАРНИХ ПРОПОЗИЦІЙ ВІД ОНЛАЙН-КРАМНИЦЬ МУЗИЧНИХ ІНСТРУМЕНТІВ
(тема)

Виконав:
студент 4 курсу, групи ІТІНФ-20-3
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)

2024 р.

Харківський національний університет радіоелектроніки

Факультет Інформаційно-аналітичних технологій та менеджменту
(повна назва)

Кафедра Інформатики
(повна назва)

Рівень вищої освіти перший (бакалаврський)

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

Тип програми освітньо-професійна

Освітня програма Інформатика
(повна назва освітньої програми)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«____»_____ 2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

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

1. Тема роботи Розробка вебплатформи для об'єднання товарних пропозицій від онлайн-крамниць музичних інструментів

затверджена наказом університету від 20 травня 2024 року № 464 Ст

2. Термін подання студентом роботи до екзаменаційної комісії 24 травня 2024 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.

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

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	Терміни виконання етапів роботи	Примітка
1	Отримання завдання на кваліфікаційну роботу	08.04.2024	
2	Аналіз завдання, підбір літератури	09.04.24-12.04.24	
3	Аналіз літератури з досліджуваної проблеми	13.04.24-18.04.24	
4	Аналіз технічних засобів	19.04.24-20.04.24	
5	Розробка методу вебскрейпінгу та product matching	21.04.24-25.04.24	
6	Програмна реалізація	26.05.24-24.05.24	
7	Оформлення пояснювальної записки	24.05.24-26.05.24	
8	Перевірка на плагіат	27.05.24	
9	Рецензування	28.05.24	
10	Підготовка презентації та доповіді	29.05.24-02.06.24	
11	Занесення роботи в електронний архів	06.06.24	
12	Попередній захист кваліфікаційної роботи	06.06.24	

Дата видачі завдання 8 квітня 2024 р.

Студент _____
(підпис)

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

РЕФЕРАТ/ABSTRACT

Пояснювальна записка до кваліфікаційної роботи: 65 с., 3 табл., 33 рис., 34 джерела.

ВЕБСКРЕЙПІНГ, БАЗА ДАНИХ, КЛІЄНТ, СЕРВЕР, ЗАСТОСУНОК, ВЕБПЛАТФОРМА, ПОСЛУГИ, ІНФОРМАЦІЯ, РЕКЛАМА, АНАЛІЗ, ІНСТРУМЕНТИ, МУЗИКА, ТОВАРИ.

Об’єктом роботи є питання об’єднання та порівняння товарних пропозицій.

Метою роботи є розробка вебплатформи для об’єднання товарних пропозицій від онлайн-крамниць музичних інструментів за допомоги методу вебскрейпінгу для збирання цінних даних про музичні інструменти.

В процесі роботи було проаналізовано технології вебскрейпінгу на базі Node.js бібліотеки Puppeteer. Розглянуто інші бібліотеки вебскрейпінгу та досліджено можливі архітектурні рішення для втілення системи об’єднання товарних пропозицій.

У результаті роботи створена комплексна програмна система для пошуку, зберігання та відображення компіляції даних від онлайн-крамниць музичних інструментів.

WEBSCRAPING, DATABASE, CLIENT, SERVER, APPLICATION, WEB PLATFORM, SERVICES, INFORMATION, ADVERTISING, ANALYSIS, TOOLS, MUSIC, PRODUCTS.

The object of the work is web platform with compiled products from the field of music production.

The purpose of the work is to develop a web-application for proving information and advertising services using the web scraping method to collect valuable data about musical instruments.

In the process of work, web scraping technologies based on Node.js library called Puppeteer were analyzed. Other web scraping libraries were considered and possible architectural solutions for the system of combining product offers implementation were investigated.

As a result of the work, a comprehensive software system was created for searching, storing and displaying a compilation of data from online stores of musical instruments.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

БД – база даних

ПЗ – програмне забезпечення

ІС – інформаційна система

JS – JavaScript

TS – TypeScript

ПО – предметна область

API – Application Programming Interface

Пайплайн – послідовність кроків або операцій, які виконуються над вихідним кодом або даними для досягнення певної мети

End user – людина, або група людей, що споживає кінцевий продукт, користується послугами

Data analysis – процес розбиття даних на компоненти, щоб знайти відповіді на актуальні питання бізнесу

Data science – розділ інформатики, який використовує наукові методи, процеси, алгоритми та системи для отримання знань та прогнозів із структурованих та нереформованих даних

W3C – World Wide Consortium, консорціум всесвітнього павутиння

ВСТУП

У сучасному цифровому світі музикальна індустрія постійно розвивається, пропонуючи широкий асортимент музичних інструментів та обладнання. Зростаюча конкуренція та швидкий темп життя ставлять перед галуззю музичного виробництва завдання ефективного просування та реалізації товарів на ринку. Більш того, потенційному клієнту, особливо починаючому музикантові, дуже легко загубитись у величезному різноманітті пропозицій. Отже, збір та аналіз інформації про товарні пропозиції може бути складним завданням.

Сьогодні все більш насущною стає потреба у зборі критично важливих даних, кількість яких непомірно росте з кожним днем, отже пошук, обробка та зберігання інформації становить актуальну та комплексну задачу, що частково вирішує вебскрейпінг.

Вебскрейпінг – це сучасний та зручний механізм отримання структурованих вебданих у власних або промислових цілях шляхом програмного аналізу вебресурсів [1]. Ручне копіювання та перегляд даних з вебсайтів – це довгий та складний процес, що вочевидь є не ефективним рішенням проблеми збору важливих даних. За допомоги різноманітних методів вебскрейпінгу можна автоматизувати та вдосконалити цей процес та інтегрувати його в пайплайн будь-якого застосунку [2].

Ефективний та автоматизований процес вебскрейпінгу було розроблено за допомоги Node.js бібліотеки Puppeteer дозволить підвищити показники роботи в сфері Data analysis, а розробка end user застосунку на основі новітніх технологій у зв’язці SvelteKit, MongoDB Cloud реалізує сучасний підхід у проєктуванні архітектурі прогресивних вебзастосунків.

1 АНАЛІЗ ОСНОВНИХ МЕТОДІВ ВЕБСКРЕЙПІНГУ ТА ОГЛЯД ІСНУЮЧИХ СЕРВІСІВ ПОРІВНЯННЯ ТОВАРНИХ ПРОПОЗИЦІЙ В ГАЛУЗІ МУЗИКАЛЬНИХ ІНСТРУМЕНТІВ

1.1 Огляд ресурсів для порівняння товарних пропозицій від різних онлайн-крамниць музикальних інструментів

Сьогодні, в епоху стрімкого розвитку інформаційних технологій та приватного бізнесу, люди зіткнулися з доволі широким та конкурентним ринком товарів та послуг в будь-яких галузях. Більш того, майже всі підприємці активно просувають свій бізнес в мережі Інтернет. Музикальне виробництво не стало винятком, тому доволі багато різних магазинів та дистриб’юторів музичного обладнання присутні на українському ринку зараз та мають онлайн-ресурс з перегляду та продажу своїх товарів [3].

Проблему перенасичення інформації в галузі товарного виробництва вирішують сервіси порівняння цін товарних пропозицій. Одним з популярніших ресурсів такої категорії в усьому світі є Google Shopping (рис. 1.1).

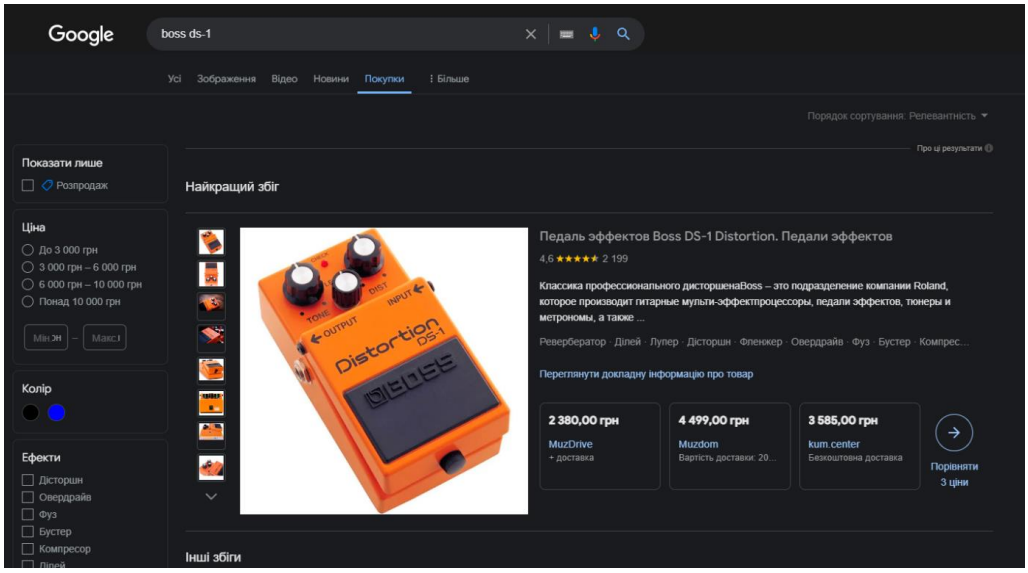

Рисунок 1.1 – Приклад порівняння цін на гітарну педаль ефектів в Google Shopping

Однак через вимушену узагальненість, фільтри в Google Shopping іноді можуть бути некоректними та надані не в повному обсязі [4].

Серед українських сервісів порівняння цін, можна виділити наступних лідерів: E-Katalog, Hotline, Magazilla. Розглянемо кожен з ресурсів.

1.1.1 E-Katalog

E-Katalog – це інтелектуальна та багатофункціональна платформа пошуку товарів у інтернет-магазинах. Сервіс охоплює такі категорії товарів: електроніка, комп’ютери, побутова техніка, автотовари, обладнання для ремонту та будівництва, туристичне спорядження, дитячі товари, тощо [5]. До системи E-Katalog наразі під’єднанні понад 3000 різних онлайн-крамниць.

Принцип роботи системи E-Katalog полягяє у прямої взаємодії сервісу та магазину. Для того, щоб пропозиції крамниці потрапили на вебсайт E-Katalog, представникам магазину потрібно заповнити форму, проконсультуватись з менеджером, щодо документів та формату прайс-листа.

Особливість полягає в спеціальному форматі прайс-листа, який має бути єдиним для того, щоб система E-Katalog правильно розпізнала та відобразила всю належну інформацію.

З мінусів можна відмітити обмежену кількість онлайн-магазинів та, як наслідок, відсутність повного обсягу товарних пропозицій, людський фактор в занесенні товарних пропозицій до сайту та певну затримку через обробку людьми. З плюсів можна відмітити велике різноманіття товарних пропозицій сучасний та зручний дизайн (рис. 1.2).

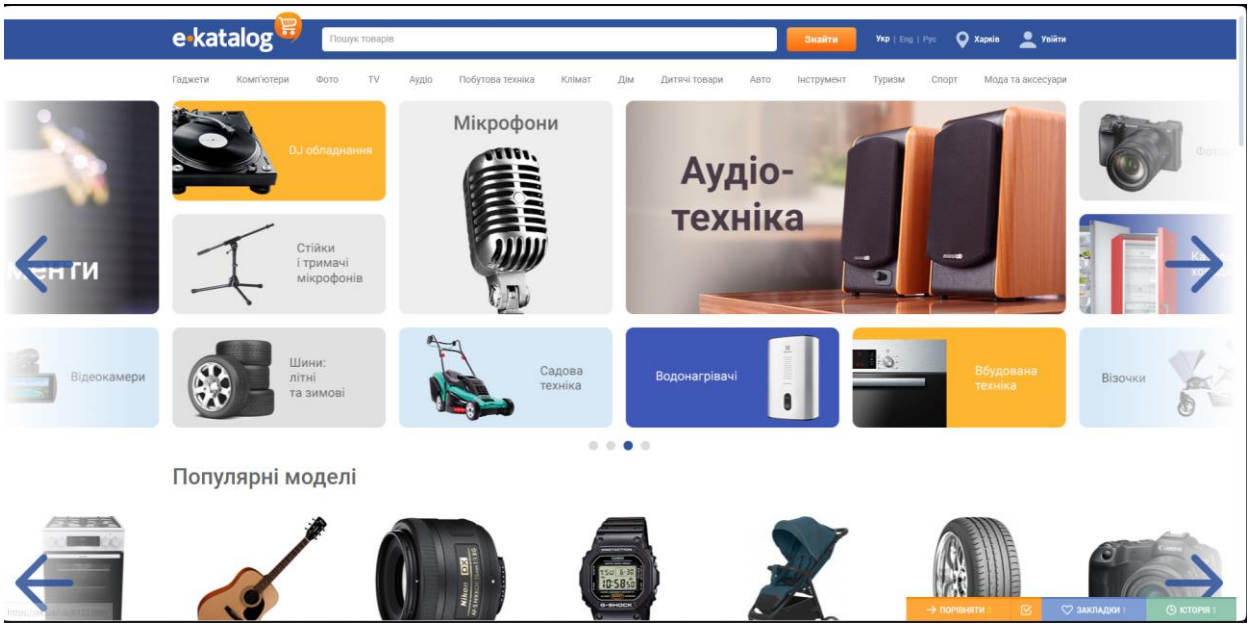

Рисунок 1.2 – Головна сторінка E-Katalog

1.1.2 Hotline

Hotline.ua – це ще один український гігант в галузі пошуку товарів та реклами магазинів. Принцип роботи дуже схожий з сервісом E-Katalog. Користувачам пропонують вибір товарів широкого спектру серед різних магазинів, продавцям – рекламні послуги своїх товарних пропозицій [6].

Магазинам потрібно заповнити заявку, та відправити її до адміністрації сервісу разом із необхідними документами.

Однією з умов потрапляння товарів того, чи іншого магазину до платформи Hotline є правильно оформлений так званий Товарний фід, який перевіряється менеджерами на коректність даних.

З недоліків можна зазначити крок участі людини в процесі додавання даних крамниці до бази Hotline, через можливу затримку у часі та помилки викликані людським фактором; та відсутності повного обсягу магазинів на платформі, через потребу звернення крамниць до сервісу.

Головна сторінка зустрічає користувача переліком всіх категорій товарів та текстовим полем пошуку (рис. 1.3).

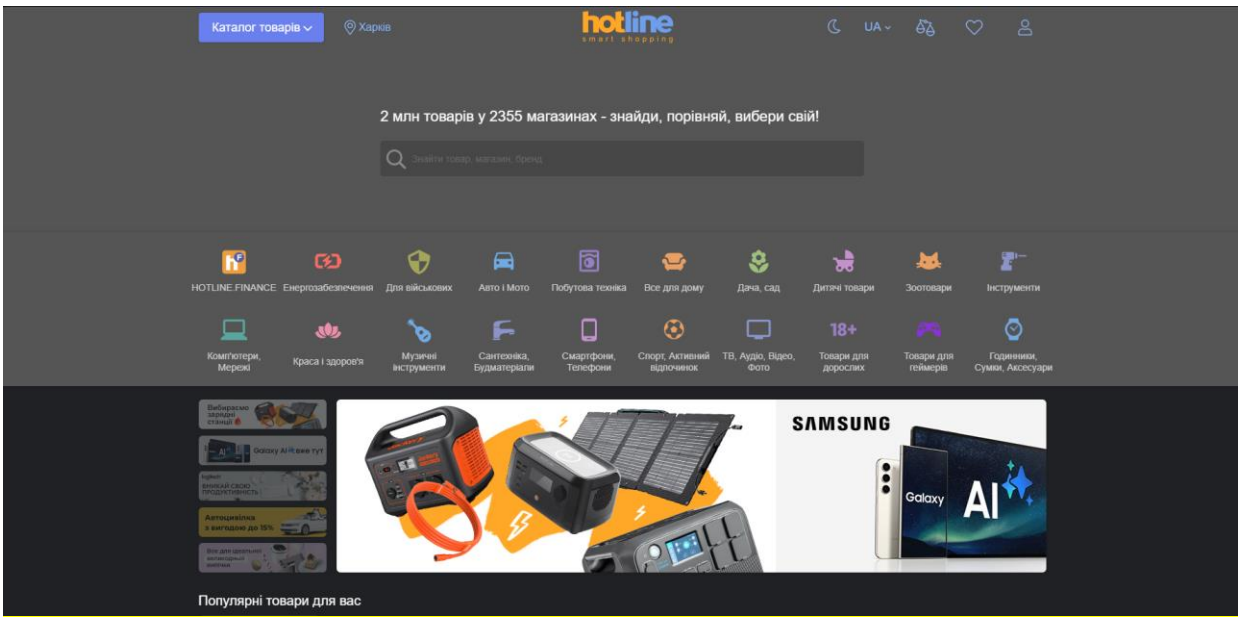

Рисунок 1.3 – Головна сторінка сервісу Hotline

1.1.3 Magazilla

Magazilla – сервіс порівняння цін від компанії власника платформи E-Katalog. Сервіс відображає товарні пропозиції категорій основних потреб побуту [7].

Так як онлайн-платформа Magazilla має спільного власника з вебсервісом E-Katalog, то процес розташування певної онлайн-крамниці реалізований таким самим алгоритмом як і у майданчика E-Katalog.

Через те, що ці дві системи мають спільний механізм підтримки та зворотнього зв’язку, Magazilla має ті самі недоліки, які має система E-Katalog, а саме – можлива затримка в доставці даних до БД, потенційні помилки скоєні людським фактором. Ще одним мінусом виступає застарілий дизайн вебсайту (рис. 1.4).

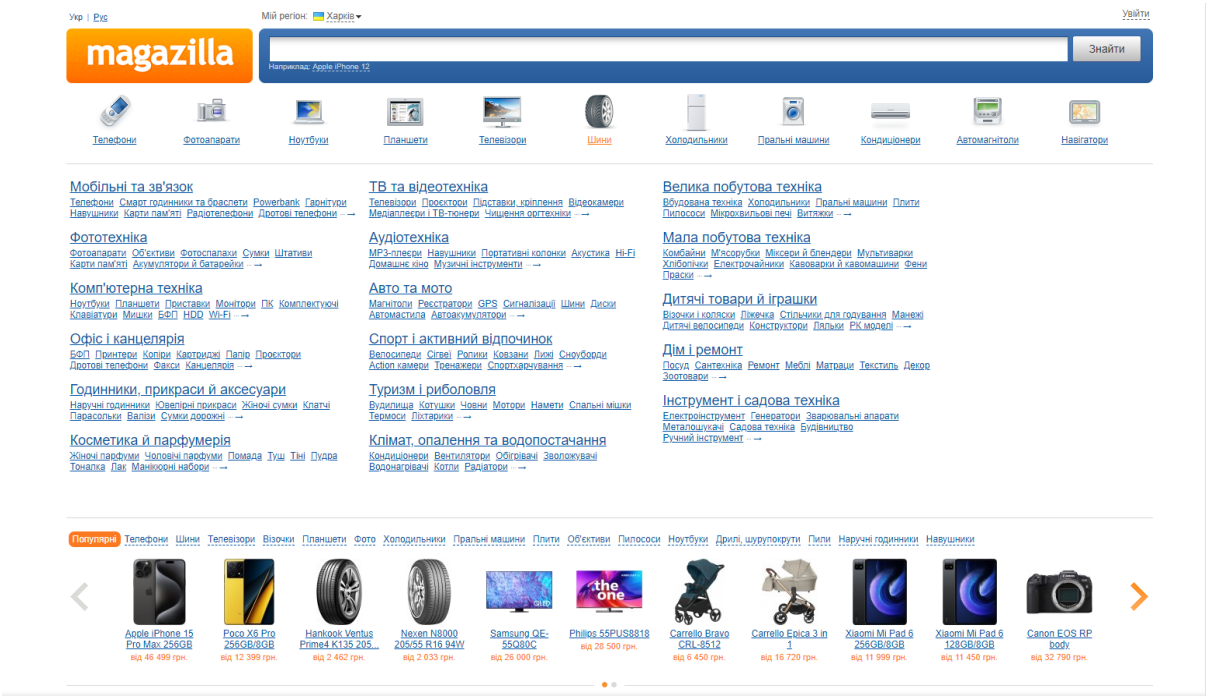

Рисунок 1.4 – Головна сторінка вебсервісу Magazilla

1.2 Огляд основних методів вебскрейпінгу

Вебскрейпінг – це технологія Data Science, яка розгортає сценарії для вилучення структурованих даних з вебсайтів. Сценарій – це комп’ютерна програма, яка автоматизує певне завдання за допомогою деяких вибраних мов програмування, таких як R, Python, JavaScript [8].

1.2.1 Мануальний вебскрейпінг

Мануальний вебскрейпінг передбачає ручний перегляд, копіювання та обробку даних із вебсайту у файл або електронну таблицю. Цей метод зазвичай використовується для невеликих завдань вилучення даних або коли автоматизація неможлива чи ресурси, витрачені на автоматизацію процесу є більшими, ніж мануальна робота [9]. Для подальшого порівняння різних

методів парсингу, розглянемо плюси та мінуси мануального вебскрейпінгу. До плюсів можна віднести:

– простота: ручний вебскрейпінг є простим у використанні, оскільки для мануального вебскрейпінгу не потрібні навички програмування чи технічні знання;

– відсутність технічних перешкод: на відміну від автоматизованих методів скрейпінгу, які можуть вимагати знання мов програмування або інструментів вебскрейпінгу, мануальний вебскрейпінг може виконувати кожен, хто має доступ до браузера;

– настроюваний: мануальний вебскрейпінг забезпечує високий ступінь налаштування щодо того, які конкретні дані витягувати та як їх упорядковувати. Користувачі можуть вручну вибирати та витягувати лише ті дані, які їм потрібні;

– підходить для невеликих завдань: мануальний вебскрейпінг підходить для невеликих завдань вилучення даних, де автоматизація невигідна або необхідна. Це швидке та просте рішення для вилучення невеликих обсягів даних.

Недоліки:

– забирає багато часу: ручне копіювання може зайняти надзвичайно багато часу, особливо для великих наборів даних або вебсайтів із великою кількістю даних. Він передбачає ручну навігацію вебсторінками, вибір даних і їх копіювання, що може бути неефективним для вилучення великих обсягів даних;

– схильність до помилок: під час мануальної роботи можуть виникнути людські помилки, наприклад випадкове копіювання неправильних даних або відсутність точок даних. Це може призвести до неточностей у отриманих даних;

– не масштабується: ручний скрейпінг не масштабується і стає непрактичним для вилучення великих обсягів даних або виконання

повторюваних завдань. Зі збільшенням обсягу даних експоненціально зростають час і зусилля, необхідні для ручного збирання;

– обмежений обсяг даних: через витрати часу та зусиль ручне копіювання можливе лише для вилучення невеликих або помірних обсягів даних. Він не підходить для завдань, що вимагають великомасштабного вилучення даних;

– не підходить для динамічного вмісту: мануальний вебскрейпінг може бути непридатним для вебсайтів із динамічним вмістом або частими оновленнями, оскільки вимагає ручного втручання для підтримки актуальності витягнутих даних.

Алгоритм проілюстровано на рисунку 1.5.

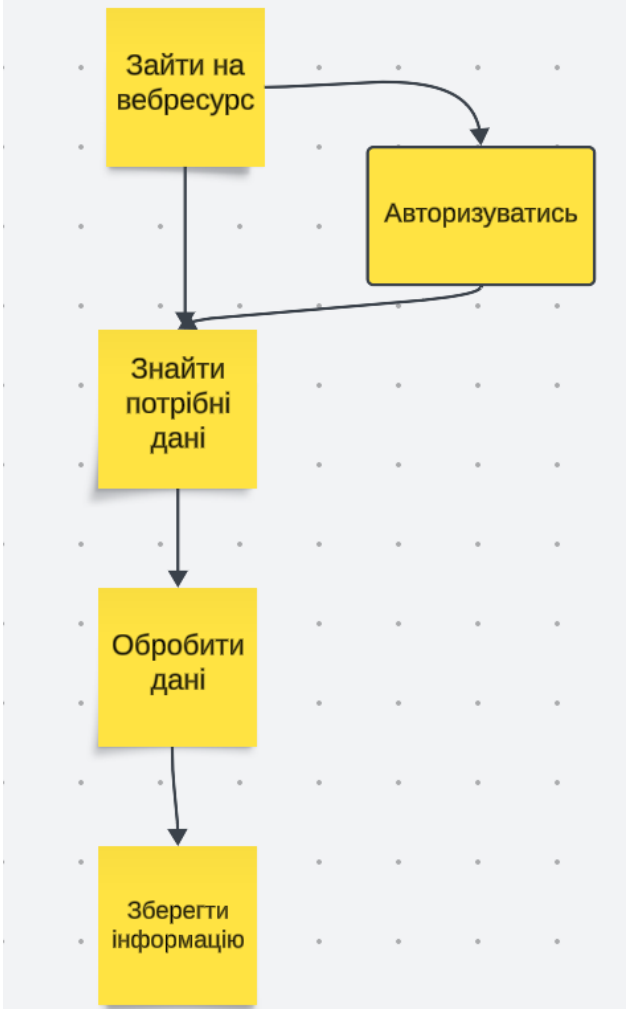

Рисунок 1.5 – Алгоритм отримання даних за допомогою мануального вебскрейпінгу

Загалом, незважаючи на те, що мануальний вебскрейпінг є простим і доступним, цей підхід не підходить для великомасштабних, промислових або повторюваних завдань вилучення даних через його трудомісткість і схильність до помилок. Автоматизовані методи збирання зазвичай є кращою опцією для таких завдань, оскільки вони забезпечують більшу ефективність, масштабованість і точність.

1.2.2 Використання API

Метод використання API для вирішення задачі вебскрейпінгу містити в собі доступ до вебданих через його програмний інтерфейс (API). Як правило, якщо у вебресурсу в наявності API, це означає, що всі значущі дані, що можуть цікавити кінцевих користувачів вже закладені в цей інтерфейс. Тому API метод забезпечує структурований і контрольований спосіб взаємодії додатків з інформацією та функціями вебсайту або вебсервісу [10].

API використовуються вебсайтами та онлайн-сервісами, щоб дозволити розробникам отримувати доступ до своїх даних програмним шляхом та автоматизувати процес оновлення інформації для кінцевих клієнтів шляхом візуалізації, надання агрегованих даних, тощо. Багато популярних вебсайтів і платформ, такі як соціальні мережи (наприклад, Twitter, Facebook), сайти електронної комерції (наприклад, Amazon, eBay) і постачальники даних (наприклад, служби погоди, постачальники фінансових даних), пропонують API для розробників, щоб отримати доступ до своїх даних.

Зазвичай, у вебслужбах зустрічаються чотири типи API, які часто використовуються: публічні, партнерські, приватні та складені [11]. У цьому контексті позначення API як «тип» означає його передбачувану сферу використання.

Публічні програмні інтерфейсі застосунку можуть мати некомерційні сервіси, що надають загальнодоступну інформацію, не захищену

юридичними документами чи авторськими правами або так звані fake API. Останні часто використовуються в навчальних цілях.

Партнерські API, доступні виключно для спеціально вибраних і авторизованих зовнішніх розробників або споживачів API, служать для оптимізації взаємодії між компаніями. Наприклад, якщо компанія має на меті вибірково розкривати дані своїх клієнтів зовнішнім фірмам CRM, партнерський API може встановити зв’язок між внутрішньою системою даних клієнтів і цими зовнішніми об’єктами, не дозволяючи використання будь-якого іншого API.

Приватні інтерфейси мають схожий принцип з партнерськими, тільки за винятком, що перші створені для використання всередині мережі розробників, що створювали API.

Складені API є інтерфейсами змішаного типу, наприклад, якщо у партнерського інтерфейсу є безкоштовна пробна версія.

Розглянемо плюси та мінуси API методу.

Плюси:

– надають доступ до вже структурованих даних у стандартизованому форматі (наприклад, JSON або XML), що полегшує вилучення та використання даних у програмах;

– використання API зазвичай потребує менше програмування та технічних знань порівняно з іншими методами;

– розроблено для забезпечення надійного доступу до даних через те, що інтерфейсом, як правило, постійно оновлюють та підтримують розробники;

– оскільки API надають доступ до даних у режимі реального часу або регулярно даних, що оновлюються, вони гарантують, що отримані дані є актуальними та точними.

Мінуси:

– іноді доволі висока вартість приватних API;

– не всі вебсайти пропонують API, а деякі API можуть мати обмеження щодо даних, які вони надають, або дозволених дій. Це може зробити обсяг даних дуже вибірковим та несумісним із задачами бізнеса;

– можуть зазнати змін або з часом вийти з підтримки, що вимагатиме від розробників оновлення свого коду для врахування цих змін. Це може призвести до накладних витрат на технічне обслуговування та потенційних збоїв у робочих процесах вилучення даних.

Отже, API метод пропонує структурований, надійний і масштабований підхід до доступу до даних із вебсайтів і онлайн-сервісів, який є дуже зручним інструментом для розробників (рис. 1.6), але не у всіх вебресурсів є така можливість.

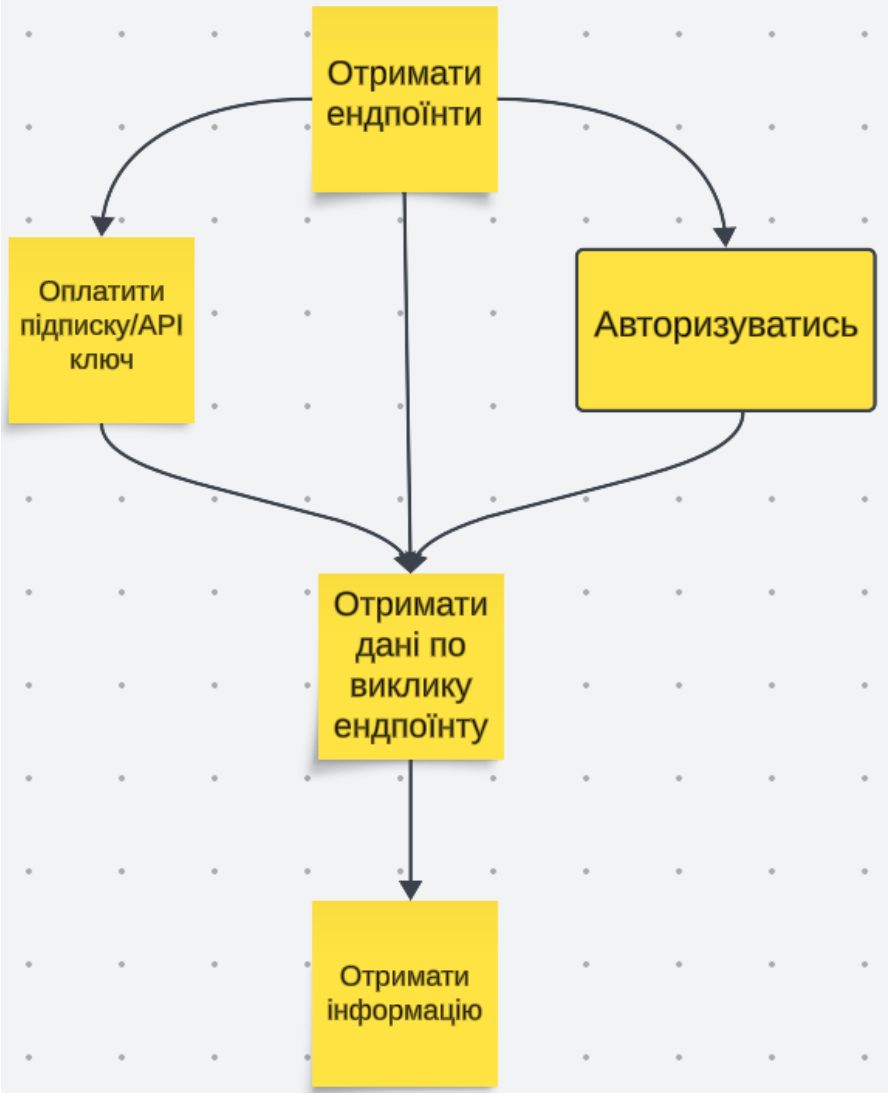

Рисунок 1.6 – Алгоритм отримання даних через API

1.2.3 DOM parsing

DOM parsing – це технологія вебскрейпінгу, яка отримує структуровані дані з готової HTML-сторінки, шляхом аналізу дерева DOM та витягування корисної інформації з його змісту.

DOM – це стандарт W3C (World Wide Consortium) [12]. DOM визначає стандарт для доступу до документів: «Модель об'єкта документа W3C (DOM) – це платформа та нейтральний інтерфейс, який дозволяє програмам та сценаріям динамічно отримувати доступ та оновлювати вміст, структуру та стиль документа».

Стандарт W3C DOM розділений на 3 різні частини:

- стандартна модель для всіх типів документів – основний DOM;
- стандартна модель для XML-документів – XML DOM;
- стандартна модель для документів HTML – HTML DOM.

Серед основних переваг технології обробки дерева DOM можна виділити велику гнучкість та доступність за рахунок того, що за допомогою скрипта DOM-парсингу можна досягти будь-якого вебсайту у відкритому доступі мережі Інтернет, що робить його доступним для подальшого автоматизованого вебскрейпінгу. Більш того, завдяки вільному та автоматичному пересуванню по вебсайту, при парсингу дерева DOM зникають обмеження на отримання даних, що цікавлять, на відміну від API методу, де обсяг даних, що надаються, є визначеним заздалегідь без можливості його зміни звичайним користувачем.

Загальний алгоритм обробки та отримання даних з вебресурсу є доволі лаконічним, але в той же час і гнучким, за рахунок вільної модифікації скрипту для парсингу DOM-контента вебсторінок.

Базовий алгоритм скрипта DOM parsing:

Крок 1. Отримання DOM-дерева вебсторінки: перший крок – отримати вебсторінку за допомогою HTTP-запитів. Після отримання вебсторінки його вміст HTML розбирається інструментом або сценарієм вебскрейпінгу.

Крок 2. Навігація по DOM: після того, як дерево DOM буде побудовано, інструмент для вебскрейпінгу може орієнтуватися по дереву, щоб знайти конкретні елементи, що містять необхідні дані. Зазвичай це робиться за допомогою селекторів CSS, XPATH або інших методів націлювання на елементи на основі їх атрибутів, імен класів або ієрархії в межах DOM.

Крок 3. Вилучення даних: після того, як потрібні елементи, отримані в форматі елементів дерева DOM, скрипт витягує відповідні дані з цих елементів у форматі, зрозумілому БД. Це може включати отримання вмісту тексту, атрибутів чи іншої інформації, пов’язаної з вибраними елементами.

Крок 4. Обробка та зберігання даних: після вилучення даних з вебсторінки, скрейпінговий скрипт може обробити його далі (наприклад, очистити або форматувати) перед тим, як зберігати його у структурованому форматі, до бази даних, електронної таблиці або JSON файлі.

Приклад алгоритму зображено на рисунку 1.7.

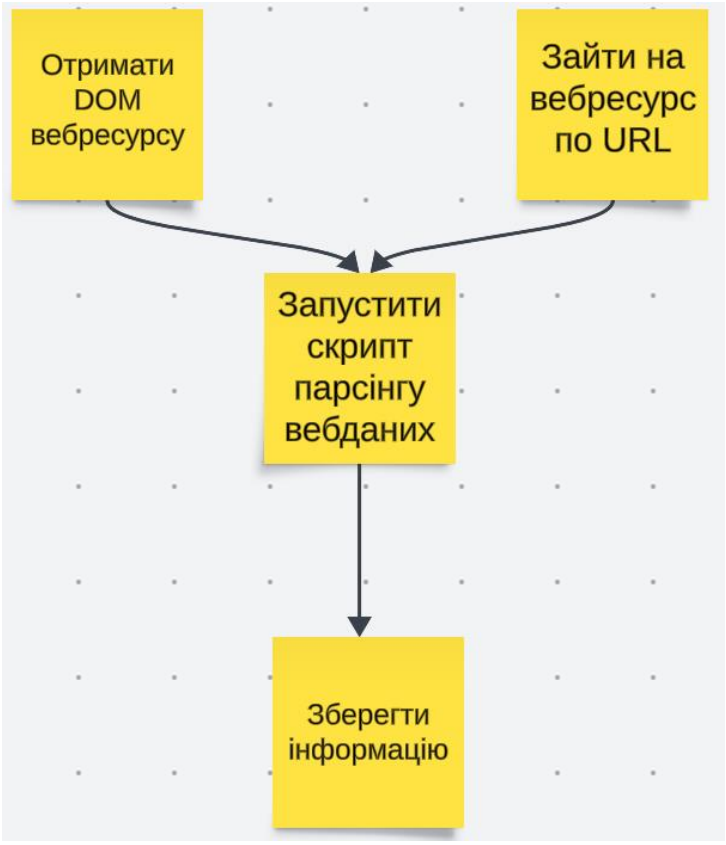

Рисунок 1.7 – Алгоритм отримання даних через парсинг дерева DOM

1.3 Постановка задачі

Таким чином, створення ресурсу з перегляду товарних пропозицій музикальних інструментів у вільному доступі є актуальним. Тому ставиться завдання розробки такої системи на базі алгоритму вебскрейпінгу дерева DOM, побудова надійної та сучасної архітектури застосунку, створення гнучкого та стабільного backend API для збору та обробки даних, розробка frontend застосунку для відображення обробленої інформації.

Об’єктом роботи є питання об’єднання та порівняння товарних пропозицій.

Метою роботи є розробка вебплатформи для об’єднання товарних пропозицій від онлайн-крамниць музичних інструментів за допомоги методу вебскрейпінгу для збирання цінних даних про музичні інструменти.

Для досягнення мети необхідно вирішити такі завдання:

– провести аналіз існуючих онлайн-крамниць музичних інструментів;

– розробити та реалізувати сучасну та надійну архітектуру системи;

– розробити та реалізувати базу даних для системи;

– розробити та реалізувати backend модуль для обробки та збору даних;

– розробити та реалізувати frontend модуль для відображення даних;

– проаналізувати та розв’язати проблему product matching [13].

2 ПРОЄКТУВАННЯ ВЕБЗАСТОСУНКУ ДЛЯ ПЕРЕГЛЯДУ ТОВАРНИХ ПРОПОЗИЦІЙ

2.1 Специфікація вимог до застосунку

Застосунок для об’єднання товарних пропозицій від онлайн-крамниць музичних інструментів – це система, яка дозволяє збирати, зберігати та відображати інформацію про музичні інструменти багатьох категорій, що продаються в різних онлайн-крамницях України. Система складається з трьох основних модулів – застосунок для автоматичного парсингу даних про товарні пропозиції, вебплатформи для перегляду товарів доступних в різних онлайн-крамницях, API для отримання та можливої обробки даних з БД.

Функціональні вимоги до парсеру:

– повинен мати підтримку різних онлайн-крамниць;

– збереження зібраних даних у MongoDB;

– повинен автоматично збирати дані;

– повинен вирішувати задачу product matching, визначати однакові товари з різних крамниць [13-15].

Функціональні вимоги до вебзастосунку:

– повинен забезпечувати зручний перегляд категорій музикальних інструментів;

– повинен мати зрозумілу класифікацію товарів;

– повинен мати фільтрацію товарів за їхніми властивостями;

– повинен відображати список різних крамниць для кожної товарної пропозиції;

– повинен відображати мінімальну та максимальну ціну при перегляді списку товарів з певної категорії;

– повинен мати пагінацію.

Функціональні вимоги до API:

– повинен мати запит на вилучення всіх товарів з певної категорії;

– повинен мати підтримку фільтрації в запиті;
– повинен мати запит на вилучення одного товару;
– повинен мати запит на вилучення всіх можливих брендів;
– повинен мати запит на вилучення всіх фільтрів.

2.2 Аналіз існуючих онлайн-крамниць музичних інструментів

Сьогодні в Україні є багато музичних магазинів, наприклад: GuiarHouse, Muzline, Jam, Muztorg, Leader, Muzdom. Крупнішими представниками онлайн-крамниць з продажу музичних інструментів можна назвати Muzline (рис. 2.1) та GuitarHouse (рис. 2.2).

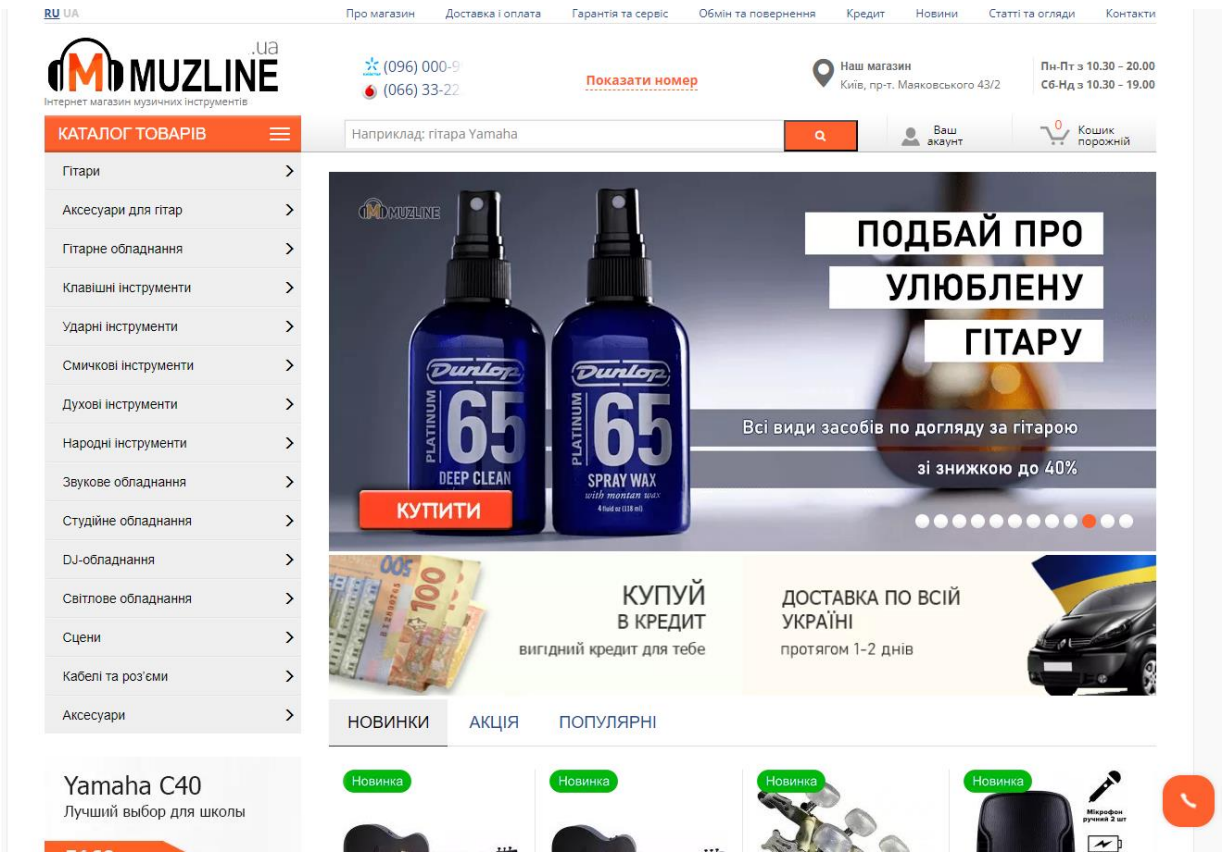

Рисунок 2.1 – Головна сторінка магазину Muzline

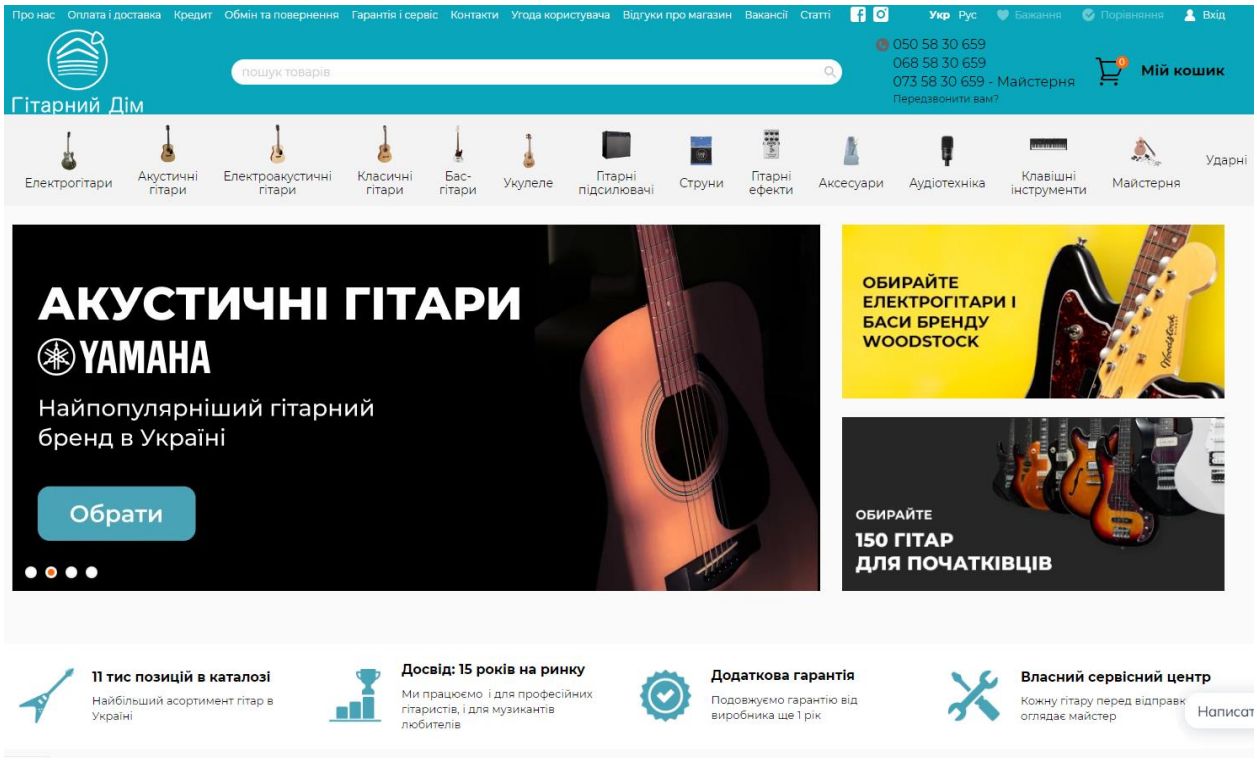

Рисунок 2.2 – Головна сторінка магазину GuitarHouse

Серед їх спільних особливостей можна виділити багатий асортимент саме гітар різного типу, наприклад, електрогітар, акустичних гітар, класичних гітар, бас-гітар, тощо. Незважаючи на їхню популярність, ціни на однакові товарні пропозиції в обох магазинах можуть бути різними, а стан наявності неочікувано змінюватись. Саме тому ці два магазини були взяті як основні торгівельні майданчики для збирання інформації та порівняння цін.

Розглянемо детально кожен з магазинів. Muzline має каталог багатьох категорій, який завжди присутній в лівій верхній частині сайту (рис. 2.3). Кожна з категорій містить в свою чергу декілька груп товарів (рис. 2.4). Фільтрація здійснена за допомогою елементів input, кожен з яких в свою чергу має власний ідентифікатор, по котрому можна визначити, за яку характеристику товару цей фільтр відповідає.

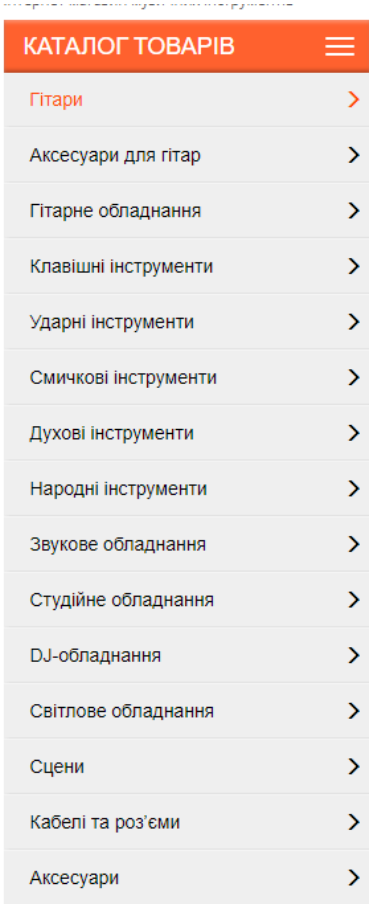

Рисунок 2.3 – Каталог категорій товарів магазину Muzline

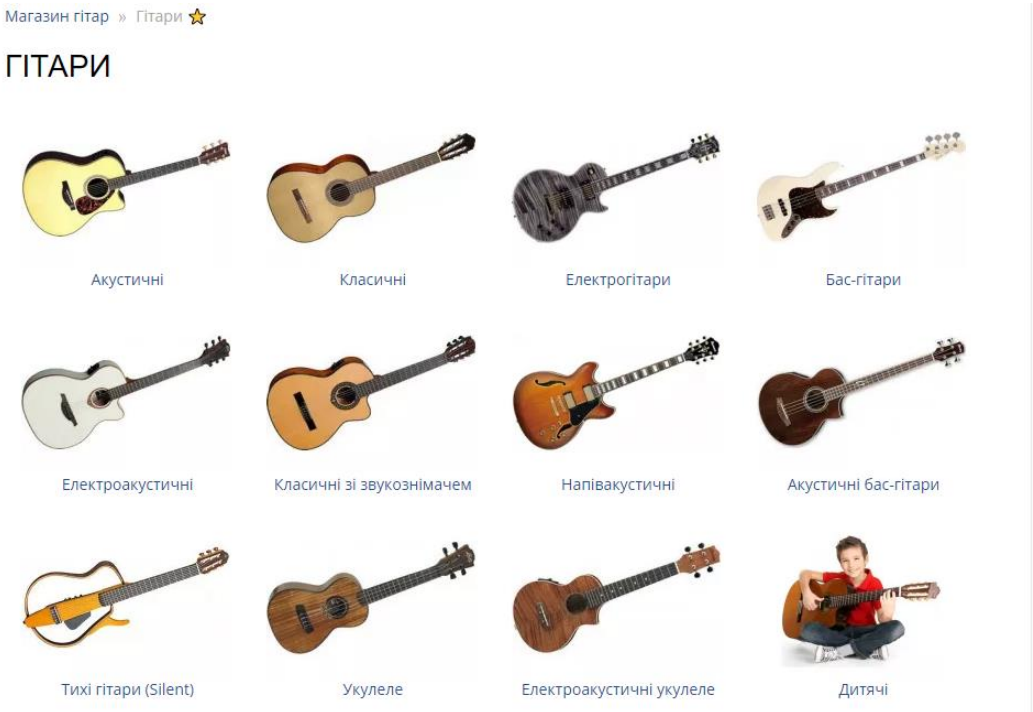

Рисунок 2.4 – Групи категорії «Гітари» магазину Muzline

Якщо перейти на сторінку групи товарів, то можна побачити список товарів цієї категорії. Також ліворуч розташований список всіх наявних фільтрів для певної групи товарів (рис. 2.5).

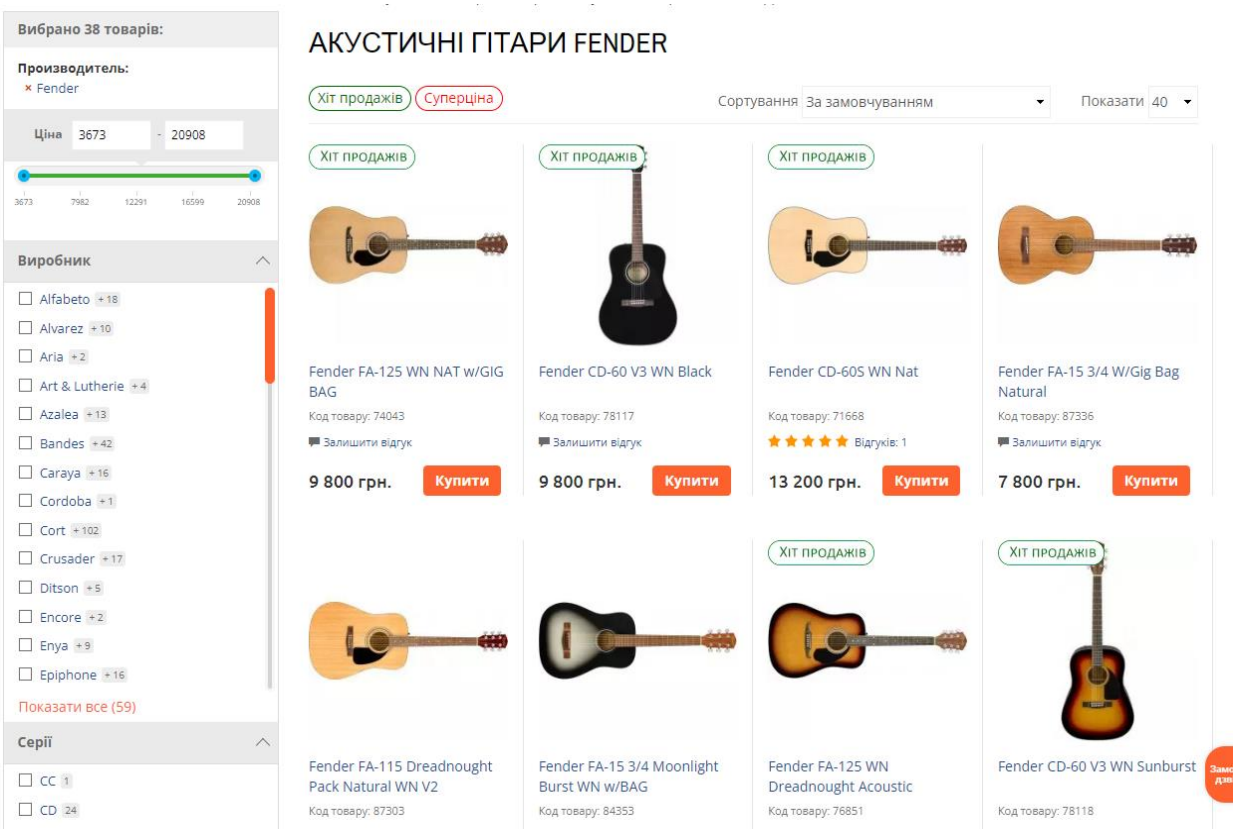

Рисунок 2.5 – Приклад списку товарів та фільтрів магазину Muzline

GuitarHouse містить всі можливі категорії товарів під хедером (рис. 2.6).

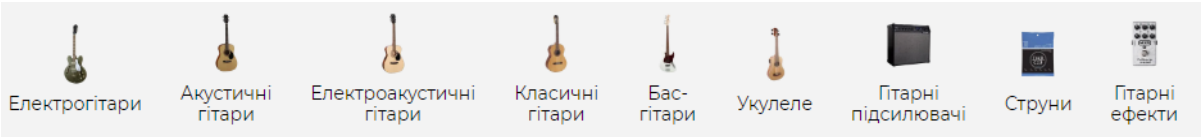

Рисунок 2.6 – Каталог категорій товарів магазину GuitarHouse

У порівнянні з магазином Muzline, GuitarHouse одразу має певні групи товарів відображеними в каталозі категорії в хедері. Приклад списку товарів та фільтрації в магазині GuitarHouse показаний на рисунку 2.7. Фільтрація

здійснена за допомоги елементів input з типом checkbox, кожен елемент має власний ідентифікатор.

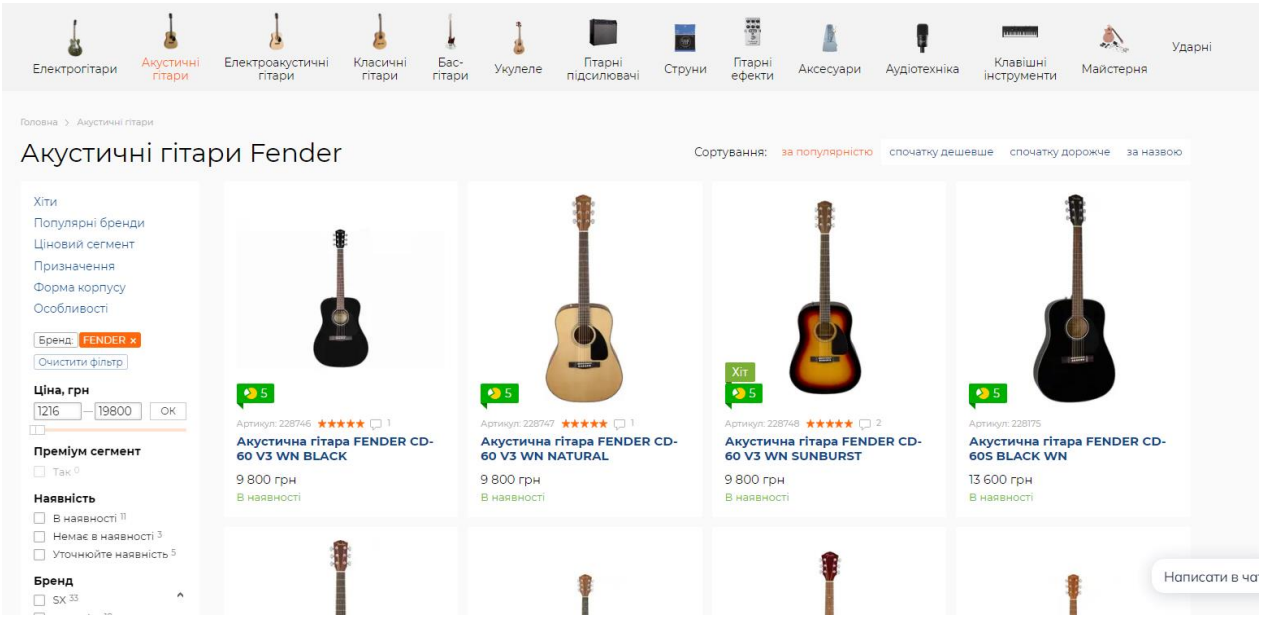

Рисунок 2.7 – Приклад списку товарів та фільтрів магазину GuitarHouse

Приклад однакових товарів з кожного з магазинів – Muzline та GuitarHouse відповідно – приведено на рисунку 2.8 та на рисунку 2.9.

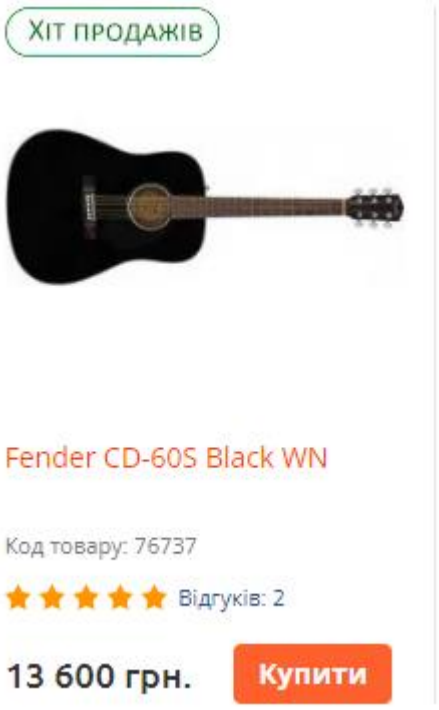

Рисунок 2.8 – Приклад товару з магазину Muzline

Рисунок 2.9 – Приклад товару з магазину GuitarHouse

Як можна побачити з рисунків 2.8 та 2.9, одна й та сама модель гітари відрізняється у відображенні на двух різних онлайн-крамницях: різна назва та різне зображення товару. У визначені однакових товарі з урахуванням можливих відмінностей полягає задача product matching.

2.3 Проєктування архітектурі застосунку

Вебзастосунок – це система, яка складається з декількох компонентів та має клієнт-серверну архітектуру [16]. База клієнту – новітній фронтенд фреймворк Svelte. Стилізація виконувалася за допомоги CSS фреймворку Tailwind CSS. Сервер написаний на базі фреймворку SvelteKit. В якості бази даних було обрано нереляційну БД MongoDB Cloud, яка розгорнута в хмарі. Парсер створено на основі Node.js бібліотеки Puppeteer.

Архітектура системи поділяється на три основні компоненти, кожен з яких виконує свої специфічні функції та взаємодіє з іншими компонентами через базу даних MongoDB та API (рис. 2.10).

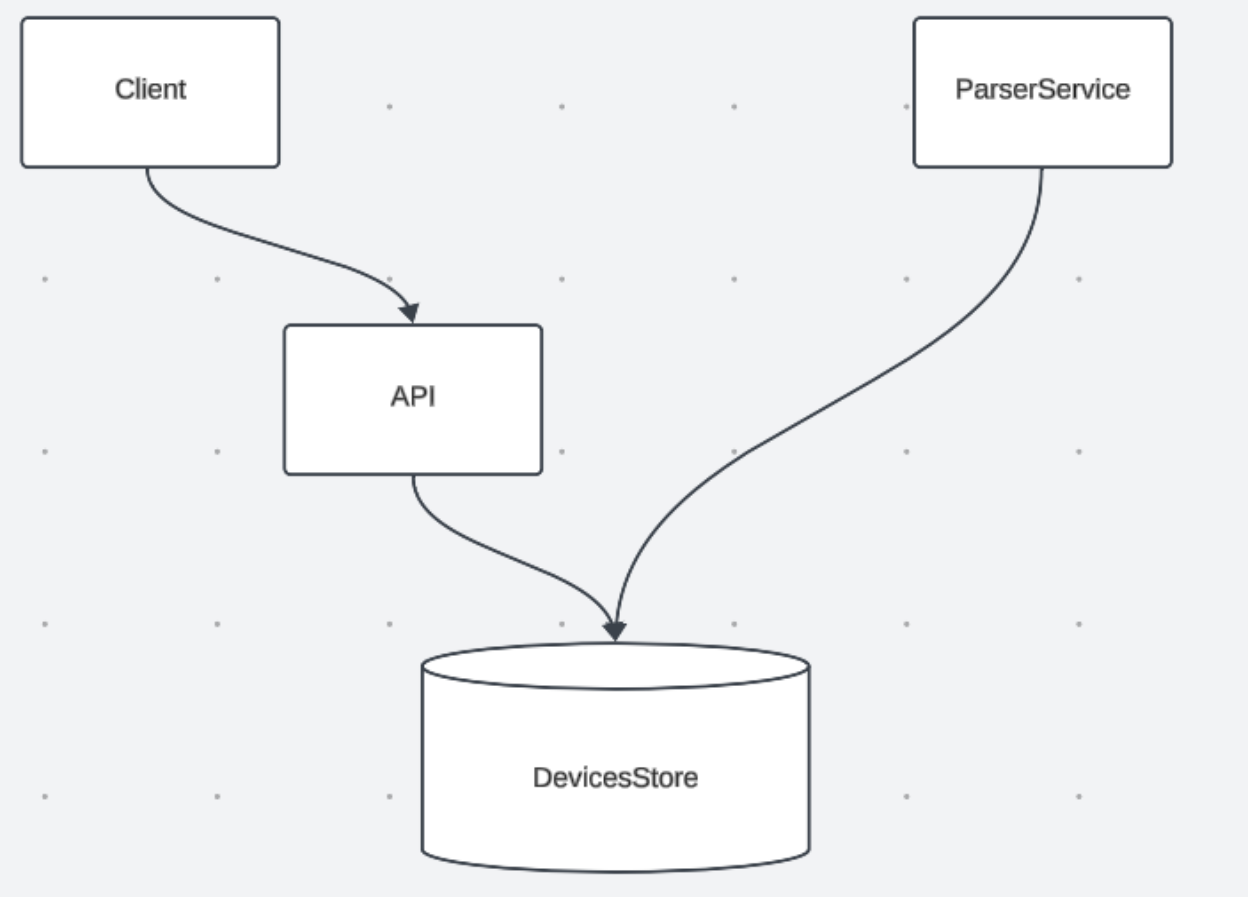

Рисунок 2.10 – Діаграма архітектури застосунку

Кожен компонент розроблений із використанням певного набору технологій, які забезпечують його ефективну роботу та інтеграцію в загальну систему.

Технології ParserService:

– Node.js: серверна платформа, що дозволяє виконувати JavaScript-код на сервері. Використовується для розробки високопродуктивних мережевих застосунків;

– MongoDB: NoSQL база даних, яка забезпечує зберігання зібраних даних у вигляді документів. Вибрана за її гнучкість і масштабованість;

– Puppeteer: бібліотека для автоматизації роботи з вебсторінками. Використовується для збору даних з онлайн-крамниць.

Технології Client:

– SvelteKit: сучасний фреймворк для створення високопродуктивних вебзастосунків. Забезпечує швидку розробку та високу продуктивність інтерфейсу;

– MongoDB: використовується для зберігання даних про товари, забезпечуючи швидкий доступ до інформації;

– Tailwind CSS: утилітарний CSS-фреймворк для створення адаптивного та привабливого інтерфейсу користувача;

– TypeScript: надбудова над JavaScript, що додає статичну типізацію. Покращує якість коду та зменшує кількість помилок.

Технології API:

– SvelteKit: використовується для розробки API, забезпечуючи високу продуктивність та зручність у розробці;

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

– TypeScript: використовується для забезпечення високої якості та надійності коду API.

Всі компоненти системи інтегруються через базу даних MongoDB та API. Парсинг-застосунок збирає та зберігає дані у MongoDB, вебзастосунок використовує ці дані для відображення користувачам, а API надає доступ до даних для зовнішніх сервісів. Така архітектура забезпечує високу гнучкість, масштабованість та ефективність системи.

2.4 Модуль парсингу товарних пропозицій

Модуль парсингу товарниї пропозицій – це модуль, який відповідає за навігацію по заданим вебсайтам, обробку та збереження даних до БД. Також

цей модуль вирішує проблему product matching шляхом розбиття назви кожного товару на окремі слова та перетворення їх до загальної структури. Після чого набір окремих слів порівнюється. Модуль використовує БД Devices.

Архітектурні особливості:

– збір даних: парсинг-застосунок збирає дані з різних онлайн-крамниць, використовуючи Puppeteer для автоматизації процесу;

– обробка даних: зібрані дані обробляються для визначення однакових товарів (product matching), що дозволяє уникнути дублювання інформації [17-19];

– збереження даних: дані зберігаються у MongoDB, що забезпечує їхню доступність для інших компонентів системи.

2.5 Вирішення проблеми product matching

Product matching, або визначення однакових товарів, є критичною задачею у багатьох системах e-commerce. Це завдання полягає у виявленні однакових або подібних товарів, представлених на різних онлайн-платформах для їх подальшого об’єднання та використання інформації в різних цілях бізнес-логіки. В контексті вебплатформи для об’єднання товарних пропозицій від онлайн-крамниць музичних інструментів, ефективне вирішення цієї проблеми є надзвичайно важливим для забезпечення точності даних, що відображаються користувачам [20].

Проблема product matching виникає, коли один і той самий товар може мати різні назви, описи або інші атрибути на різних вебсайтах [21]. Наприклад, одна і та ж електрогітара може бути названа як «електрогітара Fender Stratocaster» на одному сайті і «Fender Stratocaster електро-гітара» на іншому. Основна задача полягає у виявленні цих відповідностей і злитті інформації про один товар в одну єдину запис.

Існує кілька підходів до вирішення проблеми product matching:

– текстовий на основі евристичного підходу;
– текстовий на основі порівняння ембедінгів;
– на основі порівняння візуального контенту;
– гібридні.

Текстові методи на основі евристичного підходу – це такі методи, які порівнюють будь-яку текстову інформацію на основі спеціалізованого алгоритму в контексті певної задачі. Як правило, такі алгоритми може видати правильний результат в певній галузі, але не гарантує найкращого результати загалом. Суть алгоритму полягає у різноманітних методах порівнянь символів, слів, комбінацій.

Текстові методи на основі порівняння ембедінгів – це такі методи, що використовують ембедінги для порівняння текстової інформації [22]. Ембедінги – це векторне чисельне уявлення тексту для його подальшої обробки комп’ютером [23].

Методи на основі порівняння візуального контенту – це така група методів, що визначає схожість двох зображень на основі обчислювання ембедінгів цих зображень [24].

Гібридні методі – це поєднання всіх або декількох методів для визначення більш детального результату. Наприклад, за допомоги коефіцієнту, що визначатиме вагу кожного з методу складової гібридного підходу.

Для того, щоб визначити метод, який найефективніше вирішуватиме задачу product matching, потрібно протестувати кожен з методів на прикладі реальних даних та порівняємо результат, порахувати середнє успішних відповідей для кожного з методів та обрати той, в якому доля буде найвищою. Тестування відбуватиметься серед порівняння трьох алгоритмів – текстового на основі Bert-моделі, текстового на основі Tensorflow та евристичного. Метод на основі візуального не підходить в силу особливостей архітектури. Результати тестування наведені у таблицях 2.1 та 2.2.

Таблиця 2.1 – Результати тестування різних методів product matching на виборці однакових товарів

Назва 1 товару	**Назва 2 товару**	**Bert-модель**	**Tensorflow**	**Евристичний алгоритм**
SQUIER by FENDER BULLET STRATOCASTER TREM BSB Електрогітара	ЕЛЕКТРОГІТАРА FENDER SQUIER BULLET STRATOCASTER TREM BSB	0,94	0,92	1
Електрогітара Ibanez AZES40L-PRB	Ibanez AZES40L-PRB	0,83	0,91	1
Бас-гітара CORT Action V Plus (Black)	Cort Action V Plus BK	0,85	0,71	0,8
Гітарний комбопідсилювач Orange Crush Mini	Orange Crush Mini	0,87	0,88	1
MIDI клавіатура M-Audio Keystation 49 MK3	M-Audio Keystation 49 MK3	0,92	0,88	1
Педаль ефектів TC Electronic HOF Mini Reverb	TC Electronic HOF Mini Reverb	0,9	0,94	1
Педаль ефекту Electro-harmonix Chillswitch	Electro-Harmonix Chillswitch	0,92	0,55	1
Електроакустична гітара YAMAHA FSX315C (Tobacco Brown Sunburst)	Yamaha FSX315C (Tobacco Brown Sunburst)	0,87	0,95	1
Укулеле Parksons UK21C	Parksons UK21C	0,92	0,79	1
Акустична гітара FENDER CD-60 V3 WN BLACK	Fender CD-60 V3 WN Black	0,87	0,91	1

Як можна бачити з таблиці 2.1, результати у всіх трьох методів непогані.

Таблиця 2.2 – Результати тестування різних методів product matching на виборці різних товарів

Назва 1 товару	**Назва 2 товару**	**Bert-модель**	**Tensorflow**	**Евристичний алгоритм**
SQUIER by FENDER BULLET STRATOCASTER TREM BSB Електрогітара	Ibanez AZES40L-PRB	0,8	0,55	0
Педаль ефектів TC Electronic HOF Mini Reverb	Бас-гітара CORT Action V Plus (Black)	0,87	0,25	0
MIDI клавіатура IK Multimedia iRIG Keys 2	IK Multimedia iRig Keys 2 Mini	0,92	0,76	0,8
Orange Crush 20	Orange Crush 12	0,98	0,79	0,66
Акустична гітара EPIPHONE DR-100 NT	Epiphone DR-100 VSB	0,83	0,67	0,75
J8 ST HMG 101 SB	J8 ST HMG 101 WH	0,92	0,68	0,6
Електрогітара Fender Player Telecaster MN 3TS	Fender Player Telecaster PF PWT	0,85	0,83	0,2
Fender Player Telecaster Left Handed MN 3TS	Fender Player Telecaster MN 3TS	0,95	0,83	0,71
Parksons SPB-140 TBLB	Parksons SPB-140 3TSB	0,96	0,77	0,25
Бас-гітара CORT Action V Plus (Black)	Cort Action PJ OPB	0,84	0,46	0,4

Як можна бачити з таблиці 2.2, Bert модель показує незадовільний результат у контексті задачі визначення відмінних назв.

Середній процент успіху кожного з методів наведений у таблиці 2.3.

Таблиця 2.3 – Середнє значення успішних порівнянь

Назва методу	**Однакові**	**Різні**	**Загалом**
Bert-модель	0,889	0,108	0,4985
Tensorflow	0,844	0,341	0,5925
Евристичний алгоритм	0,98	0,563	0,7715

Як можна бачити з таблиці 2.3, по результатам тестування евристичний алгоритм має найвищі показники.

В якості вирішення проблемі product matching для цієї системи було обрано текстовий метод на основі евристичного підходу з використанням спеціального алгоритму для порівняння товарних назв. В процесі тестування методу машинного навчання на класифікації та порівнянні текстової та графічної інформації відомі бібліотеки для JS такі як Tensorflow.js, Bert-модель не показали високоефективного результату у порівнянні з евристичним методом, який ідеально підходить для задач класифікації у невеликому та вузькоспеціалізованому масштабі [25-28].

Одним з основних етапів алгоритму є усунення «зайвих» слів, які не впливають на сутність товару, але можуть змінити результати порівняння [29]. Наприклад, слова типу «електро», «акустична», «бас» тощо часто зустрічаються у назвах музичних інструментів, але не є ключовими для ідентифікації товару.

Опис алгоритму:

Крок 1. Декомпозиція рядка. Функція decomposeString приймає рядок, приводить його до нижнього регістру і розділяє на слова, видаляючи зайві символи та слова з переліку redundantKeyWords.

Крок 2. Фільтрація та сортування. Залишкові слова фільтруються і сортуються для подальшого порівняння.

Крок 3. Порівняння товарів. Функція productMatch порівнює два товарні найменування, використовуючи decomposeString, і визначає, чи вони є однаковими.

Запропонований підхід до вирішення проблеми product matching базується на символьному методі порівняння рядків, зокрема, фільтрації зайвих слів та символів. Це дозволяє ефективно ідентифікувати однакові товари, представлені на різних онлайн-крамницях, і зливати їх в одну єдину запис. Такий підхід забезпечує точність даних, що відображаються користувачам, і сприяє покращенню користувацького досвіду на вебплатформі для об’єднання товарних пропозицій від онлайн-крамниць музичних інструментів.

2.6 Модуль клієнтського вебзастосунку

Модуль клієнтського вебзастосунку – це модуль, який відповідає за відображення даних, отриманих в процесі вебскрейпінгу. Модуль використовує БД Devices.

Архітектурні особливості:

– інтерфейс користувача: вебзастосунок забезпечує зручний та інтуїтивно зрозумілий інтерфейс для перегляду категорій музичних інструментів та їхніх характеристик;

– класифікація та фільтрація: реалізована чітка класифікація товарів та можливість фільтрації за різними параметрами, що полегшує користувачам пошук потрібного товару;

– відображення пропозицій: кожен товар відображається з інформацією про ціни та наявність у різних онлайн-крамницях, що дозволяє користувачам обрати найвигіднішу пропозицію;

– пагінація: для зручності перегляду великих списків товарів реалізована пагінація.

2.7 Модуль API

Модуль API – це модуль, який відповідає за процес отримання та обробку даних з БД.

Архітектурні особливості:

– взаємодія з даними: забезпечений доступ API до даних, збережених у MongoDB, дозволяє зовнішнім застосункам отримувати та перетворювати інформацію;

– запити: реалізовані запити для отримання списків товарів, фільтрів, брендів та конкретних товарів, що забезпечує гнучкість у використанні даних;

– фільтрація: підтримка фільтрації даних на рівні API дозволяє оптимізувати процеси пошуку та відбору товарів.

2.8 Розробка дизайну вебзастосунку

Дизайн застосунку повинен бути простий та зрозумілий. Функції пошуку певних товарів повинні бути полегшені завдяки категорізації товарів, відображенню категорій на головній сторінці та у зрозумілій формі, наприклад, завдяки іконкам. З приводу UI частини дизайну, було обрано мінімалістичний стиль з використанням чорно-білого та зеленого кольорів.

Головна сторінка застосунку має демонструвати основні категорії товарів, що існують. Одразу після категорій, повинен відображатись список детальніших класів товарів (рис. 2.11).

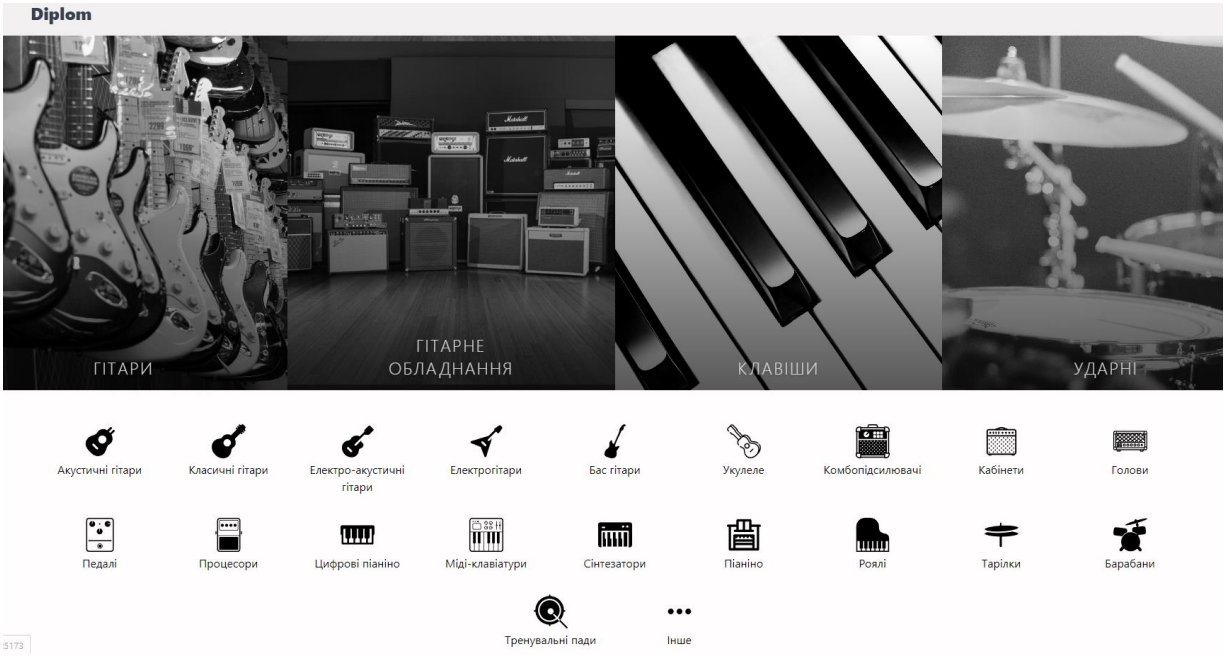

Рисунок 2.11 – Головна сторінка застосунку

Кожна категорія зі списку основних містить в собі посилання на сторінку з переліком підгруп музичних інструментів (рис. 2.12). Кожна підгрупа відображається завдяки svg-іконок для комфортного сприйняття інформації.

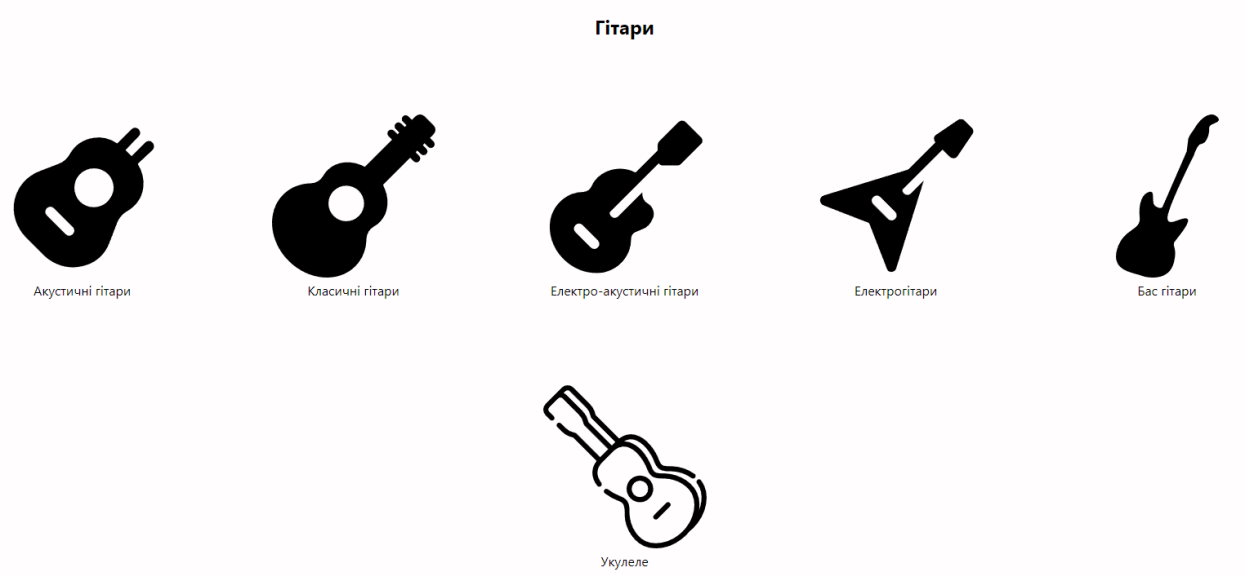

Рисунок 2.12 – Сторінка підгруп категорії

На сторінці підгрупи міститься список всіх музичних інструментів з певної групи, фільтри та пагінація (рис. 2.13).

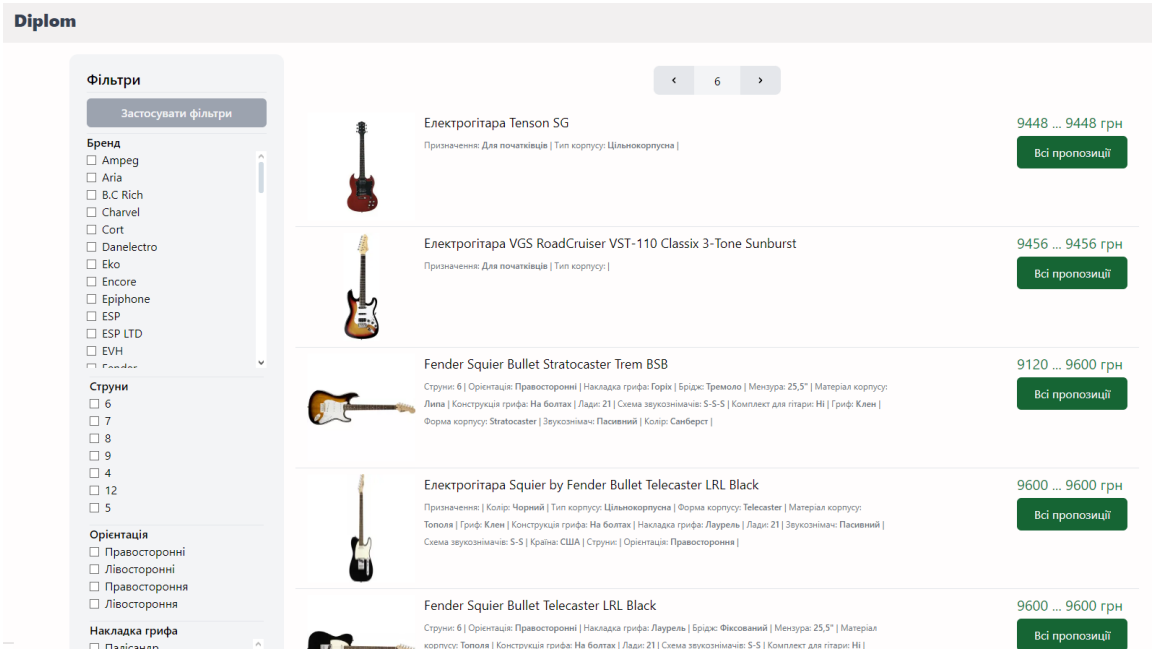

Рисунок 2.13 – Приклад списку товарів в категорії Гітари/Електрогітари

В кожній ділянці товару всередині списку товарів повинна чітко та помітно відображатись основна інформація про інструмент – назва, зображення, ціна. Маленьким текстом потрібно розташувати всі характеристики, щоб не перенавантажувати зір (рис. 2.14).

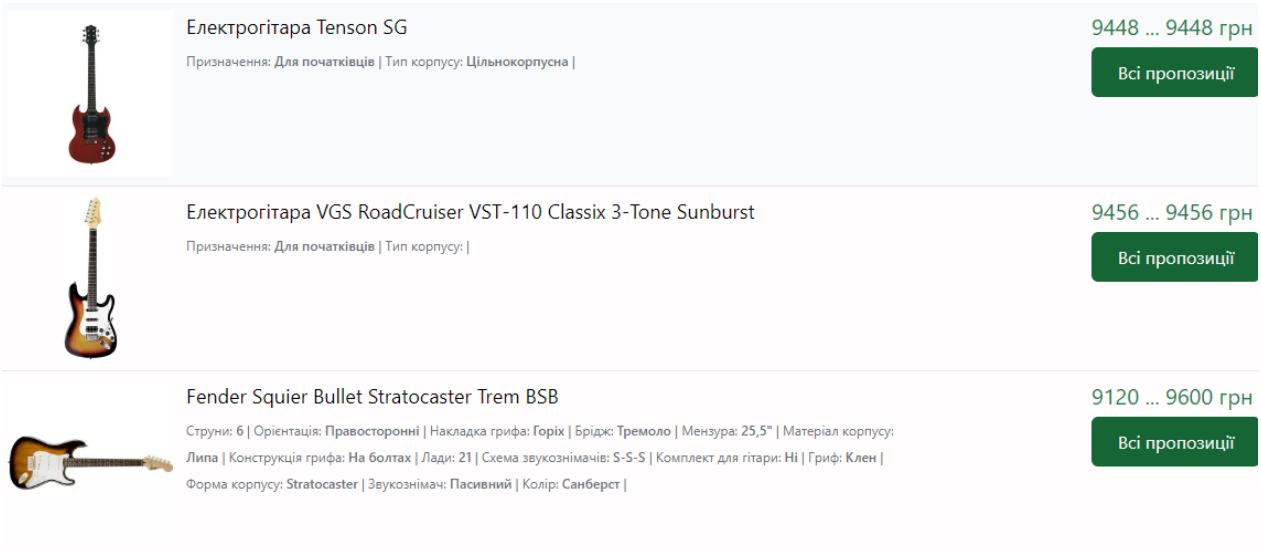

Рисунок 2.14 – Список товарів

Для зручності пошуку товарів, потрібно розмістити блок фільтрації зліва від списку товарів. Щоб процес фільтрації був зручнішим, потрібно додати кнопку застосування фільтрів, якщо вони були обрані (рис. 2.15).

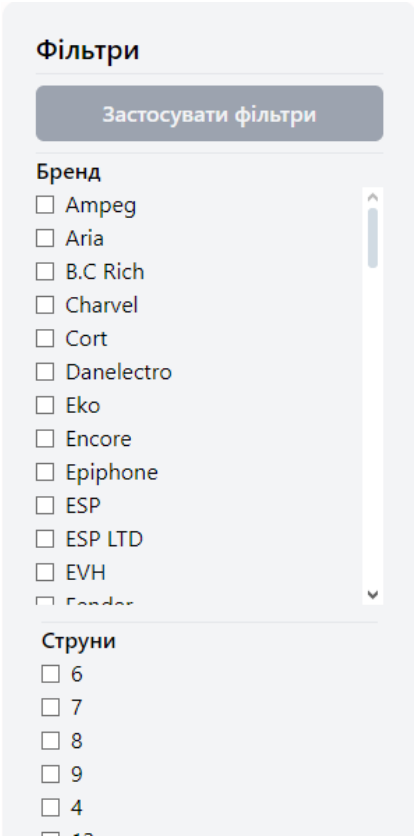

Рисунок 2.15 – Блок з фільтрами

В цілях зручності перегляду та зменшення часу завантаження даних, потрібно додати блок пагінації (рис. 2.16).

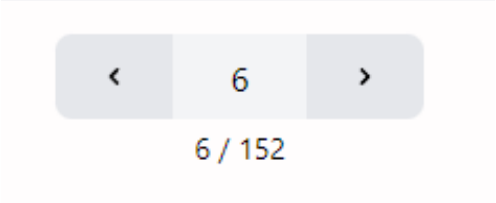

Рисунок 2.16 – Блок пагінації

Зі сторінки списку товарів має бути можливість перейти до сторінки кожного товару окремо (рис. 2.17).

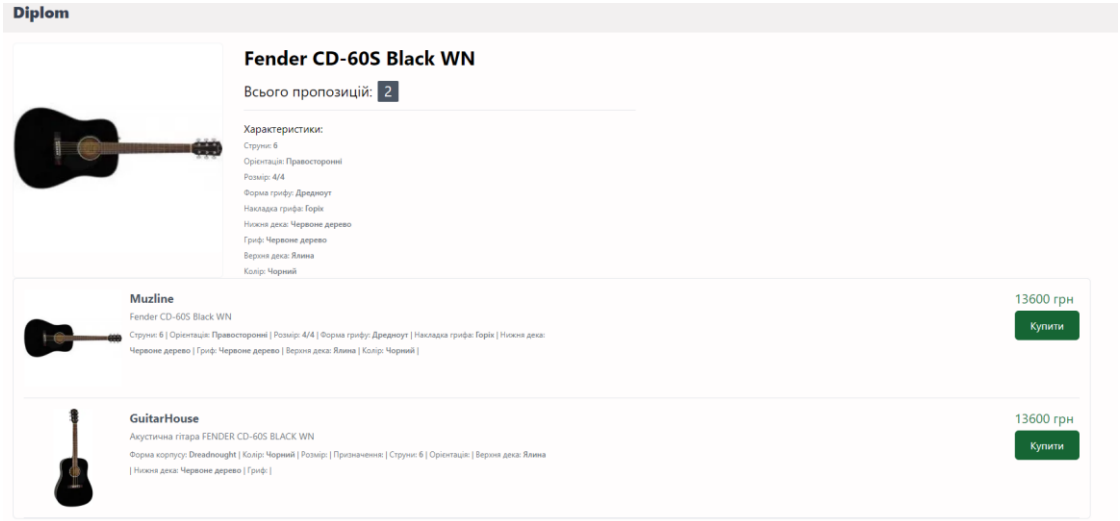

Рисунок 2.17 – Сторінка товару

На сторінці товару користувач може переглянути детальні характеристики інструменту, подивитись кількість пропозицій (рис. 2.18), доступних по цьому товару, переглянути список магазинів, в яких ця пропозиція існує з можливістю перейти до бажаної онлайн-крамниці (рис. 2.19).

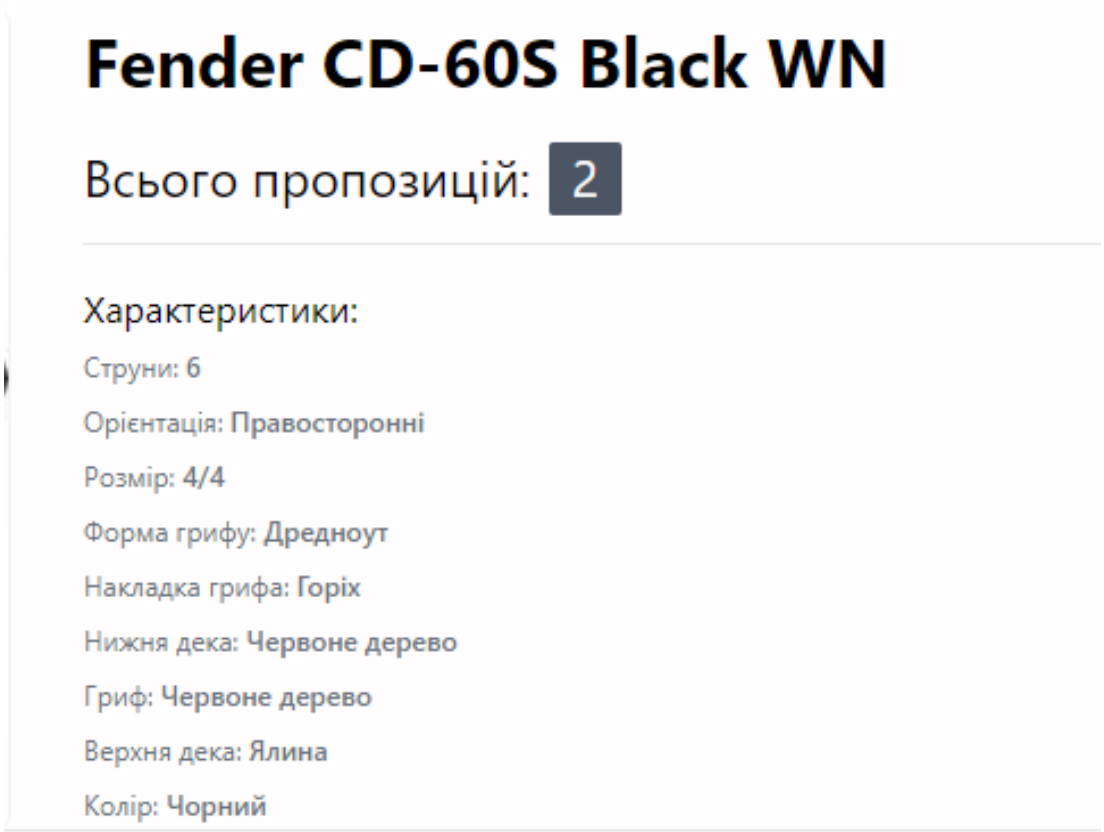

Рисунок 2.18 – Характеристика товару, кількість пропозицій

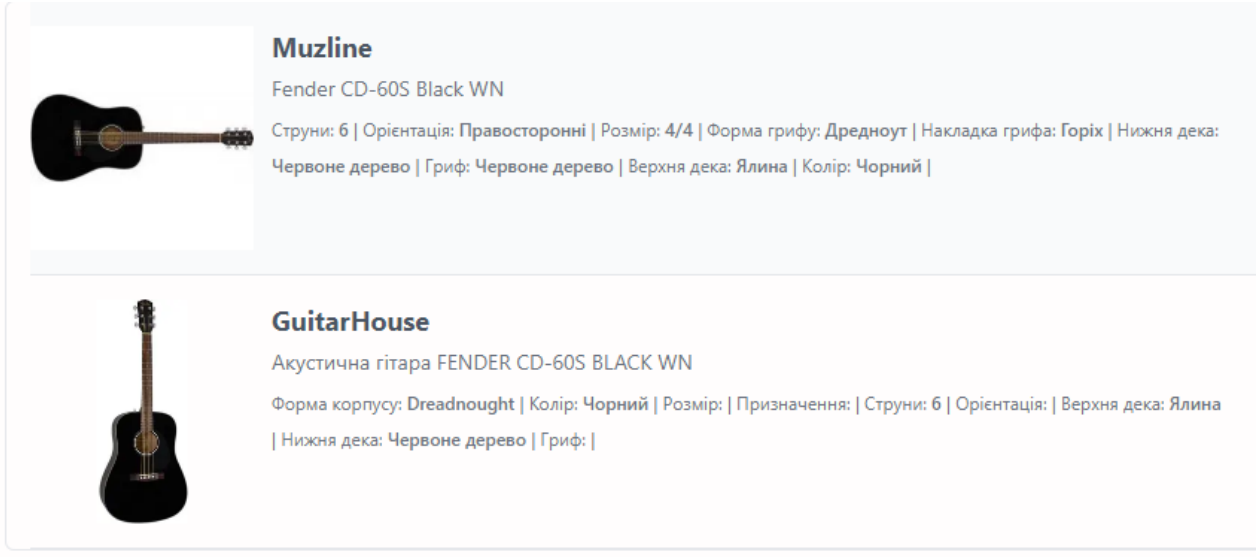

Рисунок 2.19 – Список магазинів з певним товаром

Футер має містити в собі узагальнену інформацію вебплатформи – основні категорії з можливістю перейти за посиланням на сторінку кожної з них, контактна інформація (рис. 2.20).

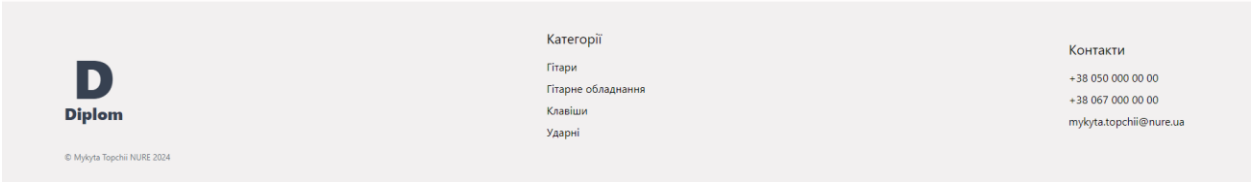

Рисунок 2.20 – Футер

3 РОЗРОБКА ВЕБЗАСТОСУНКУ ДЛЯ ПЕРЕГЛЯДУ ТОВАРНИХ ПРОПОЗИЦІЙ

3.1 Налаштування програмного забезпечення

Всі модулі застосунку побудовані на базі Node.js, тому для розробки програмного забезпечення та написання коду потребується середовище для веброзробки, наприклад JetBrains WebStorm. Також для розробки вебзастосунку та встановлення бібліотек, необхідно завантажити Node.js та npm на персональний комп'ютер Для розробки сервісу парсинга вебданих потрібно встановити бібліотеки Puppeteer (для виконання процедури вебскрейпінгу), mongodb (для спілкування з БД), dotenv (для підтримки змінних середовища). Для розробки клієнтського вебзастосунку потрібно встановити наступні залежності: sveltekit, vite, typescript, tailwind, svelte. Для розробки API модулю необхідні бібліотеки sveltekit, typescript, mongodb, vite.

В якості розподіленої системи управління версій було обрано сервіс GitHub. Деплоймент застосунку відбувається за допомоги хмарної вебплатформи Vercel, що підтримує велику кількість технологій, серед яких є Next.js, React, Angular, Vue, SvelteKit.

3.2 Середовища застосунку

У розробці програмного забезпечення середовища стосуються окремих налаштувань або конфігурацій, з якими застосунки стикаються протягом свого життєвого циклу [30]. Кожне середовище призначене для певної фази, ізольованої від інших, для підтримки різних етапів розробки, тестування та розгортання. Ці середовища гарантують належну обробку коду, конфігурації та даних програми для різних цілей.

Керуючи процесом розробки програмного забезпечення через ці середовища, розробники можуть підтримувати якість коду, перевіряти правильність роботи, забезпечувати безпеку та оптимізувати продуктивність на кожному етапі. Розділення середовищ дозволяє проводити ретельне тестування, окремі зміни та стабільний випуск для кінцевих користувачів.

Для стабільності системи, якості та безпеки даних, тестування нових функцій було створено два середовища розробки для кожного з модулів застосунку – test та prod. Таким чином, архітектура декількох середовищ мінімізує ризики того, що застосунок зламається при імплементації нових функцій або дані постраждають внаслідок будь-яких помилок.

Створимо два різні середовища для кожного з модулів. В якості середовища в MongoDB Atlas представлені проєкти. Приклад різних середовищ в модулі MongoDB Atlas показано на рисунку 3.1.

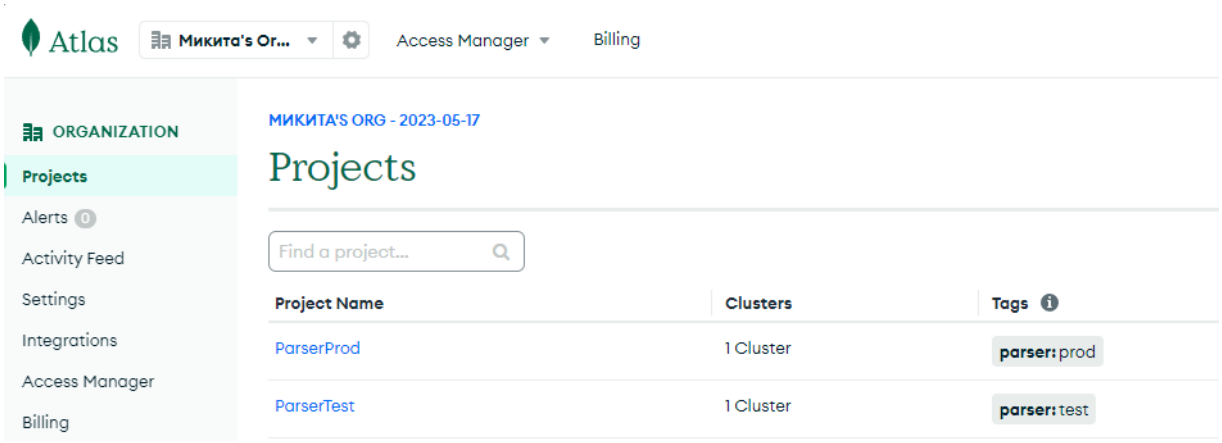

Рисунок 3.1 – Приклад різних середовищ у MongoDB Atlas

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.

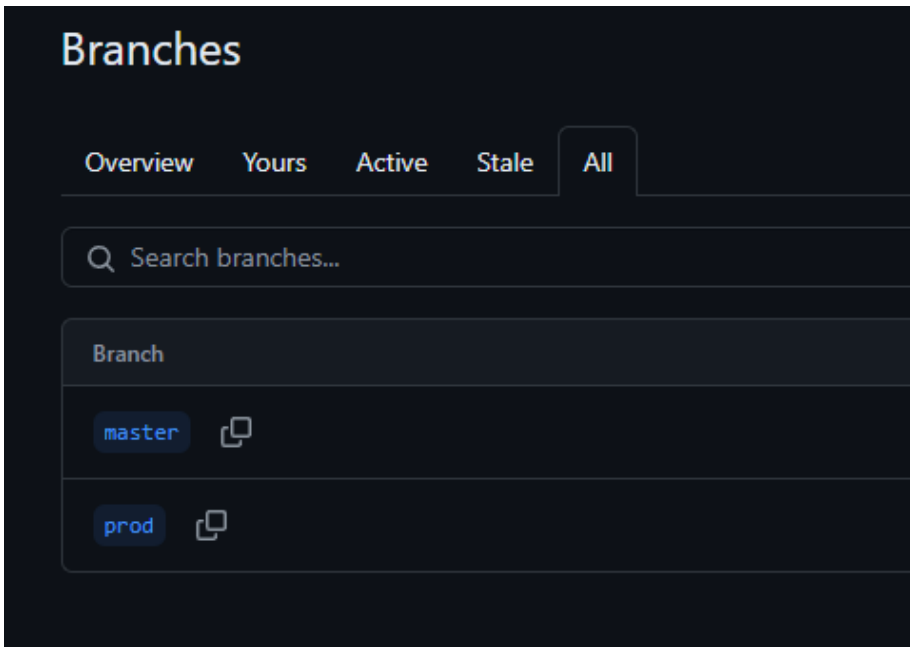

Рисунок 3.2 – Приклад різних середовищ у GitHub

3.3 Розробка парсеру вебданих товарів

Для того, щоб якісно та точно вилучити важливі дані з магазинів, потрібно визначити список магазинів, та створити конфігураційні файли для кожного з магазинів.

В якості джерела даних було обрано два найпопулярніших онлайн-магазини музичних інструментів – Muzline та GuitarHouse. Обидва з цих двох магазинів мають достатньо велику базу товарних пропозицій музичних інструментів та обладняння.

Структура проєкту складається з загальних файлів – допоміжні функції, конфігураційний файл підключення до БД, константи та файли парсингу й обробки даних кожного з онлайн-магазинів. В директорії кожного з магазинів містяться директорії з категоріями музичних товарів. Кожна директорія містить в собі 2 файли – конфігураційний файл та файл вебскрейпінгу (рис. 3.3). Загальна функція вебскрейпінгу по кожному магазину визначена у файлі з допоміжними функціями utils.js.

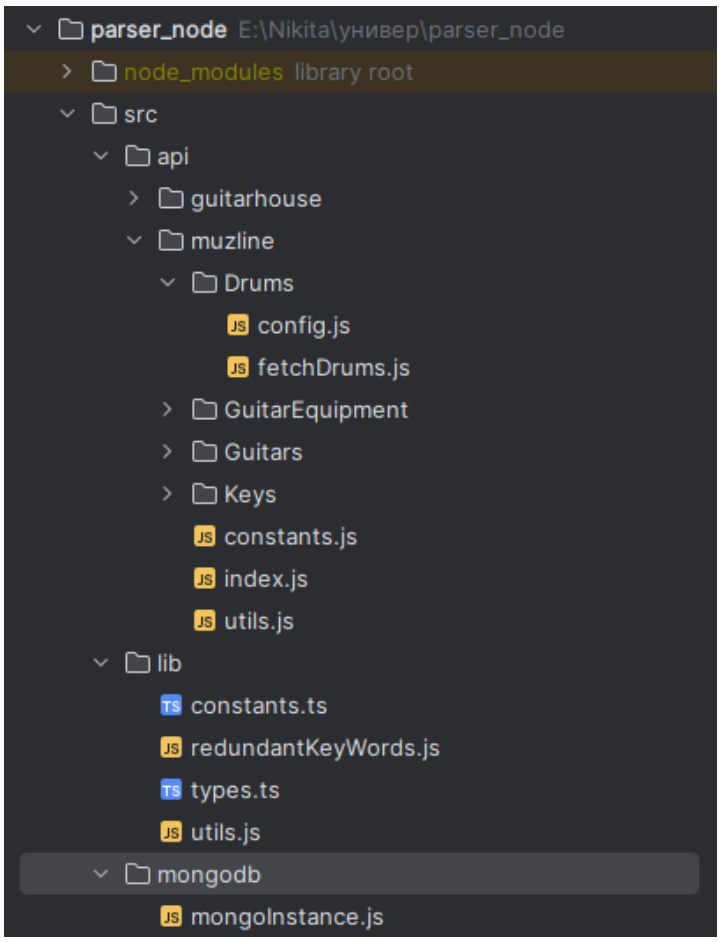

Рисунок 3.3 – Структура проєкту вебпарсера

Скрипти для автоматизованого процесу вебскрейпінгу було розроблено в пустому Node.js проєкті за допомоги безкоштовної Chromium бібліотеки Puppeteer.

Puppeteer – це високорівневий API для роботи з браузером Chrome або Chromium, що надається командою Google Chrome. Він дозволяє автоматизувати браузерні дії, використовуючи Node.js, і забезпечує повний контроль над браузером через протокол DevTools. Puppeteer використовується для різноманітних задач, таких як тестування вебзастосунків, збирання даних (скрейпінг), генерація PDF-документів і знімків екранів (скріншотів), а також для автоматизації рутинних браузерних дій [31].

Однією з ключових особливостей Puppeteer є його здатність повністю імітувати поведінку користувача в браузері. Це дозволяє виконувати складні

інтерактивні дії, такі як заповнення форм, натискання кнопок, навігація між сторінками та взаємодія з динамічним контентом. Puppeteer надає розробникам можливість писати сценарії, які можуть взаємодіяти з вебсторінками так, як це робив би реальний користувач. Це особливо корисно для тестування, оскільки дозволяє перевіряти функціональність і поведінку вебзастосунку в реальних умовах. Така категорія автоматичних тестів зветься end-to-end тестування.

Puppeteer також відзначається високою точністю у збиранні даних і генерації контенту. Ця безкоштовна бібліотека дозволяє робити знімки екранів з високою роздільною здатністю та створювати PDF-документи з вебсторінок, зберігаючи точне форматування та стиль. Це робить його ідеальним інструментом для створення автоматизованих звітів, документів та презентацій на основі вебконтенту. Крім того, Puppeteer може виконувати JavaScript-код безпосередньо у браузері, що дозволяє отримувати динамічні дані та взаємодіяти зі складними вебзастосунками.

Ще однією важливою особливістю Puppeteer є його інтеграція з інструментами для тестування, такими як Jest і Mocha. Це дозволяє розробникам легко включати Puppeteer у свої існуючі процеси тестування та автоматизації, забезпечуючи більш ефективне та надійне тестування вебзастосунків. Puppeteer також підтримує запуск тестів у безголовому режимі (headless mode), що дозволяє виконувати автоматизовані задачі на сервері без потреби у графічному інтерфейсі.

Як висновок, Puppeteer є потужним та гнучким інструментом для автоматизації браузера, який пропонує широкий спектр можливостей для тестування, збирання даних та генерації контенту. Завдяки своїй простоті використання та інтеграції з Node.js, Puppeteer стає все більш популярним серед розробників, які шукають надійні та ефективні інструменти для роботи з вебзастосунками.

Команда для встановлення бібліотеки через менеджер пакунків npm продемонстрована на лістингу 3.1.

Лістинг 3.1 Завантаження бібліотеки Puppeteer:

```
npm install puppeteer
```

Для того, щоб отримати вебдані, потрібно запустити внутрішній браузер, створити об’єкт вебсторінки, та задати посилання, по якому потрібно перейти.

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

Так як контролювати інформацію, яку надають власники тих або інших інтернет-магазинів неможливо, та структура даних є хаотичною, а кількість окремих категорій може змінюватися, ідеальним рішенням буде використовувати нереляційну базу даних MongoDB. Як первинний ключ, окрім внутрішнього ідентифікатора MongoDB _id буде виступати комбінація з артикулу товару та магазина, що пропонує товар. Також важливою інформацією для подальшого відображення будуть ціна, назва, картинка, наявність, бренд та характеристики товару. Передбачити характеристики неможливо, тому створимо об’єкт filters, що буде містити в собі всі можливі характеристики товару, всередені якого будуть поля filterName з назвою фільтру для подальшого відображення та масив зі строками values – значення

фільтру для певного товару на випадок, якщо один товар має водночас декілька властивостей певної опції (лістинг 3.3).

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).

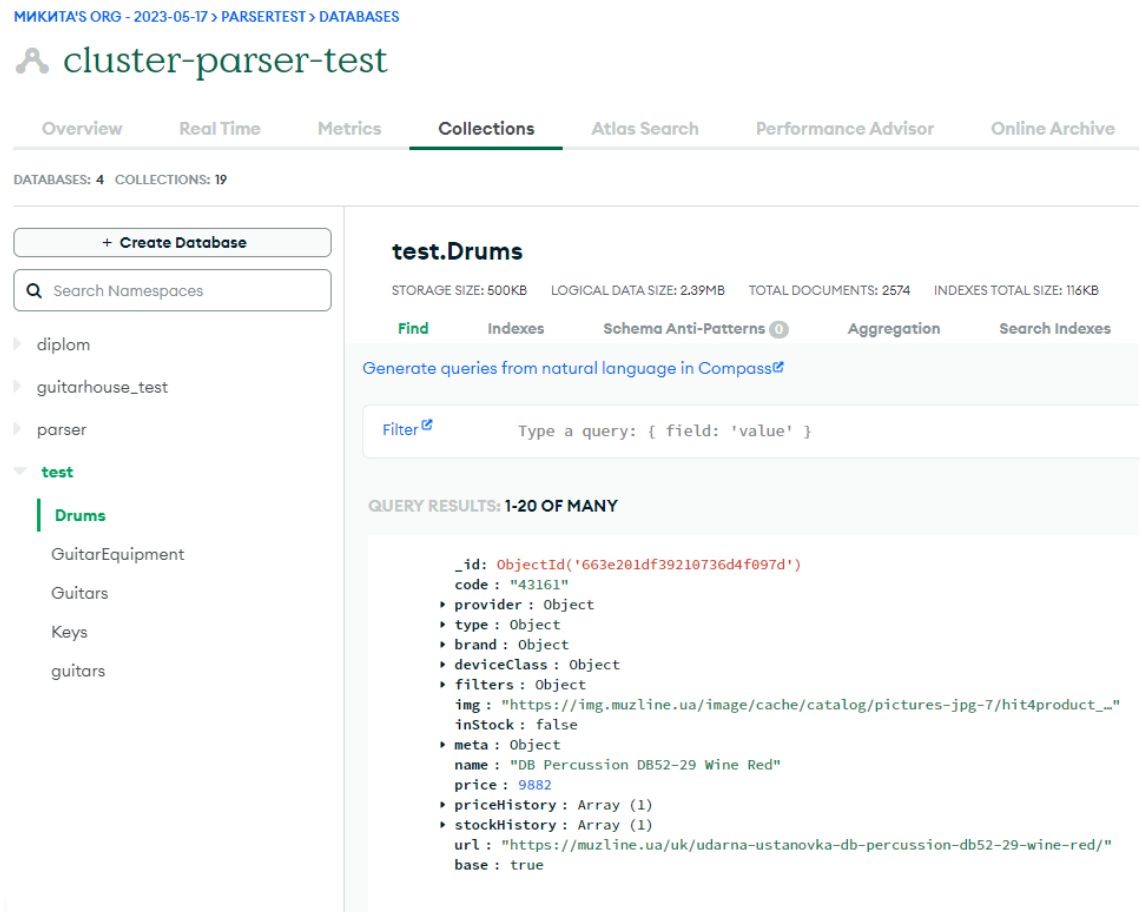

Рисунок 3.4 – Середовище MongoDB cloud – Databases view

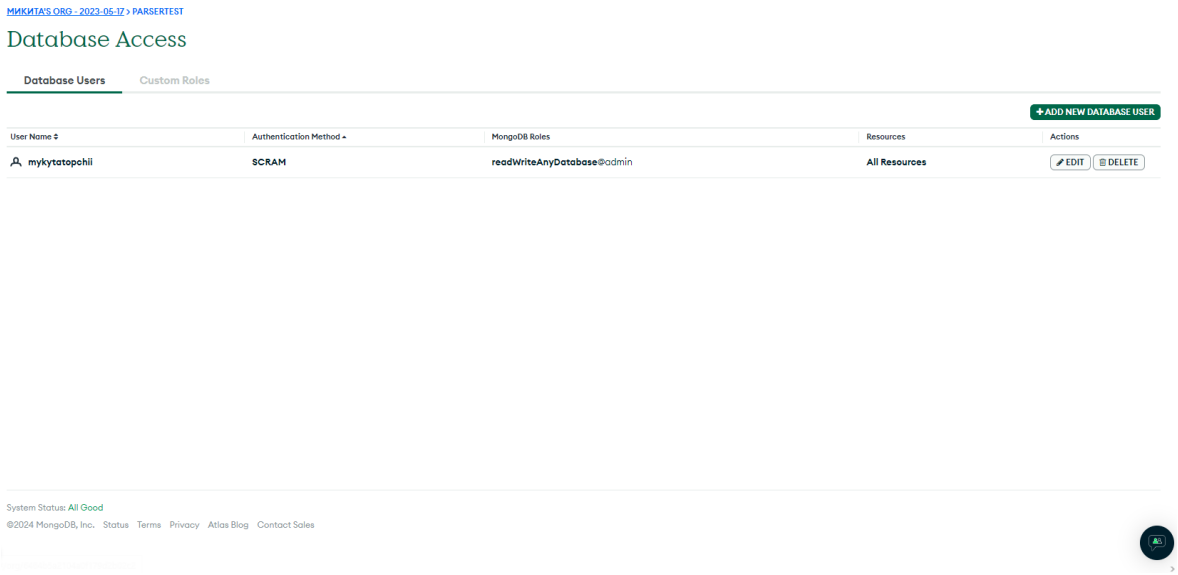

Рисунок 3.5 – Меню управління користувачами БД

3.5 Розробка вебзастосунку

Для клієнтського застосунку було обрано фронтенд фреймворк Svelte та SvelteKit. Для оформлення візуальної частини вебсайту, було обрано CSS фреймворк Tailwind CSS. Вебсайт використовує технологію динамічної маршрутизації за допомоги slug URL.

3.5.1 Svelte та SvelteKit

Svelte – це сучасний фреймворк для створення інтерфейсів користувача, розроблений Річем Харрісом. На відміну від інших популярних фреймворків, таких як React чи Vue, Svelte виконує більшу частину своєї роботи під час компіляції, а не під час виконання. Це означає, що в результаті ви отримуєте менший розмір бандла і більш швидкий час завантаження, оскільки браузеру не потрібно виконувати великий runtime для рендерингу компонентів [32].

Однією з ключових особливостей Svelte є його простота та легкість у використанні. Компоненти Svelte пишуться у файлах з розширенням .svelte, де можна змішувати HTML, CSS і JavaScript в одному місці. Це сприяє більш інтуїтивному підходу до розробки, коли вся логіка і стилі компонента зібрані разом. Реактивність у Svelte досягається за допомогою спеціальних синтаксичних конструкцій, що дозволяє автоматично оновлювати інтерфейс користувача при зміні стану без необхідності використовувати складні механізми, як у інших фреймворках.

Svelte також пропонує вбудовану підтримку стилізації компонентів. Кожен компонент може містити власні стилі, які будуть застосовуватися тільки до нього, завдяки ізольованим CSS-правилам. Це дозволяє уникнути конфліктів стилів і робить управління стилями більш зручним. Крім того,

Svelte підтримує написання CSS у зовнішніх файлах та використання препроцесорів, таких як Sass або Less, для більш складних випадків.

Ще однією важливою особливістю Svelte є його висока продуктивність. Оскільки більшість роботи виконується під час компіляції, кінцевий код, який виконується у браузері, є дуже легким і оптимізованим. Це особливо корисно для мобільних пристроїв або слабших комп'ютерів, де ресурси обмежені. Svelte також підтримує механізми, такі як код-сплітінг, що дозволяє ще більше покращити продуктивність великих вебзастосунків.

Загалом, Svelte пропонує унікальний підхід до розробки інтерфейсів користувача, який поєднує простоту та високу продуктивність. Завдяки своїм особливостям і можливостям, Svelte стає все більш популярним серед розробників, які шукають ефективні та зручні інструменти для створення сучасних вебзастосунків.

SvelteKit – це повнофункціональний фреймворк для створення вебзастосунків на основі Svelte. Він розширює можливості Svelte, пропонуючи інструменти для створення як клієнтських, так і серверних застосунків, а також підтримку статичного рендерингу, файлового роутінгу та динамічної маршрутизації.

Приклад файлового роутінгу з використанням slug у SvelteKit відображено на лістингу 3.4.

Лістинг 3.4 Приклад файлового роутінгу у 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 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 SvelteKit:

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

3.5.2 Slug технологія

Задля забезпечення надійної на ефективної навігації та розробки, було використано метод роутінгу slug. Slug – це частина URL, яка ідентифікує певну сторінку в читабельній формі. Зазвичай, slug складається з букв, цифр та дефісів, які замінюють пробіли і спеціальні символи, роблячи URL більш зрозумілим і зручним для користувачів та пошукових систем.

У SvelteKit, slug часто використовується для створення динамічних маршрутів. Це дозволяє створювати сторінки з динамічними URL, які можуть змінюватися в залежності від вмісту. У SvelteKit динамічні маршрути створюються за допомогою квадратних дужок у назвах файлів у папці src/routes. Наприклад, [slug].svelte створює маршрут, який може приймати різні значення для slug.

Дані про категорії отримуються з конфігураційного файла, а дані про певний товар – з БД. Запит до бази даних робиться в API модулі (рис. 3.6).

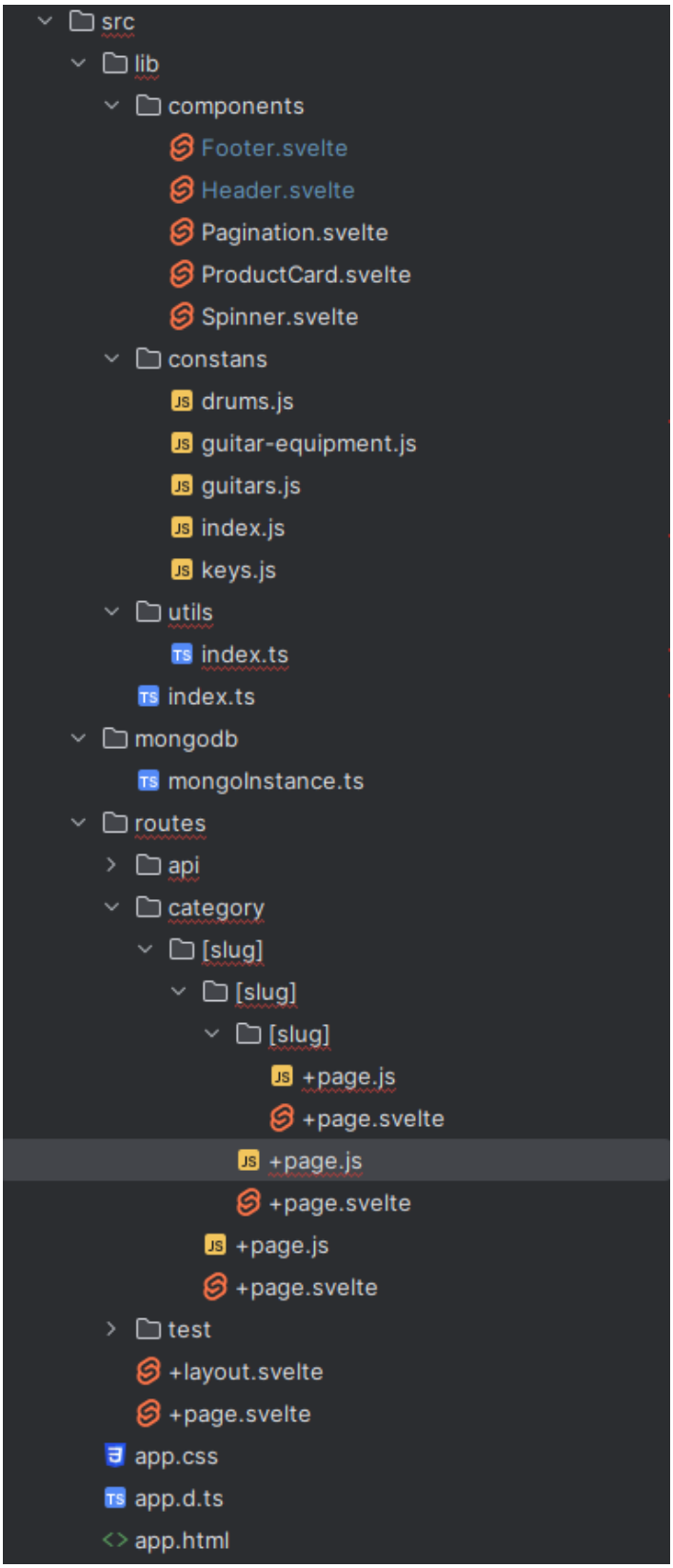

Рисунок 3.6 – Структура проєкту вебзастосунку

3.5.3 Tailwind CSS

Візуальна частина розроблена за допомоги CSS фреймворку Tailwind CSS.

Tailwind CSS – це утилітарно-орієнтована CSS-бібліотека для створення інтерфейсів користувача. Вона дозволяє використовувати набір класів для стилізації елементів без необхідності написання власного CSS-коду. Класами можна керувати безпосередньо у HTML, що робить процес розробки швидшим і більш інтуїтивним [33].

Щоб підключити Tailwind до SvelteKit проєкту, потрібно спочатку встановити tailwind, як показано на 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: [],
}
```

Якщо казати про плюси Tailwind, то можна виділити чотири основні та потужні переваги, які роблять цей фреймворк пріорітетним серед інших CSS-бібліотек для ефективної та швидкої розробки надійного вебзастосунку з сучасним та привабливим дизайном:

– прискорення розробки за рахунок використання утилітарних класів. Оскільки немає потреби постійно перемикатися між HTML та CSS-файлами;

– мінімальне навантаження стилями забезпечує режим JIT (Just-In-Time), що значно зменшує розмір кінцевого CSS-файлу, оскільки генеруються лише необхідні класи;

– стилізація за допомогою утилітарних класів робить код легко підтримуваним та зрозумілим для інших розробників;

– гнучкість та масштабованість. Tailwind CSS дозволяє створювати унікальні дизайни без обмежень, які можуть накладати готові шаблони або компоненти інших фреймворків.

Комбінація сучасних фронтенд фреймворків Svelte та Tailwind CSS успішно виконує задачу реалізації дизайну та відображення даних з реактивною фільтрацією.

3.6 Розробка API

У проекті SvelteKit важливою частиною є API, яке забезпечує взаємодію між клієнтом та базою даних MongoDB. API дозволяє отримувати необхідні дані для відображення товарів на вебплатформі, фільтрацію та пошук товарів за різними критеріями. У цьому розділі розглядається розробка API, що складається з чотирьох основних GET методів: getAllDevices, getDevicesByName, getAllBrands, getAllFilters.

API реалізовано за допомогою SvelteKit та написано на TypeScript. Основна мета API – забезпечити ефективну та швидку взаємодію з базою даних MongoDB, отримуючи та обробляючи дані для подальшого відображення на клієнті.

Для взаємодії з MongoDB використовується офіційний драйвер MongoDB для 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;
}
```

ВИСНОВКИ

У рамках практики за темою кваліфікаційної роботи було розроблено вебзастосунок для перегляду та порівняння цін товарних пропозицій, нереляційна база даних для зберігання інформації про різні товари, вебпарсер для автоматичного збору та обробки даних з різних онлайн-магазинів, вирішено задачу product matching для класифікації та порівняння одних і тих самих товарів, але з різними назвами. Для досягнення цієї мети було розглянуто декілька методів product matching, та за результатами аналізу було обрано евристичний метод з декомпозицією рядку назви товару. Такий вибір обумовлений найвищим показником точності серед інших та швидкодією. Було спроєктовано багатомодульну інформаційну систему з використання сучасних технологій веброзробки.

Виконано всі поставлені задачі, а саме:

– проведено аналіз існуючих онлайн-крамниць музичних інструментів;

– розроблено та реалізовано сучасну та надійну архітектуру системи;

– розроблено та реалізовано базу даних для системи;

– розроблено та реалізовано backend систему для обробки та збору даних;

– розроблено та реалізовано frontend застосунок для відображення даних;

– проаналізовано та розв’язано проблему product matching.

Розроблений програмний застосунок може бути корисним користувачам, що бажають подивитись товарні пропозиції різних онлайн-крамниць та порівняти ціни, наявність товарів, що їх цікавлять. Також вебзастосунок може принести користь продавцям музикальних інструментів у виді реклами їхніх онлайн-магазинів.

Застосунок має шляхи до масштабування, наприклад додавання якомога більше онлайн-крамниць України і Європи; покращення алгоритму

для вирішення задачі product matching, додавання алгоритму порівняння зображень. Також одним із важливих потенційних покращень є придбання вебхостінгу для підвищення продуктивності вебскрейпінгу та постановки скриптів на крон-роботи з метою мати найбільш актуальні дані.

Результати роботи апробовано у вигляді тез доповідей під час конференції «FOSS-2024» [34].

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Web Scraping. URL: https://apix-drive.com/ua/blog/ecommerce/web-scraping (дата звернення 01.03.2024).

2. Працюємо з Selenium для автоматизації задач у браузері. URL: https://kr-labs.com.ua/blog/pratsyuyemo-z-selenium-dlya-avtomatyzatsiyi-zadach-u-veb-brauzeri (дата звернення 02.03.2024).

3. Відгуки про музичні магазини України. URL: https://www.otzyvua.net/uk/magaziny/muzykalnye-instrumenty/ (дата звернення 03.03.2024).

4. How Google Shopping works. URL: https://support.google.com/faqs/answer/2987537?hl=en#:~:text=What%20is%20Google%20Shopping%3F,sellers%20to%20make%20their%20purchase (дата звернення 04.03.2024).

5. E-Katalog. URL: https://ek.ua/ua/ (дата звернення 05.03.2024).

6. Hotline. URL: https://hotline.ua (дата звернення 06.03.2024).

7. Magazilla. URL: https://m.ua/ua/ (дата звернення 07.03.2024).

8. SASSCAL WebSAPI: A Web Scraping Application Programming Interface to Support Access to SASSCAL's Weather Data. URL: https://datascience.codata.org/articles/10.5334/dsj-2021-024 (дата звернення 08.03.2024).

9. Повний посібник з вебскрепінгу. URL: https://www.rapidseedbox.com/uk/blog/web-scraping (дата звернення 09.03.2024).

10. Web Scraping vs API: The Best Way to Extract Data. URL: https://hasdata.com/blog/web-scraping-vs-api (дата звернення 10.03.2024).

11. What are the types of APIs and their differences. URL: https://www.techtarget.com/searchapparchitecture/tip/What-are-the-types-of-APIs-and-their-

differences#:~:text=There%20are%20four%20different%20types,any%20outside%20developer%20or%20business (дата звернення 11.03.2024).

12. Документація W3. URL: https://www.w3schools.com/ (дата звернення 12.03.2024).

13. 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.

14. А.Р. Ковтуненко, О.В. Яковлева, В.А. Любченко, & О.В. Янголенко (2020) Дослідження сумісного використання математичної морфології та згорткових нейронних мереж для вирішення задачі розпізнавання цінників. Вісник Національного технічного університету ХПІ (3). 24-31.

15. Gorokhovatskyi V., Tvoroshenko I., and 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.

16. 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.

17. Yakovleva, O., Kovač, M., Ardasov, V. & Yeremenko, I. (2023). Study on adding functionality to the Zoom online conference system for monitoring the participant activities. Scientific Journal of Bratislava University of Economics and Management «Public Administration and Regional Development, Economics, Management, and Marketing», 19(1), pp. 158-183.

18. 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.

19. 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. URL: https://isg-konf.com/the-world-of-modern-technologies-and-inventions/ (дата звернення 11.04.2024).

20. 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. URL: https://eu-conf.com/ua/events/trends-in-science-regarding-the-creation-of-new-teaching-methods/ (дата звернення 12.04.2024).

21. Gorokhovatskyi, V., Tvoroshenko, I., Kobylin, O., & Vlasenko, N. (2023). Search for visual objects by request in the form of a cluster representation for the structural image description, Advances in Electrical and Electronic Engineering, 21(1), pp. 19-27.

22. Kobylin O., Gorokhovatskyi V., Tvoroshenko I., and Peredrii O. (2020) The application of non-parametric statistics methods in image classifiers based on structural description components, Telecommunications and Radio Engineering, 79(10), pp. 855-863.

23. Daradkeh, Y.I., Gorokhovatskyi, V., Tvoroshenko, I., Gadetska, S., and Al-Dhaifallah, M. (2021) Methods of Classification of Images on the Basis of the Values of Statistical Distributions for the Composition of Structural Description Components, IEEE Access, 9, pp. 92964-92973.

24. Daradkeh, Y.I., Tvoroshenko, I., Gorokhovatskyi, V., Latiff, L.A., and Ahmad, N. (2021) Development of Effective Methods for Structural Image Recognition Using the Principles of Data Granulation and Apparatus of Fuzzy Logic, IEEE Access, 9, pp. 13417-13428.

25. Daradkeh Y.I., Gorokhovatskyi V., Tvoroshenko I., and Al-Dhaifallah M. (2022) Classification of Images Based on a System of Hierarchical Features, Computers, Materials & Continua, 72(1), pp. 1785-1797.

26. Кобилін, О.А., & Творошенко, І.С. (2021). Методи цифрової обробки зображень: навч. посібник. Харків: ХНУРЕ.

27. Путятін, Є.П., Гороховатський, В.О., & Матат, О.О. (2006). Методи та алгоритми комп’ютерного зору: навч. посібник.

28. Гороховатський, В.О., & Творошенко, І.С. (2021). Методи інтелектуального аналізу та оброблення даних: навч. посібник.

29. Гороховатський В.О., Творошенко І.С. (2022) Аналіз багатовимірних даних за описом у формі множини компонент: монографія. Харків: ХНУРЕ, 124 с.

30. Multiple environments to improve your developer experience. URL: https://thomaspoignant.medium.com/multiple-environments-to-improve-your-developer-experience-727f72dc39da (дата звернення 01.04.2024).

31. Документація Puppeteer. URL: https://pptr.dev/category/introduction (дата звернення 02.04.2024).

32. Документація SvelteKit. URL: https://kit.svelte.dev (дата звернення 05.04.2024).

33. Документація Tailwind CSS. URL: https://tailwindcss.com (дата звернення 10.04.2024).

34. Топчій М. А., Яковлева О.В. (2024). ОГЛЯД БІБЛІОТЕКИ PUPPETEER ДЛЯ ВИРІШЕННЯ ЗАДАЧІ DATA SCRAPING З ЦІЛЛЮ ЗБОРУ ЦІННИХ ДАНИХ. FOSS-2024, с. 101.