**Міністерство освіти і науки України**

**Луцький національний технічний університет**

**Факультет комп'ютерних та інформаційних технологій**

**Кафедра інженерії програмного забезпечення**

# КВАЛІФІКАЦІЙНА РОБОТА ЗА СТУПЕНЕМ ВИЩОЇ ОСВІТИ «БАКАЛАВР»

## ОПТИМІЗАЦІЯ МЕХАНІЗМУ ПОШУКУ В ВЕБ-ЗАСТОСУНКУ ЗА ДОПОМОГОЮ ELASTICSEARCH ДЛЯ МУЛЬТИПЛАТФОРМЕННОГО ВИКОРИСТАННЯ

## OPTIMIZING THE SEARCH ENGINE IN A WEB APPLICATION USING ELASTICSEARCH FOR MULTI-PLATFORM USE

спеціальність 121 Інженерія програмного забезпечення
(шифр і назва спеціальності)

освітня програма «Інженерія програмного забезпечення»
(назва освітньої програми)

Виконав: здобувач вищої освіти
Групи ІПЗс-21
**Киричук Олександр Олександрович**
_____
(підпис)

Керівник:
к.т.н., доцент
Повстяна Юлія Славомирівна
_____
(підпис)

Кваліфікаційну роботу
допущено до захисту
«8» червня 2024 р.
к.т.н., доцент
Гарант освітньої програми:
Ліщина Наталія Миколаївна
_____
(підпис)

Луцьк – 2024 року

**ЛУЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ**

Факультет *комп'ютерних та інформаційних технологій*
Кафедра *інженерії програмного забезпечення*
Ступінь вищої освіти: *бакалавр*
Галузь знань: *12 Інформаційні технології*
Спеціальність: *121 Інженерія програмного забезпечення*
Освітня програма: *«Інженерія програмного забезпечення»*

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____
«2» 02 2024 р.

# ЗАВДАННЯ
## НА КВАЛІФІКАЦІЙНУ РОБОТУ ЗДОБУВАЧУ ВИЩОЇ ОСВІТИ

*Киричуку Олександру Олександровичу*
(прізвище, ім'я, по батькові)

1. Тема кваліфікаційної роботи: Розробка оптимізації механізму пошуку в веб-застосунку за допомогою ElasticSearch для мультиплатформенного використання

Керівник роботи: *к.т.н., доцент, Повстяна Юлія Славомирівна*
затверджені наказом закладу вищої освіти від «*30» грудня 2023 р.* № *477/01-02*
2. Строк подання здобувачем вищої освіти кваліфікаційної роботи *«5» червня 2024 р.*
3. Вихідні дані до роботи: *технічне та програмне забезпечення ЕОМ, вимоги до розробки програмного забезпечення, ергономічні вимоги до функціонування програмного засобу.*

4. Зміст розрахунково-пояснювальної записки (перелік питань, що потрібно розробити): *Аналіз сучасного стану проблеми, існуючих методів і засобів її розв'язання, аналіз, визначення вимог до розроблюваного програмного забезпечення та проектування програмного забезпечення, опис функціонального наповнення об'єкта проектування, практична реалізація об'єкта проектування.*

5. Перелік графічного матеріалу: *24 рисунка; 1 таблиця.*

6. Консультанти розділів роботи

| Розділ | Прізвище, ініціали та посада консультанта | Підпис | |
|---|---|---|---|
| | | завдання видав | завдання прийняв |
| *Аналіз предметної області* | *Повстяна Ю. С.* | | |
| *Специфікації вимог до розробленої системи* | *Повстяна Ю. С.* | | |
| *Розробка об'єкта проєктування* | *Повстяна Ю. С.* | | |
| *Нормоконтроль* | *Повстяна Ю. С.* | | |
| *Гарант ОП* | *Ліщина Н. М.* | | |
| *Показник запозичень тексту* | | 4,6 % | |
| *Академічна доброчесність* | *Ящук А. А.* | | |

7. Дата видачі завдання *«2» лютого 2024 р.*

КАЛЕНДАРНИЙ ПЛАН

| № З/п | Назва етапів кваліфікаційної роботи бакалавра | Строк виконання етапів роботи | Примітка |
|---|---|---|---|
| 1 | *Огляд літературних джерел по темі кваліфікаційної роботи бакалавра* | *07.02.2024 р.* | |
| 2 | *Аналіз проблеми розробки та впровадження об'єкту проектування* | *27.02.2024 р.* | |
| 3 | *Обґрунтування вибору шляхів, технологій (алгоритмів) і засобів вирішення поставленого завдання* | *12.03.2024 р.* | |
| 4 | *Розробка функціонально-структурної схеми роботи об'єкта проектування та проектування бази даних* | *17.04.2024 р.* | |
| 5 | *Практична реалізація об'єкта проектування та розробка бази даних* | *18.05.2024 р.* | |
| 6 | *Тестування та налагодження об'єкта проектування* | *01.06.2024 р.* | |
| 7 | *Здача чистового варіанту кваліфікаційної роботи бакалавра на кафедру* | *05.06.2024 р.* | |
| | | | |
| | | | |

Здобувач вищої освіти _____ (підпис) *Киричук О. О.* (прізвище, ініціали)

Керівник кваліфікаційної роботи _____ (підпис) *Повстяна Ю. С.* (прізвище, ініціали)

Міністерство освіти і науки України
Луцький національний технічний університет

**Рецензія**
**на кваліфікаційну роботу бакалавра**

**Здобувач вищої освіти:**

Киричук Олександр Олександрович

**Тема:**

Оптимізація механізму пошуку в веб-застосунку за допомогою ElasticSearch для мультиплатформенного використання.

**Коротка характеристика кваліфікаційної роботи:**

Кваліфікаційна робота присвячена дослідженню та вдосконаленню механізмів пошуку в сучасних веб-застосунках. Автор детально розглядає можливості та переваги використання Elasticsearch як інструменту для підвищення продуктивності та ефективності пошукових запитів. Робота включає аналіз існуючих рішень, детальний опис інтеграції Elasticsearch у веб-застосунки та проведення експериментів для оцінки продуктивності на різних платформах.

**Самостійні розробки і пропозиції автора:**

Автором проведено комплексний аналіз архітектури Elasticsearch та її можливостей для оптимізації пошукових механізмів. В роботі представлено власний підхід до інтеграції Elasticsearch, що використовуються в мультиплатформених веб-застосунках. Зокрема, запропоновано методику налаштування індексів і аналізаторів для досягнення максимальної швидкості та точності пошуку.

**Практичне значення роботи:**

Практичне значення роботи полягає в тому, що запропоновані підходи до оптимізації пошукових механізмів можуть бути впроваджені в різних веб-застосунках, що використовують мультиплатформенні технології. Це

дозволить в свою чергу використовувати оптимізований еластичний пошук в різних системах.

**Загальний висновок:**

Кваліфікаційна робота на тему "Оптимізація механізму пошуку в веб-застосунку за допомогою Elasticsearch для мультиплатформенного використання" демонструє знання автора та інноваційний підхід до вирішення поставленої задачі. Запропоновані методики значно покращують продуктивність та якість пошукових функцій у веб-застосунках, що має велике практичне значення. Робота заслуговує на високу оцінку та може бути корисною для подальшого впровадження у реальних проектах.

Рецензент: Лісунов В.О., к.т.н., доц.
(прізвище, ім'я, по-батькові, посада)

**Рецензент кваліфікаційної роботи** _____
(підпис)

«7» 06 2024 р.

Міністерство освіти і науки України
Луцький національний технічний університет
Факультет комп'ютерних та інформаційних технологій
Кафедра інженерії програмного забезпечення

**ВІДГУК**
**КЕРІВНИКА НА КВАЛІФІКАЦІЙНУ РОБОТУ БАКАЛАВРА**

Здобувач вищої освіти: Киричук Олександр Олександрович

група ІПЗс-21

**Тема кваліфікаційної роботи**

Оптимізація механізму пошуку в веб-застосунку за допомогою ElasticSearch для мультиплатформенного використання

**Керівник** к.т.н., доц. Повстяна Ю. С.

**Актуальність теми.**

Тема кваліфікаційної роботи є надзвичайно актуальною в сучасних умовах розвитку веб-технологій. Зі зростанням обсягів інформації та вимог до швидкості обробки даних, ефективний пошук стає критично важливим для забезпечення задоволеності користувачів та конкурентоспроможності веб-застосунків.

**Об'єкт дослідження**

Об'єктом дослідження є механізми пошуку в веб-застосунках, зокрема інтеграція та оптимізація Elasticsearch для забезпечення ефективного мультиплатформенного використання.

**Характеристика теоретичного рівня, наявності самостійних розробок і практичної значимості роботи.**

Теоретичний рівень роботи є високим. Автор провів ґрунтовний аналіз існуючих методів пошуку, детально вивчив архітектуру та можливості Elasticsearch. Робота демонструє глибоке розуміння теоретичних аспектів пошукових систем та їхньої оптимізації.

**Загальний висновок**

Роботу виконано у повному обсязі у відповідності до поставлених завдань, а здобувач Киричук О. О. заслуговує на оцінку «відмінно».

**Керівник**_____ **Повстяна Ю.С.**
(підпис) (прізвище, ім'я, по батькові)

«7» 06 2024 р.

# АНОТАЦІЯ

Киричук О. О. Оптимізації механізму пошуку в веб-застосунку за допомогою ElasticSearch для мультиплатформенного використання. Рукопис.

Кваліфікаційна робота бакалавра ОП «Інженерія програмного забезпечення». Луцький національний технічний університет. Луцьк, 2024.

Кваліфікаційна робота бакалавра складається зі вступу, трьох розділів, висновків і пропозицій, списку використаних джерел та додатків.

У першому розділі здійснено аналіз сучасного стану проблеми веб-застосунків із можливістю оптимізації пошуку і сформульованого завдання. В другому розділі визначено вимоги до розроблюваної системи, а також засоби, методи і алгоритми розробки. У третьому розділі розроблено інтернет-магазин із оптимізованим механізмом еластичного пошуку за допомогою сервісу ElasticSearch. У висновках узагальнено інформацію, відображену у попередніх частинах. Результати розробки демонструють можливості взаємодії досить швидко отримувати результати пошуку по товарам мультиплатформенній системі.

Ключові слова: ElasticSearch, еластичний пошук, оптимізація, Laravel, REST API, Blade.

# ABSTRACT

Kyrychuk O. O. Optimizing the search engine in a web application with ElasticSearch for cross-platform use. Manuscript.

Bachelor's qualifying thesis of the educational program "Software Engineering". Lutsk National Technical University. Lutsk, 2024.

The bachelor's qualifying thesis consists of an introduction, three sections, conclusions and proposals, a list of used sources and annexes.

In the first chapter, an analysis of the current state of the problem of web applications with the possibility of optimization search and the task is formulated. The requirements are defined in the second section and the task was formulated. The second section defines the requirements for the developed system, as well as means, methods and development algorithms. In the third chapter, an online store with an optimized elastic search mechanism using the ElasticSearch service is developed. The conclusions summarize the information presented in the previous parts. The development results demonstrate the possibilities of interaction to quickly receive search results for goods in a multi-platform system.

Keywords: ElasticSearch, elastic search, optimization, Laravel, REST API, Blade.

# ЗМІСТ

# ВСТУП

Пошук по сайту є цінним інструментом для пошуку товарів в будь-якому інтернет-магазині і не тільки. Добре розроблений пошук допомагає інтернет-магазинам продавати більше товарів, відповідно більше заробляти. Статистика показує, що потенційні покупці купують більше товарів, коли заходять на сайт та шукають потрібну річ через пошуковий інструмент, ніж переходячи по категоріях.

Великі інтернет-магазини використовують саме оптимізований механізм пошуку, а не SQL запити.  ElasticSearch надає ці можливості за рахунок свого алгоритму, який індексує всі можливі товари та зберігає це все в оперативній пам’яті, що є набагато кращою практикою, ніж робити окремі запити до бази даних. Зазвичай, такі запити не тільки дуже важкі, а й ще доволі сильно помиляються, що для сайту  є дуже вагомою проблемою.

Метою роботи є розробка оптимізованого механізму пошуку в веб-застосунку за допомогою ElasticSearch для мультиплатформенного використання, яка буде надавати більш правильний та оптимізований пошуковий запит потенційному клієнту.

Завдання дослідження:

– ознайомитись з документацією PHP. Вивчення основних функцій та розуміння застосування їх;

– вивчення та отримання практичних навиків при перегляді документації Laravel. Ознайомлення з концепціями та можливостями фреймворку;

– побудова схеми бази даних з потрібними зв’язками для кращого розуміння реалізації веб-сайту;

– реалізація адмін-панелі для інтернет-магазину «CozaStore», за допомогою безкоштовного адміністративного шаблону. CRUD операції для керування вмістом контенту;

– реалізація REST API сайту, щоб була можливість розробникам реалізовувати в своїх проєктах наше API;

– реалізація користувацького інтерфейсу frontend;

– реалізація backend частини за допомогою PHP та Laravel. Створення фасадів для написання бізнес-логіки яке буде застосовуватися в контролерах та відповідні реалізовані маршрути;

– встановлення та вбудовування API ElasticSearch на сайт.

Об’єкт дослідження – процес і результат дослідження подібних мультиплатформенних систем з еластичним пошуком.

Предмет дослідження – функції та алгоритми розробки мультиплатформенної системи з оптимізованим еластичним пошуковим сервісом ElasticSearch.

Новизна роботи полягає у розробці оптимізованого механізму пошуку для мультиплатформенної системи з більш швидкою та точною віддачею даних для клієнта.

# РОЗДІЛ 1
# АНАЛІЗ ОПТИМІЗАЦІЇ МЕХАНІЗМУ ПОШУКУ В ВЕБ-ЗАСТОСУНКУ ЗА ДОПОМОГОЮ ELASTICSEARCH ДЛЯ МУЛЬТИПЛАТФОРМЕННОГО ВИКОРИСТАННЯ І ПОСТАНОВКА ЗАВДАНЬ НА КВАЛІФІКАЦІЙНУ РОБОТУ

## 1.1 Аналіз сучасного стану проблеми

В наший час все доволі швидко розвивається, хоча здається тільки 76 років тому, створили першу малу, електронну обчислювальну машину, яку згодом назвали «Комп'ютер». В сьогоденні, любий користувач, який вміє користуватися базовими функціями що надає комп'ютер, може створити свій перший простий сайт, виділивши на це певний діапазон часу, який не буде і таким вже великим.

Розробники, які створювали сайти за допомогою таких інструментів та мов програмування як HTML, CSS, JS, PHP, з досвідом покращували свій код та розуміли, що потрібно зробити розробку сайтів ще простішою. Згодом почали з'являтися системи управління контентом сайту, іншими словами CMS.

CMS (Content Management System) з роками розвиваються та стають все оптимізованими та більш користувацькими для просто користувача. Вони доволі добре показують себе з розробкою не великого функціоналу та простого дизайну. Але в двигунах сайтів є і мінуси, такі як, наприклад: платні розширення, не гнучкий та функціональний код, потрібен повний контроль розширень та конфлікти між ними.

Звісно сьогодні багато сайтів зроблені на цих системах, по типу WordPress, Wix, Shopify, BigCommerce, Drupal, але якщо потрібен сайт з великою бізнес-логікою та якісним дизайном то ці варіанти вже не підходять. І тут, вже ми повертаємося до інструментів розмітки та мов програмування.

Зазвичай, для оптимізації frontend частини використовуються такі інструменти як Gulp, Webpack, Parcel, або Grunt. Ці інструменти допомагають

мінімізувати CSS та JavaScript код. Також, вони добре справляються з оптимізацією картинок.

Frontend частина здається легкою, але насправді, там потрібно знати ще більше мабуть, ніж на backend стороні. Для того щоб, бути затребуваним frontend розробником потрібно знати: HTML, CSS, JS, Angular, Node.js, Vue.js, React, API, JSON і тому подібне. І щоб, це все знати, де що застосовувати, потрібно мати багато практики та розуміння документації.

Щоб реалізовувати backend частину, потрібно вибрати стек технології. Тут доволі не великий вибір, це PHP або Python. Python є доволі хорошою мовою програмування та також має свій фреймворк Django. В PHP теж є свій фреймворк під назвою Laravel, але головним його плюсом є те, що він на ринку більше затребуваний ніж Django, або той же Symfony, тому я вибрав саме його, як за основу стек розробки.

Чому ж саме пошукова система? Тому що, щоб створити хороший сайт, потрібно враховувати багато як великих так і малих дрібниць. На мою думку пошукова система є доволі затребуваною частиною сайту, оскільки важливо, щоб, користувач міг зайти на сайт та знайти все, що йому потрібно. Звісно, майже на кожному сайті є пошук, але чи він є гнучким, це вже друге питання.

Що ж мається під словами гнучкий пошук? Простими словами, це пошук який шукає подібні слова в реченні не в залежності де вони знаходяться, тобто, якщо ми візьмемо запити SQL, то вони шукають збіги саме в тому порядку, якому ви його напишете. Альтернативою також є PostgreSQL, який дозволяє встановити розширення «pg_trgm», який розбиває речення на слова та шукає схожі собі, але тут постає питання оптимізації, якої тут не буде.

Існує багато пошукових інструментів, але виділити з них можна саме такі як:

1) ElasticSearch – пошукова система з відкритим кодом, яка не тільки шукає слова, але й має багато іншого функціоналу, як наприклад агрегація. На рисунку 1.1 зображено головний інтерфейс сервісу ElasticSearch [1];

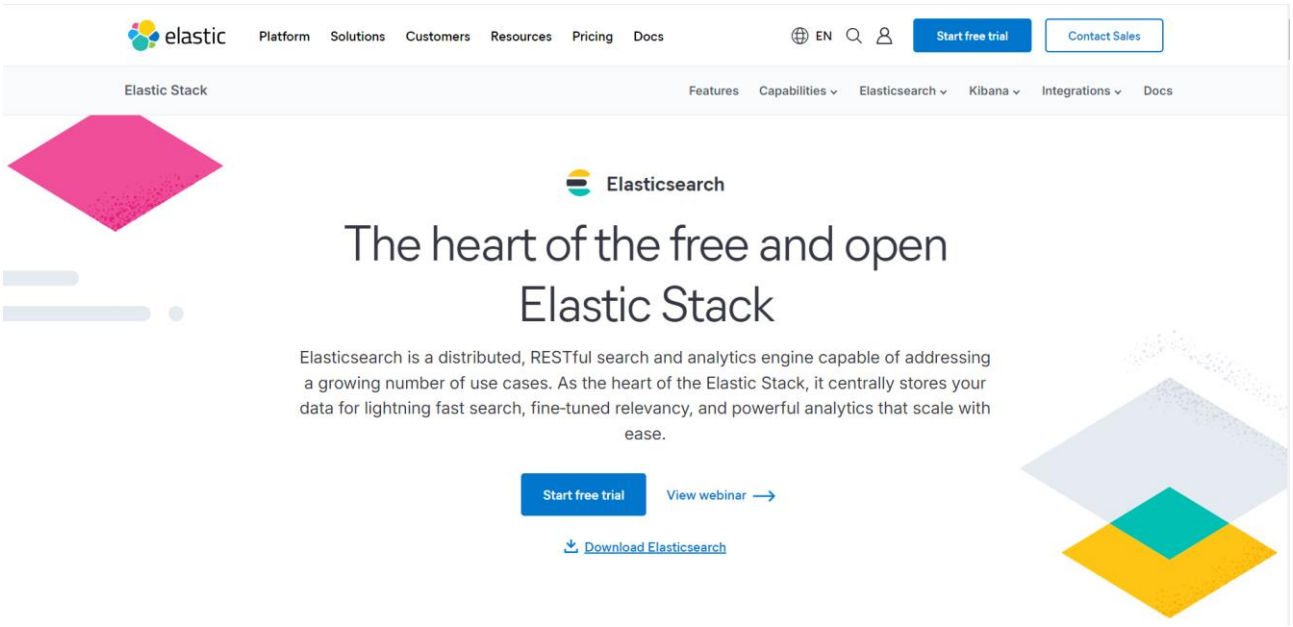


Рисунок 1.1 – Вигляд головного інтерфейсу ElasticSearch [1]

2) Algolia – пошукова система на веб-сайти, яка сканує його, щоб надавати швидкі та релевантні результати за пошуковими запитами відвідувачів. На рисунку 1.2 зображено вигляд головної сторінки документації, для реалізації її в свій продукт;

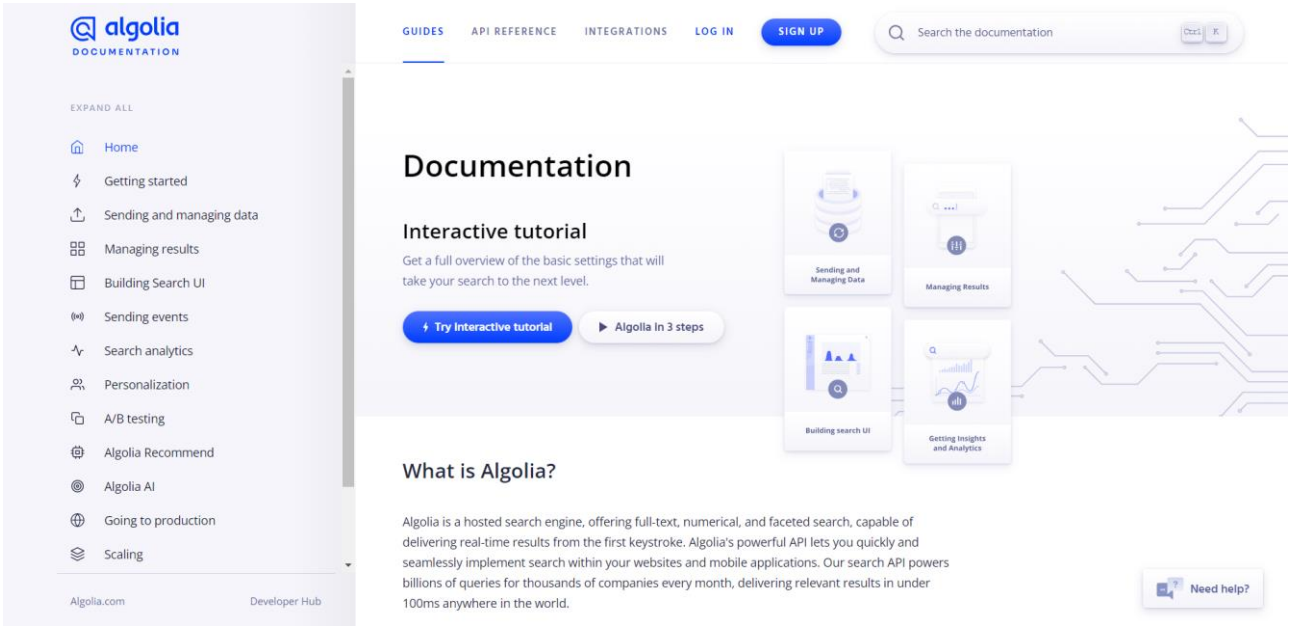


Рисунок 1.2 – Вигляд документації пошукового сервісу Algolia [2]

3) Typesense – це пошукова система з відкритим кодом, яка аналізує вже готовий sitemap сайту та має семантичний і векторний пошук. На рисунку 1.3 показано головний інтерфейс Typesense з прикладом реалізованого пошуку;

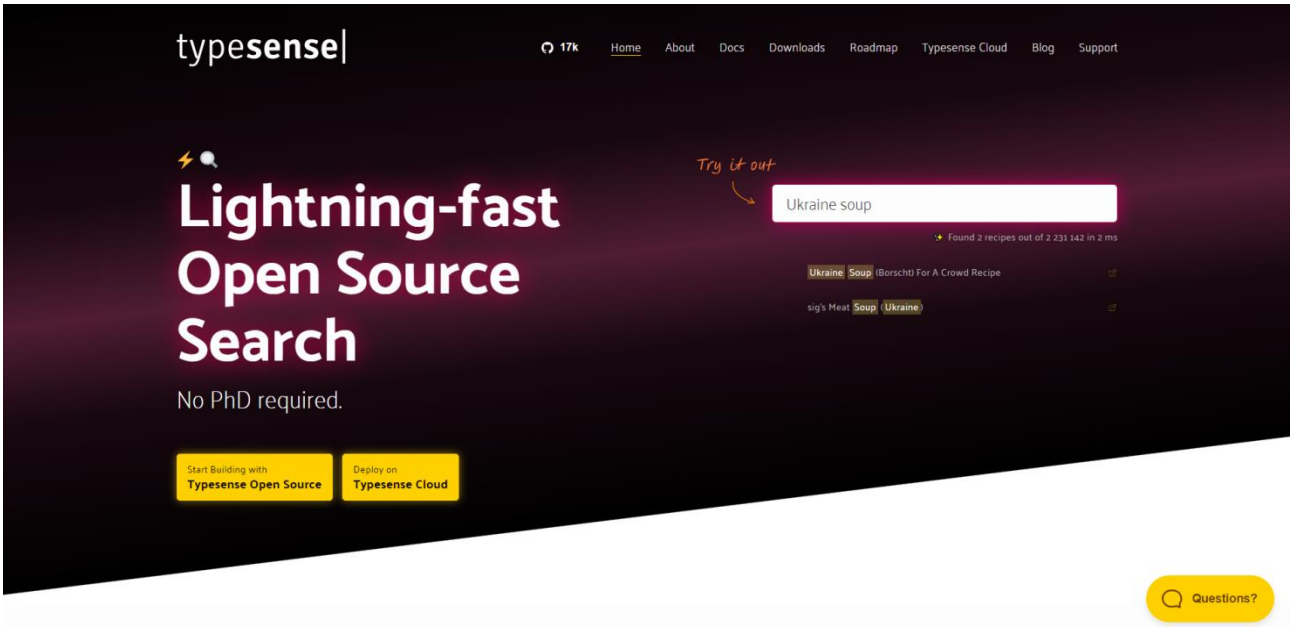


Рисунок 1.3 – Вигляд головного інтерфейсу пошукового сервісу Typesense [3]

4) Meilisearch – це доволі швидка пошукова система з відкритим кодом яка повертає результати за 50 мілісекунд. Але вона дозволяє шукати максимум за 10 словами в пошуку. На рисунку 1.4 зображено реалізація пошуку за допомогою мови програмування PHP та веб-сервісу Meilisearch [4].

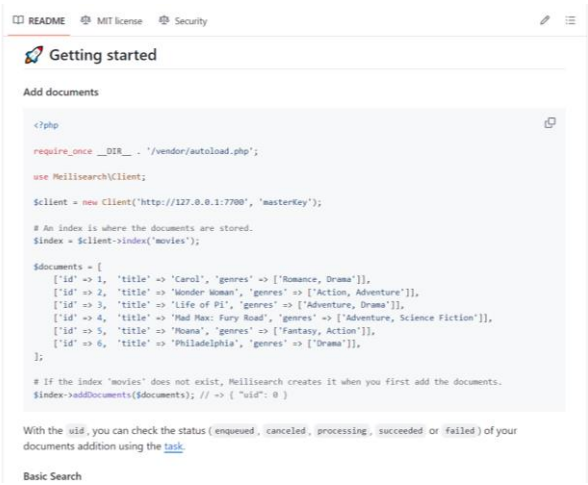


Рисунок 1.4 – Приклад реалізації пошукової системи Meilisearch [4]

В таблиці 1.1 описані об’єктивні порівняння ключових функцій пошукових систем, які описані в документації.

Таблиця 1.1 – Порівняння ключових функцій пошукових систем

| | Typesense | Algolia | ElasticSearch | Meilisearch |
|---|---|---|---|---|
| Кількість документів | Обмежено лише допустимою кількістю оперативної пам’яті | Необмежено | Обмежено лише доступною кількістю пам'яті на жорсткому диску | Залежить від дискового простору та продуктивністю оперативної пам'яті |
| Максимальна кількість індексів | Без обмежень | Без обмежень | Без обмежень | 200 для Linux/MacOS; 20 для Windows |
| Максимальний розмір індексу | Обмежено лише кількістю оперативної пам’яті | 128 ГБ | Необмежений | 500Гб |
| Максимальна кількість слів на поле | Без обмежень | Без обмежень | Без обмежень | Без обмежень |
| Максимальний розмір запису | Без обмежень | 10 КБ | Без обмежень | 2 ГБ |
| Кількість ключів API | Без обмежень | 5000 | Без обмежень | Без обмежень |

Мною було вибрано таку пошукову систему як ElasticSearch, тому що, вона є безкоштовною, та якщо, якісно налаштувати індексацію то швидко справляється з пошуком по сайті.

В лістингу 1.1 зображено запит до сервера ElasticSearch для створення індексу з назвою «products» та визначення типів документів з властивостями «name» та «age».

Лістинг 1.1 – Демонстрація запиту до ElasticSearch для індексації продуктів за її властивостями

```
curl -XPUT localhost:9200/products -d '{
  "mappings": {
    "mytype": {
      "properties": {
        "name": {
          "type": "string"
        },
        "age": {
          "type": "long"
        }
      }
    }
  }
}'
```

Кінець лістингу 1.1

Laravel було започатковано 2011 року Тейлором Отвеллом на основі мови програмування PHP. Вже, на тих стадіях фреймворк ставав популярним своїм елегантним синтаксисом та великим набором функцій, які дозволяли розробникам швидше та якісніше виконувати свою роботу.

Laravel є високорівневим фреймворком, який дозволяє легко та лаконічно описувати свій код, звісно, якщо писати по правилам документації та спільноти. Laravel вже зміг оновити свій синтаксис до 11 версії, що ж звісно не може не тішити. Ось декілька сервісів, які написані за допомогою Laravel:

– Laracasts – навчальний ресурс, який надає користувачам більше поглибитись у веб–розробку, надаючи їм доступ до Laravel, Vue.js, React та багато чого іншого, що зображено на рисунку 1.5;

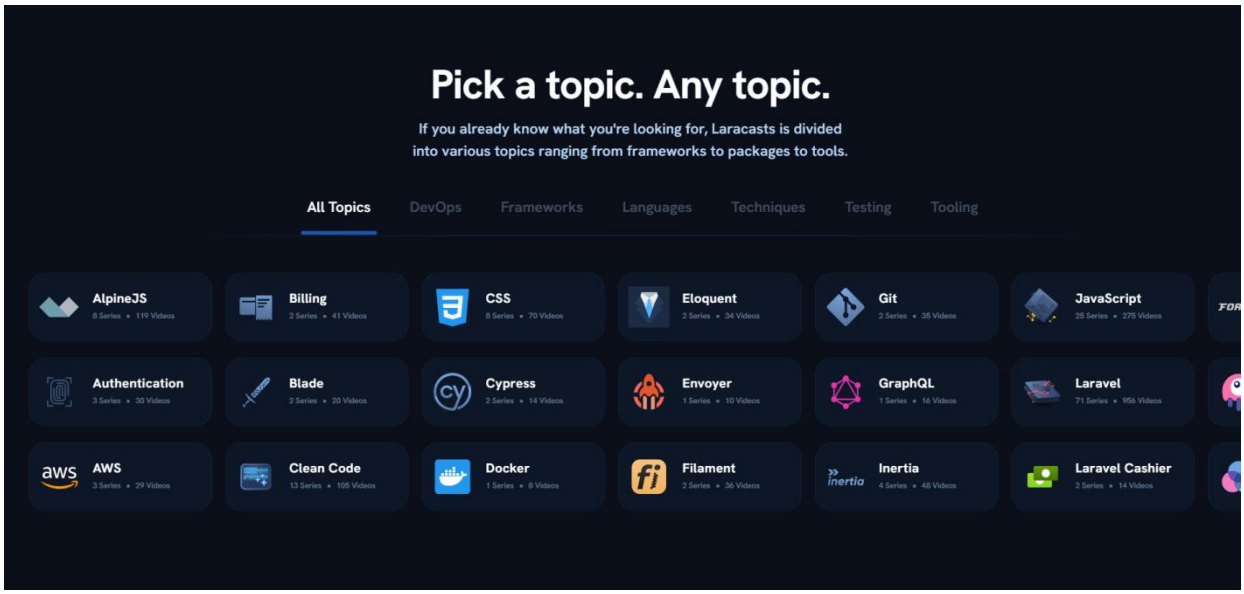


Рисунок 1.5 – Laracasts та курси, які розміщенні на ньому [5]

– October CMS – система управління контентом веб-сайтів та веб-додатків. На рисунку 1.6 зображено приклад реалізації та роль даної CMS системи;

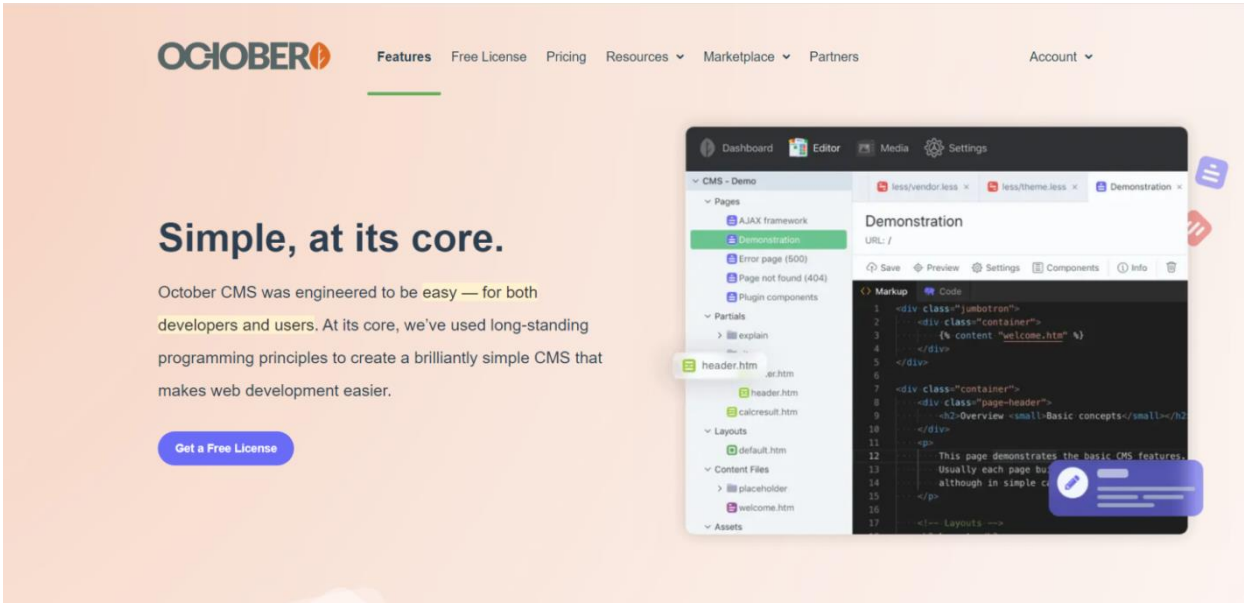


Рисунок 1.6 – October CMS та її роль [6]

– Invoice Ninja – надійний, гнучкий, але напрочуд доступний інструмент для виставлення рахунків і керування витратами, яким щодня користуються

понад 200 000 індивідуальних підприємців і власників малого бізнесу, який зображений на рисунку 1.7;

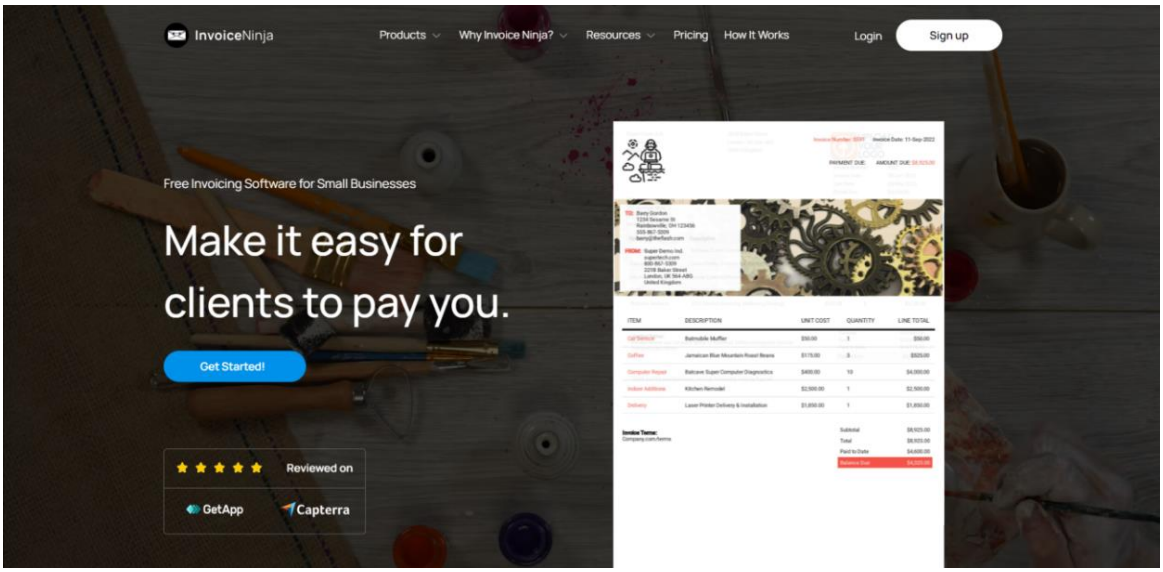


Рисунок 1.7 – Вигляд при перенаправленні на сайт Invoice Ninja [7]

– Attendize – це платформа, яка надає API для реалізації керуваннями подій та продажу квитків. Вона має широкий набір функцій, щоб організація з легкістю могла використовувати даний функціонал (рис. 1.8);

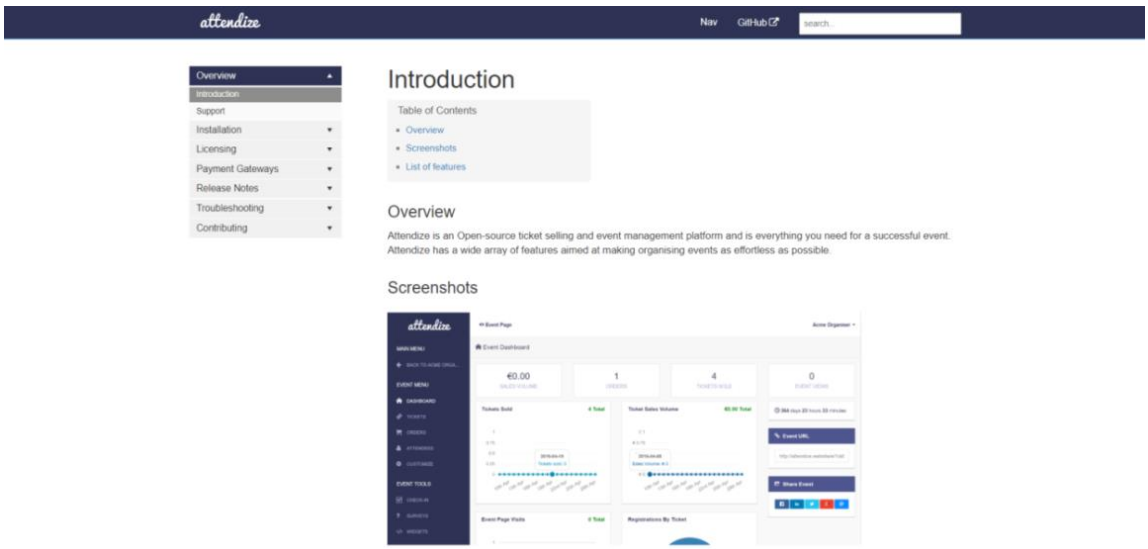


Рисунок 1.8 – Інтерфейс користувацької інтеграції API Attendize [8]

– Flarum – сучасний форум для обговорення різним тем (рис. 1.9).

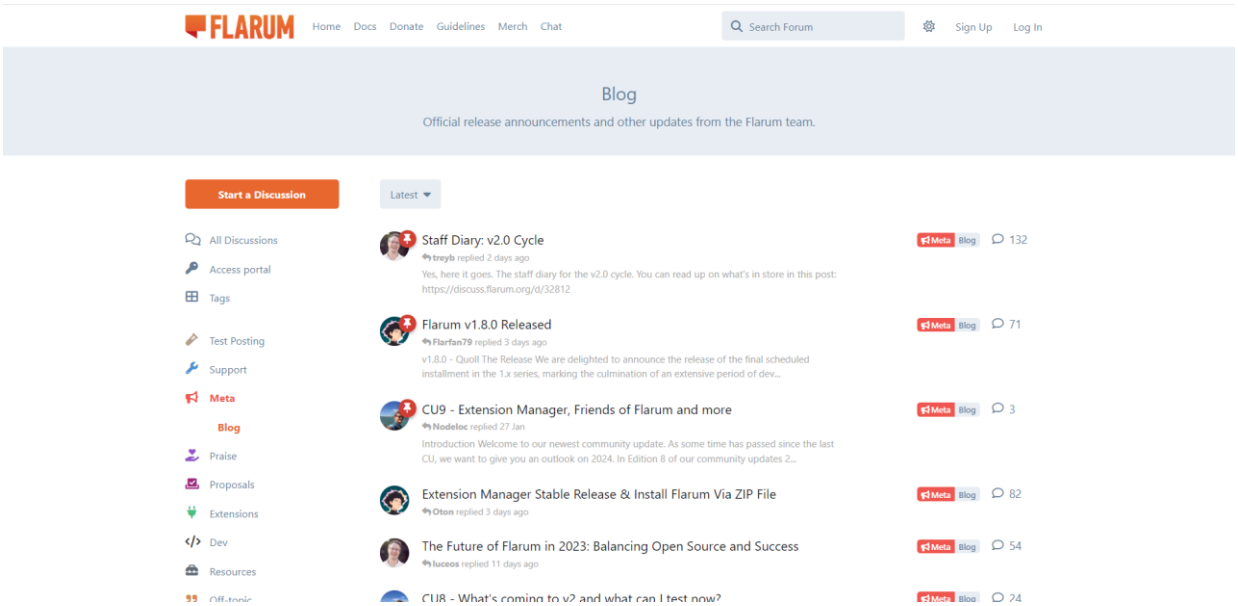


Рисунок 1.9 – Інтерфейс користувацької інтеграції API Attendize [9]

В Laravel звісно є ж своя документація, яка є доволі лаконічною, яку зображено на рисунку 1.10.

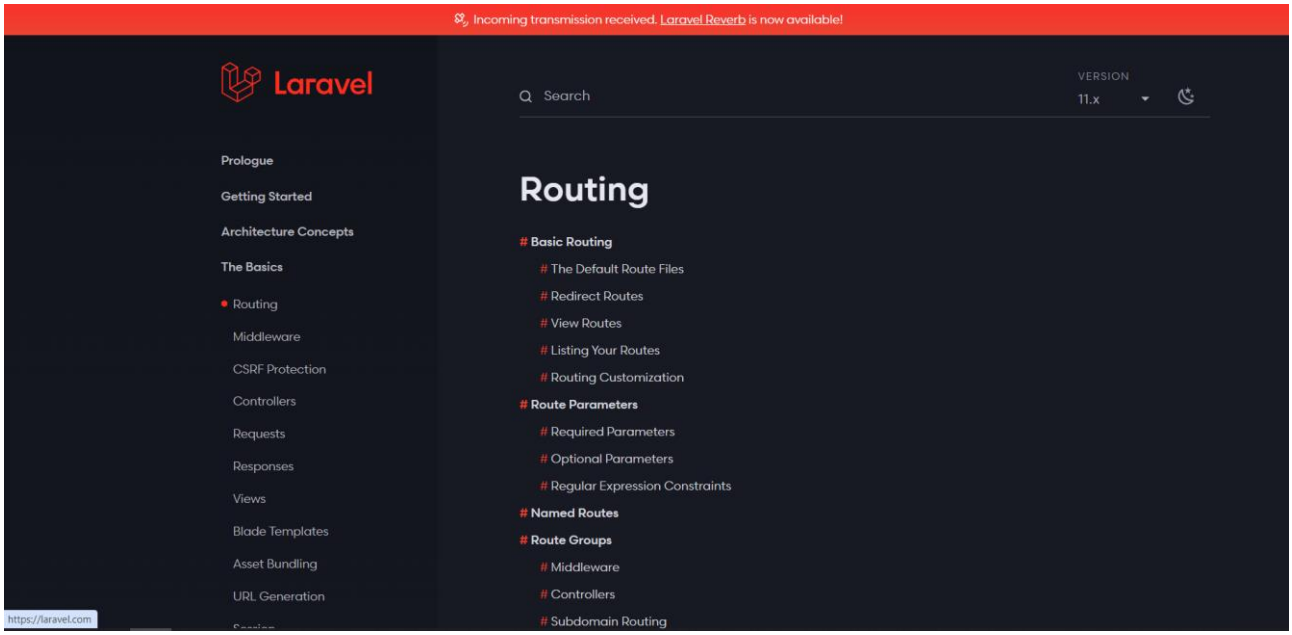


Рисунок 1.10 – Вигляд документації Laravel [10]

## 1.2 Постановка завдання на кваліфікаційну роботу бакалавра

Ціллю, моєї кваліфікаційної роботи є розробка веб-сайту та пошукової системи за допомогою PHP фреймворку Laravel, API ElasticSearch та допоміжних бібліотек.

Щоб виконати цю ціль, потрібно реалізувати наступне описане, в наступних пунктах:

– ознайомитись з документацією PHP. Вивчення основних функцій та розуміння застосування їх;

– вивчення та отримання практичних навиків при перегляді документації Laravel. Ознайомлення з концепціями та можливостями фреймворку;

– побудова схеми бази даних з потрібними зв’язками, для кращого розуміння реалізації веб-сайту;

– реалізація адмін-панелі, для інтернет-магазину. CRUD операції для керування вмістом контенту;

– реалізація REST API сайту, щоб була можливість розробникам реалізовувати в своїх проєктах наше API;

– реалізація користувацького інтерфейсу клієнтської частини сайту;

– реалізація серверної частини;

– встановлення та вбудовування API ElasticSearch на сайт.

## Висновки до розділу 1

Якщо, дотримуватися цього плану, то за 2-3 місяці буде розроблено повноцінний інтернет-магазин з реалізованою адмін-панелю, користувацьким інтерфейсом та повноцінним пошуком з яким користувачам буде приємніше знаходитися на сайті. Звісно. з часом будуть реалізовуватися нові фішки, такі як наприклад Ajax на формах зворотного зв’язку і тому подібне, що зробить веб-ресурс тільки приємнішим для користування.

# РОЗДІЛ 2
# СПЕЦИФІКАЦІЯ ВИМОГ ДО РОЗРОБЛЮВАНОЇ СИСТЕМИ

## 2.1 Аналіз, визначення вимог до розроблюваного програмного забезпечення та проектування програмного забезпечення

Для чіткого розуміння, які повинні бути кроки, щоб реалізувати певний програмний продукт, потрібно для початку відобразити цикл його за допомогою UML-діаграми.

Загалом частіше використовують наступні 11 видів UML-діаграм, для розробки моделей:

- діаграма класів;
- діаграма об’єктів;
- діаграма компонентів;
- діаграма складової структури;
- діаграма варіантів використання;
- діаграма послідовності;
- діаграма комунікації;
- діаграма станів;
- діаграма діяльності;
- діаграма розміщення;
- діаграма пакетів.

Мною було вибрано діаграму послідовності, для її реалізації, потрібно мати загальне розуміння як її правильно графічно відобразити, які є тонкощі при її реалізації та загалом розмітку PlantUML для гнучкості.

Щоб реалізувати дану модель, я проаналізував та описав, які основні функції повинні бути на сайті, після чого розробив власну діаграму послідовностей для інтернет-магазину «CozaStore», що зображено на рисунку 2.1.

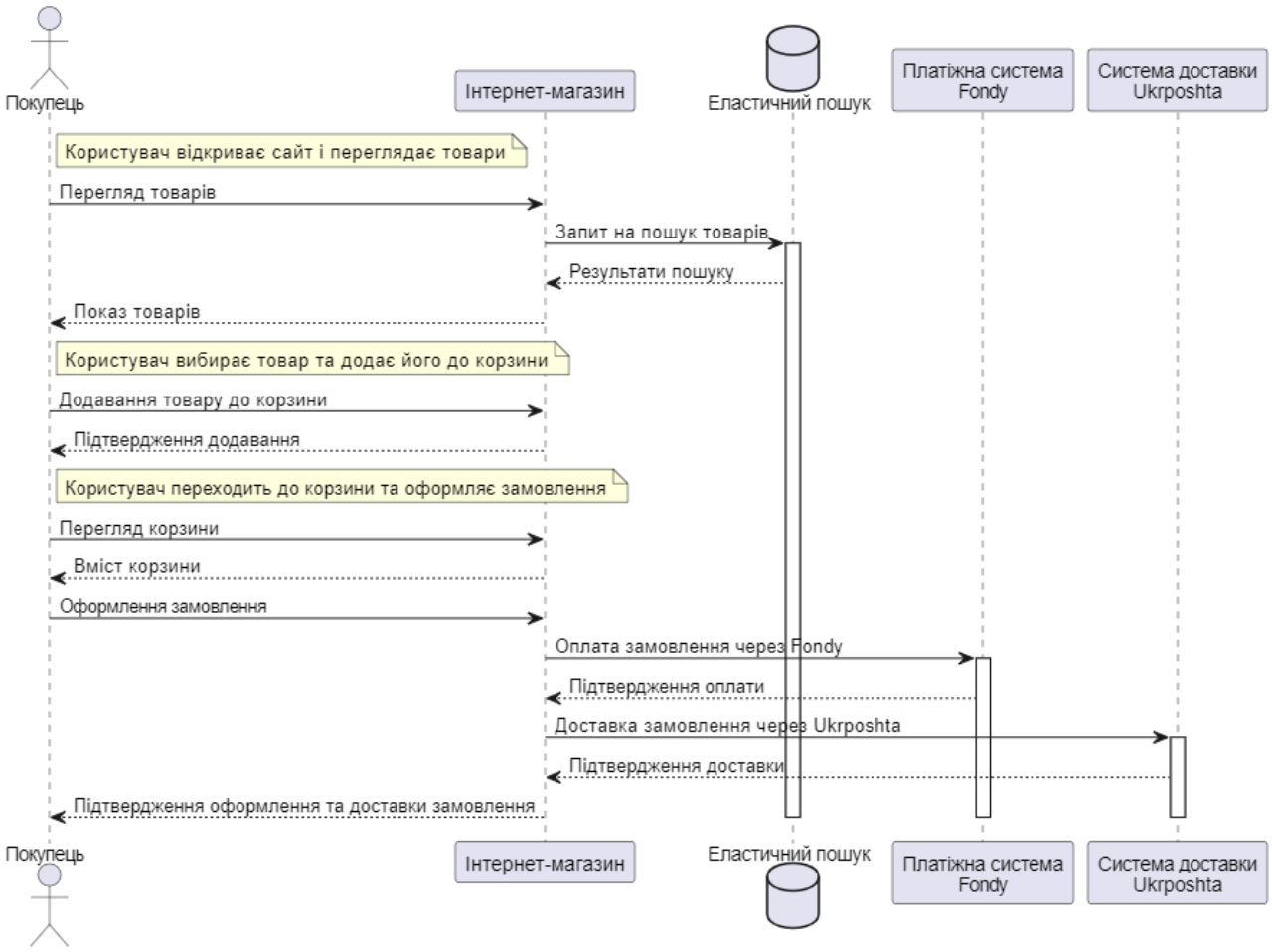


Рисунок 2.1 – Вигляд діаграми послідовностей для інтернет-магазину «CozaStore»

Діаграми є дуже важливою частиною перед початком розробки будь-якого проєкту. Вони дають змогу орієнтуватися, що за чим потрібно реалізовувати. Дана діаграма показує, який функціонал повинен бути реалізованим для завершення проєкту.

## 2.2 Вибір засобів, методів і алгоритмів вирішення поставленого завдання

При розумінні, який повинен бути функціонал сайту, потрібно проаналізувати за допомогою яких стеків, засобів, методів та алгоритмів реалізовувати даний програмний продукт. В розділі 1 описано загально

прийняті технології розробки, за яких допомогою є можливість розробити любий інтернет-ресурс з будь-якою бізнес логікою.

За основу буде взято повноцінний фреймворк Laravel. Чому ж не CMS? Тому що, на CMS системі не вийде створити інтернет-ресурс з оптимізованою логікою та гнучким кодом, який без всяких зайвих рухів, можна замінити та протестити, що ж в іншому варіанті не сильно і так просто вийде, тому що, там все зв’язано розширеннями, та будь-які зміни можуть порушити їх роботу.

Керування веб-сайтом буде відбуватися за допомогою адмін-панелі AdminLTE 3 [11]. В ній є всі необхідні компоненти для зручної реалізації форм, інпут полів, чек боксів, кнопок та тому подібне. Також, головним плюсом даної панелі є те, що вона оновляється та повністю у вільному доступі для розробників.

Frontend частину буде реалізовано за допомогою HTML, CSS, JS, JQuery та Bootstrap технологій. Також хорошою практикою є реалізація frontend частини за допомогою фреймворків Vue.JS [12] та React [13]. Але на це піде більше часу. Загалом, шаблонізатор Blade від Laravel справляється з цим завданням відмінно.

Blade – системний шаблонізатор від фреймворку Laravel. Він дозволяє розробникам, за допомогою мови програмування PHP, виводити контент з бази даних своїм синтаксисом. В шаблонізаторі не дозволено писати логіку, тому він легкий в використанні. Blade має свій алгоритм для захисту інформації, яка знаходяться в БД. Він надає розробникам, розробляти веб-сайти з надійним та безпечним для користування функціоналом.

Розроблення схеми бази даних буде реалізовано за допомогою програмного забезпечення MySQL Workbench [14]. Workbench – це інструмент, який надає користувачам змогу візуально створювати схеми бази даних з всіма необхідними полями та їх типізацією (рис. 2.2). Це програмне забезпечення, розроблене кампанією MySQL, яке підходить під всі операційні системи.

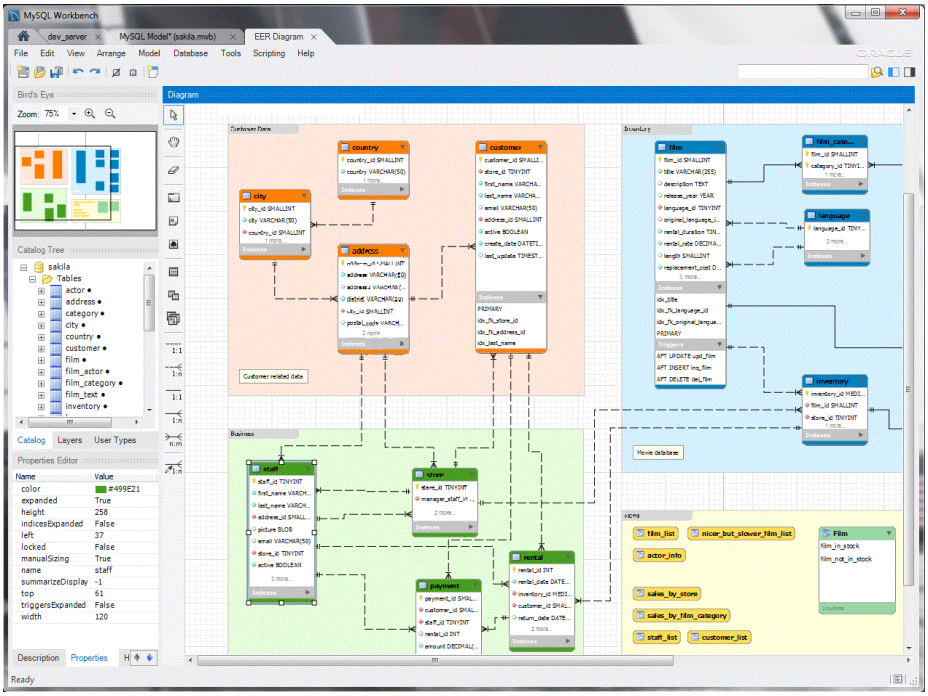


Рисунок 2.2 – Візуальний інтерфейс ПЗ MySQL Workbench [14]

Backend частина. Як ми знаємо, реалізація API посторонніх ресурсів відбувається за допомогою посилань. Ми користуємося API постійно, просто цього не помічаємо. Наприклад, розробник розробив сайт з онлайн-конвертором валют та хоче, щоб цим функціоналом користувалися не тільки на його ресурсі, а й була змога реалізовувати це і на другий програмних продуктах. Тоді на допомогу приходить вибраний фреймворк та REST API. За допомогою цих стеків, розробник може розробити посилання, за допомогою якого любий розробник зможе отримати дані валют, які йому були необхідні [15]. Як виглядає звернення через REST API показано на рисунку 2.3.

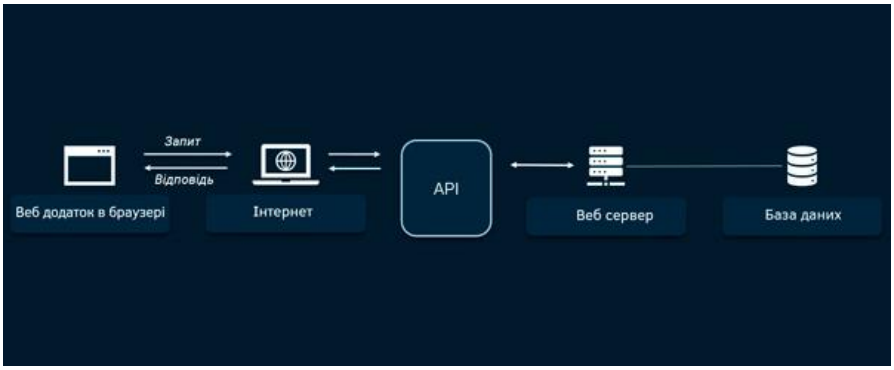


Рисунок 2.3 – Візуалізація роботи API [15]

Тому, реалізація backend частини буде відбуватися за допомогою фреймворку Laravel. Laravel на дає змогу оптимізовано реалізовувати своє API та використовувати його іншим розробників. Реалізація онлайн-оплати буде реалізовано за допомогою API Fondy, підтягування даних з самою доставкою буде відбуватися завдяки API Ukrposhta.

Пошуковий функціонал для інтернет-магазину буде реалізовано за допомогою API та індексних документів ElasticSearch.

**Висновки до розділу 2**

Проаналізувавши розділ 2 можна зробити висновок, що реалізація інтернет-магазину не є і такою вже простою справою. На розробку даного програмного забезпечення йде багато часу не тільки розробки, а й аналізу, що, де і як потрібно краще зробити. Якщо не продумати правильно алгоритм та візуалізацію самого сайту то на стадії розробки або навіть при її закінченні може виникнути багато нюансів, для реалізації яких потім прийдеться потратити час та написання багато лишнього коду, який потім буде заважати просуванню сайту.

Тому, для правильної реалізації потрібно дотримуватися наступні кроки розробки даного програмного забезпечення:

– проаналізувати, що потрібно для споживачів, іншими словами проаналізувати ринок на популярні товари;

– реалізація діаграми послідовності для повного розуміння який функціонал потрібно розробити;

– розробка бази даних за допомого програмного забезпечення Workbench, для візуального розуміння структури та її реалізації;

– розробка адмін частини веб-сайту;

– розробка API для надання можливості користування функціоналом сайту іншим розробникам;

– створення frontend частини веб-сайту за допомогою: HTML, CSS, JS та JQuery;

– створення функціоналу еластичного пошуку за допомогою API ElasticSearch та його документів;

– реалізація платіжної API Fondy;

– реалізація підтягування областей, міст, відділень та змога роздрукувати TTN за допомогою API Ukrposhta.

# РОЗДІЛ 3
# РОЗРОБКА МЕХАНІЗМУ ПОШУКУ В ВЕБ-ЗАСТОСУНКУ ЗА ДОПОМОГОЮ ELASTICSEARCH ДЛЯ МУЛЬТИПЛАТФОРМЕННОГО ВИКОРИСТАННЯ

## 3.1 Практична реалізація об’єкта проєктування

Відповідно до архітектури другого розділу, головними пунктами для розробки інтернет-магазину можна вважати:

– розробка frontend частини;

– розробка backend частини;

– розробка адмін-панелі;

– інтеграція API ElasticSearch.

Розробка frontend частини буде реалізовуватися за допомогою редактора вихідного коду Visual Studio Code (рис 3.1) [16]. Даний редактор доволі зручний в використанні та не багато затратний в ресурсах продуктивності комп’ютера. Великим плюсом редактора є, можливість встановлювати додаткові розширення.

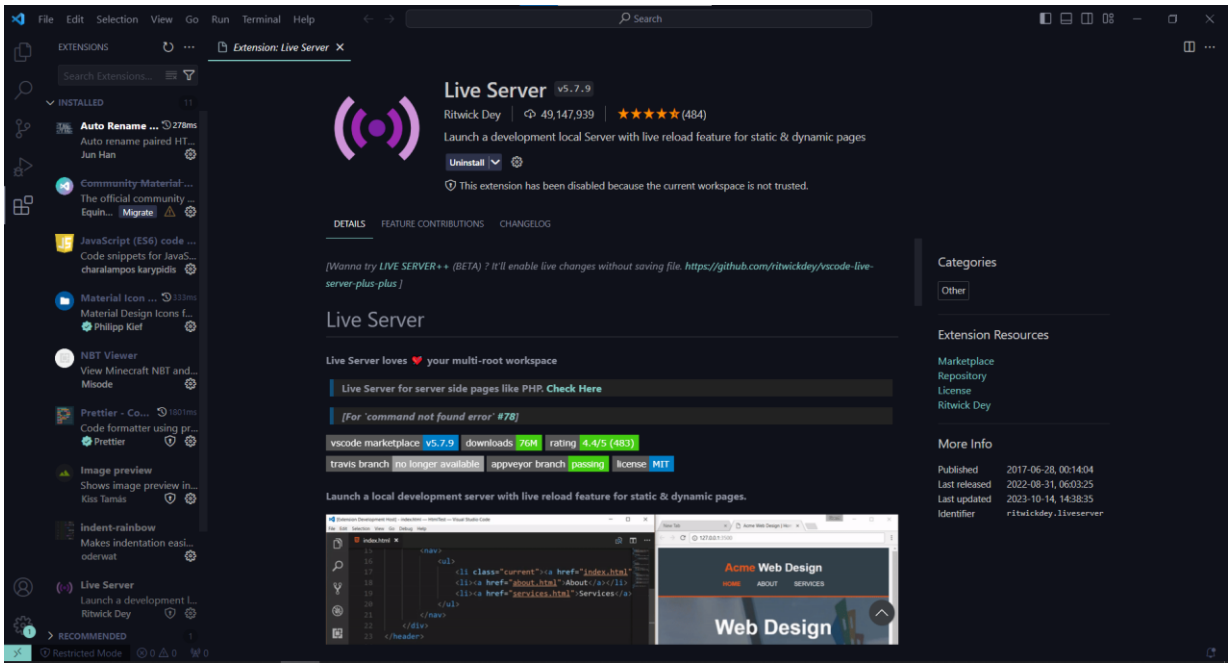


Рисунок 3.1 – Інтерфейс VS Code з популярним розширенням Live Server [16]

При створенні проєкту, потрібно відразу продумати архітектуру файлів. Розумно продумана архітектура клієнтської частини – це легко-розширювальна система, яка в подальшому буде легка в розумінні, як для frontend, так і для backend розробників.

Архітектуру frontend частини я вирішив розділити по папкам, а саме:

– css – всі стилі сторінок, які були написані власноруч;

– fonts – шрифти, які будуть необхідні для веб-застосунку;

– images – картинки та svg;

– js – самописні JavaScript скрипти;

– vendor – підключені готові скрипти, для легшої реалізації слайдів, popup меню, паралаксу і тому подібне.

Сторінки будуть лежати в кореневій теці проєкту. Це дозволить backend розробникам, легко орієнтуватися в структурі. При початку розробки, це не так видно, але згодом, розробник все більше починає плутатися у своєму же проєкті.

На рисунку 3.2 зображено інтерфейс сторінки варіацій з фінальними результатом клієнтської частини.

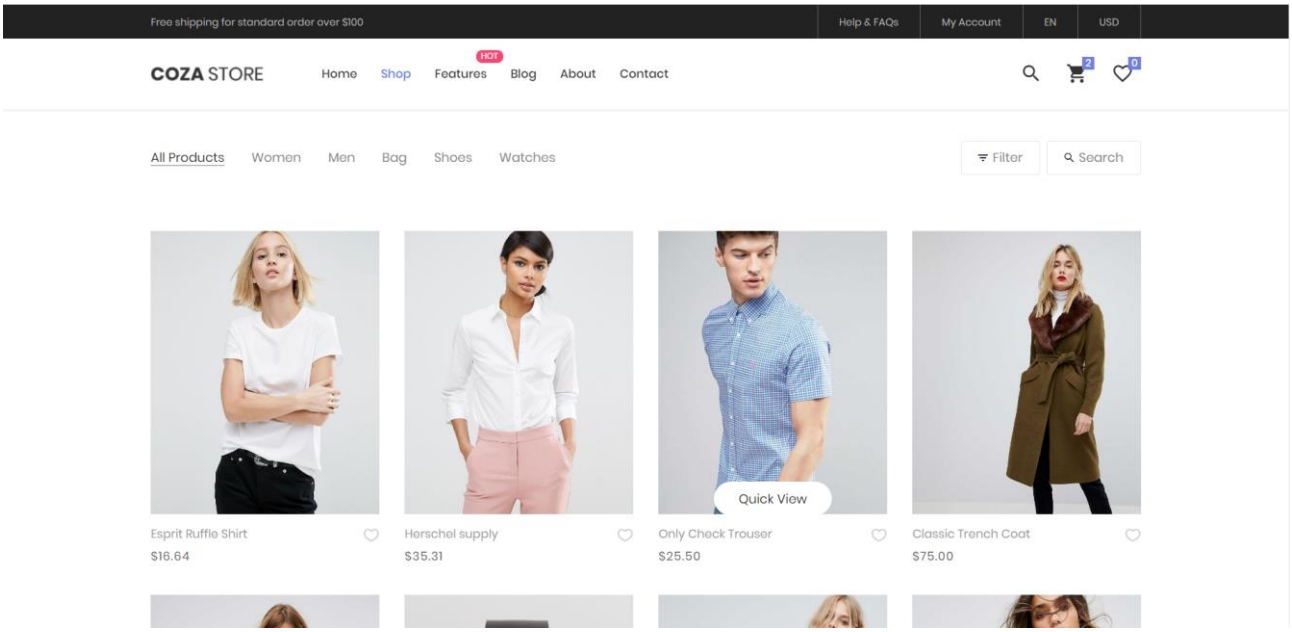


Рисунок 3.2 – Зовнішній вигляд інтерфейсу варіацій

Загалом, сайти можна розділити на такі два види:

– статичні – це сайти, в яких ніяк не міняється контент всередині, тобто вони розроблені тільки на HTML, CSS та JavaScript;

– динамічні – це сайти, в яких міняється контент через адмін-панель менеджера. Він реалізований на мові програмування, по типу Python або PHP. Це сайт який має базу даних та бізнес-логіку.

Оскільки, мені потрібно реалізувати механізм пошуку в веб-застосунку за допомогою ElasticSearch, то це має бути динамічний веб-сайт.

Щоб, почати реалізовувати динамічний веб-сайт, потрібно створити проєкт. Для цього потрібно перейти до документації Laravel, де розписані команди та деякі нюанси. Наприклад, для повноцінного встановлення потрібно переконатися, що у вас встановлено PHP потрібної версії. Після чого, за декілька команд можна створити новий проєкт. Laravel надає можливість встановити себе як глобально, тобто, на всю систему Windows, а бо ж, локально на проєкт, що підходить найбільше, тому що, так, всі пакети та залежності буде встановлено в сам проєкт, що зменшує ризики не працюючого веб-застосунку на другому комп’ютері.

В лістингу 3.1 показано, як встановити проєкт локально з відкриттям стартового шаблону.

Лістинг 3.1 – Команди для встановлення та запуску початкового меню

```
composer create-project laravel/laravel cozastore
cd cozastore
php artisan serve
```

Кінець лістингу 3.1

На рисунку 3.3 зображено, як буде виглядати стартове вікно тільки новоствореного проєкту за допомогою Laravel.

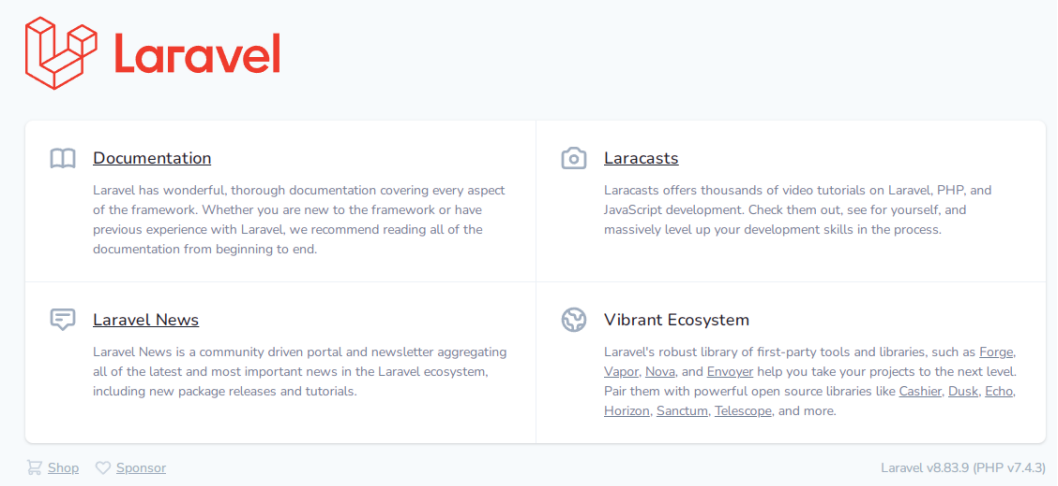


Рисунок 3.3 – Зовнішній вигляд створеного проєкту на Laravel

Далі нам потрібно реалізувати адмін-панель з базою даних, про неї докладніше можна прочитати в 3 пункті цього розділу. Щоб, проєкт було зв’язано з сховищем даних, потрібно в файлі «.env» прописати для цього дані, які надає нам сервіс. На лістингу 3.2 більш детально показано як підключитися до БД та мати можливість взаємодіяти з нею.

Лістинг 3.2 – Встановлення зв’язку з БД

```
DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=cozastore
DB_USERNAME=root
DB_PASSWORD=
```

Кінець лістингу 3.2

Щоб, розпочати інтегрувати HTML розмітку в Blade шаблонізатори, потрібно архітектурно перекинути сторінки в папку «resources/views/client» та розподілити структуровану по директоріях. На рисунку 3.4 зображено архітектура папок для клієнтської частини.

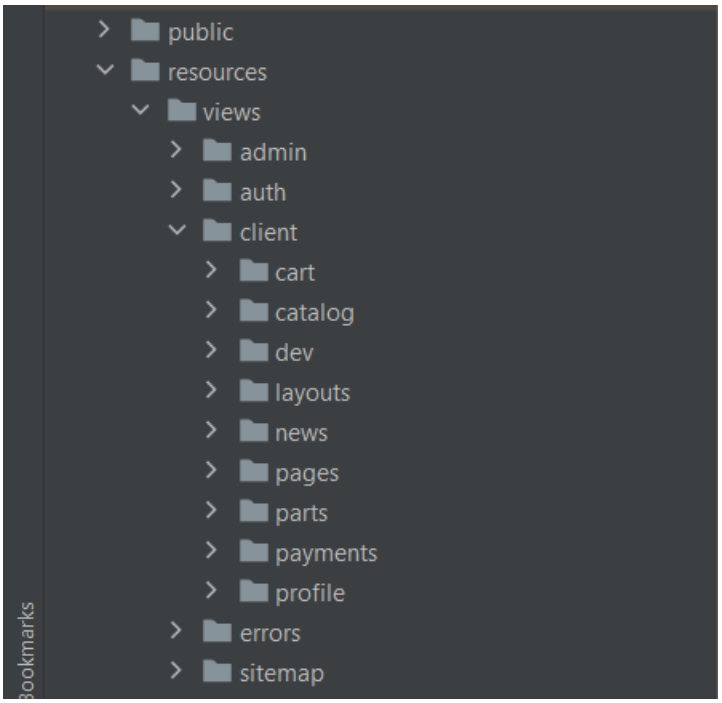


Рисунок 3.4 – Архітектура клієнтської частини папок зі сторінками всередині

В папці «layouts» знаходиться шляхи до стилів та скриптів. Також, там розподілено верхню шапку сайту та підвал, щоб потім міняти дані тільки в одному шаблоні, а не по всіх.

Щоб вивести товари динамічно, шаблонізатор Blade надає можливість писати PHP код всередині HTML розмітки. На лістингу 3.3 показано як це реалізовується.

Лістинг 3.3 – Команди для встановлення та запуску початкового меню

```
@foreach($products as $product)
  <div class="col-sm-6 col-md-4 col-lg-3 p-b-35 isotope-item women">
    <div class="block2">
      <div class="block2-pic hov-img0" style="height: 250px">
        <img src="{{ $product?->getFirstMediaUrl('photo', 'preview') ?:
          asset('storage/media/default/marketplace.png') }}"
          alt="IMG-PRODUCT" style="height: 100%">
      </div>
      <div class="block2-txt flex-w flex-t p-t-14">
        <div class="block2-txt-child1 flex-col-l ">
          <a href="{{ route('product.page', [$product->category->slug,
$product->slug]) }}"
            class="stext-104 cl4 hov-cl1 trans-04 js-name-b2 p-b-6">
            {{ $product->name }}
          </a>
```

```
          <span class="stext-105 cl3">
            {{ $product->price }}₴ <del style="font-size: 12px">{{
            $product->old_price }}₴</del>
          </span>
        </div>
      </div>
    </div>
  </div>
@endforeach
```

Кінець лістингу 3.3

В лістингу 3.3 все розпочинається з PHP циклу «foreach», який іде по всім товарам, що повертаються з контролера, який зображено в лістингу 3.4. Зміна «$product» містить один товар в кожному циклі, та вже з нього беруться дані, які необхідні для користувача, сайтом. Далі виводяться картинку за допомогою функції «getFirstMediaUrl», в яку передається медіа колекція та назва конверсії. Тернарним оператором перевіряємо, якщо не буде картинки в товарі то показуємо просто статичну. За допомогою route та query запиту з передавання slug, формуємо посилання на сторінку одного товару. Також за допомогою «Route Binding» показуємо ім’я товару і ціни.

Лістинг 3.4 – Метод products в контролері

```
public function products(string $categorySlug)
{
    $category = Category::where('slug', $categorySlug)->firstOrFail();

    $categories = Category::whereParentId($category->id)->get()
      ->toTree();

    $products = $category->products()->filterable()->with('media',
      'category')->paginate(20);

    return view('client.catalog.products', ['category' => $category,
      'products' => $products, 'categories' => $categories]);
}
```

Кінець лістингу 3.4

Реалізація переходів за посиланнями виконується в файлах:

– «admin.php» – маршрутизація по сторінках в адмін-панелі;

– «api.php» – маршрутизація для реалізації API частини;

– «web.php» – маршрутизація по сторінках для клієнтської частини;

– «web–auth.php» – маршрутизація для сторінок, які зв’язані з авторизацією.

Всі ці файли, забезпечують функціоналість веб-сайтів якими ми їх бачимо. За допомогою маршрутизації реалізовуються API веб-сайти, які виглядають красиво. В коді роутів, прописуються, як буде виглядати посилання та який контролер з методом буде за це відповідати. Також хорошою практикою є реалізація посередників, які не допускають, щоб простий користувач не зміг перейти до адмін-панелі будь яким шляхом. Посередники можуть виконувати будь-які завдання, які повинен. В лістингу 3.5 показано маршрутизацію для авторизаційних методів, яке реалізовано в файлі «web–auth.php».

Лістинг 3.5 – Реалізація маршрутизації для авторизаційної функціональсті

```
public function products(string $categorySlug)
{
    $category = Category::where('slug', $categorySlug)->firstOrFail();

    $categories = Category::whereParentId($category->id)->get()
      ->toTree();

    $products = $category->products()->filterable()->with('media',
      'category')->paginate(20);

    return view('client.catalog.products', ['category' => $category,
      'products' => $products, 'categories' => $categories]);
}
```

Кінець лістингу 3.5

Наступним кроком є реалізація оптимізованого механізму пошуку товарів за допомогою сервісу ElasticSearch в веб-сайті. Для цього я використаю пакет від автора «ezimuel's» з найменуванням «elasticsearch/elasticsearch» [17]. За його

допомогою стане набагато простіше зв’язуватися з API ElasticSearch. На рисунках 3.5-3.6 зображено порівняння, як виконується пошук за допомогою ElasticSearch та простих SQL запитів.

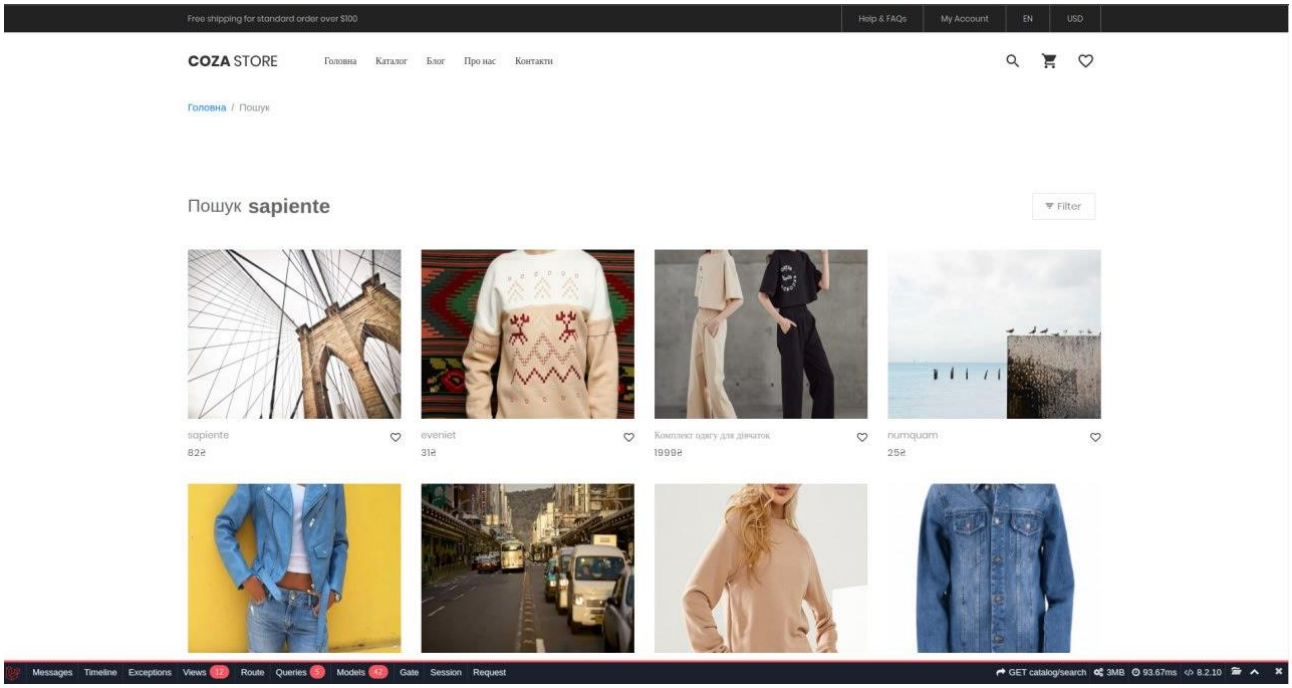


Рисунок 3.5 – Пошук за допомогою сервісу ElasticSearch

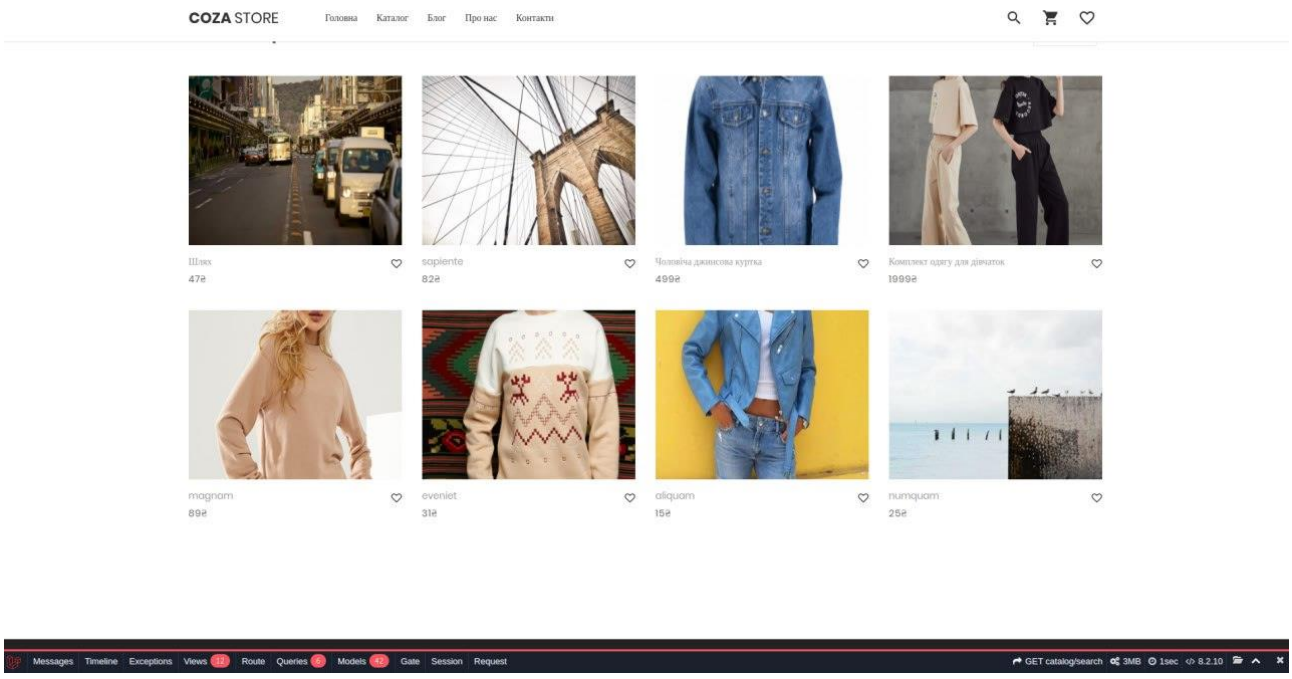


Рисунок 3.6 – Пошук за допомогою SQL запитів

На цих рисунках, можна побачити на скільки оптимізованіше працює пошук за допомогою механізму ElasticSearch, ніж це просто реалізовано за допомогою SQL запитів. Також, на цих рисунках можна побачити на скільки краще ElasticSearch справляється з пошуком в плані збігів в словах.

Код, який дозволяє шукати товари еластично показано в додатку А та лістингу 3.6. Реалізовано так, щоб коли ElasticSearch не відповідає, то пошук відбувається за допомогою SQL. Загалом тут також реалізовано агрегацію та сортування. ElasticSearch також іноді використовують тільки для того, щоб реалізувати агрегацію, тому, що, це доволі важкі запити через SQL, а ElasticSearch ці дані просто індексує десь один раз в день (ліст. 3.6), для оновлення своїх внутрішніх даних, і ніяких запитів тоді не іде до бази даних при пошуку, відповідно сайт набагато оперативніше працює, а це вже означає, що більше користувачів залишаться на сайті, тобто не покинуть його за його довге завантаження, та й Google таке краще піднімає в перші місця пошуку.

Лістинг 3.6 – Реалізація індексування сутностей, для подальшого пошуку

```
class UpdateIndexElasticSearch
{
    use AsAction;

    private $elasticsearch;

    public function __construct(Client $elasticsearch)
    {
        $this->elasticsearch = $elasticsearch;
    }

    public function handle($model, $operation = 'index')
    {
        try {
            if ($operation === 'index') {
                $this->elasticsearch->index([
                    'index' => $model->getSearchIndex(),
                    'type' => $model->getSearchType(),
                    'id' => $model->getKey(),
                    'body' => $model->toSearchArray(),
                ]);
            } elseif ($operation === 'delete') {
                $this->elasticsearch->delete([
                    'index' => $model->getSearchIndex(),
```

```
                    'type' => $model->getSearchType(),
                    'id' => $model->getKey(),
                ]);
            }
        } catch (\Exception $e) {
            Log::info($e->getMessage());
        }
    }
}
```

Кінець лістингу 3.6

При реалізації оптимізованого еластичного пошуку виникало багато моментів, які потрібно було враховувати та виправляти. Наприклад, коли віддавалися посортовані id товарів, я їх виводив за допомогою бази даних. Дані можна віддавати і за допомогою того що проіндексовано, але це є не хорошою практикою, тому що, тоді буде писатися багато перевірок, що негативно вплине на оптимізацію сайту. Момент виникав в тому, що дані витягувалися не в тому порядку, що приходили від пошуку, метод «whereIn» їх сортував по своєму, тому потрібно було використати «orderByRaw», що вирішило дану проблему.

Укрпошта – це компанія, яка надає послуги поштового зв’язку. Це найбільша пошта на території України. За її допомогою доставляються посилки по всій Україні, і не тільки. Вона надає доволі приємну документацію в двох виглядах.

Для реалізації API Укрпошти я не використовував жодного пакету, тому іноді було потрібно зв’язуватися з підтримкою яка доволі швидко відповідала. В документації Укрпошти є можливість створювати TTN для швидкої відправки посилок. Також, це API надає можливість отримувати області, міста, всіх доступних відділень та пунктів доставки. Відповідно, Укрпошта надає маршрутизацію для можливості реалізації трекінгу відправлень. На рисунку 3.7 показано сформований TTN, який готовий до використання.

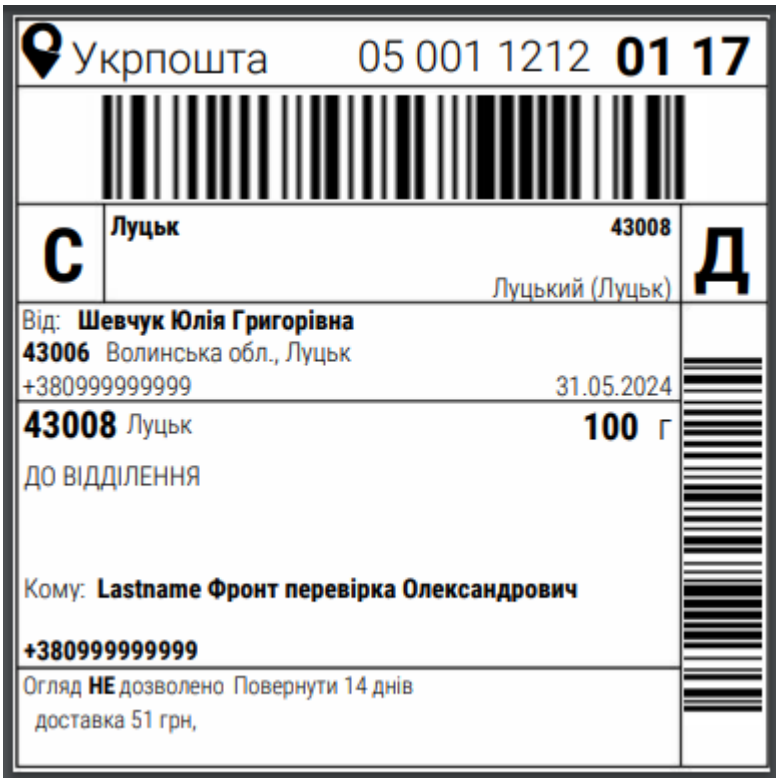


Рисунок 3.7 – Сформований ТТН від Укрпошти

Таким вагомим мінусом даної API, є те, що, для того щоб реалізувати отримання даних областей, населених пунктів та відділень потрібно мати Bearer токени. Реалізований функціонал створення ТТН, показано в додатку Б.

### 3.2 Розробка бази даних

База даних – це архітектурне збереження даних, яке використовуються для виведення певної інформації. За допомогою її, будь-який веб-сайт стає набагато динамічнішим та гнучким. Без бази даних не можливе створення будь-якого динамічного веб-сайту, мобільного додатку або ж програми.

Дані можна зберігати в таких сервісах як Microsoft Access, SQlite, MySQL і тому подібне. Веб-сайт «CozaStore» використовує реляційну базу даних MySQL, як і в більшості других веб-сайтів.

Сьогодні, світ дуже швидко розвивається, тому дані, які потрібно якось більше шифрувати, є можливість зберігати на других серверах, які відповідають за це. Наприклад єдиний портал державних послуг «Дія» використовує саме

такий підхід. VoIP-застосунок обміном повідомлення та відеозвінками Viber, зберігає дані на вашому ж пристрої, що з однієї сторони добре, що всі дані належать тільки вам, але при переносі даних можуть виникнути проблеми з відновленнями їх.

При реалізації веб-застосунку з еластичним пошуком було використано вебдодаток «phpMyAdmin». Загалом, «phpMyAdmin» це графічний інструмент, написаний за допомогою мови програмування PHP (рис. 3.8). Він був розроблений для полегшення користування збереженими даними адміністраторам. Оскільки без графічного інструменту потрібно більше розбиратися командами терміналів, що займає теж не мало часу, щоб розібратися в цьому.

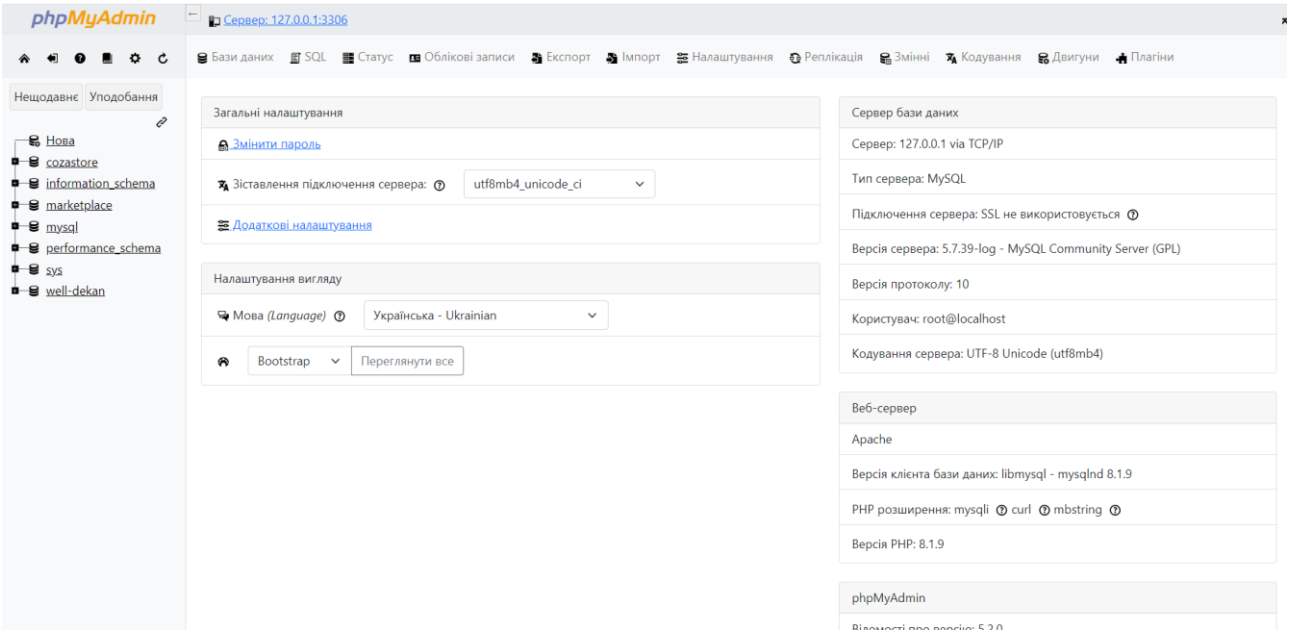


Рисунок 3.8 – Графічний інструмент «phpMyAdmin»

Реалізація будь-якого динамічного застосунку розпочинається з роздумів над схемою бази даних. Оскільки різні веб-сайти, мобільні застосунки, комп’ютерні програми потребують збереження різних даних, тому перед тим як реалізовувати зв’язки та поля в базі даних потрібно спочатку зобразити їх графічно. Тут можна використовувати багато різних сервісів, програм, але на мою думку самим хорошим інструментом для цього є MySQL Workbench. Він

дозволяє реалізовувати приблизні схеми база даних зі зв'язками, та експортувати їх в будь-який зручний для вас формат.

Побудова схеми даних для «CozaStore» зайняла не мало часу, оскільки потрібно було продумати, які потрібні рядки для її, а, що краще розділити на різні таблиці. В кінцевому результаті вийшла доволі  не маленька схема бази даних, що зображена на рисунку 3.9.

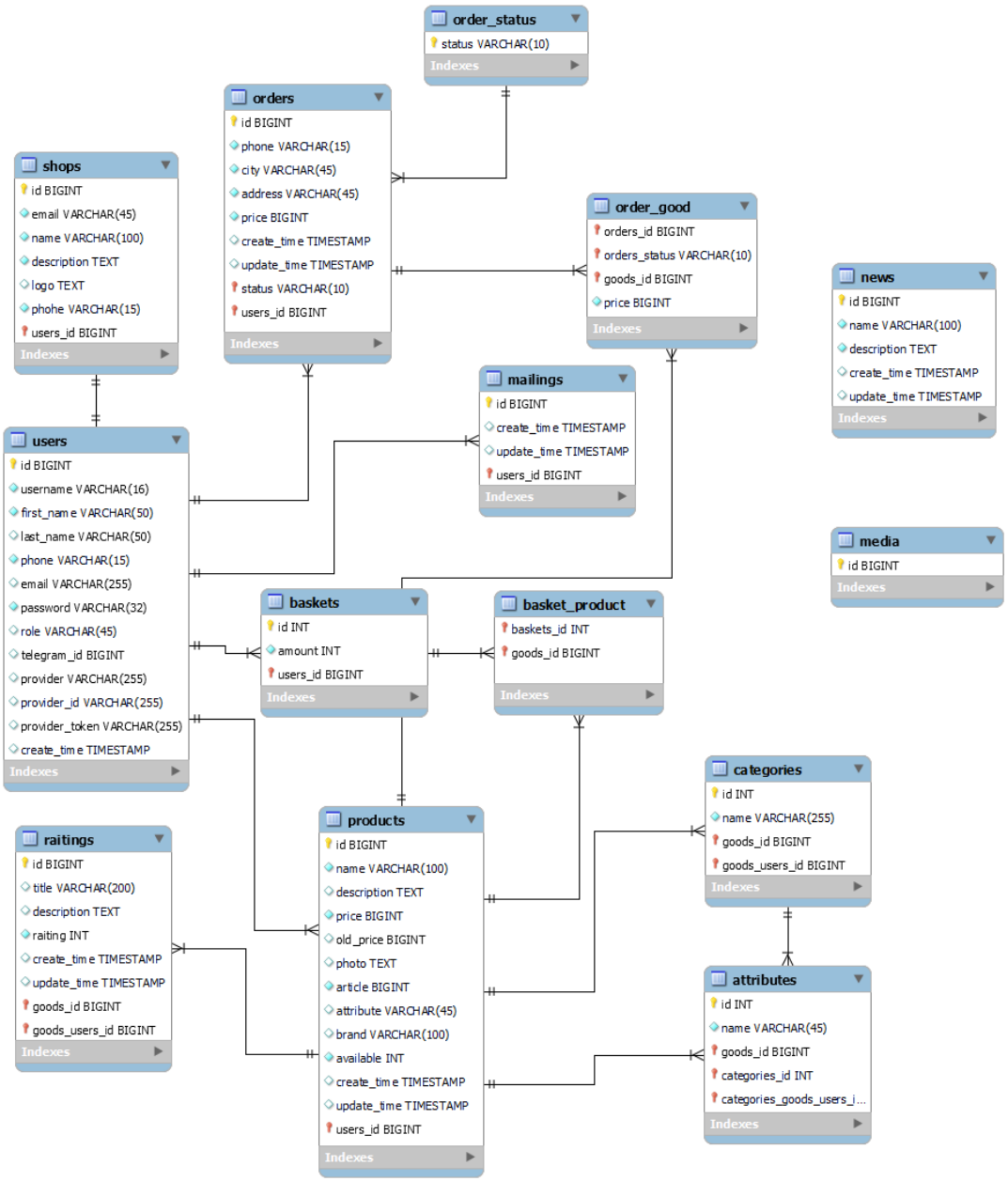


Рисунок 3.9 – Схема бази даних

Далі, потрібно це все реалізувати в моделях та міграціях Laravel. Laravel надає доволі широкі можливості для роботи з базами даних. В створених моделях описуються рядки, які повинні бути, але якщо, прочитати добре документацію, то можна знайти метод, який дозволяє прописати тільки одне поле а не всі, це називається магією Laravel. В лістингу 3.7 описано як реалізовуються зв’язки з різними сутностями.

Лістинг 3.7 – Зв’язки між сутностями

```
//Масове заповнення атрибутів в Моделі Product
protected $guarded = [
    'id',
];

protected $casts = [
    'added' => 'array',
];

// Відношення One-to-Many до Моделі Shop
public function shop()
{
    return $this->belongsTo(Shop::class);
}

// Відношення Many-to-Many до Моделі Comment
public function properties()
{
    return $this->belongsToMany(Property::class);
}

// Відношення Many-to-One до Моделі Category
public function category()
{
    return $this->belongsTo(Category::class, 'category_id');
}
```

Кінець лістингу 3.7

Для підключення даних між фреймворком Laravel та базою даних, використовуються міграції. В них описується більш строга типізація, для того, щоб користувачі не зберігали в базі дані, які не підходять під ті поля (ліст. 3.8).

Лістинг 3.8 – Реалізація міграцій для повної функціональності

```
return new class extends Migration
{
/**
 * Run the migrations.
 */
    public function up(): void
    {
        Schema::create('products', function (Blueprint $table) {
            $table->id();
            $table->string('slug')->index();
            $table->string('name');
            $table->text('description')->nullable();
            $table->float('price', 11, 2)->unsigned(); // unsigned не
відємне значення
            $table->float('old_price', 11, 2)->nullable();
            $table->string('article')->nullable();
            $table->boolean('available')->default(true)->nullable();
            $table->text('youtube')->nullable();

            //Звязок з Міграцією Shops
            $table->foreignId('shop_id')->nullable()->constrained()-
>cascadeOnDelete();
            //Звязок з Міграцією Categories
            $table->foreignId('category_id')->constrained();

            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     */
    public function down(): void
    {
        Schema::dropIfExists('products');
    }
};
```

Кінець лістингу 3.8

Laravel відразу сам генерує стовпці та сутності в графічному інтерфейсі «phpMyAdmin» за допомогою з’єднаних даних, які прописані в файлі «.env».

### 3.3 Розробка документації для програмного продукта

Розробка документації, є обов'язковою частиною розробки будь-якої API частини веб-сайтів. Документація надає розробникам розуміння по якому роуті потрібно звертатися, щоб отримати певні дані, які потрібно передати query параметри.

Реалізація Elasticsearch та Укрпошти також відбувається за допомогою API. Що там, що там, документація є доволі великою, але добре розписаною. Сервісам вигідно робити своє API, тому що, це для них їх заробіток, що саме їх сервісом будуть користуватися.

Для реалізації API роутів, для надання даних розробникам, Laravel надає можливість, це писати в контролері та ресурсах. Контролери я описував раніше, а ресурси це всього лише контролювання, як будуть приходити дані, при зверненні по маршруту. В лістингу 3.9 показано, як за допомогою Laravel описуються ресурси, за допомогою яких, реалізовуються інші застосунки.

Лістинг 3.9 – Опис формату даних для API роуту

```
class ProductResource extends JsonResource
{
    /**
     * Transform the resource into an array.
     *
     * @return array<string, mixed>
     */
    public function toArray(Request $request): array
    {
        return [
            'id' => $this->id,
            'name' => $this->name,
            'description' => $this->description,
            'price' => $this->price,
            'old_price' => $this->old_price,
            'youtube' => $this->youtube,
            'shop_id' => $this->shop_id,
            'favorite' => \Favorite::isFavorite($this->resource),
            'created_at' => $this->created_at,
            'category' => $this->whenLoaded('category',
CategoryResource::make($this->category)),
            'photos' => $this->whenLoaded('media', fn() =>
```

```
MediaResource::make($this->getFirstMedia('photo')))
        ];
    }
}
```

Кінець лістингу 3.9

Для опису документації, я обрав пакет APIDOC [18]. Він є безкоштовним та легкий у використанні з добре описаною документацією. В лістингу 3.10 показано, як встановити пакет, за допомогою команди.

Лістинг 3.10 – Опис документації для розробників

```
composer require --dev mpociot/laravel-apidoc-generator
```

Кінець лістингу 3.10

В лістингу 3.11 показано як за допомогою цього пакету описується документація.

Лістинг 3.11 – Опис документації для розробників

```
/**
 * @api {get} api/products 1. List
 * @apiName GetProducts
 * @apiGroup Products
 *
 * @apiExample {curl} Example url:
 *     /api/products/search?q=ratione&category=2&order=desc&sort=name
 *
 *
 * @apiParam {Number} [page] Number Page
 * @apiParam {Number} [per_page] Per Page
 * @apiParam {String} [q] Search string
 * @apiParam {Number} [category] Category ID
 * @apiParam {String=desc,asc} [order] Sort Direction
 * @apiParam {String=name,article,price} [sort] Sort Field
 */
```

Кінець лістингу 3.11

На рисунку 3.10 проілюстровано, як виглядає зовнішньо загалом вся документації.

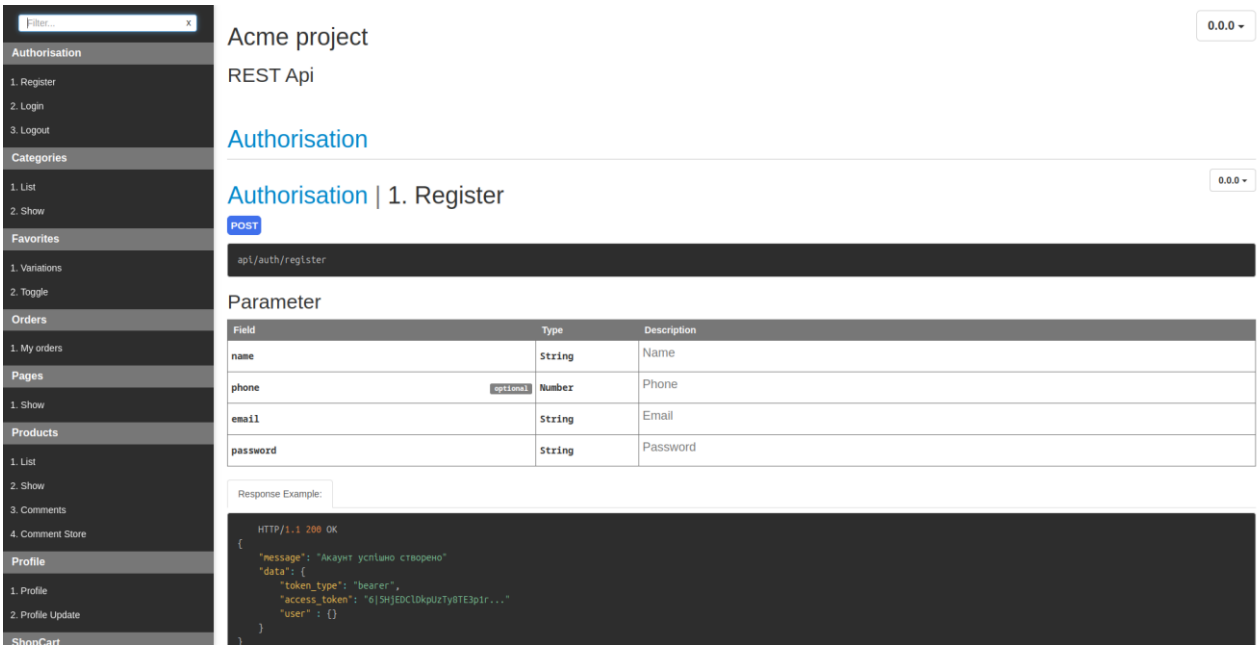


Рисунок 3.10 – Зовнішній вигляд API документації

В даній документації, вже описано авторизацію користувача, категорії, улюблені, замовлення, продукти, дані користувача та багато інакшого. Це надає іншим розробникам користуватися даною API.

### 3.4 SEO інформаційно-комп’ютерної системи

SEO – це великий обсяг роботи, який робиться для покращення видимості сайту в пошуковиках. SEO є важливою частини розробки веб-сайту, бо саме за її допомогою можна отримати багато безкоштовної цільової аудиторії. Чому ж безкоштовної? Вся справа в алгоритмах пошукових систем. При індексуванні сайтів, система по своїх алгоритмах дивиться на скільки сайт є добре оптимізованим, та наскільки якісно написаний його вміст всередині.

Загалом робота пошукових систем являє собою такі 3 основні етапи:

– сканування – це процес, під час якого, пошукові роботи переглядають та аналізують сторінки веб-сайтів;

– індексування – це процес обробки сканованих сторінок і додавання їх до бази даних;

– ранжування – це процес розташування сторінок у результатах пошуку на основі їхньої відповідності пошуковому запиту. Алгоритми, які виконують цю роботу, є дуже складними.

Потрібно розуміти, що ми живемо в епоху швидкого розвитку, тому правила, алгоритми і тому подібне розвивається та міняється доволі швидко. Якщо порівняти алгоритми над SEO 2018 року і в сьогоденні, то це дужу великі зміни.

Оскільки мною було вибрано розробку мультиплатформенного веб-застосунку, то тут SEO є головним аспектом для підняття його в рейтингу. Для реалізації SEO-метатегів було використано пакет від розробника «fomvasss», а саме «Laravel-seo». Для використання даного пакету, потрібно підключити Trait «HasSeo», та в моделі викликати метод «registerSeoDefaultTags». В лістингу 3.12 більш детально описано реалізація SEO-метатегів за допомогою пакету.

Лістинг 3.12 – Реалізація SEO-метатегів за допомогою пакету

```
namespace App\Models;

use Fomvasss\Seo\Models\HasSeo;

class Product extends Model implements HasMedia
{
  public function registerSeoDefaultTags(): array
  {
    return [
      'title' => $this->seo->tags['title'] ?? $this->name,
      'description' => $this->seo->tags['description'] ?? $this-
        >description,
      'og_image' => $this->getFirstMediaUrl('photo') ??
        asset('/public.default.marketplace') ?? null,
    ];
  }
}
```

Кінець лістингу 3.12

Для того, щоб, відрендерити метатеги для пошукових систем пакет дає змогу використовувати як статично задані теги, так і динамічно. Для статично

заданих використовується метод «setDefault». Оскільки сторінка товару є динамічно реалізованою сутністю, то я використовуватиму метод «setModel», а саме «\Seo::setModel($product)». Тобто, при відкритті любого товару, буде відкриватися унікальна інформація в тегах. На рисунку 3.11 показано, як за допомогою адмін-панелі, можна задати свої метатеги для кожного товару. По замовчуванні, теги заповнюються з інформації товару.

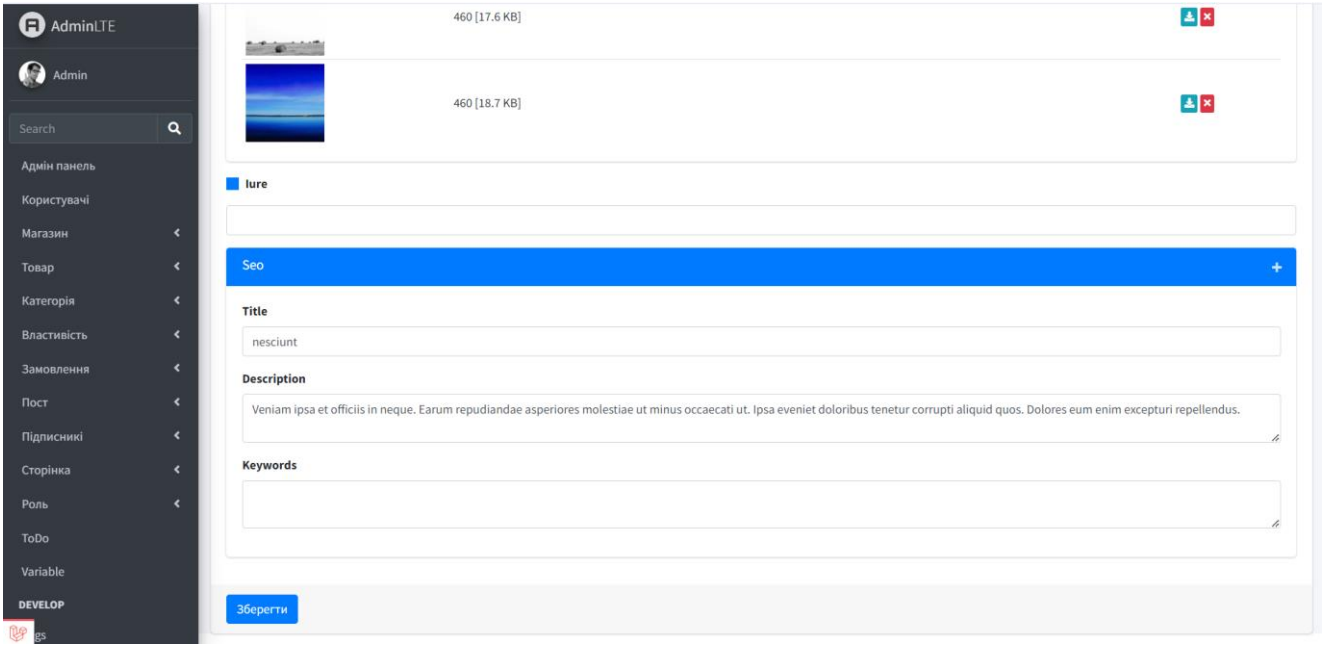


Рисунок 3.11 – Зміна мета тегів за допомогою адмін-панелі

В адмін-панелі задаються мета теги для контролю інформації, тому що, алгоритми працюють так, коли переповнено інформацією в них, то система в такому випадку просто ігнорує цей вміст, відповідно, товар, статтю чи будь-яку іншу сутність, просто не буде проіндексовано.

На рисунку 3.12 зображено вигляд SEO мета тегів одного товару, та як вони виводять на веб-сайті, в консолі розробника.

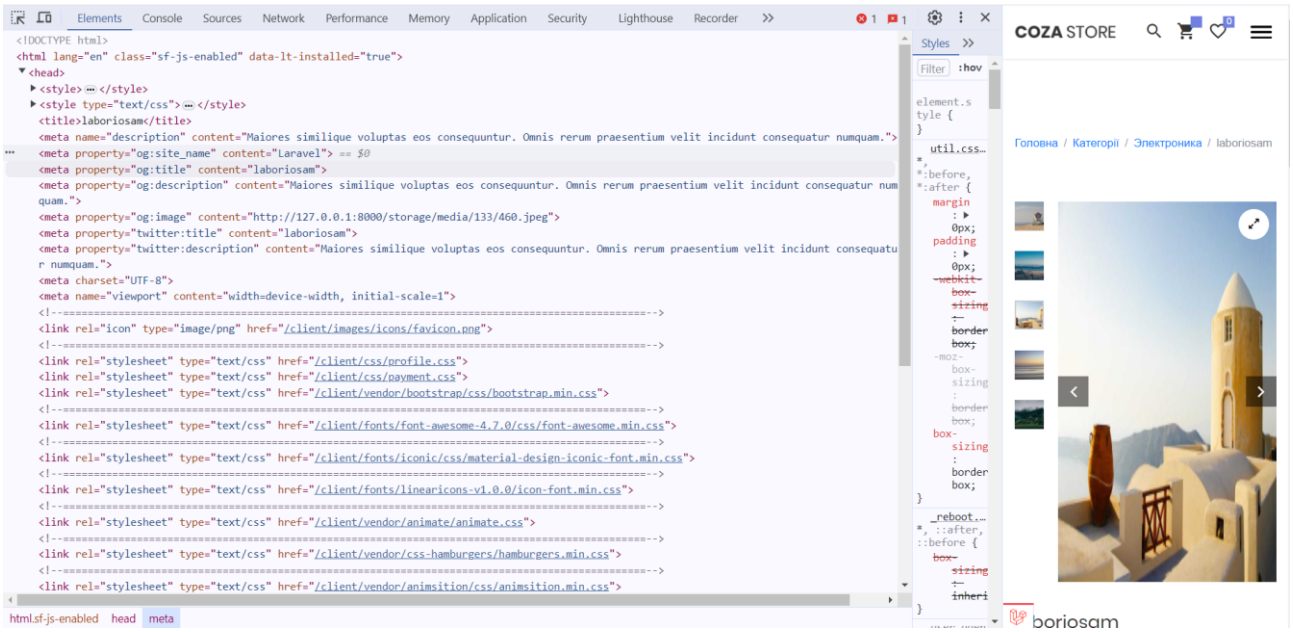


Рисунок 3.12 – Вигляд SEO мета тегів на сторінці товару

Завершуючи про SEO мета теги, можу сказати, що їх доволі легко реалізовувати, якщо знати ціль проєкту, та, що туди має входити.

**Висновки до розділу 3**

В цьому розділі, я розробив правильно реалізовану архітектуру файлів у frontend частині з HTML розміткою, яка зроблена для кращого читання другим розробникам. Backend частина навчила мене багато чому, як наприклад, що реалізація бізнес-логіки повинна бути винесена у сервісні класи, іншими словами «Фасади». Реалізована маршрутизація, надала мені більшого розуміння, як загалом працює будь який динамічний веб-сайт. Загалом, за допомогою аналізу інформації, я зрозумів, як правильно повинна бути реалізована архітектура та чому саме таким чином.

При розробці оптимізованого механізму системи пошуку, в веб-застосунку за допомогою ElasticSearch, для мультиплатформенного використання я зрозумів, наскільки важливо дотримуватися алгоритмів розробки, для якомога більшої оптимізації, не тільки еластичного пошуку, а й будь-якого функціоналу, тому що, це, як пришвидшує роботу платформи так і

більш виглядає для читання іншим розробникам. При реалізації еластичного пошуку за допомогою сервісу ElasticSearch, я зрозумів, наскільки важливим є реалізований пошук на сайті для клієнтів, оскільки саме за допомогою нього, шукаються ті чи інші сутності.

При реалізації, на мою думку найважливішої частини, а саме бази даних, я зрозумів, що дуже важливо, перед тим, як почати реалізовувати веб-сайт або ж якусь систему, дуже важливо розробити схему, за допомогою якої, на наступних етапах розробки стане набагато зрозуміліше, який функціонал повинний бути реалізований.

# ВИСНОВКИ

У цій кваліфікаційній роботі бакалавра було поставлено за мету розробити механізм пошуку в веб-застосунку за допомогою ElasticSearch для мультиплатформенного використання. Для досягнення цієї мети потрібно було дотримуватися порядку виконання завдання. Для дотримання цих порядків було проаналізовано та створено пункти, які допоможуть реалізувати цю ціль.

На початку виконання роботи, було проаналізовано та ознайомлено з документацією PHP та її основних функцій, таких як «array», «array_column», «array_chunk» та інших. Далі, було вивчено та отримано, практичні навики в Laravel, а саме, що таке «Helpers», MVC модель, міграції та інше. В наступному етапі, було побудовано схему бази даних з зв’язками за допомогою інструменту «MySQL Workbench». Також, було реалізовано адмін-панель за допомогою безкоштовного адміністративного шаблону «AdminLTE 3», з системою CRUD, для керування контентом сайту. В наступному етапі було розроблено REST API та системи «Resources», для коректного надання даних розробникам. Було реалізовано користувацький інтерфейс, за допомогою таких технологій як: HTML, CSS, JS, JQuery та Bootstrap. Наступним кроком, було реалізовано серверну частину, за допомогою PHP та фреймворку Laravel. Були створені фасади для написання та винесення з контролерів бізнес-логіки. Далі було встановлено та вбудовано REST API ElasticSearch, за допомогою пакету «elasticsearch/elasticsearch». Останнім, що було зробленим, це реалізація створення ТТН за допомогою документації Укрпошти та запитів.

Отже, кінцевою реалізацією стало впровадження ефективного механізму пошуку в веб-застосунку, який значно підвищує швидкість та точність пошукових запитів. Також, створено інтуїтивно зрозумілий користувацький інтерфейс та зручну адмін-панель для керування контентом.

# СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. ElasticSearch: The Official Distributed Search & Analytics Engine. Elastic. URL: https://www.elastic.co/elasticsearch (дата звернення: 21.04.2024).

2. Пошуковий сервіс з ШІ. Algolia. URL: https://www.algolia.com/ (дата звернення: 22.04.2024).

3. Typesense. Typesense | Open Source Alternative to Algolia + Pinecone. URL: https://typesense.org/ (дата звернення: 23.04.2024).

4. Meilisearch Documentation. URL: https://www.meilisearch.com/docs (дата звернення: 24.04.2024).

5. Laracasts. Laracats – платформа з відео уроками для вивчення Laravel : веб–сайт. URL: https://laracasts.com/ (дата звернення: 30.04.2024).

6. October CMS. October – PHP CMS platform based on the Laravel Framework. URL: https://octobercms.com/features (дата звернення: 30.04.2024).

7. Invoice Ninja – Create. Send. Get Paid. Free Invoicing Software for Small Businesses | Invoice Ninja. URL: https://invoiceninja.com/ (дата звернення: 30.04.2024).

8. Attendize | Attendize documentation site. Introduction. URL: https://www.attendize.com/index.html (дата звернення: 30.04.2024).

9. Blog – Flarum Community. Flarum Community. URL: https://discuss.flarum.org/t/blog (дата звернення: 30.04.2024).

10. Laravel – PHP Фреймворк. Laravel – The PHP Framework: веб–сайт. URL: https://laravel.com/ (дата звернення: 30.04.2024).

11. AdminLTE 3 | Dashboard. Free Bootstrap Admin Template – AdminLTE.IO. URL: https://adminlte.io/themes/v3/ (дата звернення: 20.04.2024).

12. Vue.js | Vue.js. – Прогресивний JavaScript фреймворку URL: https://ua.vuejs.org/ (дата звернення: 16.05.2024).

13. React. URL: https://react.dev/ (дата звернення: 18.05.2024).

14. MySQL Workbench. URL: http://surl.li/uiaad (дата звернення: 20.05.2024).

15. API: що це простими словами, приклади. PROSEO. URL: http://surl.li/uiaaf (дата звернення: 20.05.2024).

16. Microsoft. Visual Studio Code – Code Editing. Redefined. URL: https://code.visualstudio.com/ (дата звернення: 21.05.2024).

17. Packagist. URL: http://surl.li/uiaak (дата звернення: 22.05.2024).

18. apiDoc. URL: https://apidocjs.com/ (дата звернення: 26.05.2024).

# ДОДАТКИ

# Додаток А

## Реалізація функціоналу еластичного пошуку

```
public function buildQuery($params)
    {
        // Підключення до API Elasticsearch
        $client = ClientBuilder::create()
            ->setHosts(['elasticsearch:9200'])
            ->build();

        $name = request('q');
        $sort = request()->query('sort');
        $sort_order = request()->query('order');
        $price_at = request()->query('price_at');
        $price_to = request()->query('price_to');

        $offset = $params['offset'] ?? 0;
        $perPage = $params['perPage'] ?? 1000;

        $query = [
            'body' => [
                'size' => $perPage,
                'from' => $offset,
            ],
        ];

        // Пошук по імені та опису продукта
        if ($name) {
            $query['body']['query'] = [
                'bool' => [
                    'should' => [
                        [
                            'multi_match' => [
                                'fields' => ['name^3', 'description'],
                                'query' => $name,
                            ],
                        ],
                    ],
                ],
            ];
        }

        // Ці переліки потрібні для того щоб перевіряти чи правильно
введені поля по яких дозволяється сортувати
        $sortableFields = ['created_at', 'price'];
        $sortOrders = ['asc', 'desc'];
        // Сортування за такими полями: created_at, price
        if (in_array($sort, $sortableFields) && in_array($sort_order,
$sortOrders)) {
```

```
            $query['body']['sort'] = [
                $sort => [
                    'order' => $sort_order,
                ],
            ];
        }

        if ($price_at && $price_to) {
            $query['body']['aggs'] = [
                'price_ranges' => [
                    'range' => [
                        'field' => 'price',
                        'ranges' => [
                            ['from' => $price_at, 'to' => $price_to],
                        ],
                    ],
                    'aggs' => [
                        'price_stats' => [
                            'stats' => [
                                'field' => 'price',
                            ],
                        ],
                    ],
                ],
            ];
        }

        if ($price_at && $price_to) {
            $query['body']['query']['bool']['filter'][] = [
                'range' => [
                    'price' => [
                        'gte' => $price_at,
                        'lte' => $price_to,
                    ],
                ],
            ];
        }

        $query['body']['aggs'] = [
            'attributes' => [
                'terms' => [
                    'field' => 'attributes.name.keyword',
                ],
                'aggs' => [
                    'properties' => [
                        'terms' => [
                            'field' =>
'attributes.properties.name.keyword',
                        ],
                    ],
                ],
            ],
```

```
        ];

        // Видає інформацію про: took, timed_out, _shards, hits
        $response = $client->search($query);
//        dd($response);

        // Тут масиви в якому лежить інформація про: _index, _type, _id,
_score, _source
        $productsElasticsearch = $response['hits']['hits'];

        $productsAggregation =
$response['aggregations']['attributes']['buckets'];

        // Тут id в типу string
        $productIds = array_map(static function ($product) {
            return $product['_id'] ?? null;
        }, $productsElasticsearch);

        // Перевірка, чи $productIds порожній
        $productIds = !empty($productIds) ? $productIds : [0];

        // Шукаємо продукти в БД, для того щоб була завжди оновлення інфа
        $collections = Product::whereIn('id', $productIds)
            ->orderByRaw('FIELD(id, ' . implode(',', $productIds) . ')')
            ->get();


        return [
            'products' => $collections,
            'aggregations' => $productsAggregation,
        ];
    }
```

## Додаток Б

## Реалізація функціоналу створення TTN для Укрпошти

```
public function ttnStore(Request $request)
    {
        $response_senderAddress = Http::withToken(env('Ukrposhta'))
            ->post('https://dev.ukrposhta.ua/ecom/0.0.1/addresses', [
                'postcode' => intval(request()-
>shipping['ukrposhta']['sender']['postcode']),
            ]);

        $response_recipientAddress = Http::withToken(env('Ukrposhta'))
            ->post('https://dev.ukrposhta.ua/ecom/0.0.1/addresses', [
                'postcode' => intval(request()-
>shipping['ukrposhta']['recipient']['postcode']),
            ]);

        $response_senderClient = Http::withToken(env('Ukrposhta'))
            ->withQueryParameters([
                'token' => '8c06fcce-69e7-4f56-944a-9fde92ed9696'
            ])
            ->post('https://dev.ukrposhta.ua/ecom/0.0.1/clients', [
                'firstName' => request()->sender_firstName,
                'lastName' => \request()->sender_lastName,
                'addressId' => json_decode($response_senderAddress-
>body())->id,
                'phoneNumber' => request()->sender_phoneNumber,
                'type' => "INDIVIDUAL"
            ]);

        $response_recipientClient = Http::withToken(env('Ukrposhta'))
            ->withQueryParameters([
                'token' => '8c06fcce-69e7-4f56-944a-9fde92ed9696'
            ])
            ->post('https://dev.ukrposhta.ua/ecom/0.0.1/clients', [
                'firstName' => request()->recipient_firstName,
                'lastName' => \request()->recipient_lastName,
                'addressId' => json_decode($response_recipientAddress-
>body())->id,
                'phoneNumber' => request()->recipient_phoneNumber,
                'type' => "INDIVIDUAL"
            ]);

        $response_shipment = Http::withToken(env('Ukrposhta'))
            ->withQueryParameters([
                'token' => '8c06fcce-69e7-4f56-944a-9fde92ed9696'
            ])
            ->post('https://dev.ukrposhta.ua/ecom/0.0.1/shipments', [
                'sender' => [
                    'uuid' => json_decode($response_senderClient-
```

```
>body())->uuid
                ],
                'recipient' => [
                    'uuid' => json_decode($response_recipientClient-
>body())->uuid
                ],
                'deliveryType' => "W2W",
                'parcels' => [
                    [
                        'weight' => \request()->weight,
                        'length' => \request()->length,
                        'width' => \request()->width,
                        'height' => \request()->height,
                        'declaredPrice' => \request()->declaredPrice,
                        'description' => \request()->description,
                    ]
                ],
                'type' => \request()->type,
                'paidByRecipient' => \request()->paidByRecipient,
//                'availablePaymentTypes' => \request()-
>availablePaymentTypes,
            ]);

        $response_form119 = Http::withToken(env('Ukrposhta'))
            ->withQueryParameters([
                'token' => '8c06fcce-69e7-4f56-944a-9fde92ed9696',
            ])
            ->get('https://dev.ukrposhta.ua/forms/ecom/0.0.1/shipments/'
. json_decode($response_shipment->body())->uuid . '/sticker');

        Storage::put('sticker.pdf', $response_form119->body());
    }
```