–ЕКОНОМІКО-ТЕХНОЛОГІЧНИЙ ІНСТИТУТ ІМЕНІ РОБЕРТА ЕЛЬВОРТІ
КАФЕДРА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

«Допущено до захисту»
Завідувач кафедри інформаційних технологій
_____ Бондар О.П.
«19» ____06____ 2024 р.

Кваліфікаційна робота

на здобуття ступеня вищої освіти «бакалавр» зі спеціальності 122 «Комп'ютерні науки»

на тему:

«Розробка інтернет-магазину дронів за допомогою Laravel»

Приймак Дмитро Олександрович

Керівник кваліфікаційної роботи:
Неділько Віталій Миколайович, кандидат технічних наук, доцент

Роботу рекомендовано до захисту
на засіданні кафедри інформаційних технологій
Протокол №10 від «06» 06 2024р.
Завідувач кафедри інформаційних технологій
_____ Бондар О.П.

Роботу захищено на засіданні ЕК
з оцінкою
добре / С / 74
(за національною шкалою, шкалою ECTS, бали)

Протокол №8 від «20» 06 2024 р.
Голова ЕК _____

Кропивницький – 2024

АНОТАЦІЯ

Приймак Д.О. Розробка інтернет-магазину дронів за допомогою Laravel – Кваліфікаційна робота зі спеціальності 122 «Комп'ютерні науки». – Економіко-технологічний інститут імені Роберта Ельворті, Кропивницький, 2024.

Пояснювальна записка: 90 сторінок, 65 рисунків, 1 таблицю, 20 джерел, 1 додаток.

Об'єктом дослідження є інтернет-магазин дронів.

Мета роботи: є аналіз електронної комерції, тобто діяльності інтернет-магазинів, аналіз відмінностей від звичайних магазинів, розгляд аналогічних товарів на ринку та розробка інтернет-магазину «Дрони» для продажу дронів різних категорій.

Відповідно до поставленого завдання в роботі проведено дослідження процесу для створення інтернет-магазину. В результаті виконання цієї роботи створений прототип інтернет-магазину для продажу дронів.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ/СКОРОЧЕНЬ

CSS – Cascading Style Sheets

HTML – HyperText Markup Language

PHP – Hypertext Preprocessor

MySQL – система управління базами даних (СУБД)

UML – Unified Modeling Language

ER – Entity-Relationship model

ВСТУП

Стрімкий розвиток технологій у галузі безпілотних літальних апаратів (БПЛА) або дронів відкриває широкі перспективи для їх застосування в різноманітних сферах діяльності. Дрони знаходять своє застосування в таких галузях, як сільське господарство, будівництво, кінематограф, доставка вантажів, а також у сфері безпеки та оборони. Зростаючий попит на ці інноваційні пристрої зумовлює необхідність створення зручних та функціональних платформ для їх реалізації, серед яких важливу роль відіграють інтернет-магазини.

Актуальність даної кваліфікаційної роботи полягає у розробці сучасного та зручного у використанні інтернет-магазину дронів, який би задовольняв потреби користувачів у швидкому та ефективному пошуку, вибору та придбанні необхідного обладнання. Створення такої платформи дозволить підвищити доступність та зручність придбання дронів, а також сприятиме популяризації їх використання в різних сферах діяльності.

Практичне значення роботи полягає у можливості швидкого впровадження та використання розробленого інтернет-магазину, що забезпечить зручний доступ до широкого асортименту дронів. Крім того, створена платформа може слугувати базою для подальшого розвитку та вдосконалення функціоналу інтернет-магазину відповідно до потреб користувачів та тенденцій ринку.

На сьогоднішній день існує значна кількість інтернет-магазинів, що пропонують дрони та супутні товари. Проте, більшість з них мають обмежений функціонал, незручний інтерфейс або недостатньо широкий асортимент продукції. Тому розробка сучасного, зручного у використанні та функціонального інтернет-магазину дронів є актуальною проблемою, вирішенню якої присвячена дана кваліфікаційна робота.

Мета роботи - є аналіз електронної комерції, тобто діяльності інтернет-магазинів, аналіз відмінностей від звичайних магазинів, розгляд аналогічних товарів на ринку та розробка інтернет-магазину «Дрони» для продажу дронів різних категорій.

Для досягнення поставленої мети необхідно виконати такі завдання:

1. Проаналізувати існуючі рішення у сфері інтернет-магазинів дронів та визначити їх переваги та недоліки.

2. Вивчити особливості та функціональні можливості фреймворку Laravel для розробки веб-додатків.

3. Спроектувати структуру та функціонал інтернет-магазину дронів з урахуванням вимог користувачів та сучасних тенденцій у галузі електронної комерції.

4. Реалізувати розроблений проект з використанням фреймворку Laravel та відповідних технологій.

5. Протестувати роботу інтернет-магазину та усунути виявлені помилки та недоліки.

6. Розробити документацію та інструкції для користувачів та адміністраторів інтернет-магазину.

Об’єктом дослідження є інтернет-магазин дронів.

Предметом дослідження є засоби та методи програмної реалізації веб-сайтів – інтернет-магазинів, з урахуванням критерій:

- доступність інтернет-магазину, простота;
- швидке завантаження інформації;
- зручне розташування елементів на сторінці;
- відсутність складнощів для замовлення товарів або виконання будь-яких інших дій.

Під час виконання кваліфікаційної роботи будуть використані такі методи дослідження:

1. Аналіз та узагальнення існуючих рішень у сфері інтернет-магазинів дронів.

2. Об'єктно-орієнтоване програмування та проектування для розробки структури та функціоналу інтернет-магазину.

3. Тестування та налагодження розробленого додатку для забезпечення його коректної роботи.

Інформаційною базою для виконання кваліфікаційної роботи слугуватимуть наукові публікації, статті, навчальні матеріали, документація фреймворку Laravel, а також інші відкриті джерела інформації, пов'язані з темою дослідження.

1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1. Значення інтернет-магазинів

Інтернет-магазини є невід'ємною частиною сучасного електронного комерції. Вони надають можливість клієнтам здійснювати покупки онлайн, без необхідності фізичного відвідування магазину. Це зручно, швидко та ефективно.

Одним з основних переваг інтернет-магазинів є доступність товарів для широкої аудиторії. Клієнти можуть замовити продукцію з будь-якого місця, де є доступ до Інтернету. Це особливо важливо для тих, хто проживає в віддалених районах або має обмежену можливість фізично відвідувати магазини.

Інтернет-магазини також дозволяють клієнтам порівнювати ціни та характеристики товарів, що допомагає зробити обґрунтований вибір. Крім того, вони забезпечують гнучкість у виборі способу оплати та доставки, що дозволяє задовольнити індивідуальні потреби кожного клієнта.

Проте, варто враховувати деякі ризики, пов'язані з інтернет-магазинами, такі як можливість шахрайства або недостатня якість товарів. Тому перед здійсненням покупки важливо ретельно перевіряти надійність та репутацію магазину [2].

Інтернет-магазини в сучасному світі відіграють важливу роль у розвитку електронної комерції та забезпеченні зручного доступу до товарів та послуг для споживачів. Вони мають значний потенціал як для підприємств, так і для споживачів з точки зору зручності, швидкості та доступності.

По-перше, інтернет-магазини надають підприємствам можливість потрапити на глобальний ринок, не витрачаючи великі кошти на фізичні магазини та інфраструктуру. Це розширює їхні можливості залучення клієнтів та збільшення обсягів продажів, відкриваючи нові перспективи для зростання та розвитку бізнесу.

У другу чергу, для споживачів, інтернет-магазини стають нескінченним джерелом товарів і послуг, які можна придбати зручно та швидко, не виходячи з дому. Це особливо важливо в умовах зростаючого темпу життя та постійного дефіциту часу. Крім того, доступність інтернет-магазинів з будь-якого пристрою, що має підключення до мережі, робить процес покупок ще більш зручним та привабливим для клієнтів.

Інтернет-магазини мають велике значення, оскільки вони надають можливість зручного порівняння товарів, читання відгуків та отримання детальної інформації про продукцію та послуги. Це дозволяє споживачам робити обдумані рішення щодо покупок. Такий підхід спонукає до конкуренції та підвищення якості товарів та обслуговування, сприяючи взаємовигідним відносинам між покупцями та продавцями [20].

З урахуванням цих важливих факторів, розробка інтернет-магазину для дронів стає важливим та перспективним завданням, яке сприятиме не тільки розвитку електронної комерції в цілому, але й задоволенню конкретних потреб споживачів у сфері дронів та пов'язаних з ними товарів та послуг.

1.2. Класифікація інтернет-магазинів

У сучасному світі інтернет-магазини стали невід’ємною частиною електронної комерції. Вони надають споживачам зручний спосіб купувати товари та послуги безпосередньо через Інтернет. Залежно від різних критеріїв, інтернет-магазини можна розділити на різні категорії.

Класифікація інтернет-магазинів важлива для розуміння їх різноманітності та функціональних характеристик.

Залежно від різних критеріїв інтернет-магазини можна розділити на кілька категорій залежно від їх специфіки та цільової групи.

- Перша класифікація базується на спеціальності інтернет-магазину. Деякі магазини спеціалізуються на певних товарах, наприклад електроніці, одязі та косметиці. Інші магазини можуть пропонувати широкий асортимент товарів у різних категоріях. Ця класифікація

допомагає споживачам знайти потрібний магазин, який спеціалізується на конкретному продукті чи послузі. Інтернет-магазини дронів належать до цієї категорії, оскільки вони спеціалізуються на продажу певних типів товарів.

- Другий підхід до класифікації полягає у визначенні типу бізнес-моделі, що лежить в основі інтернет-магазину. Серед таких типів можна виділити маркетплейси, де товари продаються з багатьох різних джерел; прямі продавці, що реалізують свою власну продукцію; підписні сервіси, які надають доступ до товарів або послуг за певну щомісячну плату тощо. Ця класифікація може вплинути на репутацію та довіру споживачів до магазину.

- Третя класифікація вимагає масштабу діяльності інтернет-магазину. Певні онлайн-крамниці мають глобальне охоплення, пропонуючи свої товари та послуги по всьому світу. Натомість, інші обмежуються локальним ринком, обслуговуючи лише визначені регіони чи територію. Ця категоризація є важливою для споживачів, які шукають продукцію залежно від їхнього продажу та конкретних потреб.

Класифікація інтернет-магазинів допомагає споживачам зорієнтуватися у різноманітності пропозицій, знайти потрібний магазин та здійснити покупку зручним способом. Вибір інтернет-магазину залежить від індивідуальних потреб та вимог кожного споживача.

1.3. Аналіз інтернет-магазинів аналогів

Аналіз інтернет-магазинів-аналогів є важливою складовою частиною розробки будь-якого проекту, в тому числі, інтернет-магазина дронів. Цей процес дозволяє нам краще зрозуміти ринок, ідентифікувати сильні та слабкі сторони конкурентів і виявити можливості для вдосконалення нашого продукту.

Під час аналізу ми досліджували інтернет-магазини, що спеціалізуються на продажі дронів та супутнього обладнання. Ми вивчали їхні веб-сайти, функціонал, ціноутворення, асортимент товарів

Одним із найбільш популярних магазинів дронів є “DronTech”. Головна сторінка сайту представлена на рис.1.1

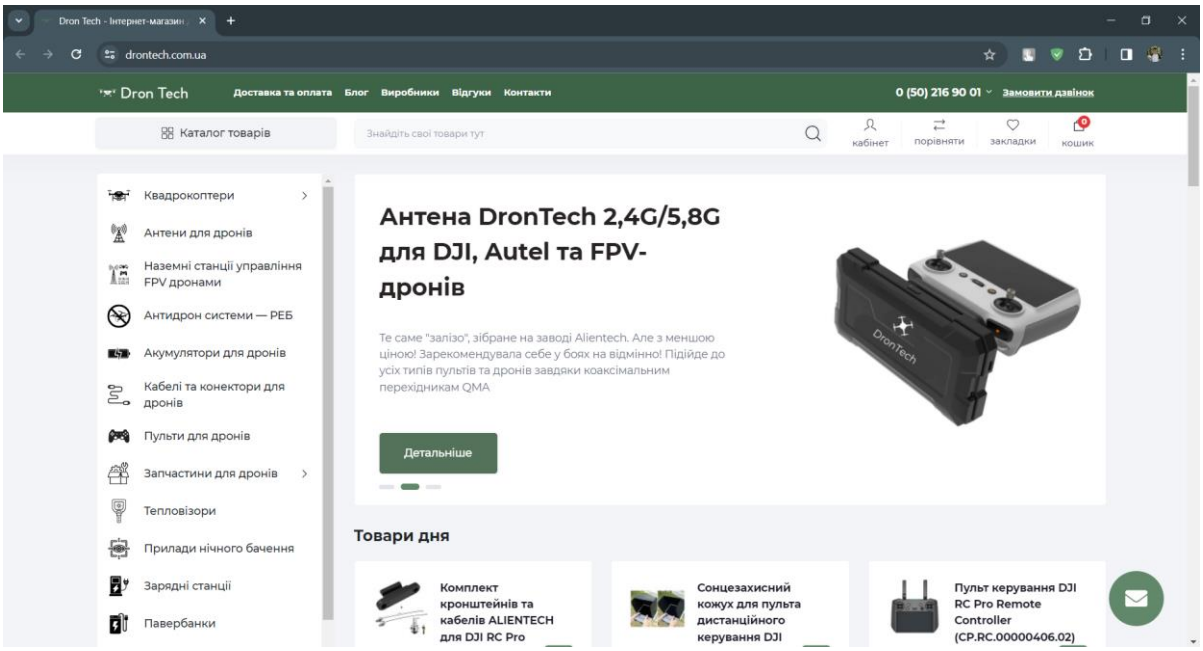

Рис.1.1. – Головна сторінка DronTech

На мою думку, один із мінусів цього сайту є великий банер, який займає, майже, всю ширину сторінки. На ньому представлені випадкові товари, які не зрозуміло, з якої категорії товарів.

Але в свою чергу, це маркетинговий хід. Фото в будь-якому випадку приваблює очі покупця та він починає цікавитися цим товаром. Можливо, є великий відсоток користувачів, які натискають на цей банер та переглядають детальнішу інформацію про певний товар.

Схожий, за функціоналом магазин – “ProDrone”. Він також продає дрони та аксесуари до них. Головна сторінка представлена на рисунку 1.2

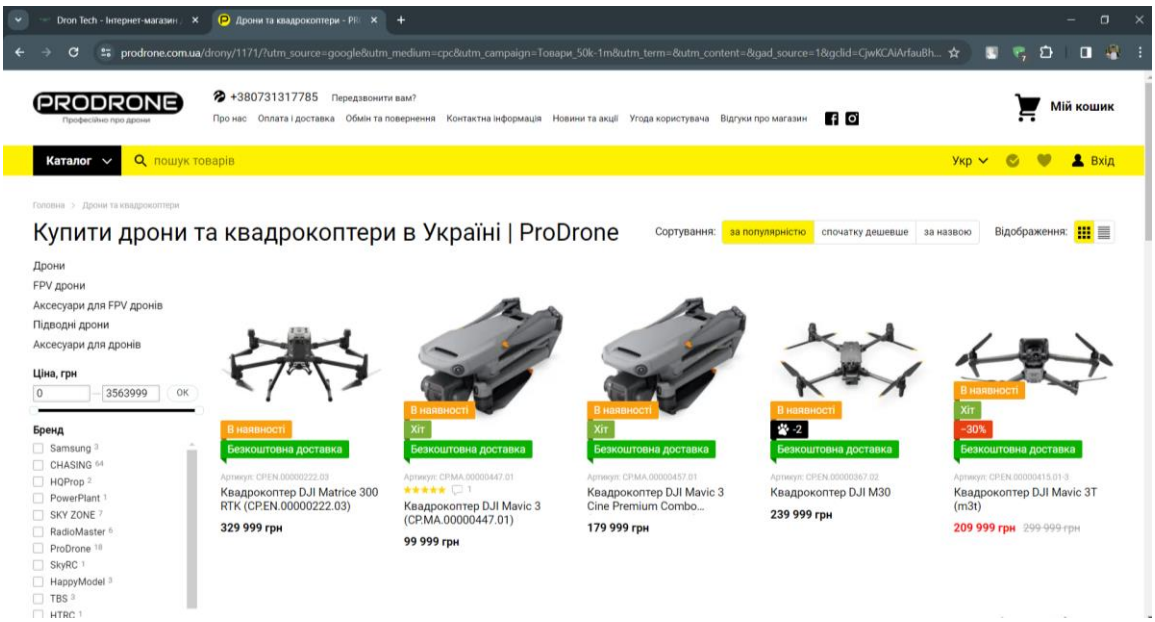

Рис.1.2. – Головна сторінка ProDrone

Перевагою цього сайту є зручний інтерфейс, який дає змогу користувачу відразу фільтрувати товари по потрібним йому пунктам. Також зліва на сторінці, можливо вибрати бренд товару. Вгорі реалізовано меню, яке представляє покупцю інформацію про магазин, оплату та доставку товару тощо.

Але є один недолік, на мій погляд. Не виділене меню, щоб відобразити категорії дронів. Дрони можна вибрати по категоріям, але це меню не дуже помітне для покупця. На мою думку, дану ділянку сторінку, яка представлена на рис.1.3, можна було б якось виокремити.

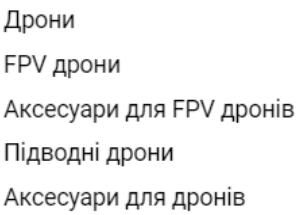

Рис.1.3. Категорії на сайті ProDrone

Також на даному сайті реалізована сторінка «Про нас». В цьому розділі надається відвідувачу сайту інформацію про магазин, а саме:

1. Який перелік товар і від яких брендів можна замовити

2. Сервіс, який можна отримати в результаті замовлення

Але, я вважаю, що даної інформації про магазин мало, адже, продавець повинен викликати довіру до себе та привабити покупця.

Наступний магазин, який ми аналізували – «FlyDron». Даний магазин продає не тільки дрони та аксесуари, а ще і запчастини до них. Я вважаю, що це перевага над іншими магазинами, тому що, не прийдеться шукати продавців, в яких будуть деталі саме до вашого дрона. Відразу можна звернутися до цього інтернет-магазину та фахівці повинні підібрати вам деталь, яку потрібно відремонтувати.

Головна сторінка даного магазина представлена на рис.1.4

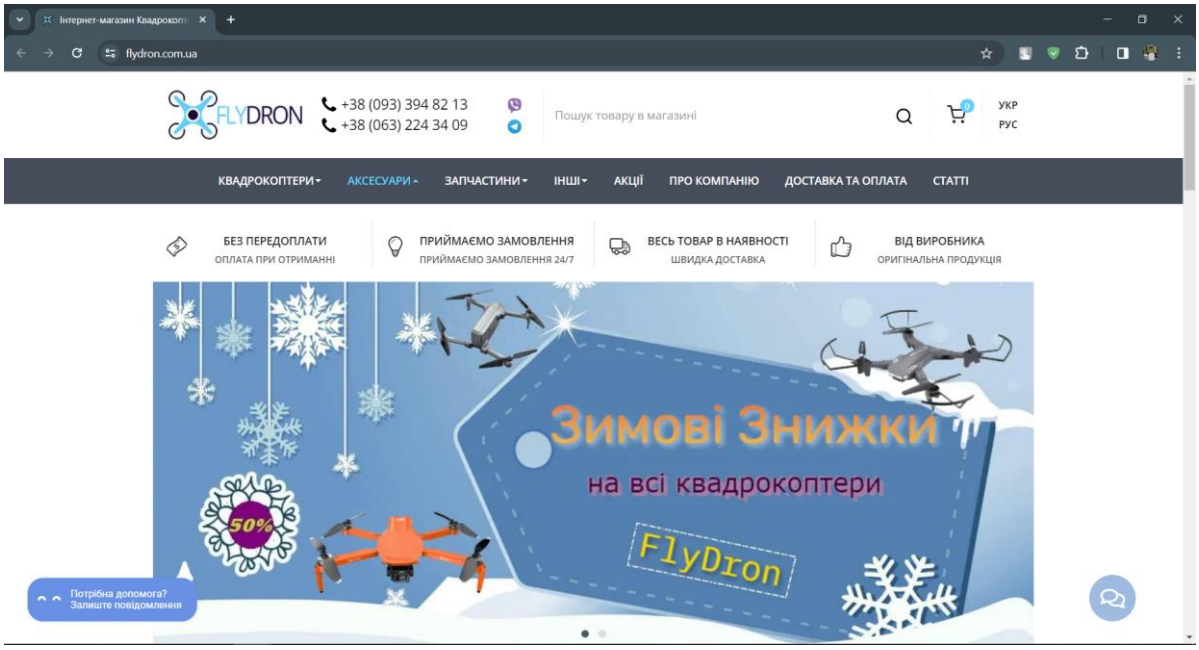

Рис.1.4. – Головна сторінка FlyDron

Перейдемо до огляду інтерфейсу інтернет-магазину. Зверху розміщена інформація із назвою магазину, контактними даними, пошуком по сайту, корзиною та мультимовою.

Нижче, реалізована навігація сайту. Можна перейти на різні категорії товарів, які нас цікавлять та переглянути їх. Також представлено інформацію про акції, компанію, доставку та оплату. І цікавий розділ «Статті». Там

знаходиться інформація про різні новинки або цікаві факти про технології дронів.

Є змога перейти в чат із консультантами або замовити дзвінок із повідомленням, яке вас хвилює. Згодом, повинен зателефонувати оператор та пояснити вам необхідну інформацію.

Із мінусів, це знову ж таки, великий банер, який займає, майже, всю ширину екрана та надає не зовсім цікаву інформацію для користувача.

Якщо ми полистаємо сторінку трохи вниз, то побачимо вигляд сторінки, який знаходиться на рис.1.5

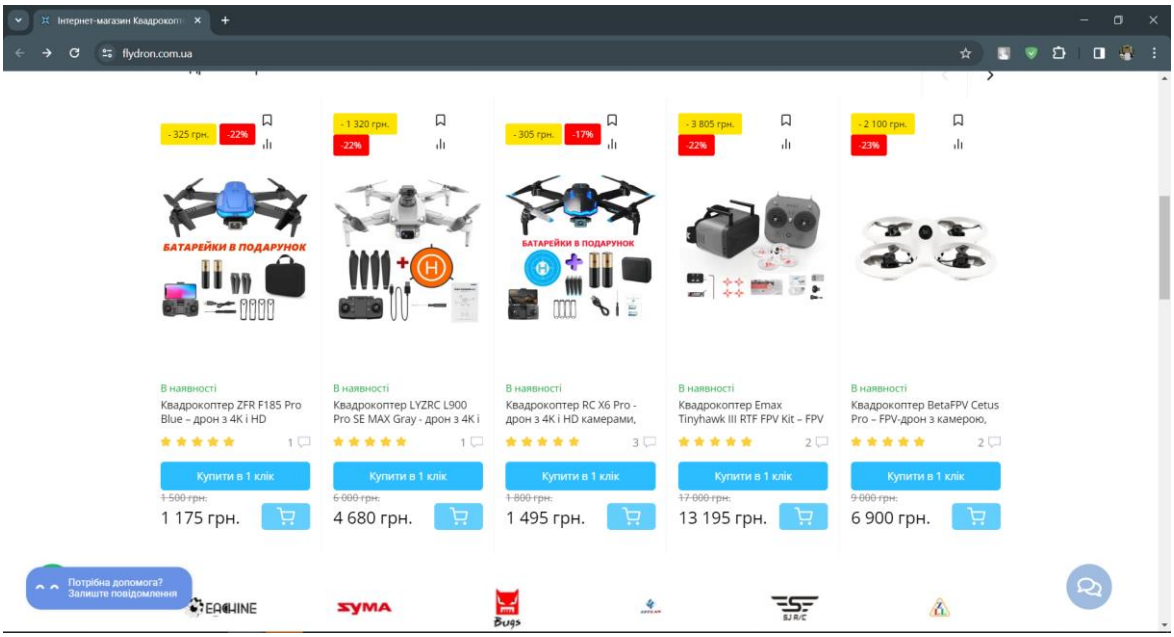

Рис.1.5. – Статичні товари на FlyDron

Даний відрізок сторінки ніяк не змінюється, а лише відображає статичні товари та фірми дронів. Також зберігається змога перейти в чат та залишити повідомлення для консультантів магазину.

Якщо ми підемо ще дальше вниз, то побачимо наступне на рис.1.6

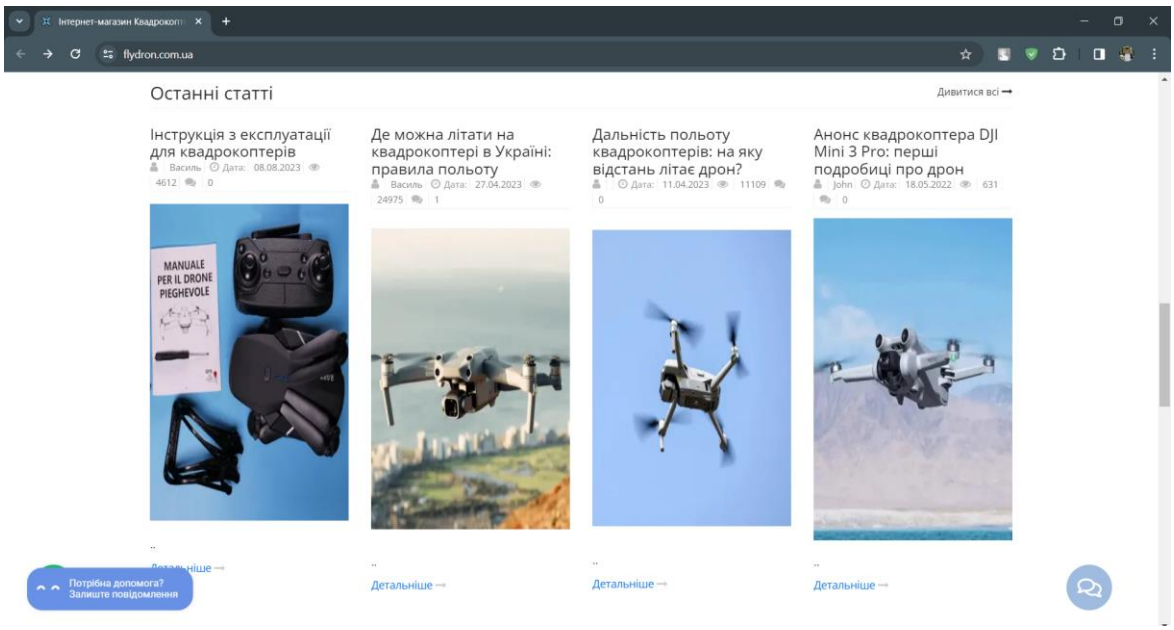

Рис.1.6. – Статті FlyDron

Маємо змогу переглянути останні добавлені статті на сайт. Але бачимо, що фотографії із текстом мають різну висоту, що не надає привабливого вигляду.

Після фото є три маленьких точок, які не зрозуміли, що мають на увазі.

Отже, із недоліків даного сайту є недоопрацювання над CSS частиною, відрізок сайту, де відображена лише статика та великий занятий простір банером.

Із перевагом я б хотів відмітити, це продаж запчастин до дронів, змога в реальному часу поспілкуватися онлайн-чатом із продавцем.

Розглянемо останній інтернет-магазин по нашій тематиці. Наший вибір зупинився на «EcoDrive». Скажемо так, що це не зовсім інтернет-магазин, який спеціалізується саме на дронах. Але дрони також є в списку товарів. Це може бути один із мінусів, бо консультанти не завжди можуть знати всі характеристики та специфіки товарів. Я вважаю, що тоді потрібно спеціально відділяти спеціалістів для консультування покупців. Так буде ефективність продажів та задоволення клієнтів набагато вище.

Головна сторінка сайту представлена на рисунку 1.7

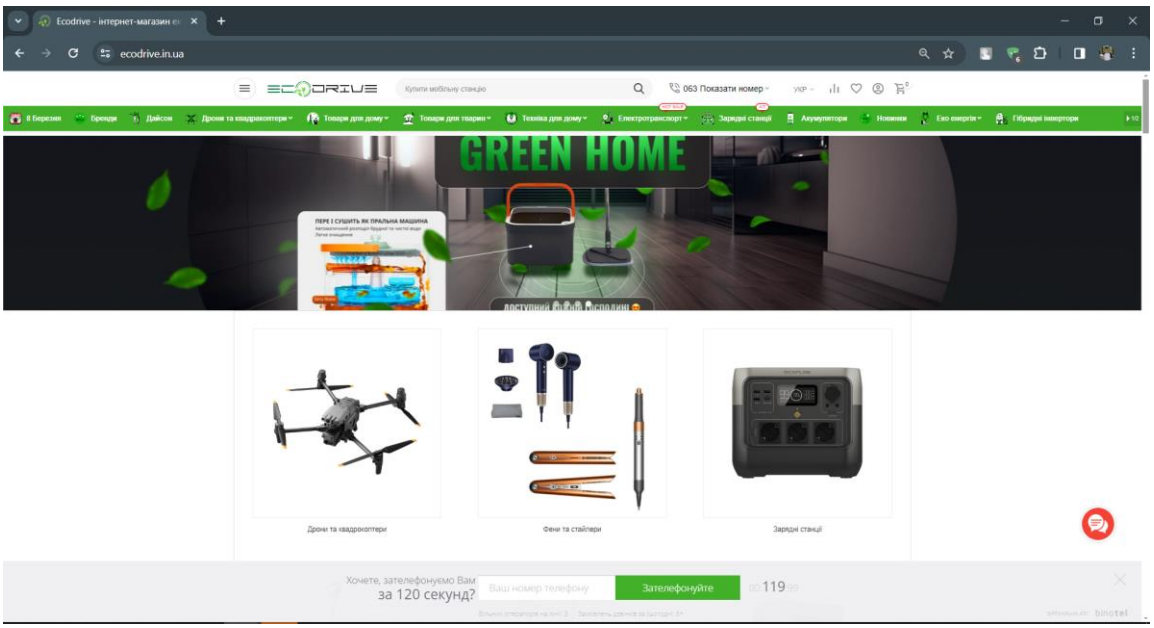

Рис.1.7. Головна сторінка EcoDrive

Спочатку ми побачимо зверху меню навігації. В ньому маємо змогу натиснути на меню, яке представлене у вигляді трьох рисок. Після цього відкривається дане меню.

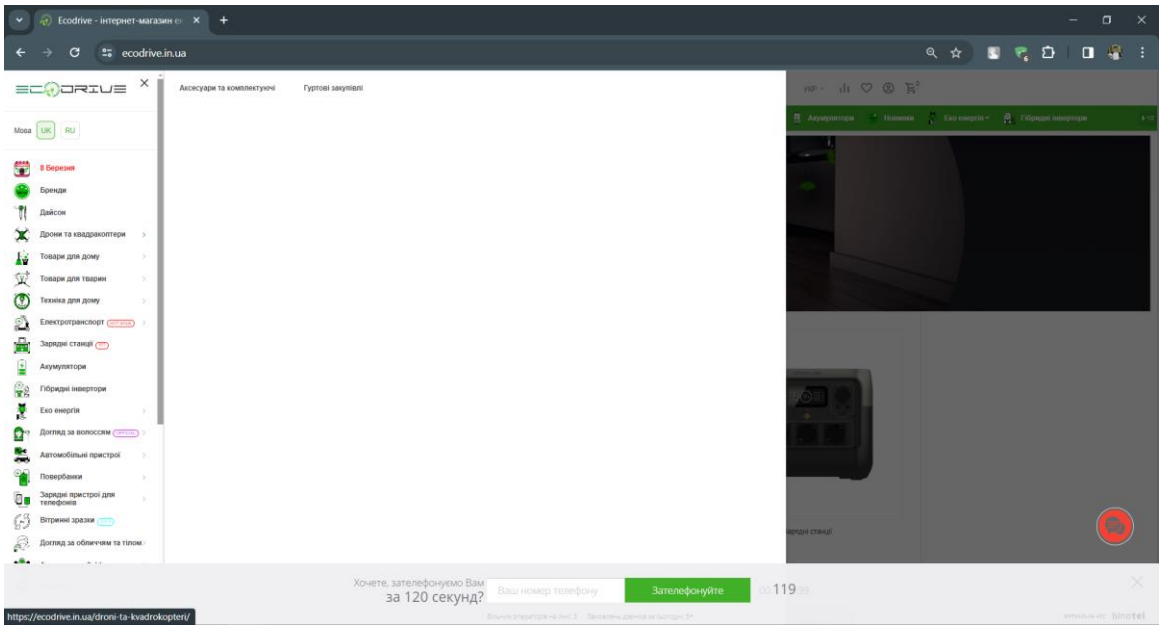

Рис.1.8. Меню сайту EcoDrive

Але таке саме меню ми можемо побачити і на головній сторінці сайту. Тому, це меню неефективне, бо доводиться зайвий раз клікати, щоб побачити його.

Після цього меню ми маємо змогу побачити реалізацію «пошук». Після тестування, пошук працює правильно. Виводиться інформація по ключовому слові, яке шукається в категоріях та товарах.

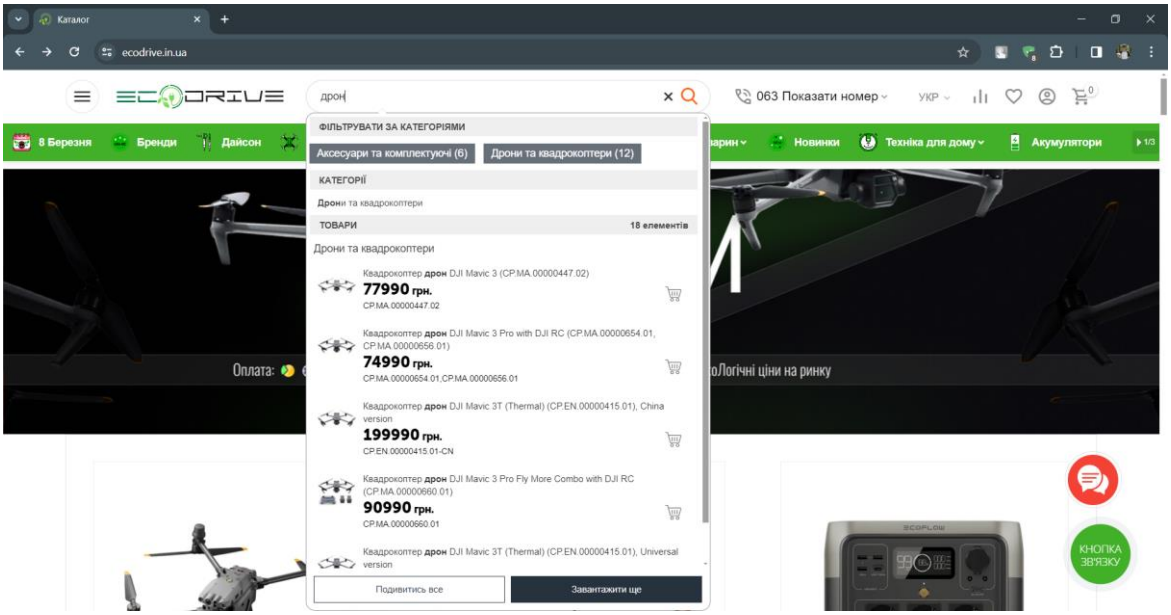

Рис.1.9. Пошук

Далі йдуть меню із контактними даними магазину, вибором мови, меню порівняння, особистим кабінетом та корзиною. Внизу сторінки знаходяться інтерактивні кнопки для того, щоб швидко зв’язатися із продавцями.

Перейдемо до категорії дронів на цьому сайті. Це нас цікавить найбільше. Переглянемо цей розділ на рисунку 1.10

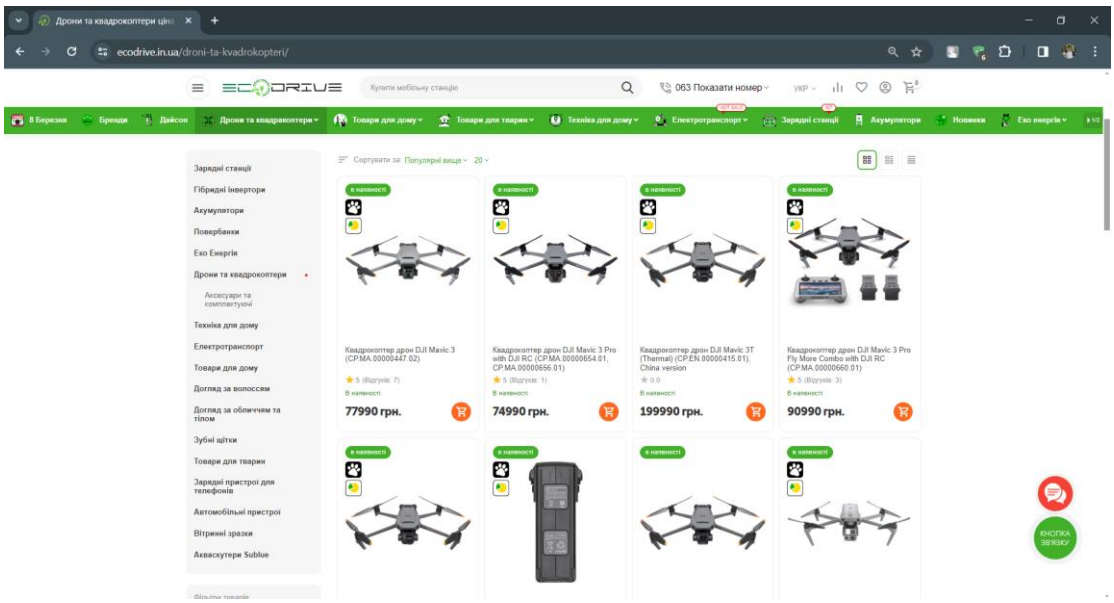

Рис.1.10. – Категорія дронів

Представлені в стовпчик по чотири товара дрони з інформацією про наявність цих дронів в магазині. Також можемо вибрати фільтри, по яким хочемо відфільтрувати дрони по певним критеріям.

Але недолік саме на цій сторінці в тому, що зліва в нас представлені категорії, які відносяться до загальної сторінки сайту, а не саме до категорії «Дрони». Ми вважаємо, що ці категорії в цьому розділі не потрібні.

Тепер перейдемо до перегляду певного товару. Ця сторінка буде представлена на рисунку 1.11

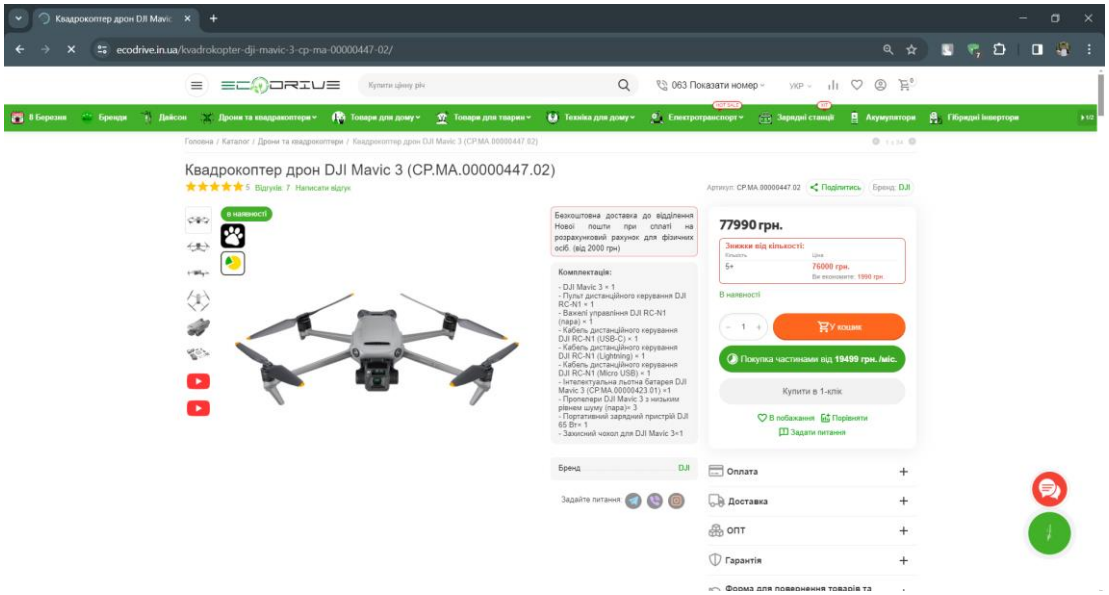

Рис. 1.11. Перегляд товару

Ця сторінка надає нам інформацію про певний товар. Вгорі написана назва товару та арикул. Нижче зліва представлені фото дрона, а справа від нього комплектація, яка буде йти при замовленні. Ще правіше ціна товару та можливість замовити його.

Перейдемо нижче цієї розмітки сайту. Переглянемо її на рисунку 1.12

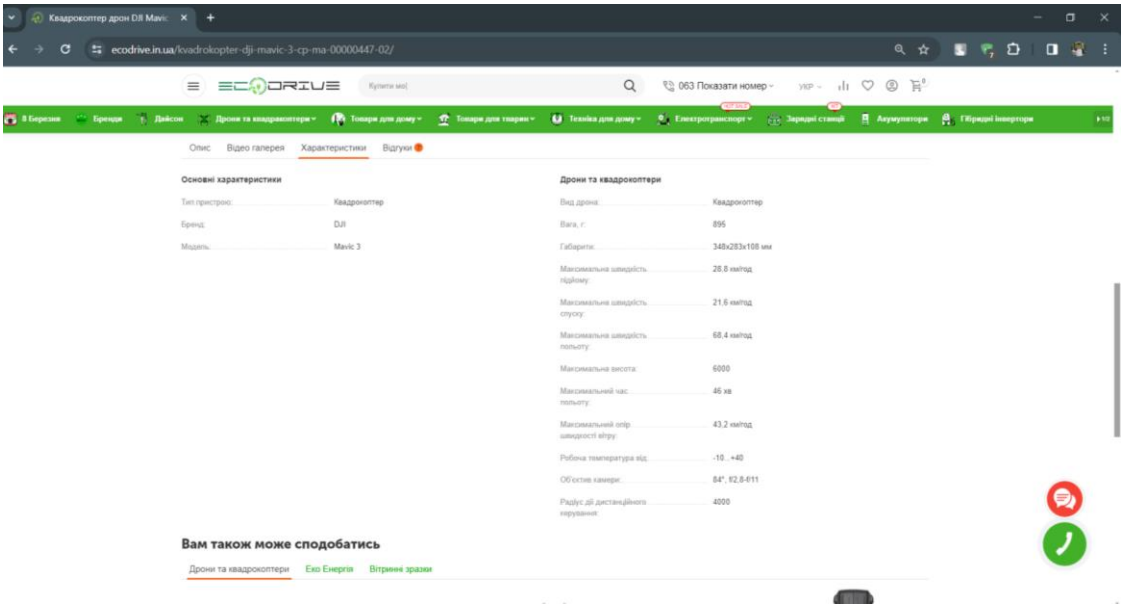

Рис.1.12. Меню товара

Маємо можливість вибрати конкретну інформацію про цей товар, а саме: опис, галерея, характеристики та відгуки про товар. Це дуже зручно, що вся ця інформація не відображається відразу на одній сторінці. Ми можемо вибрати, що саме нам потрібно переглянути та не засмічуємо сторінку не потрібною нам інформацією. Це перевага.

Підведемо підсумок по цьому сайті. Із недоліків ми виокремлили про не потрібне меню в навігації, при перегляді товарів в певній категорії зліва зберігається меню із категоріями для загальної сторінки. Із переваг – це при перегляді списків товарів є інформація про наявність цього товару в магазині. А при перегляді конкретного товару є змога вибрати потрібну нам інформацію про товар та не розбиратися в купі іншого тексту.

Підведемо загальний підсумок нашого аналізу.

Отже, нами було розглянуто та проаналізовано чотири інтернет-магазина. Було знайдемо як слабкі, так і сильні сторони кожного із магазинів. Ми виділили основні ключові моменти для впровадження або навпаки, не використання, у власному сайті.

Основні критерії:

1. Веб-сайт та інтерфейс:
    - зручність користування та наявність фільтрації;
    - наявність інформації про продавця, сторінка «Про нас».
2. Продуктовий асортимент:
    - різноманіття товарів, може включати дронів різних категорій;
    - наявність сортування дронів по категоріям.
3. Маркетинговий хід:
    - не використання великих банерів, які займають собою всю сторінку;
    - використання привабливих фішок при відображення списку товарів для зацікавлення покупця.

Також потрібно приділяти важливий відсоток своєї уваги для особистого кабінету користувача. Він повинен бути водночас простий та функціональний.

В результаті, після проведення всіх досліджень та аналізу аналогів було прийнятне рішення розробити власний веб-сайт інтернет-магазину, який не повинен мати недоліків, які були знайдені під час аналізу та повинен включати в себе всі переваги.

2. ПРОЕКТУВАННЯ ІНТЕРНЕТ-МАГАЗИНУ

2.1. Структура інтернет-магазину

В сучасному світі електронна комерція дуже популярна, і кількість онлайн-платформ зростає з кожним днем. У зв'язку з цим у покупців є багато варіантів здійснення онлайн-покупок. Однак вибір зазвичай залежить від швидкості та зручності взаємодії з веб-ресурсом. Для багатьох клієнтів час є найціннішим ресурсом, тому вони очікують швидкої відповіді від вашого сайту, щоб зробити покупку. Якщо вони зіткнуться з повільним або неефективним веб-сайтом, вони негайно перейдуть на інший ресурс, де процес покупки більш приємний і ефективний. Однак залучення та утримання клієнтів залежить не лише від швидкості, а й від простоти використання [3, 248 с.]. Структура веб-сайту відіграє важливу роль у створенні позитивного досвіду для користувачів. Наприклад, якщо ваш веб-сайт містить корисну інформацію, але навігація по ньому складна або неприваблива, користувачі не залишаться на вашому сайті надовго. Те саме стосується інтернет-магазину. Клієнти очікують легко та швидко знайти потрібні їм товари. Якщо структура магазину їм не підходить, то вони можуть дуже швидко знайти інші альтернативи.

Структура веб-сайту – це деревоподібне розташування блоків сайту, що забезпечує практичну та логічну структуру. Якщо розглядати магазин більш детально, це схематичне розташування блоків категорій, карток товарів, кошика, особистого кабінету, формою для реєстрації та авторизації користувача, а також не менш важливими аспектами. Отже, розглядаючи спроектовану структуру сайту можна більш наочно побачити загальну картину, яка дасть зрозуміти про доцільність певних блоків та можливі їх зміни.

Якщо оцінювати загальний вигляд веб-сторінок сайтів, які займаються онлайн-продажами, то можна зробити висновок, що, майже, всі ресурси мають

одинакові функціональні частини, але мають відмінність у зовнішньому вигляду сайту [11].

При проектуванні та розробці веб-сайту велика частина уваги приділяється головній сторінці, тому, що це найперше, що бачить відвідувач сайту перед своїми очима, коли переходить по посиланню. Вона повинна приваблювати клієнта своїм зовнішнім виглядом та відобразити те, що теоретично шукати покупець.

Щоб покупець відчував комфорт, коли відвідує ваший веб-сайт та здійснював дії, які допомогли б йому вибрати потрібний товар, сайт, перш за все, повинен мати логічну та добре продуману структуру. Якщо ваший сайт структурований неправильно та не відображає всю важливість даних про товар, то це призведе до втрати клієнта. Тому дуже важливим та першочерговим завданням для розробника веб-сайту є проектування гнучкої, логічної та добре візуалізованої структури.

Фахівці виділяють чотири типи структури організації інформації: лінійну(також називають послідовна), ієрархічну, мережну і комбіновану.

Тепер розглянемо кожну з них детально та виберемо для себе найбільш практичну.

Проаналізуємо перший тип розміщення інформації на сайті – лінійну структуру.

Лінійна структура організації є простим і легким способом відображення вмісту сторінки на сайті. У цій моделі сторінки не поділяються на розділи чи категорії, а розміщуються одна за одною зберігаючи певну послідовність. Користувачі переміщуються на вашому веб-сайті, переходячи від однієї сторінки до іншої за допомогою посилань або кнопок навігації.

Даний варіант чудово підійшов б для невеликих веб-сайтів, які мають невелику кількість інформації. Лінійна структура полегшує навігацію, оскільки відвідувачі сайту мають змогу легко прокручувати весь вміст сторінки та переходити по потрібним посиланням без зайвих перешкод. Також ще один прикладів, де б чудово підійшла дана структура – це навчальні сайти.

Там користувач буде поступово здійснювати операції для перегляду навчального матеріалу або виконання тестових завдань і на сторінці будуть відображатися тільки ті посилання, які потрібні для підтримання шляху навігації. Ці веб-сайти вимагають від користувача виконання дій, які пов’язані між собою та мають залежність. Приклад даної структури представлений на рисунку 2.1

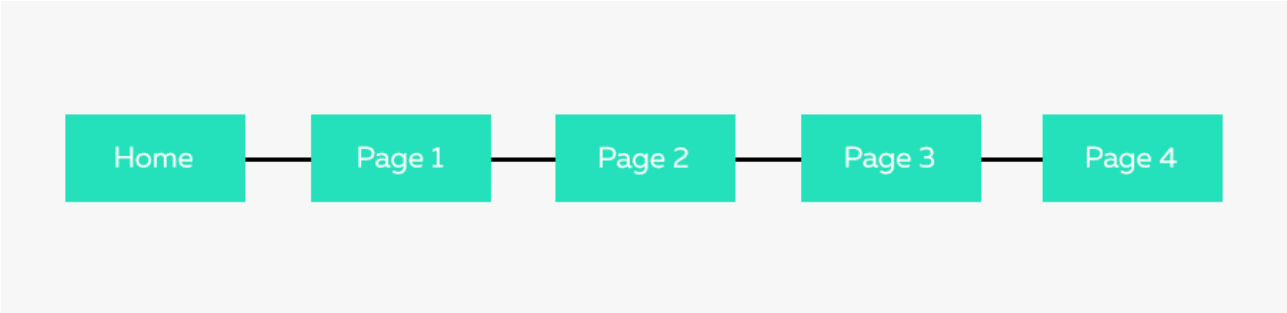

Рис.2.1. – Лінійна структура

Головною перевагою лінійної структури є її простота. Це дуже легко реалізувати і не потрібно використовувати складні системи управлінням контенту або навігації. Також це ефективна структура для сайтів, які мають однорідну інформацію або продукти. Але ця структура буде неефективна для великих веб-сайтів, в яких міститься величезна кількість товару або текстової інформації. В цьому випадку користувачу буде складно знайти потрібну йому інформацію.

Наступний тип структури – ієрархічна структура. Цей тип структури є дуже популярним та найкращим способом для організації і відображення дуже великих інформаційних масивів. Ієрархічна структура особливо добре підходить для організації веб-сайту, які організовані навколо головної сторінки. З головної сторінки можна переходити в підкатегорії або підрозділи. Візуальний приклад такої структури зображений на рисунку 2.2

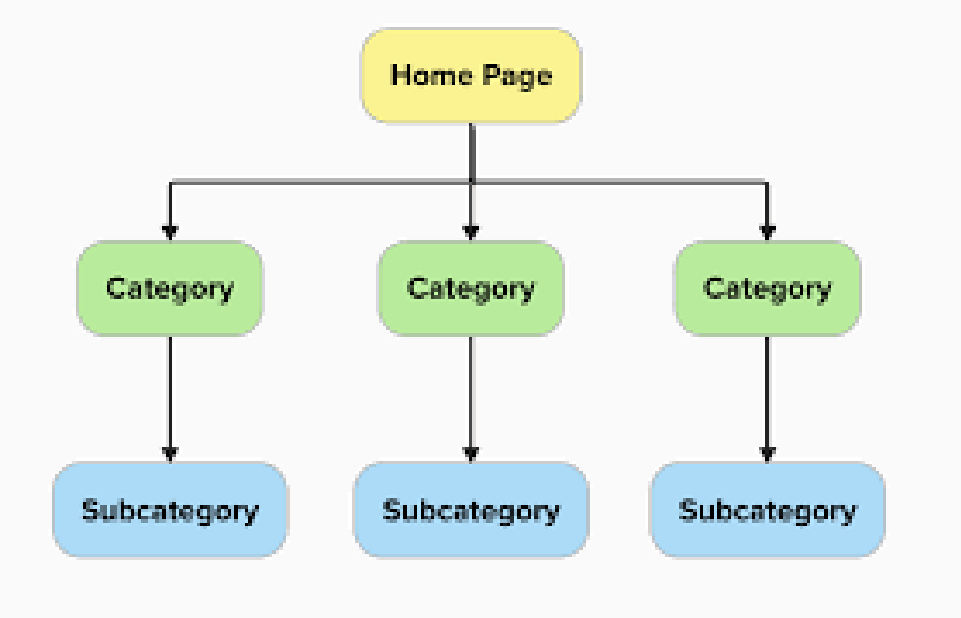

Рис.2.2. – Ієрархічна структура

Це найпростіша форма ієрархічної структури – з головної сторінки розходяться набір сторінок. Якщо реалізовувати сайт по даному рисунку, то він буде мати однорівневу ієрархію. В навігації буде міститися простий список підсторінок і посилання на головну сторінку [1, 59 с.].

Велика кількість людей звикли кожного дня будувати велику кількість інформації в ієрархічну структуру, оскільки більшість із них працюють у компаніях, які будують структури керівництва або створюють плани на бізнес-проекти. Також і більшість веб-сайтів використовують деревоподібну архітектуру. Дане розташування категорій і підкатегорій дає значні переваги для складних організацій сайтів, оскільки більшість людей знайомі з ієрархічною структурою та можуть легко створити уявну модель структури сайту.

Третій тип структури – мережева структура. Даний тип структура передбачає створення системи навігації, де навігація сайту має зв’язки між усіма елементами, що дозволяє зручно та швидко переходити між сторінками сайту без завантаження додаткових проміжних сторінок. Візуалізований приклад даного вида представлений на рисунку 2.3

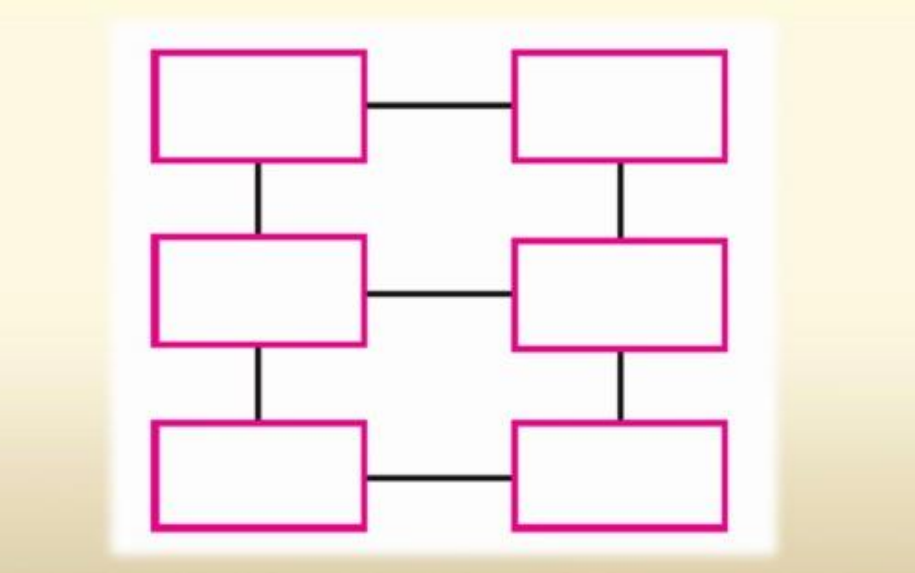

Рис.2.3. – Мережева структура

Мережева структура широко поширена в Інтернеті. Прикладом такої структури є веб-сайт, який є великим інформаційний порталом.

Та останній із розглянутих нами типів структур – комбінована структура. Це поєднання в собі двох видів структур. Наприклад, сайт може включати в себе спроектовану ієрархічну структуру, але при відображенні та реалізації певних сторінок використати ту саму ж структуру буде не практично. Тому в цьому випадку можна замінити цю структуру іншою, яка буде більше краще та логічніше підходити до цієї задачі. При використанні даного сайту із комбінованою структурою, переходити між сторінками можна у будь-який доступний спосіб.

Отже, при здійсненні аналізу типів структур сайтів та проектуванні власного інтернет-магазину, було обрано для використання комбіновану структуру. Основною перевагою комбінованої структури є можливість створити просту та зручну у використанні користувача структуру сайту.

На рисунку 2.4 спроектована структура інтернет-магазину для продажу дронів.

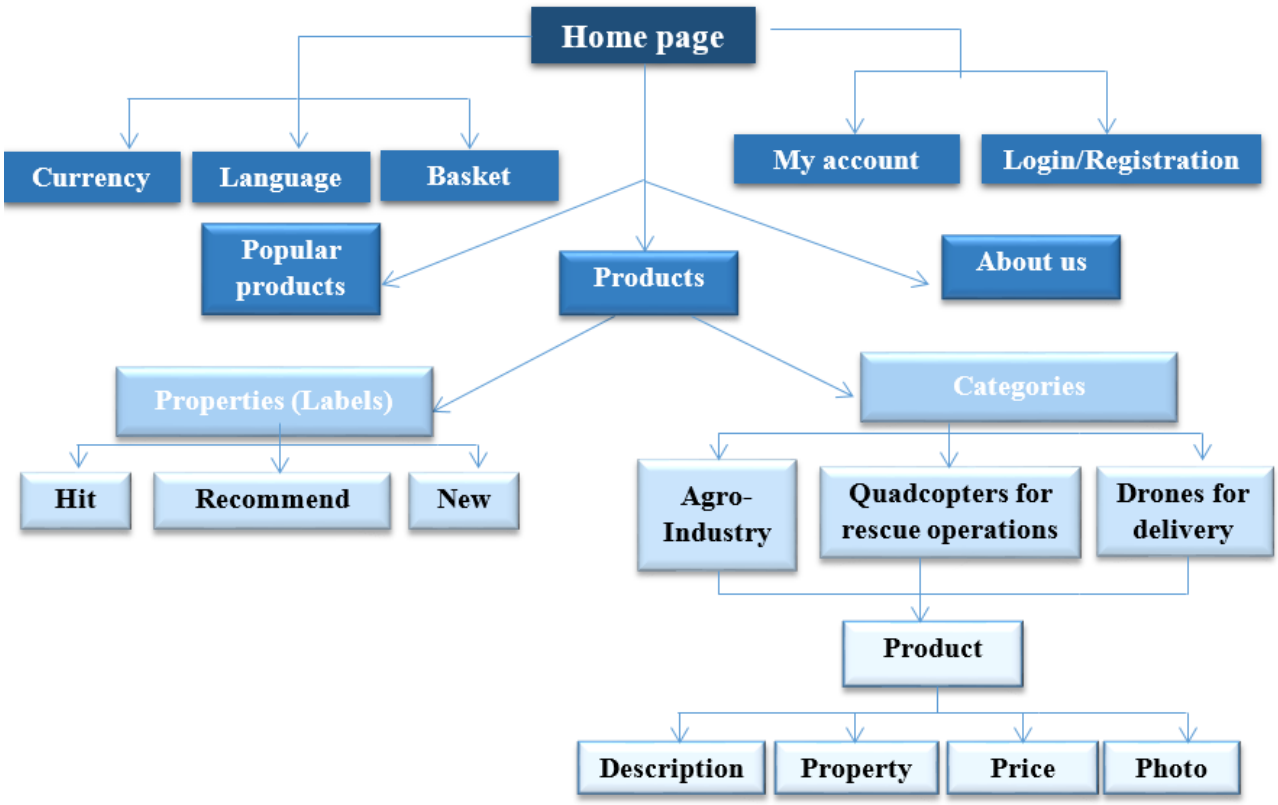

Рис. 2.4. – Проект інтернет-магазину

Розробці та проектуванні головної сторінки сайту приділяють особливу увагу, бо це найперше, що бачить покупець, коли заходить на сторінку інтернет-магазину. Майже, завжди, цю сторінку забезпечують навігацією по сайту і, можливо, роблять для зручності користувача зручні функції. Але, ми вважаємо, що основним із критеріїв та оцінки зручності головної сторінки сайту – є розміщення цікавого, інформативного та корисного контенту.

Сторінки сайту будуть поділятися на три важливі категорії – навігаційна панель, основний інформаційний блок та нижня частина сайту, або іншими словами, футер. Навігаційна панель буде поєднувати в собі і частини хедеру. Хедер – це «шапка» сайту, яка дублюється на всій решті сторінок. В навігаційні панель буде розміщуватися назва інтернет-магазину та дуже важливі компоненти для користувача. Це буде включати в себе можливість змінити валюту, мову, відкрити корзину для перегляду доданих товарів у замовлення, вхід або реєстрацію користувача та доступ до особистого кабінету

зареєстрованого користувача. Одним із критеріїв при розробці буде враховуватися зручність та зрозумілість. Тому найкращим рішенням буде розмістити назви підсторінок одним рядком, а зміну мови або валюти випадаючим меню.

Основний блок інформації буде містити в собі контент для перегляду користувачем. Також буде можливість перейти до більш детальної інформації про товар та ознайомитися із нею. Або, якщо товар на даний момент недоступний, залишити заявку для менеджерів інтернет-магазину і тоді обов'язково вони повідомлять, коли товар знову з'явиться в продажі.

Футер – елемент, який знаходиться в самому низу сторінки. В ньому, зазвичай, міститься інформація про авторські права, назву категорій та назви продуктів, які найбільше мають популярність у покупців.

В будь-якому магазину, звичайному чи інтернет, будь-які дії із покупкою товару виконує менеджер або адміністратор. Тому і в нашому інтернет-магазині потрібно реалізувати зручну та функціональну адміністративну панель. Вона повинна бути зрозумілою та не затримувати нашого адміністратора у роздумах, яку кнопку натиснути, щоб зробити ту чи іншу дію.

Тому при розробці структури інтернет-магазину, не менш важливо, спроектувати структуру адміністративної панелі, яка дозволяє виконувати всі необхідні операції, пов'язані з реалізацією електронної комерції та взаємодією з покупцями. На рисунку 2.5 зображено структуру адміністративної панелі, яка буде зручною, щоб наповнити магазин контентом та взаємодіяти із замовленнями.

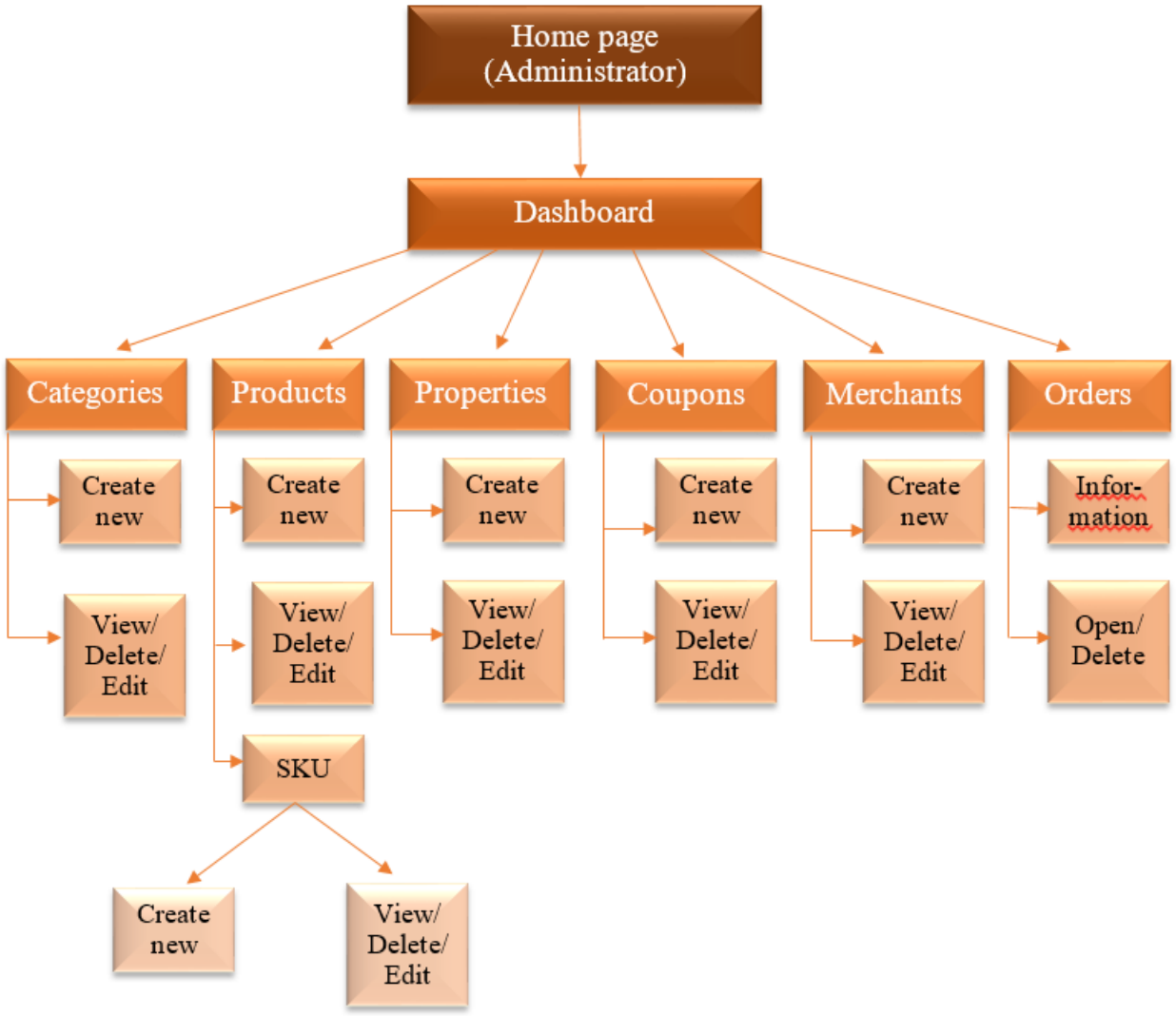

Рис. 2.5. – Структура адміністративної панелі

Тут також основну функцію виконує головна сторінка управління адміністративною панеллю. В ній розташовані всі потрібні адміністратору функції та пункти управління товарами і наповненням інтернет-магазину. Також у розділі «Products» ми добавили дуже корисний пункт меню, який має назву «Skus».

SKU – це абревіатура, що означає "Stock Keeping Unit" або одиниця управління запасами. Це ідентифікатор, який привласнюється кожному товару окремо та має унікальність. За допомогою цієї функції дуже зручно відстежувати та керувати запасами товару. Кожен товар має власну інформацію про виробника, характеристики, модель тощо. SKU додає

адміністратора полегшене меню, яке допомагає йому оформляти замовлення, відстежувати кількість запасу товару та робити по товару звіти.

2.2. Моделювання основних принципів роботи веб-сайту

Для проектування сайту найкраще підходить UML. Проектування нам потрібно для того, щоб сайт був зрозумілий та функціональний.

Мова UML — це мова візуального моделювання загального призначення, розроблена для специфікації, візуалізації, проектування та документування програмних компонентів, бізнес-процесів та інших систем [17].

Розробка діаграми UML в основному використовується на всіх етапах життєвого циклу розробки бізнес-процесів і прикладних програм. Мови графічного моделювання підтримують різні типи діаграм, а також включають різні варіанти для опису та відображення різних аспектів складних систем, що робить цю мову універсальним вибором для написання проектів програмного забезпечення. Вона стала чудовим інструментом. UML можна застосовувати в різних сферах нашого життя(фінанси, Інтернет, авіакосмічна сфера, охорона здоров’я тощо).

UML – діаграма представляє графічне зображення моделі системі, коли проект розробляється, реалізується або вже існує. Вона містить в собі графічні елементи, які пов’язані з потоками(також називають шляхи) – в сукупності вони представляють собою елементи моделі UML розробленої системи.

Так, як існують різні види діаграм, то вони відрізняються між собою первинними графічними символами. В діаграмі, де основні символи класи – має назву діаграма класів. А в діаграмі, де зображено випадки використання проекту з дійовими особами – має назву діаграма випадків використання. У випадку, коли діаграма показує поділ програмної системи на компоненти та зв’язки між ними – має назву діаграму компонентів.

Також за документацією UML можна змішувати різні типи діаграм в одну. Наприклад, використовувати комбінацію об’єктів та профілю елементів.

З цього можна зробити висновок, що в основному діаграми дуже сильно не відрізняються між собою. Але потрібно бути уважним, бо деякі інструменти мають обмеження на набір графічних елементів і можуть використовуватися тільки в певному виду діаграми [14].

Метою цієї мови є полегшення розробки програмних продуктів, оскільки діаграми достатньою мірою представляють структуру моделі та можуть бути легко переведені в програмний код.

В мові UML є дві основні категорії – структурні діаграми та діаграми поведінки.

Розглянемо спочатку поняття «структурні діаграми».

Структурні діаграми відповідають за відображення статичних станів системи. За допомогою цих діаграм можуть виділити основну частину системи, яку проектують. Якщо сказати іншими словами, то структурні діаграми мають показати, що буде відбуватися в системі. Також вони описують, як об’єкти будуть взаємодіяти між собою.

Найбільш доцільніше та точніше для проектування інтернет-магазину буде використання діаграми класів, яка відноситься до блоку структурні діаграми, діаграми варіантів використання та діаграму послідовності процесів. Тепер потрібно розібратися з кожною більш детальніше.

Діаграма класів – зображає структуру розробленої системи на рівні класів та інтерфейсів, відображає їх зв’язки та обмеження. Діаграми класів є основними будівельними блоками будь-якого об’єктно-орієнтованого методу. Оскільки класи є будівельними блоками програм на основі ООП, діаграми класів мають гарну структуру для представлення класів, успадкування, зв’язків і всього, що є в контексті ООП.

Цей клас описує різні типи об’єктів і статичні зв’язки між ними.

Основною метою для чого використовуються діаграми класу є:

- відображення структури програми;
- ідентифікація класу – дає розуміння, які мають бути основні класи в програмі та їхні атрибути і методи;

- зв’язки між класами – зображення взаємодію між класами, успадкування тощо;
- допомога при проектуванні – важливий інструмент для проектування програми.

Зображення діаграми класів буде представлено на рисунку 2.6

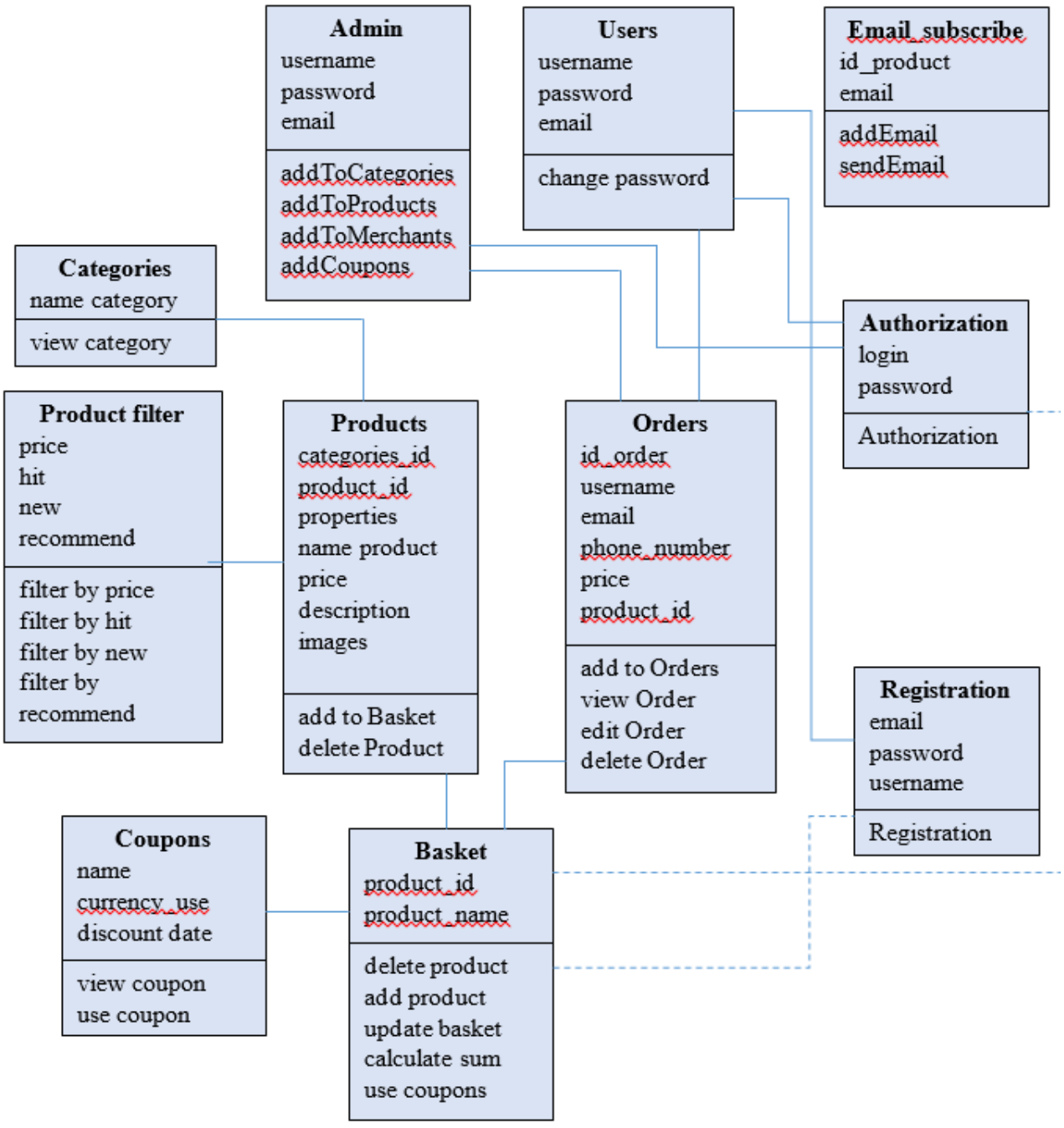

Рис. 2.6. - Діаграма класів

Детальніший опис цієї діаграми буде представлений у таблиці 2.1. Там ми опишемо кожний клас, атрибути та операції, які можна використовувати.

Таблиця 2.1 – Опис діаграми класів

Параметр	Значення
Клас Admin	
Коментар	Клас, який є адміністратором інтернет-магазину
Атрибути	username – ім’я адміністратора password – пароль для входу email – поштова адреса адміністратора для входу
Дії	addToCategories – додавання нової категорії addToProducts – додавання нового продукту addToMerchants – додавання постачальника addToCoupons – додавання купона
Клас Users	
Коментар	Клас, який є користувача інтернет-магазину
Атрибути	username – ім’я користувача password – пароль для входу email – поштова адреса користувача для входу
Дії	change password – зміна паролю
Клас Email_subscribe	
Коментар	Клас, який відповідає за сповіщення користувача при появі товара
Атрибути	id_product – ідентифікатор товару email – поштова адреса користувача для сповіщення
Дії	addEmail – добавлення поштової адреси SendEmail – відправка на поштову адресу
Клас Orders	
Коментар	Клас, який показує замовлення здійсненні користувачем
Атрибути	id_order – ідентифікатор замовлення username – ім’я користувача email – поштова адреса користувача phone_number – номер телефону користувача price – ціна замовлення product_id - ідентифікатор товарів, які замовили
Дії	add to Orders – добавити до замовлення view Order – переглянути замовлення edit Order – змінити замовлення delete Order – видалити замовлення
Клас Authorization	
Коментар	Клас для авторизації користувачів

Продовження табл.2.1

Атрибути	login – поштова адреса користувача password – пароль користувача
Дії	Authorization – метод авторизації
Клас Registration	
Коментар	Клас для реєстрації користувачів
Атрибути	username – ім’я користувача password – пароль для реєстрації email – поштова адреса користувача для реєстрації
Дії	Registration – метод реєстрації
Клас Basket	
Коментар	Клас для перегляду замовлень користувача
Атрибути	product_id – ідентифікатор товару product_name – назва товару
	delete product – видалити товар з кошику add product – додати товар в кошик update basket – оновити кошик calculate sum – сума замовлення use coupons – використання купонів
Клас Products	
Коментар	Клас для перегляду товарів
Атрибути	categories_id – ідентифікатор категорії product_id – ідентифікатор товару properties – властивості товару name product – ім’я продукту price – ціна товару description – опис товару images – фото товару
Дії	add to Basket – добавити товар до корзини delete Product – видалити товар
Клас Categories	
Коментар	Клас для перегляду категорій
Атрибути	name category – назва категорії
Дії	view category – перегляд категорії
Клас Product filter	
Коментар	Клас для фільтрування товарів по параметрам
Атрибути	price – сортування по ціні hit – сортування по кількості продажів new - сортування по новизні recommend - сортування по рекомендації магазину

Продовження табл.2.1

Дії	filter by price – вибір фільтру по ціні filter by hit - вибір фільтру по продажам filter by new - вибір фільтру по новизні filter by recommend - вибір фільтру по рекомендації магазину
Клас Coupons	
Коментар	Клас для застосування знижок
Атрибути	name – код купона currency_use – валюта використання discount date – дійсний до
Дії	view coupon – перегляд купона use coupon – використання купона

Для з'єднання класів між собою використовуються зв'язки. Розглянемо кожний з них та з'ясуємо, які зв'язки ми повинні використати. В залежності від відносин між класами використовують види зв'язків:

1. асоціація – зв'язок, який з'єднує два класи та показує, що один із класів може використовувати функції та властивості іншого. Наприклад, можна зробити класи «Студент» та «Курс». Тут студент може брати участь у багатьох курсах.
2. агрегація – зв'язок між класами, де один клас є частиною іншого. Наприклад, клас «Команда» включає в себе клас «Гравець».
3. композиція – зв'язок між класами, де один клас повністю володіє іншим класом. Наприклад, клас «Працівник» може бути складовою частиною класу «Компанія». Якщо клас «Компанія» припиняє своє існування, то й, відповідно, «Працівник» також.
4. залежність – взаємозв'язок, де один клас залежить певним чином від іншого класу. Наприклад, клас «Замовлення» використовує клас «Клієнт» для отримання інформації про особисті дані замовника.
5. реалізація – відношення між класом та інтерфейсом користувача, де клас реалізує функції та можливості, які описані в інтерфейсі.

У нашій схемі представлені такі види зв'язків між класами:

- між класами «Products» і «Product filter» - агрегація;
- між класом «Products» та «Basket», «Categories» - композиція;
- між класом «Admin» та «Orders» – асоціація;
- між класом «Admin» та «Authorization» – агрегація;
- між класом «Basket» та «Authorization», «Registration» – залежність;
- між класом «Users» та «Orders», «Email_subscribe», «Authorization», «Registration» – асоціація;
- між класами «Authorization» та «Registration» - залежність;
- між класом «Basket» та «Orders» - асоціація.

Наступна наша діаграма - це діаграма варіантів використання.

Діаграма варіантів використання, або Use Case Diagram, є одним із основних типів графічних моделей в методології UML (Unified Modeling Language). Вона застосовується для розробки функціональних вимог до системи з урахуванням того, як користувачі спілкуються з нею.

Основна мета цієї діаграми - визначити різні сценарії або ситуації, які можуть виникнути під час взаємодії користувачів з системою. Кожен варіант використання, або use case, описує конкретну можливість або дію, яку може здійснити користувач за допомогою системи.

Основні переваги використання діаграм варіантів використання включають:

- Зрозумілість: Ці діаграми зрозумілі як для технічних, так і для не-технічних учасників проекту, що допомагає усім сторонам будувати спільне розуміння системи.
- Визначення вимог: Вони допомагають визначити функціональні вимоги до системи, описуючи, як система буде використовуватися користувачами.
- Основа для подальших аналізів і проектування: Ці діаграми є основою для подальшого аналізу, проектування і тестування системи.

- Виявлення акторів та їх ролей: Діаграми допомагають ідентифікувати акторів (користувачів або зовнішні системи) та їх взаємодію з системою.

Для того, щоб розробити цю діаграму потрібно спочатку визначити ролі, які будуть в нашому інтернет-магазину. Можна виділити такі особи:

- адміністратор магазину;
- користувач – не зареєстрований або не авторизований;
- авторизований користувач.

На рисунку 2.7 можна переглянути діаграму варіантів використання інтернет-магазину.

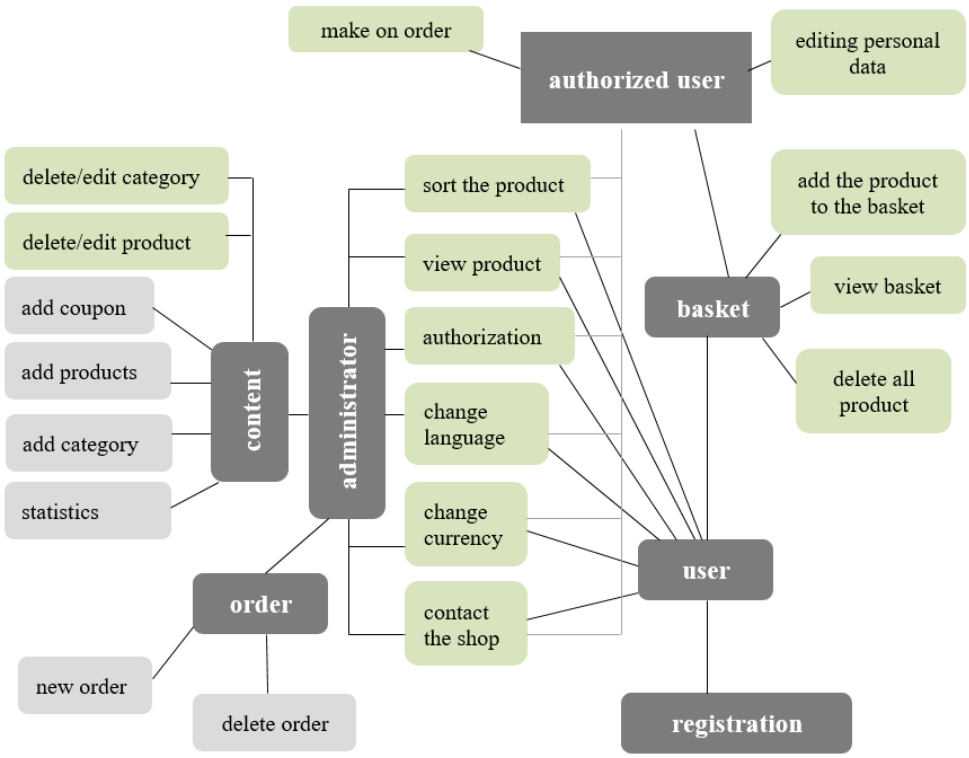

Рис. 2.7. – Діаграма варіантів використання

Для звичайного користувача – User, будуть доступні такі функції:

1. можливість побачити контактні дані інтернет-магазину;
2. змінити валюту для перегляду цін товарів;
3. змінити мову інтернет-магазину;

4. авторизуватися або зареєструватися;

5. переглянути товари;

6. відсортувати товари за власним критеріям по доступним фільтрам;

7. добавити товар у корзину.

Це дуже не поганий функціонал для звичайного користувача, але у авторизованого має бути трохи більше можливостей для роботи із замовленнями. Тому, авторизація(Authorized user) дає такі можливості:

1. управління замовленням;

2. використання купонів на оформлення замовлень;

3. редагування профілю користувача.

Детально розглянемо пункт «управління замовленням». Користувач при перегляді товарів добавляє їх у корзину, а згодом, оформлює замовлення. Добавити товар у корзину можна зі сторінки, де висвітлені всі товари певної категорії, на головній сторінці або при перегляді детальної інформації одного товару при натисканні кнопки «У корзину». Оформлене замовлення можна переглянути у особистому кабінеті у пункті «Мої замовлення». Далі авторизований користувач має можливість переглянути детальну інформацію про замовлення, а саме: фото товару, назву товару, ціну та валюту, коли зробив замовлення і номер телефону, який вказав. А також має можливість видалити замовлення із власного кабінету. Це означає, що замовлення скасоване та не буде далі оброблятися інтернет-магазином.

Адміністратор(Administrator) – це менеджер інтернет-магазину, тому він має найбільше функціоналу. В нього особистий кабінет в рази більше, ніж в просто авторизованого користувача. Давайте розглянемо детально кожну із його можливостей.

Особистий кабінет включає в себе функції, які доступні в авторизованого користувача, але тут менеджер має можливість видаляти будь-яке замовлення будь-якого користувача. А також переглядати їх. Йому

доступна інформація про користувача, який зробив це замовлення, про товар та про дату оформлення замовлення.

Також менеджер управляє контентом. А саме:

- добавлення нові товари;
- видалення товарів зі списку;
- змінює ціну на товари;
- змінює властивості товару та його рекомендації;
- добавлення нових категорій;
- видалення категорій;
- оформлення купонів;
- добавлення в базу даних постачальників інтернет-магазину.

Розглянемо функціонал про купони. Купони – це система лояльності або знижок для покупців інтернет-магазину. Купони можна отримати при перегляді рекламної кампанії або він прийде розсилкою на електронну пошту. Менеджер в своєму особистому кабінеті добавляє ці купони для використання. В цьому адміністратор заповнює такі поля:

1. код купона;
2. сума знижки;
3. знижка у відсотках чи від суми замовлення;
4. дата дії купони;
5. кількість разів використання.

Остання із запропованих нами діаграм UML – діаграма послідовності. Для чого ця діаграма? Вона корисна тим, що видно, які процеси мають йти послідовно один за одним. Це дуже зручно для програмування серверної частини інтернет-магазину. Дану діаграму можна переглянути на рисунку 2.8.

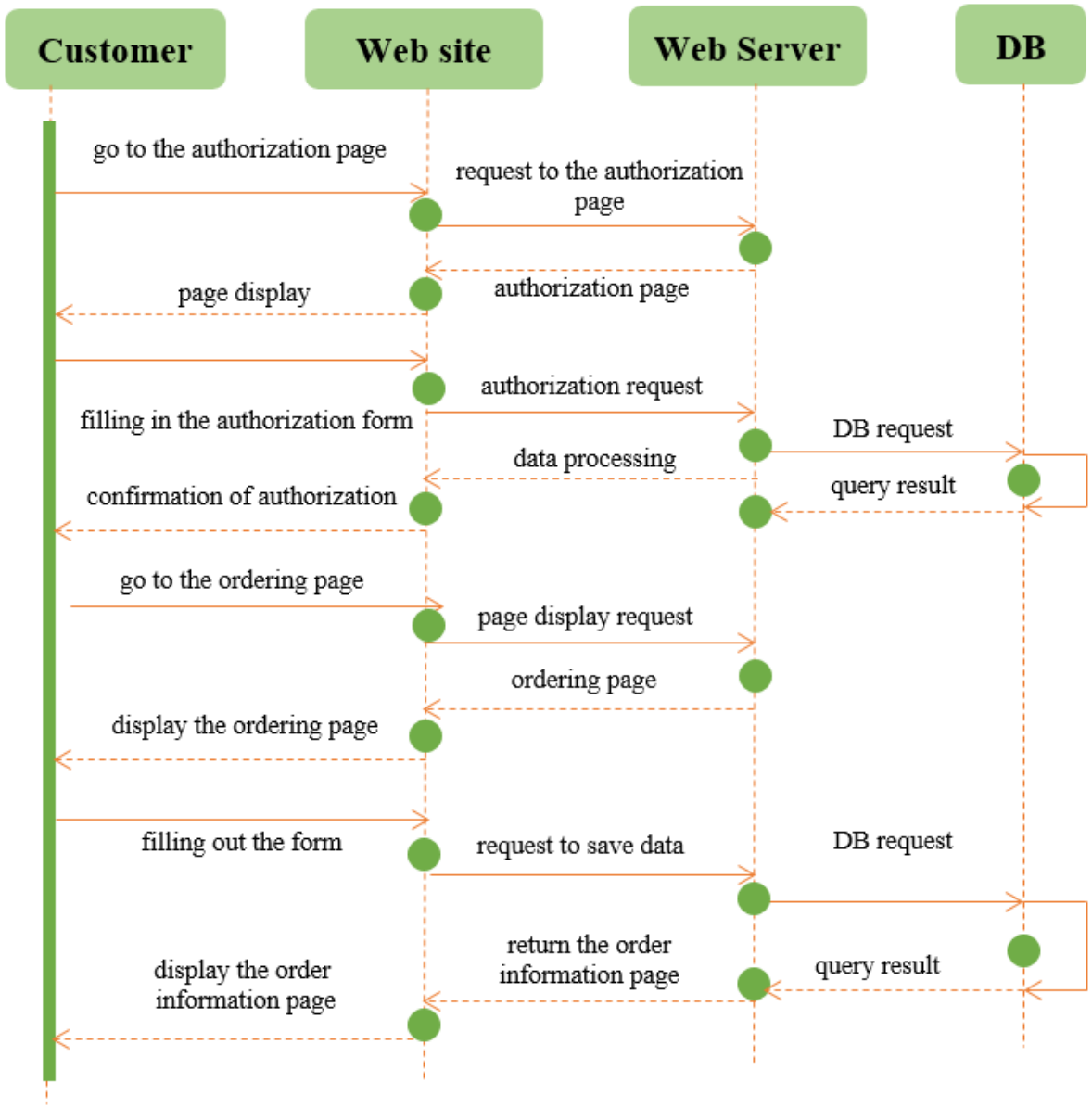

Рис. 2.8. – Діаграма послідовностей

На даному рисунку розміщено чотири ключові об’єкти: Customer, Web site, Web Server та Database. Вертикальні лінії зображують час роботи для певного об’єкта.

Послідовність дій така: Користувач натискає кнопку підтвердження. Потім, коли ви відвідуєте веб-сайт, веб-сайт надсилає запит сторінки на сервер, який потім виводить сторінку та відображає її користувачеві, який заповнює форму автентифікації та натискає кнопку підтвердження.

Потім запит надсилається на веб-сервер, який надсилає запит до бази даних, і запит обробляється.

Потім він отримує результати з веб-сервера БД.

Потім дані обробляються, а результати надсилаються на веб-сервер.

Підтвердження успішної авторизації. Коли покупець отримує доступ до сторінки замовлення, на веб-сервер надсилається запит на відображення цієї сторінки, і в результаті запиту користувачеві повертається сторінка та, відповідно, її інтерфейс.

Наступною дією користувача є введення даних у відповідну форму. Запит на збереження даних надсилається на веб-сервер, який потім робить запит до бази даних, обробляє дані та повертає успішні результати введення на сервер.

Веб-сервер повертає сторінку, що містить інформацію про замовлення, яке відображається клієнту.

2.3. Проектування структури бази даних

Для ефективної роботи з даними важливо розробити структуру бази даних, яка дозволяє аналізувати предметну область і виявляти можливі проблеми. На основі цього аналізу створюється концептуальна модель бази даних.

Під час концептуального проектування вимоги до даних збираються та аналізуються, щоб визначити основні сутності (таблиці), їхні атрибути та зв’язки між ними. Результатом є концептуальна модель, яка відображає структуру бази даних у формі моделі сутності-зв’язку. Ця модель включає структуру таблиці та логічні зв’язки. Кожна таблиця складається зі стовпців, імена атрибутів мають бути унікальними, а ключі таблиці та типи даних визначені відповідно до вимог.

Спеціалізованою сферою діяльності Drone Online Shop є комерційна онлайн-платформа з продажу дронів, спрямована на надання користувачам легкого доступу до необхідних товарів. Цей бізнес діє як посередник між продавцями, менеджерами і власниками та їхніми клієнтами та покупцями, надаючи можливості для ефективного обміну інформацією між цими

сторонами. Автоматизація інформаційних процесів необхідна для забезпечення швидкої та ефективної взаємодії між усіма залученими сторонами. Це допомагає виконувати завдання швидко, злагоджено та якісно.

Метою проектування структури цієї предметної області є створення системи обліку користувачів, їх замовлень і товарів. В інтернет-магазині основним завданням є спонукання користувачів додавати товари в кошик, тому на початковому етапі проектування бази даних визначаються три основні компоненти: «користувачі», «замовлення» і «товари».

Сутності визначають основні типи інформації, що зберігається в базі даних (у реляційній базі даних кожна сутність відповідає таблиці). Суб'єкти цієї предметної області включають: користувачів, замовлення, продукти, категорії, фотографії, розсилка сповіщень про наявність товару, купони та зв'язатися з нами тощо.

ER-діаграма сутностей буде представлена на рис.2.9.

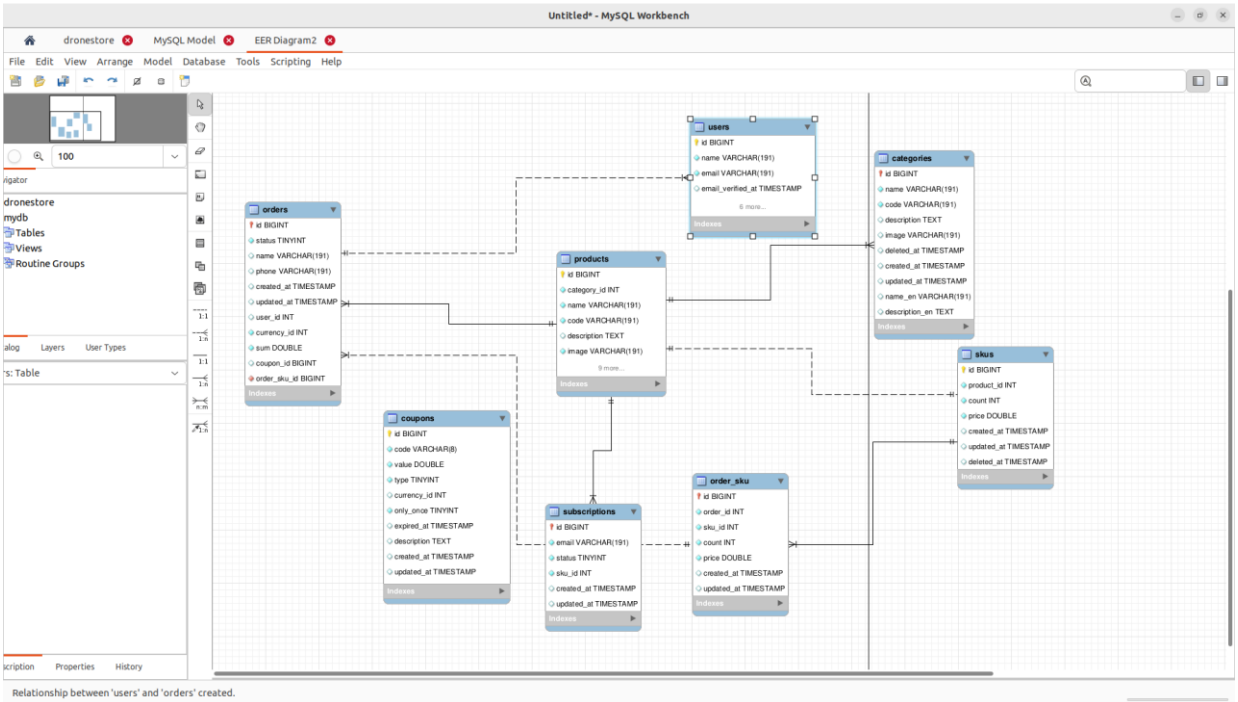

Рис.2.9. - ER-діаграма сутностей

Сутність “users” - одна із ключових та головних. Саме вона регулює, хто із користувач є просто користувачем або адміністратором інтернет-магазину. Вона зберігає детальну інформацію про користувача, а саме:

- “id” - ідентифікатор користувача;
- “name” - ім’я користувача;
- “email” - електронну адресу користувача;
- “password” - пароль для авторизації.

Сутність “coupons” - це набір атрибутів, які показують про систему знижок для оформлення замовлень. Основні атрибути:

- “id” - ідентифікатор купона;
- “code” - код купона для оформлення знижки;
- “value” - відсоток, на скільки буде застосована знижка;
- “currency_id” - валюта, для якої знижка;
- “only_once” - скільки разів можна використати купон.

Не менш важлива сутність “products”. Вона має зв’язок із великою кількістю таблиць та є одною із головною в нашій схемі. Включає в себе такі атрибути:

- “id” - ідентифікатор товару;
- “category_id” - ідентифікатор категорії, до якої належить товар;
- “name” - назва товару;
- “cod” - код товару;
- “description” - опис товару;
- “image” - фото товару;
- “new” - характеристика товару по новизні;
- “hit” - характеристика товару по кількості продажів;
- “recommended” - характеристика товару по рекомендації від магазину;
- “skus_id” - ідентифікатор характеристики SKU.

Для того, щоб поєднати дві сутності, а саме “products” і “skus” використовується зв’язок “один до одного”, бо один товар може мати лише одну товарну пропозицію.

Даний зв’язок можна переглянути на рисунку 2.10.

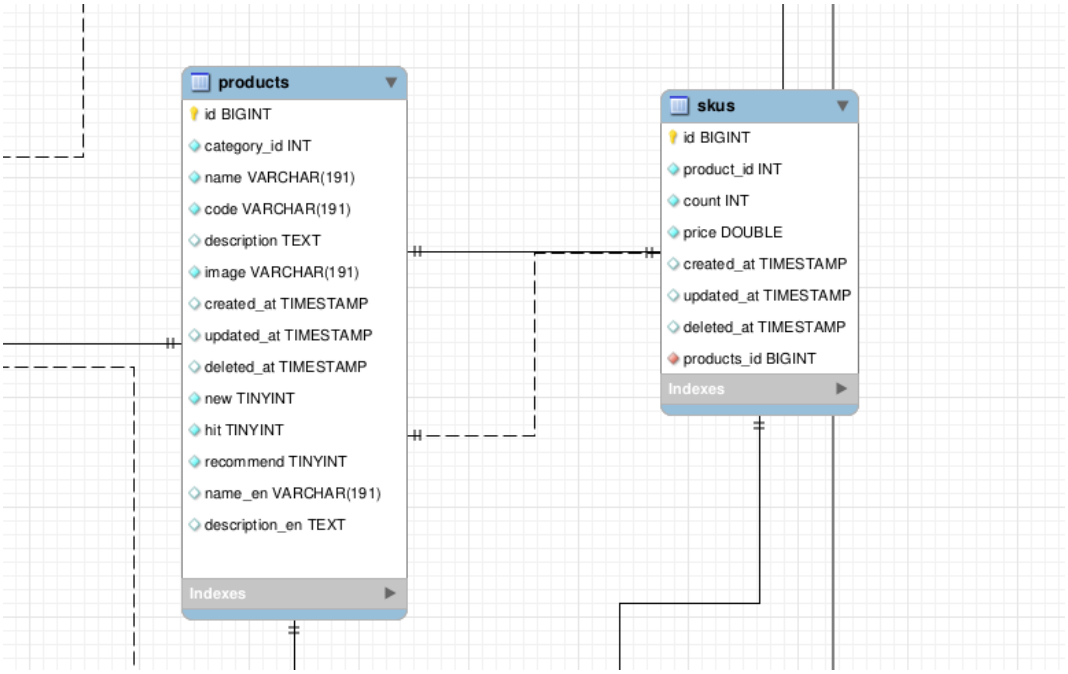

Рис.2.10. - ER-діаграма сутностей “products” та “skus”

Розглянемо сутність “orders” - вона містить детальну інформацію про замовлення та про дані клієнта. Таблиця включає в себе такі атрибути:

- “id” - ідентифікатор замовлення;
- “status” - статус виконання замовлення;
- “name” - ім’я клієнта;
- “phone” - номер телефону клієнта;
- “user_id” - ідентифікатор клієнта;
- “currency_id” - ідентифікатор валюти замовлення;
- “sum” - сума замовлення;
- “coupon_id” - ідентифікатор купона;
- “order_sku_id” - ідентифікатор товарної пропозиції замовлення.

Дана сутність має зв’язок “один до одного” із сутність “coupons”. Бо в замовленні може бути використано лише один раз купон. Це можна переглянути на рисунку 2.11.

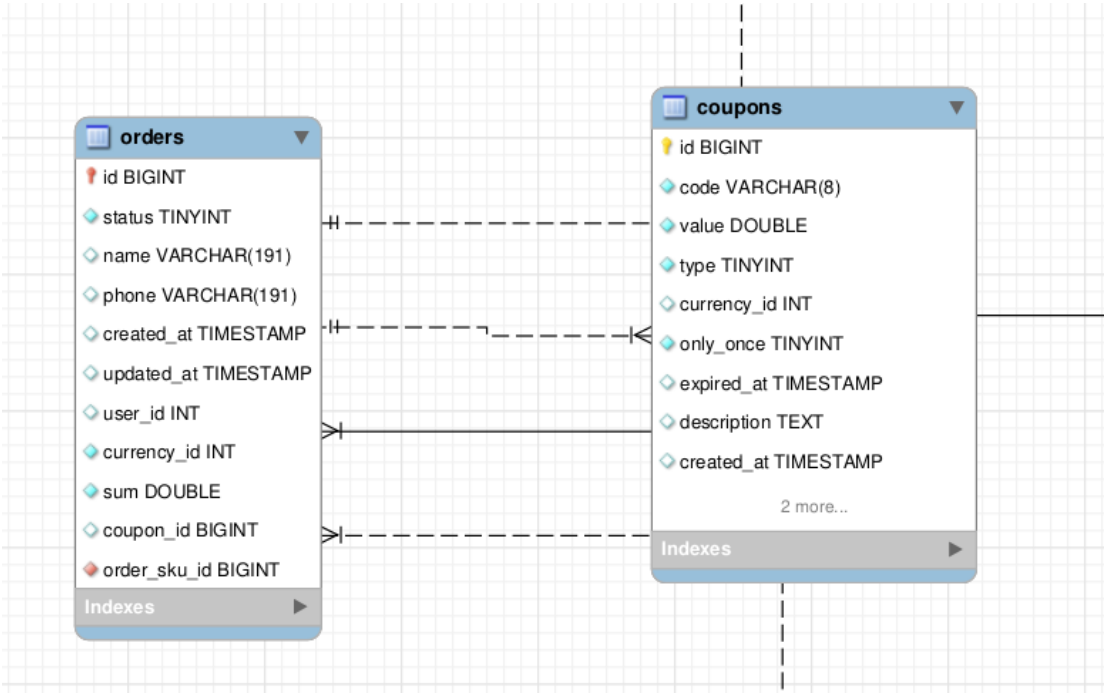

Рис.2.11. - ER-діаграма сутностей “products” та “coupons”

Сутність “subscriptions” - відповідає за сповіщення користувача про наявність товару. Має такі атрибути:

- “id” - ідентифікатор сповіщення;
- “sku_id” - ідентифікатор товарної пропозиції.

Поєднується зв’язком “багато до багатьох” із сутність “products”, так як може бути надіслано багато сповіщень про появу у продажу багатьох товарів. Зв’язок зображений на рисунку 2.12.

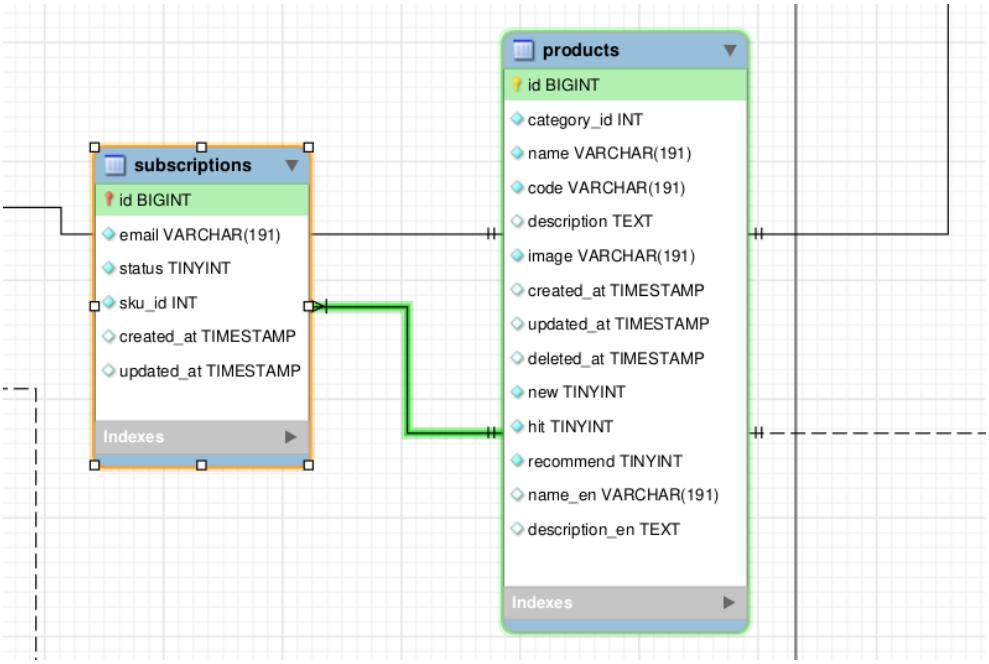

Рис.2.12. - ER-діаграма сутностей “products” та “subscriptions”

Сутність “categories” - відповідає за типи категорій, в яких знаходяться товари. Дає детальну інформацію про категорії. Включає в себе такі атрибути:

- “id” - ідентифікатор категорії;
- “name” - назва категорії;
- “cod” - код категорії;
- “description” - опис категорії;;
- “image” - фото категорії.

Із сутністю “products” має зв’язок “один до багатьох”, так як в одній категорії може бути багато товарів. Зв’язок зображений на рис.2.13

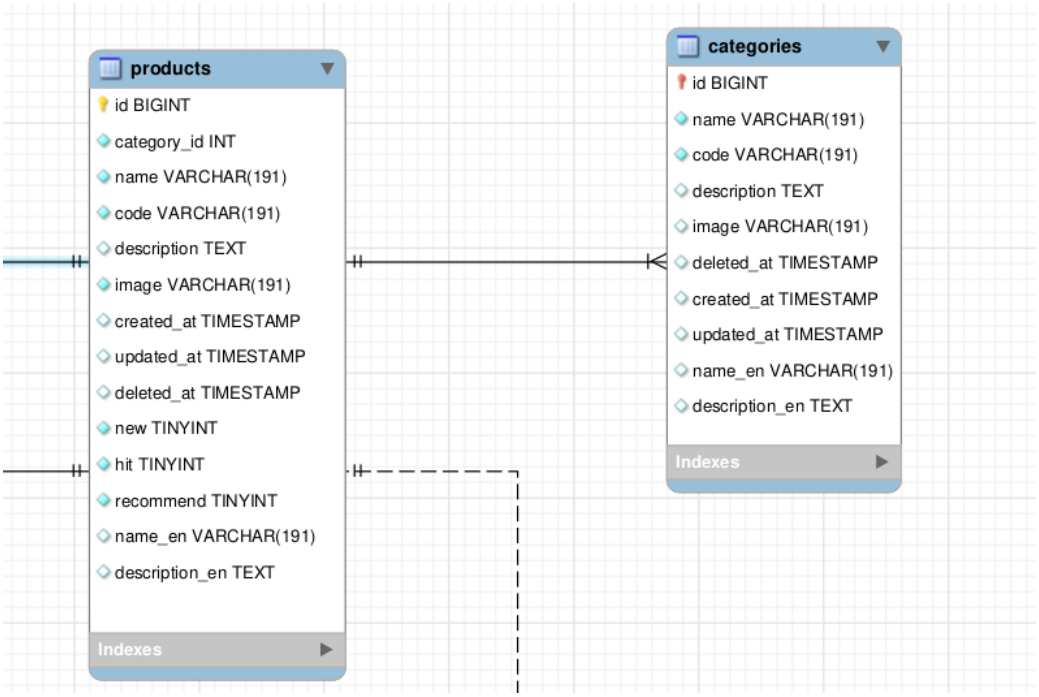

Рис.2.13. - ER-діаграма сутностей “products” та “categories”

3. ПРОГРАМНА РЕАЛІЗАЦІЯ ІНТЕРНЕТ-МАГАЗИНУ

3.1. Вибір технологій для розробки

Спочатку для розробки веб-сайту потрібно вибрати мову програмування. Наший вибір відразу зупинився на мові програмування PHP. Це скриптова мова програмування, яка дозволяє легко інтегрувати процес розробки з різними сторонніми допоміжними засобами, на прикладі фреймворків [19]. Також це мова, яка має кросплатформність та дозволяє не задумуватися над вибором операційної системи для розробки інтернет-магазину.

Якщо говорити про популярність мов програмування, то тут, PHP програє свою позицію JavaScript. Дану тенденцію ми можемо переглянути на рисунку 3.1

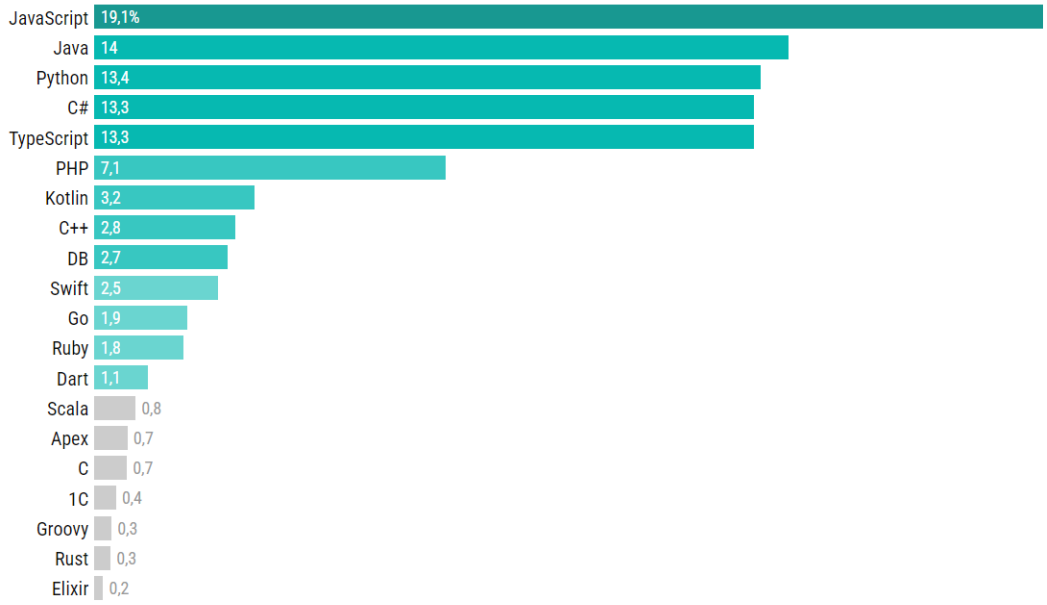

Рис.3.1. – Популярність мов програмування

Дана статистика взята за 2023 рік і показує використання мови програмування у всіх сферах IT. Але якщо говорити саме про розробку веб-ресурсів, то PHP з JavaScript тут лідери. До того ж, ними можна доповнювати один одного.

Тепер потрібно розглянути різні фреймворки, які найбільше підходять для розробки інтернет-магазину та вибрати той, який найбільше відповідає нашим потребам. Laravel і Symfony є двома найпопулярнішими фреймворками PHP, які часто використовуються для веб-розробки інтернет-магазинів.

Що таке фреймворк?

Фреймворк - це структура або платформа програмного забезпечення, яка надає розробникам набір заздалегідь підготовлених компонентів, модулів і інструментів для швидкої і ефективної розробки програмного забезпечення [18].

Фреймворки забезпечують загальну структуру для створення програм та додатків, дозволяючи розробникам уникати написання коду з нуля і зосередитися на реалізації конкретної функціональності.

Вони також забезпечують стандартизований спосіб організації та керування проектами, полегшуючи підтримку, розширення та покращення програмного забезпечення.

Давайте розглянемо основні характеристики та порівняємо їх з точки зору комфорту та адаптивності.

Laravel:

Laravel – один із найефективніших та зручних у використанні фреймворків. Він надає розробникам широкий набір інструментів для реалізації різних функцій та можливостей [13, 15 с.]. Ось деякі з переваг Laravel

- Простота використання: Laravel має чітку та добре задокументовану структуру, що робить його привабливим для новачків у веб-розробці.

- Велика спільнота: Laravel має активну та велику спільноту розробників, яка постійно додає нові функції та плагіни до фреймворку.

- Екосистема інструментів: Laravel пропонує широкий спектр готових рішень і пакетів, які спрощують розробку та підтримку проектів.

Symfony:

Symfony також є потужним фреймворком PHP із багатьма функціями та можливостями [9]. Ось деякі особливості Symfony:

- Гнучкість і розширюваність: Symfony дозволяє розробникам налаштовувати фреймворк під свої потреби та розширювати його за допомогою багатьох пакетів та компонентів.

- Стабільність та надійність: Symfony має довгий життєвий цикл та гарантовану підтримку, що робить його надійним вибором для великих та складних проектів.

- Висока продуктивність: Symfony має високу продуктивність і швидкість, що дозволяє розробникам створювати швидкі та ефективні програми.

Порівняння та оцінка:

Обидва фреймворки мають свої переваги та недоліки, і вибір між ними залежить від конкретних вимог проекту та власних уподобань розробника. Якщо ви швидко починаєте проєкт або маєте веб-додаток малого та середнього розміру, Laravel може бути кращим вибором. З іншого боку, Symfony може бути більш підходящим для великих та складних проектів, де важлива гнучкість та розширюваність фреймворку.

Ми вибрали Laravel для розробки нашого інтернет-магазину з кількох важливих причин:

1. **Простота у використанні:** Laravel забезпечує чітку та елегантну синтаксичну структуру, яка спрощує розробку та підтримку коду. Добре задокументований API робить процес розробки простішим і ефективнішим, особливо для команд із різним рівнем досвіду веб-розробки.

2. **Швидкість розробки:** Laravel має велику кількість готових компонентів та пакетів, які дозволяють швидко впроваджувати різні функції і можливості у проекті. Це дозволяє нам швидше випустити продукт на ринок та швидше реагувати на зміну вимог і потреб користувачів.

3. **Активна спільнота та розширюваність:** Laravel має велику та активну спільноту розробників, яка постійно додає нові ідеї та функції у

фреймворк. Це робить його дуже гнучким та розширюваним, дозволяючи нам легко впроваджувати нові функції та інтегрувати пакети сторонніх розробників у наш проект.

4. Безпека: Laravel має вбудовані механізми безпеки, такі як міграції, мідлвари та захист CSRF, щоб запобігти різним типам атак і захистити користувачів та їхні дані [13, 24 с.].

Мідлвари (Middleware) - це проміжний рівень програмного забезпечення, який дозволяє обробляти запити до того, як вони досягнуть основної логіки додатка. Вони можуть виконувати різні завдання, наприклад: перевірка аутентифікації користувача, обробка сесій, логування запитів, фільтрація введених даних та багато іншого. Мідлвари допомагають покращувати безпеку додатків, продуктивність та функціональність, а також полегшує розробку та обслуговування.

CSRF-захист (Cross-Site Request Forgery) - це тип атаки на безпеку веб-додатків, у якій зловмисник використовує авторизовану сесію користувача для виконання небажаних дій на веб-сайті без його належного дозволу. Це може включати зміну конфіденційної інформації, здійснення фінансових операцій або інші дії, які можуть завдати шкоди користувачам або веб-додатку. Захист CSRF полягає у створенні та перевірці унікального маркера для кожного запиту, гарантуючи, що запит надіслано користувачем, а не зловмисником, таким чином запобігаючи успішним атакам.

5. **Доступність готових рішень:** Laravel має велику кількість готових пакетів і рішень для всіх ваших потреб у розробці, що дозволяє нам ефективно та швидко впроваджувати необхідні вам функції у нашому інтернет-магазині.

Для ефективного управління базою даних нашого інтернет-магазину ми вибрали **«phpMyAdmin»** на основі тих же критеріїв: простий у використанні інтерфейс, багатофункціональність, доступність, активна спільнота та підтримка, а також високий рівень безпеки. Інструменти Laravel і phpMyAdmin допомагають забезпечити ефективну та безпечну розробку та

керування нашим онлайн-магазином, тим самим гарантуючи високу якість і надійність обслуговування для ваших користувачів.

3.2. Реалізація бази даних

Враховуючи складність і вимоги сучасних онлайн-магазинів дронів , вибір найкращої бази даних є одним із найважливіших рішень у процесі розробки . Для ефективної роботи нашого магазину та забезпечення високої продуктивності системи необхідно вибрати базу даних , яка відповідає вимогам проекту.

При аналізі можливих варіантів баз даних було враховано кілька важливих критеріїв .

1. **Сумісність з Laravel:** Оскільки весь онлайн-магазин розроблено на Laravel, база даних має бути максимально інтегрована з цією структурою . Забезпечення швидкості та ефективності роботи системи .

2. **Масштабованість:** наший магазин має намір постійно розвиватися, тому обрана нами базу даних має бути легко масштабованою, щоб відповідати зростаючим потребам користувачів і обсягам даних .

3. **Безпека:** враховуючи конфіденційність особистих і фінансових даних користувачів, бази даних повинні мати високий рівень безпеки та підтримувати шифрування даних .

4. **Швидкість:** важливо, щоб запити до бази даних виконувалися швидко. Це особливо важливо під час інтенсивних навантажень , коли багато запитів виконуються одночасно.

Враховуючи ці фактори, MySQL було обрано як найкращу базу даних для нашого онлайн-магазину.

MySQL — це потужна та надійна реляційна база даних для зберігання та керування великими обсягами даних в Інтернет-додатках і веб-службах. Дана база даних розроблена корпорацією Oracle, є однією з найпопулярніших у світі систем керування базами даних (СУБД) [15, 275 с]].

Реляційна база даних (RDB) — це тип бази даних на основі моделі даних, у якій дані організовані у вигляді таблиць із рядками та стовпцями.

Такий підхід до зберігання та організації даних дозволяє встановлювати зв’язки між різними таблицями, полегшувати маніпулювання інформацією та забезпечувати узгодженість даних.

Ця реляційна база даних відповідає всім перерахованим вище критеріям.

Для того, щоб управляти БД ми вже раніше вибрали phpMyAdmin.

Тепер перейдемо до створення таблиць. Щоб створити таблицю в нашому інтернет-магазині я використовував міграції. У фреймворку Laravel це дуже корисний функціонал для роботи із базами даними. Міграції - це як версійний контроль для бази даних, що дозволяє розробнику легко створювати, змінювати та ділитися схемою бази даних.

Наприклад, розглянемо процес створення таблиці 'products' (рис.3.2).

2024_01_25_133637_create_products_table.php

```php
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
    /**
     * Run the migrations.
     */
    public function up(): void
    {
        Schema::create( table: 'products', function (Blueprint $table) {
            $table->id();
            $table->integer( column: 'category_id'); //ID категорії
            $table->string( column: 'name'); //Назва товару
            $table->string( column: 'code'); //Код товару
            $table->text( column: 'description')->nullable(); //Опис товару
            $table->string( column: 'image');
            $table->double( column: 'price')->default( value: 0); //Ціна товару
            $table->timestamps();
            $table->softDeletes();
        });
    }

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

Рис.3.2. - Створення таблиці 'products'

В результаті виконання даного коду, ми можемо побачити результат в системі управління нашою базою даних. Ця таблиця буде відповідати за зберігання даних про товар (рис.3.3). Дана таблиця містить наступні атрибути:

- id - унікальний ідентифікатор товару;
- category_id - унікальний ідентифікатор категорії товару;
- name - назва товару;
- code - код товару;
- description - опис товару;
- image - фото товару;
- price - ціна товару.

	#	Ім'я	Тип	Зіставлення	Атрибути	Нуль	За замовчуванням	Коментарі	Додатково	Дія
☐	1	**id** 🔑	bigint		UNSIGNED	Ні	*Немає*		AUTO_INCREMENT	Змінити Знищити Більше
☐	2	**category_id**	int			Ні	*Немає*			Змінити Знищити Більше
☐	3	**name**	varchar(191)	utf8mb4_unicode_ci		Ні	*Немає*			Змінити Знищити Більше
☐	4	**code**	varchar(191)	utf8mb4_unicode_ci		Ні	*Немає*			Змінити Знищити Більше
☐	5	**description**	text	utf8mb4_unicode_ci		Так	*NULL*			Змінити Знищити Більше
☐	6	**image**	varchar(191)	utf8mb4_unicode_ci		Ні	*Немає*			Змінити Знищити Більше
☐	7	**created_at**	timestamp			Так	*NULL*			Змінити Знищити Більше
☐	8	**updated_at**	timestamp			Так	*NULL*			Змінити Знищити Більше
☐	9	**deleted_at**	timestamp			Так	*NULL*			Змінити Знищити Більше

Рис.3.3. - Створена таблиця 'products'

Згодом, при розробці та вдосконалення нашого магазину, потрібно було внести зміни в цю таблицю. Для того, щоб це зробити ми знову використовували зручний механізм міграції. Я добавив в дану таблицю три нових поля для відображення характеристику товару для покупця. Розглянемо процес додавання нових атрибутів (рис.3.4).

2024_01_29_080358_alter_table_products.php

```php
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
    /**
     * Run the migrations.
     */
    public function up(): void
    {
        Schema::table( table: 'products', function (Blueprint $table) {
            $table->tinyInteger( column: 'new')->default( value: 0);
            $table->tinyInteger( column: 'hit')->default( value: 0);
            $table->tinyInteger( column: 'recommend')->default( value: 0);
        });
    }

    /**
     * Reverse the migrations.
     */
    public function down(): void
    {
        Schema::table( table: 'products', function (Blueprint $table) {
            $table->dropColumn(['new', 'hit', 'recommend']);
        });
    }
};

```

Рис.3.4. - Додавання нових атрибутів в таблицю 'products'

В результаті таблиця 'products' буде мати такий фінальний вигляд після внесення всіх змін (рис.3.5)

	#	Ім'я	Тип	Зіставлення	Атрибути	Нуль	За замовчуванням	Коментарі	Додатково	Дія
☐	1	**id** 🔑	bigint		UNSIGNED	Ні	*Немає*		AUTO_INCREMENT	Змінити Знищити Більше
☐	2	**category_id**	int			Ні	*Немає*			Змінити Знищити Більше
☐	3	**name**	varchar(191)	utf8mb4_unicode_ci		Ні	*Немає*			Змінити Знищити Більше
☐	4	**code**	varchar(191)	utf8mb4_unicode_ci		Ні	*Немає*			Змінити Знищити Більше
☐	5	**description**	text	utf8mb4_unicode_ci		Так	*NULL*			Змінити Знищити Більше
☐	6	**image**	varchar(191)	utf8mb4_unicode_ci		Ні	*Немає*			Змінити Знищити Більше
☐	7	**created_at**	timestamp			Так	*NULL*			Змінити Знищити Більше
☐	8	**updated_at**	timestamp			Так	*NULL*			Змінити Знищити Більше
☐	9	**deleted_at**	timestamp			Так	*NULL*			Змінити Знищити Більше
☐	10	**new**	tinyint			Ні	0			Змінити Знищити Більше
☐	11	**hit**	tinyint			Ні	0			Змінити Знищити Більше
☐	12	**recommend**	tinyint			Ні	0			Змінити Знищити Більше
☐	13	**name_en**	varchar(191)	utf8mb4_unicode_ci		Так	*NULL*			Змінити Знищити Більше
☐	14	**description_en**	text	utf8mb4_unicode_ci		Так	*NULL*			Змінити Знищити Більше

Рис.3.5. - Таблиця 'products'

Аналогічно в інтернет-магазині ми створюємо всі інші таблиці в БД та взаємодіємо із ними.

Для того, щоб коректно всі дані із нашого середовища розробки відображалися в СУБД, нам потрібно налаштувати функції та скрипти.

Спочатку потрібно вказати ім’я БД, логін, пароль для входу адміністратора та отримання доступу до серверів бази даних (рис.3.6).

```
DB_CONNECTION=mysql
DB_HOST=mysql
DB_PORT=3306
DB_DATABASE=dronestore
DB_USERNAME=sail
DB_PASSWORD=password
```

Рис.3.6. - Налаштування параметрів для сервера БД

Наступним кроком буде конфігурація файлу із вказаними параметрами для коректного підключення бо сервера. Для цього ми будемо використовувати файл database.php. Процес налаштування доступу до сервера зображений на рисунку 3.7.

```
'mysql' => [
    'driver' => 'mysql',
    'url' => env( key: 'DATABASE_URL'),
    'host' => env( key: 'DB_HOST', default: '127.0.0.1'),
    'port' => env( key: 'DB_PORT', default: '3306'),
    'database' => env( key: 'DB_DATABASE', default: 'forge'),
    'username' => env( key: 'DB_USERNAME', default: 'forge'),
    'password' => env( key: 'DB_PASSWORD', default: ''),
    'unix_socket' => env( key: 'DB_SOCKET', default: ''),
    'charset' => 'utf8mb4',
    'collation' => 'utf8mb4_unicode_ci',
    'prefix' => '',
    'prefix_indexes' => true,
    'strict' => true,
    'engine' => null,
    'options' => extension_loaded( extension: 'pdo_mysql') ? array_filter([
        PDO::MYSQL_ATTR_SSL_CA => env( key: 'MYSQL_ATTR_SSL_CA'),
    ]) : [],
],
```

Рис.3.7. - Налаштування доступу до сервера БД

Отже, після успішного підключення до СУБД тепер переходимо до розробки функціоналу інтернет-магазину.

3.3. Програмна розробка інтернет-магазину

Інтернет-магазин має структуру клієнт-сервер. Тому, в цьому проекті, я використав для клієнта - шаблонізатор Blade, Bootstrap, JS. А для серверної частини - PHP, Laravel.

Тепер розглянемо розробку основних функцій.

Основне, що полягає в функціоналі інтернет-магазину - це авторизація та реєстрація. На рисунку 3.8 буде зображено процес реєстрації для серверної частини сайту.

```
class RegisterController extends Controller
{
    /*...*/

    use RegistersUsers;

    protected function redirectTo()
    {
        if (Auth::user()->isAdmin()) {
            return route( name: 'home');
        } else {
            return route( name: 'person.orders.index');
        }
    }


    /** Create a new controller instance. ...*/
    public function __construct()
    {
        $this->middleware( middleware: 'guest');
    }

    /** Get a validator for an incoming registration request. ...*/
    protected function validator(array $data)
    {
        return Validator::make($data, [
            'name' => ['required', 'string', 'max:255'],
            'email' => ['required', 'string', 'email', 'max:255', 'unique:users'],
            'password' => ['required', 'string', 'min:8', 'confirmed'],
        ]);
    }

    /** Create a new user instance after a valid registration. ...*/
    protected function create(array $data)
    {
        return User::create([
            'name' => $data['name'],
            'email' => $data['email'],
            'password' => Hash::make($data['password']),
        ]);
    }
}
```

Рис.3.8. - Реєстрація користувача

Цей файл, який має шлях “app/Http/Controllers/Auth/RegisterController.php” в Laravel відповідає за обробку реєстрації нових користувачів. Він використовує трейт RegistersUsers,

який надає методи для відображення форми реєстрації та обробки запиту реєстрації [7].

Основні частини цього файлу:

- Constructor (__construct): Цей метод використовується для встановлення початкового стану об’єкта. У цьому випадку встановлюється гостьовий посередник, який перевіряє, чи має користувач дозвіл перед доступом до методів контролера.
- Валідатор (validator): Даний метод виконує валідацію вхідних даних, тобто виконує перевірку того, що ввів користувач. Це може бути ім’я, пароль, пошта тощо.
- Створення (create): Метод створює нового користувача після того, як дані були перевірені. Створюється новий екземпляр для моделі User та відбувається збереження в БД.
- Перенаправлення (redirectTo): Метод перенаправляє користувач на потрібну сторінку після реєстрації в залежності від ролі користувача. Якщо це адмін - на домашню сторінку, якщо звичайний користувач - сторінка замовлень.

Цей контролер використовується разом з видом resources/views/auth/register.blade.php для відображення форми реєстрації та маршрутом register в routes/web.php для обробки запиту реєстрації.

Частину коду шаблонізатора реєстрації для користувача буде зображена на рисунку 3.9.

```
<div class="container">
    <div class="row justify-content-center">
        <div class="col-md-8">
            <div class="card">
                <div class="card-header">{{ __('Register') }}</div>

                <div class="card-body">
                    <form method="POST" action="{{ route('register') }}">
                        @csrf

                        <div class="row mb-3">
                            <label for="name" class="col-md-4 col-form-label text-md-end">{{ __('Name') }}</label>

                            <div class="col-md-6">
                                <input id="name" type="text" class="form-control @error('name') is-invalid @enderror" name="name" value="{{ old('name') }}" required autocomplete="name" autofocus>

                                @error('name')
                                    <span class="invalid-feedback" role="alert">
                                        <strong>{{ $message }}</strong>
                                    </span>
                                @enderror
                            </div>
                        </div>

                        <div class="row mb-3">
                            <label for="email" class="col-md-4 col-form-label text-md-end">{{ __('Email Address') }}</label>

                            <div class="col-md-6">
                                <input id="email" type="email" class="form-control @error('email') is-invalid @enderror" name="email" value="{{ old('email') }}" required autocomplete="email">

                                @error('email')
                                    <span class="invalid-feedback" role="alert">
                                        <strong>{{ $message }}</strong>
                                    </span>
                                @enderror
                            </div>
                        </div>

                        <div class="row mb-3">
                            <label for="password" class="col-md-4 col-form-label text-md-end">{{ __('Password') }}</label>
```

Рис.3.9. - Шаблонізатор для реєстрації

Є кілька ключових етапів, які варто розглянути, щоб зрозуміти сам механізм реєстрації в Laravel.

1. **Відображення форми реєстрації:** Коли користувач відвідує сторінку реєстрації (маршрут /register), метод “showRegistrationForm” в RegisterController відображає відповідний вид. Вид register.blade.php містить HTML форму для введення даних користувача, таких як ім'я, електронна пошта та пароль [5].

2. **Відправлення форми реєстрації:** Після того, як користувач заповнить форму та натисне на кнопку "Register", дані із форми будуть відправлятися на сервер через HTTP POST запит.

3. **Валідація даних:** Коли сервер отримує дані форми, метод “register” в RegisterController викликає метод “validator”, який виконує перевірку по вимогам на вхідні дані. Якщо дані не проходять валідацію, користувача повертають назад до форми реєстрації з повідомленнями про помилки.

4. **Створення нового користувача:** Якщо дані проходять валідацію, метод “register” викликає метод “create”, який створює новий екземпляр моделі User з вхідними даними та зберігає його в базі даних.

5. **Авторизація користувача:** Після створення нового користувача, метод “register” авторизує користувача за допомогою методу “login” в Auth фасаді.

6. **Перенаправлення користувача:** Метод “register” перенаправляє користувача на визначену сторінку після реєстрації. Це може бути домашня сторінка або будь-яка інша сторінка, визначена в методі “redirectTo” в RegisterController.

Метод авторизації (рис.3.10) є аналогічним.

```php
<?php

namespace App\Http\Controllers\Auth;

use ...

class LoginController extends Controller
{
    /*...*/

    use AuthenticatesUsers;

    protected function redirectTo()
    {
        if (Auth::user()->isAdmin()) {
            return route( name: 'home');
        } else {
            return route( name: 'person.orders.index');
        }

    }
    /**
     * Where to redirect users after login.
     *
     * @var string
     */


    /**
     * Create a new controller instance.
     *
     * @return void
     */
    public function __construct()
    {
        $this->middleware( middleware: 'guest')->except( methods: 'logout');
    }
}
```

Рис.3.10. - Авторизація користувача

Цей контролер використовується разом з видом resources/views/auth/login.blade.php для відображення форми реєстрації та маршрутом login в routes/web.php для обробки запиту авторизації.

Частину коду шаблонізатора авторизації для користувача буде зображена на рисунку 3.11.

```
@section('content')
<div class="container">
    <div class="row justify-content-center">
        <div class="col-md-8">
            <div class="card">
                <div class="card-header">{{ __('Login') }}</div>

                <div class="card-body">
                    <form method="POST" action="{{ route('login') }}">
                        @csrf

                        <div class="row mb-3">
                            <label for="email" class="col-md-4 col-form-label text-md-end">{{ __('Email Address') }}</label>

                            <div class="col-md-6">
                                <input id="email" type="email" class="form-control @error('email') is-invalid @enderror" name="email" value="{{ old('email') }}" required autocomplete="email" autofocus>

                                @error('email')
                                    <span class="invalid-feedback" role="alert">
                                        <strong>{{ $message }}</strong>
                                    </span>
                                @enderror
                            </div>
                        </div>

                        <div class="row mb-3">
                            <label for="password" class="col-md-4 col-form-label text-md-end">{{ __('Password') }}</label>

                            <div class="col-md-6">
                                <input id="password" type="password" class="form-control @error('password') is-invalid @enderror" name="password" required autocomplete="current-password">

                                @error('password')
                                    <span class="invalid-feedback" role="alert">
                                        <strong>{{ $message }}</strong>
                                    </span>
                                @enderror
                            </div>
                        </div>
```

Рис.3.11. - Шаблонізатор для авторизації

Як і процес реєстрації, авторизація має декілька етапів для проходження цієї процедури. Розглянемо основні етапи:

1. **Відображення форми входу:** Коли користувач відвідує сторінку входу (маршрут /login), відповідна форма відображається методом «showLoginForm» з LoginController. Вид login.blade.php містить HTML форму для введення даних користувача, таких як електронна пошта та пароль.

2. **Відправлення форми входу:** Коли користувач заповнює форму та натискає кнопку "Login", дані форми надсилаються на сервер через HTTP POST запит до маршруту /login.

3. **Валідація даних:** Після того, як сервер отримає дані із форми метод “login” із LoginController викликає метод “validateLogin”, який буде перевіряти чи відповідають вхідні дані вимогам. Якщо дані не проходять валідацію, користувача повертають назад до форми входу з повідомленнями про помилки.

4. **Авторизація користувача:** Якщо дані проходять перевірку, метод «login» викликає метод «attempt», щоб перевірити, чи існує в базі

даних користувач із вказаною адресою електронної пошти та паролем. Якщо такий користувач існує, він авторизований у системі.

5. **Перенаправлення користувача:** Метод “login” перенаправляє користувача на визначену сторінку після входу. Це може бути будь-яка сторінка, яка буде визначена в методі “redirectTo” в LoginController.

Тепер перейдемо до розгляду одної із найбільших основної функції для інтернет-магазину - можливість додавання товару до корзини (рис.3.12).

```
public function basketAdd(Sku $skus) // Функція для додавання товару в корзину
{
    $result = (new Basket ( createOrder: true))->addSku($skus);

    if($result){
        session()->flash('success', __( key: 'basket.added') .$skus->product->__('name'));
    }else{
        session()->flash('warning', $skus->product->__('name').__( key: 'basket.not_available_more'));
    }

    return redirect()->route( route: 'basket');
}
```

Рис.3.12. - Додавання товару в корзину

Робота цієї функції полягає в таких етапах:

1. **Параметр функції:** Функція приймає один параметр $skus, який є екземпляром класу Sku. Це SKU (Stock Keeping Unit), який користувач хоче додати до своєї корзини.

2. **Створення нового екземпляра Basket:** Відбувається створення нового замовлення, якщо його ще не існує.

3. **Додавання SKU до корзини:** Відбувається виклик методу “addSku” з об'єкта Basket, щоб передати йому ідентифікатор SKU. Якщо SKU успішно додано, то повертається true, або навпаки, повертається false, якщо не можна добавити (як приклад, якщо товар вже не доступний).

4. **Повідомлення користувачу:** Якщо товар успішно додано, то виводиться повідомлення про успіх, якщо ні, то про помилку.

5. **Перенаправлення користувача:** На завершення, відбувається перенаправлення користувача назад на сторінку корзини.

Також є можливість видалити товар із корзини (рис.3.13).

```
public function basketRemove(Sku $skus) // Функція для видалення товару з корзини
{
    (new Basket ())->removeSku($skus);

    session()->flash('warning', __( key: 'basket.removed').$skus->product->__('name'));

    return redirect()->route( route: 'basket');
}
```

Рис.3.13. - Метод для видалення товару в корзину

Даний метод звертається до функції “removeSku”(рис.3.14), яка і видаляє товар із корзини, потім виводить повідомлення, що товар із такою назвою видалений та повертає на сторінку корзини.

```
public function removeSku(Sku $sku)
{
    if ($this->order->skus->contains($sku)) { /*Перевірка чи є вже такий продукт в корзині*/
        $pivotRow = $this->order->skus->where('id', $sku->id)->first(); /*Отримання елементу з корзини*/
        if ($pivotRow->countInOrder < 1) { /*Перевірка чи кількість товару в корзині менше 1*/
            session()->flash('success', __( key: 'basket.removed') . $sku->product->__('name'));
        } else { /*Якщо кількість товару в корзині більше 1*/
            $pivotRow->countInOrder--; /*Зменшення кількості товару в корзині на 1*/
        }
        }
}
```

Рис.3.14. - Функція для видалення товару в корзину

Ця функція спочатку перевіряє, чи дійсно цей товар знаходить в корзині. Якщо так, то в залежності від кількості товару в корзині відбувається або повністю видалення із замовлення товару, або зменшення на одну кількість.

Наступна дуже цікава та корисна функція для закордонних клієнтів - отримання ціни товарів у різних валютах. Гривні, долари та євро - грошові одиниці, для яких можна отримати ціну на даний момент часу. Для реалізації

цієї можливості нам потрібні функції, щоб відбувалася конвертація валют (рис.3.15) та отримання поточного курсу валют (рис.3.16).

```php
public static function convert($sum, $originCurrencyCode = self::DEFAULT_CURRENCY_CODE, $targetCurrencyCode = null) //Конвертація валют
{
    self::loadContainer(); //Завантаження валют

    $originCurrency = self::$container[$originCurrencyCode]; //Отримання валюти по коду
    if($originCurrency->code != self::DEFAULT_CURRENCY_CODE){ //Якщо валюта не дефолтна
        if($originCurrency->rate != 0 || $originCurrency->updated_at->startOfDay() != Carbon::now()->startOfDay()) /*Для обновлення курсу валюти кожного дня*/ {
            CurrencyRates::getRates(); //Отримання курсу валют
            self::loadContainer(); //Завантаження валют
            $originCurrency = self::$container[$originCurrencyCode]; //Отримання валюти по коду
        }
    }

    if (is_null($targetCurrencyCode)) { //Якщо цільова валюта не вказана
        $targetCurrencyCode =  self::getCurrencyFromSession(); //Отримання валюти з сесії
    }
    $targetCurrency = self::$container[$targetCurrencyCode]; //Отримання валюти по коду
    if($originCurrency->code != self::DEFAULT_CURRENCY_CODE){ //Якщо валюта не дефолтна
        if($targetCurrency->rate == 0 || $targetCurrency->updated_at->startOfDay() != Carbon::now()->startOfDay())  { //Для обновлення курсу валюти кожного дня
            CurrencyRates::getRates();  //Отримання курсу валют
            self::loadContainer(); //Завантаження валют
            $targetCurrency = self::$container[$targetCurrencyCode]; //Отримання валюти по коду
        }
    }

    return round( num: $sum / $originCurrency->rate * $targetCurrency->rate,  precision: 2); //Повертаємо конвертовану суму
}
```

Рис.3.15. - Конвертація валют

Для конвертації валют я створив такий етап реалізації:

Спочатку функція отримує три основні параметри - суму, код валюти, яку конвертують, код валюти, в яку конвертують. Якщо курс валюти не є актуальним, то викликається метод “getRates”, який отримує актуальні курси валют з API та відбувається оновлення в БД.

```php
<?php

namespace App\Services;

use Exception;
use GuzzleHttp\Client;

class CurrencyRates
{
    public static function getRates() //Отримання курсу валют
    {
        $baseCurrency = CurrencyConversion::getBaseCurrency(); //Отримання базової валюти

        $url = config( key: 'currency_rates.api_url') . '?base=' . $baseCurrency->code; //Формування URL для отримання курсу валют

        $client = new Client(); //Створення клієнта
        $response = $client->request( method: 'GET', $url); //Відправлення запиту на сервер

        if($response->getStatusCode() !== 200) { //Якщо статус відповіді не 200
            throw new Exception( message: 'Error getting currency rates'); //Викидаємо помилку
        }

        $rates = json_decode($response->getBody()->getContents(),  associative: true)['rates']; //Отримання курсу валют

        foreach (CurrencyConversion::getCurrencies() as $currency) { //Для кожної валюти
            if (!$currency->isMain()) { //Якщо валюта не головна
                if (!isset($rates[$currency->code])) { //Якщо курс валюти не вказаний
                    throw new Exception( message: 'There is a problem with currency ' . $currency->code); //Викидаємо помилку
                } else { //Якщо курс валюти вказаний
                    $currency->update(['rate' => $rates[$currency->code]]); //Оновлюємо курс валюти
                    $currency->touch(); //Оновлюємо час оновлення
                }
            }
        }
    }
}
```

Рис.3.16. - Отримання актуальних курсів валют

В цій функції спочатку отримується інформація про базову валюту, потім відбувається формування URL для API. Після цього йде GET запит. Отримує функція відповідь від API, перетворює у JSON формат та отримує актуальні курси валют. Посилання на API зображено на рисунку 3.17.

```php
<?php

return [
    'api_url' => 'https://bank.gov.ua/NBUStatService/v1/statdirectory/exchange?json',
];
```

Рис.3.17. - API для курс валют

Для сервісу лояльної підтримки клієнтів та програми знижок було прийнято рішення розробити сервіс для купонів, які будуть надавати клієнтам знижки у відсотках або конкретної суми від замовлення.

На рисунку 3.18 представлений клас Coupon, який представляє купони в базі даних. В цій моделі існують методи для роботи із купонами.

```php
class Coupon extends Model
{
    use HasFactory;

    protected $fillable = ['code', 'value', 'type', 'currency_id', 'only_once', 'expired_at', 'description'];

    protected $dates = ['expired_at'];

    public function orders()
    {
        return $this->hasMany( related: Order::class);
    }

    public function currency()
    {
        return $this->belongsTo( related: Currency::class);
    }

    public function isAbsolute()
    {
        return $this->type === 1;
    }

    public function isOnlyOnce()
    {
        return $this->only_once === 1;
    }

    public function availableForUse() // Функція для перевірки чи купон можна використовувати
    {
        $this->refresh();
        if (!$this->isOnlyOnce() || $this->orders->count() === 0) {
            return is_null($this->expired_at) || Carbon::parse($this->expired_at)->gte(Carbon::now());
        }
        return false;
    }

    public function applyCost($price, Currency $currency = null) // Функція для застосування купона
    {
        if ($this->isAbsolute()) {
            return $price - CurrencyConversion::convert($this->value, $this->currency->code, $currency->code);
        } else {
            return $price - ($price * $this->value / 100);
        }
```

Рис.3.18. - Модель Coupon

Для оброблення HTTP-запитів використовується контролер “CouponController” [18] (рис.3.19).

```php
class CouponController extends Controller
{
    /**
     * Display a listing of the resource.
     */
    public function index() // Виведення списку купонів
    {
        $coupons = Coupon::paginate(10);
        return view( view: 'auth.coupons.index', compact( var_name: 'coupons'));
    }

    /**
     * Show the form for creating a new resource.
     */
    public function create()
    {
        return view( view: 'auth.coupons.form');
    }

    /**
     * Store a newly created resource in storage.
     */
    public function store(CouponRequest $request) // Додавання нового купона
    {
        $params = $request->all();
        foreach (['type', 'only_once'] as $fieldName) {
            if (isset($params[$fieldName])) {
                $params[$fieldName] = 1;
            }
        }

        if (!$request->has( key: 'currency_id')) {
            unset($params['currency_id']);
        }

        Coupon::create($params);
        return redirect()->route( route: 'coupons.index');
    }

    /**
     * Display the specified resource.
     */
    public function show(Coupon $coupon) // Перегляд інформації про купон
    {
        return view( view: 'auth.coupons.show', compact( var_name: 'coupon'));
    }

```

Рис.3.19. - Контролер “CouponController”

Цей контролер потрібний для того, щоб обробляти запити.

3.4. Інструкція для користувача

Результатом дипломного проекту є створений інтернет-магазин для продажу дронів. Сайт має корисний та зрозумілий функціонал, привабливий дизайн та гнучкий інтерфейс.

Головна сторінка сайту відкриває доступ до всіх інших. На цій сторінці зображено головне меню сайту, вибір мови, валюти, фільтр, товари та статистика внизу сторінки. Головну сторінку сайту можна поділити на три частини.

Перша частина - містить головне меню. Користувач може обрати, яку інформацію він хоче отримати, а саме: інформацію про інтернет-магазин,

відображення всіх товарів або категорій, перейти на сторінку своїх замовлень, вибір грошової одиниці та мови, перейти на сторінку авторизації (рис. 3.20).

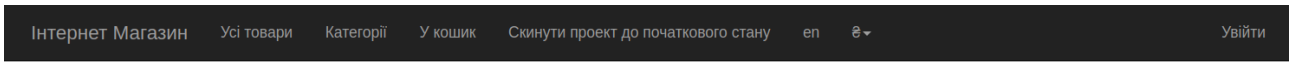

Рис. 3.20. - Головне меню сайту

Друга частина - це фільтр, по якому можна відібрати товари та відображення основних характеристик про кожний товар (рис. 3.21).

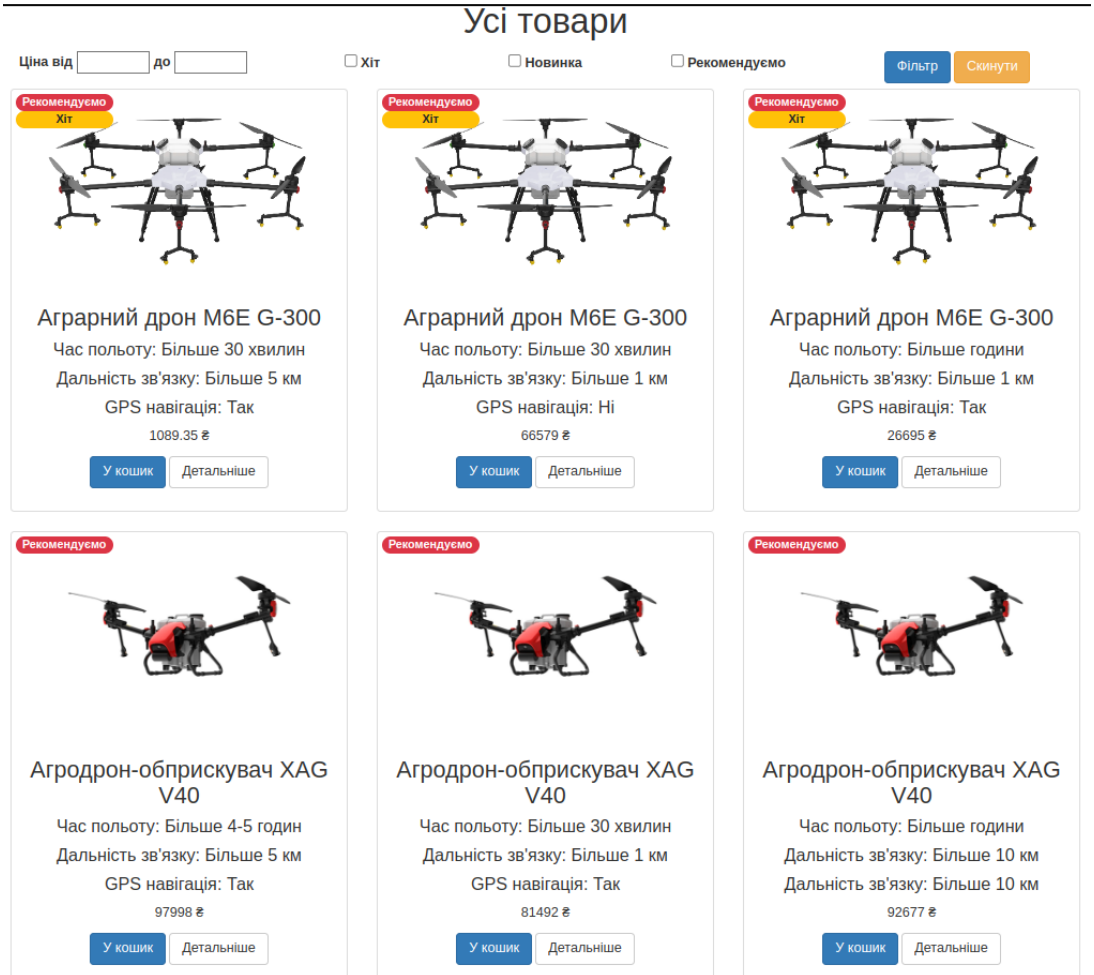

Рис. 3.21. - Друга частина

Третя частина - включає в себе навігацію по сторінкам товарів, список для швидкого доступу до категорій товару та список товарів, які найбільше продаються (рис. 3.22).

« Попередня Наступна »

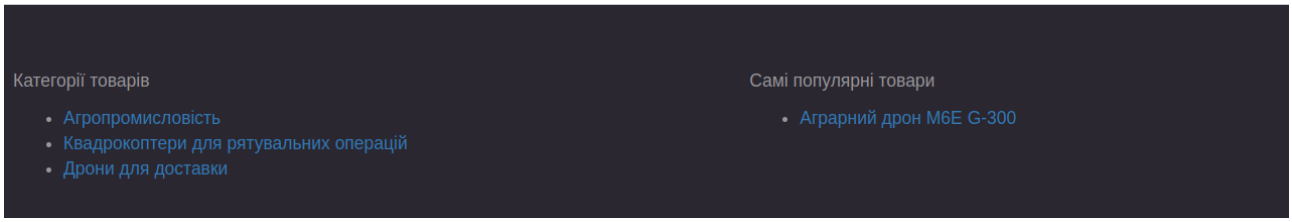

Рис. 3.22. - Третя частина

Коли користувач натискає на головному меню праворуч кнопку “Увійти”, він переходить на сторінку авторизації користувачів. На цій сторінці відображається форма (рис. 3.23), яку потрібно заповнити даними, які були вказані при реєстрації.

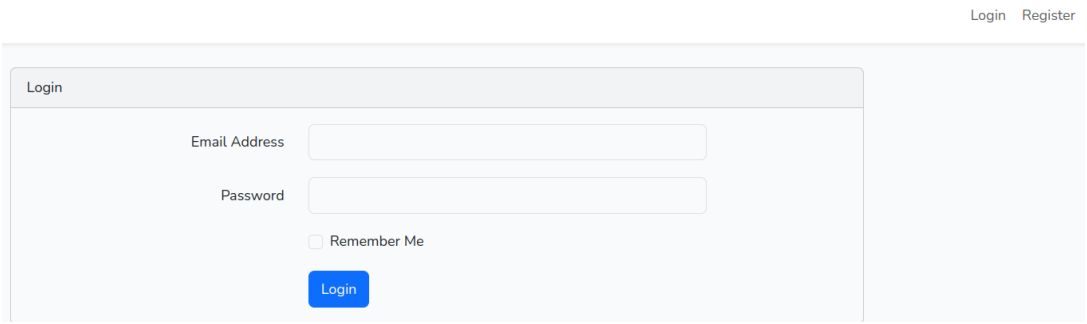

Рис. 3.23. - Сторінка авторизації

Якщо користувач ще не був зареєстрований в інтернет-магазині, то, знову ж таки, праворуч знаходиться кнопка “Register”. Коли користувач натисне на неї, йому знову відкривається форма для заповнення, але там потрібно буде заповнити більше даних про себе (рис. 3.24).

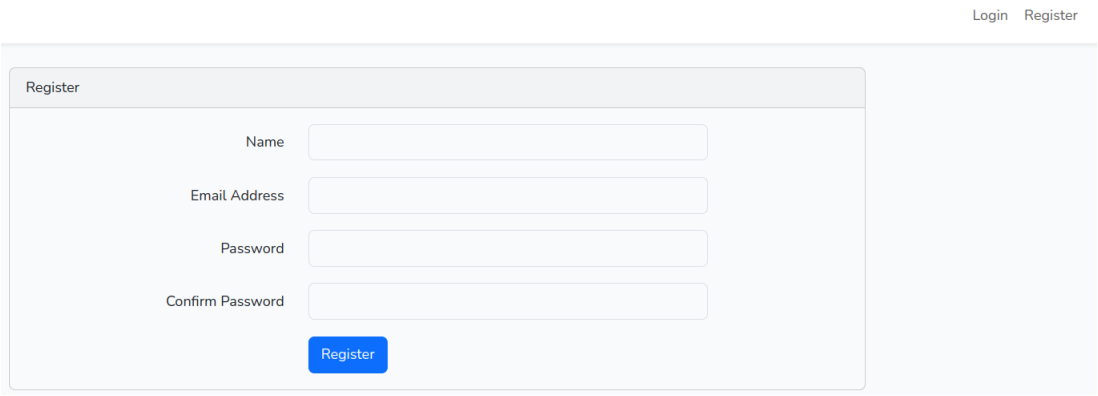

Рис. 3.24. - Сторінка реєстрації

Після того, як користувач авторизується, він автоматично попадає на сторінку, де відображаються всі його замовлення (рис. 3.25). Також на цій сторінці буде доступно повернутися на сторінку сайту, де він може створити нове замовлення або вийти із особистого кабінету.

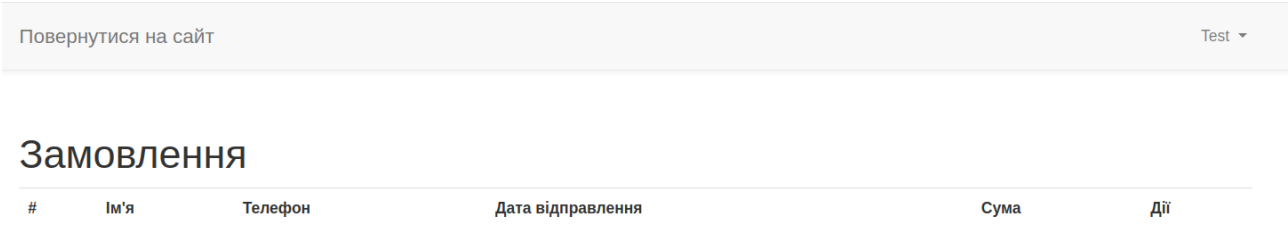

Рис. 3.25. - Сторінка замовлень користувача в особистому кабінеті

Для того, щоб додати товар у кошик, користувачу потрібно натиснути на кнопку “У кошик” на головній сторінці сайту (рис. 3.20) або на кнопку “Додати в кошик” на сторінці, де відображається конкретно інформація про певний товар (рис. 3.26).

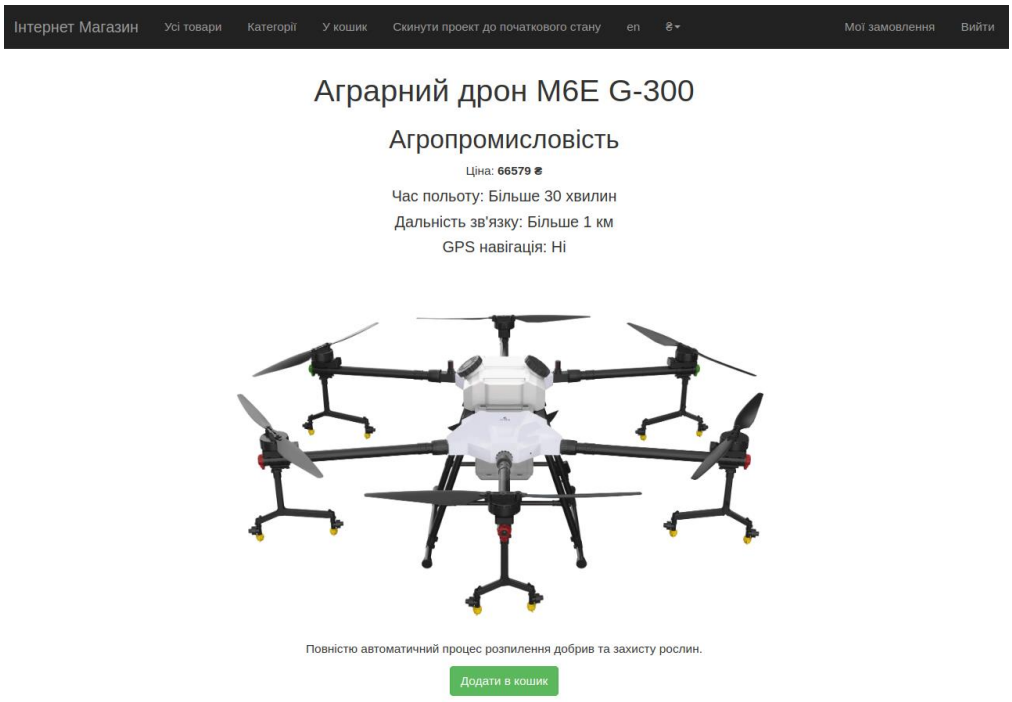

Рис. 3.26. - Інформація про певний товар

Після натискання на кнопку додавання товару до замовлення, користувач перейде на сторінку, яка має назву “Корзина” та вгорі отримає повідомлення про те, що цей товар додано до корзини (рис. 3.27). На цій сторінці можна добавити або видалити одиницю товару, застосувати купон та оформити замовлення до кінця.

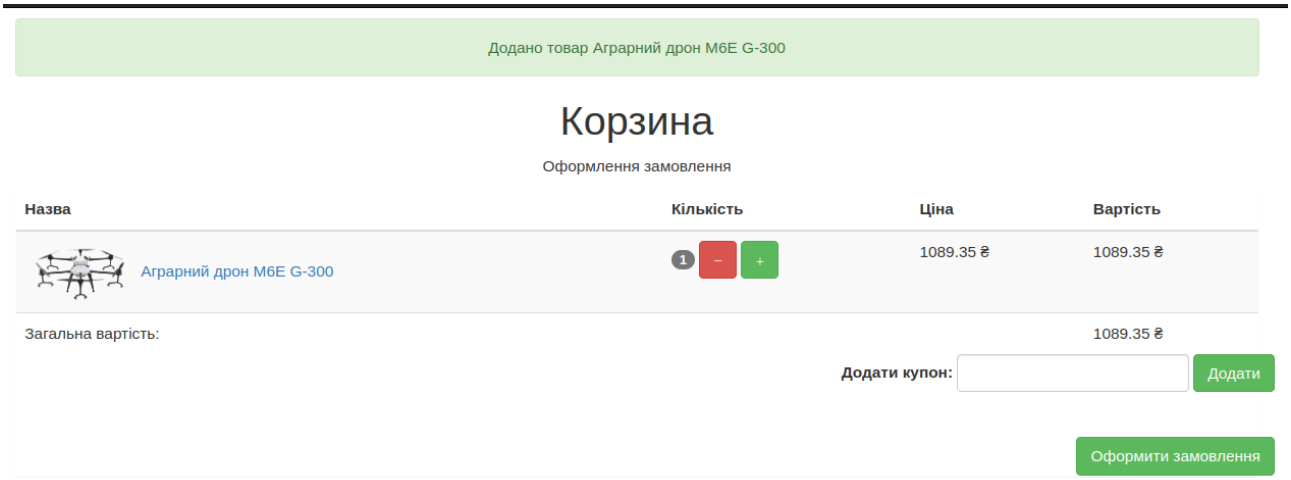

Рис. 3.27. – Корзина

Коли натискаємо на кнопку “Оформити замовлення”, ми переходимо на нову сторінку (рис.3.28). Перед нами з’являється форма, яку ми повинні заповнити та натиснути на кнопку “Підтвердіть замовлення”. На цій сторінці ми можемо побачити суму замовлення.

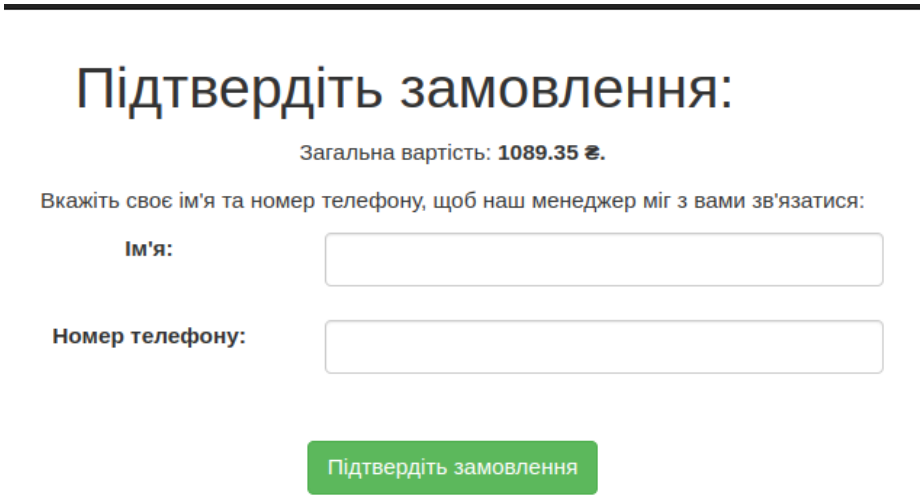

Рис. 3.28. – Інтерфейс під час підтвердження замовлення

Коли у вас з’явилося, як мінімум, одне замовлення, воно відразу буде відображатися у вашому особистому кабінеті. Щоб перейти до нього, натисніть кнопку “Мої замовлення”, яке знаходиться вгорі праворуч. У власному кабінеті ми маємо змогу виконувати дії із власним замовленням - це відкрити його та переглянути детальну інформацію або видалити (рис. 3.29). Також нам доступна інформація про: ім’я, яке ми вказували при оформленні замовлення, номер телефону, дата оформлення, сума замовлення.

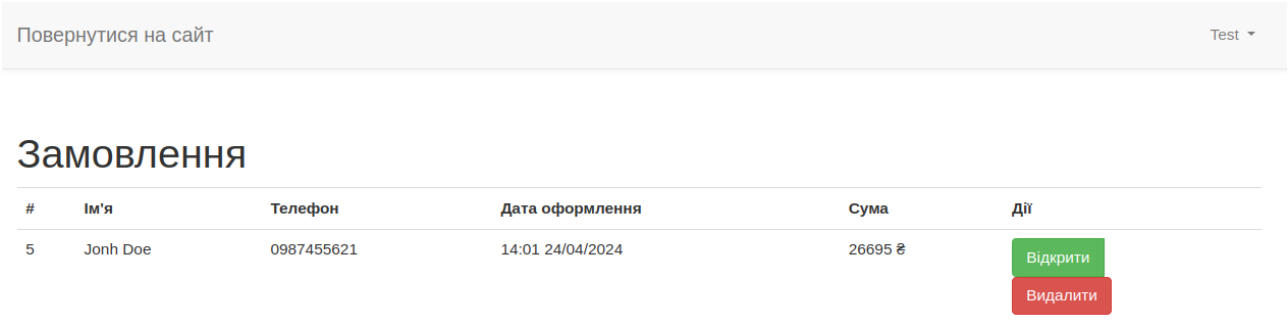

Рис. 3.29. – Інформація про власні замовлення

Коли натискаємо на кнопку “Відкрити”, то переходимо на сторінку, де детально вказана інформація про замовлення. Також маємо змогу повернутися назад до перегляду всіх замовлень (рис. 3.30.).

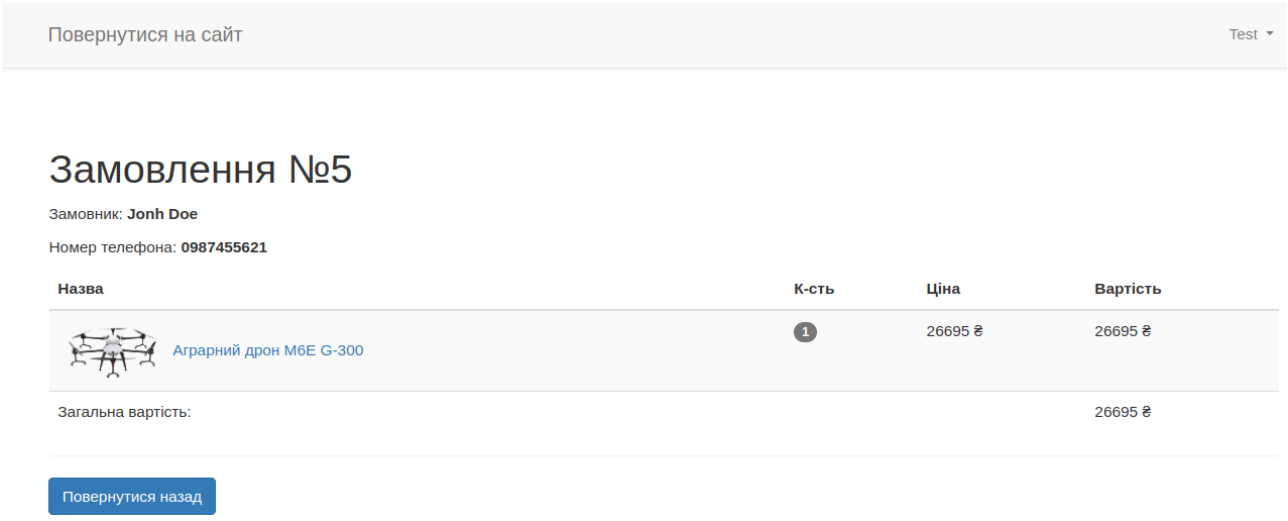

Рис. 3.30. - Детальна інформація про замовлення

Тепер перейдемо до розділу “Категорії” (рис.3.31). Ця сторінка сайту нам покаже, які доступні категорії товарів, ми можемо переглянути та містить короткий опис кожної категорії.

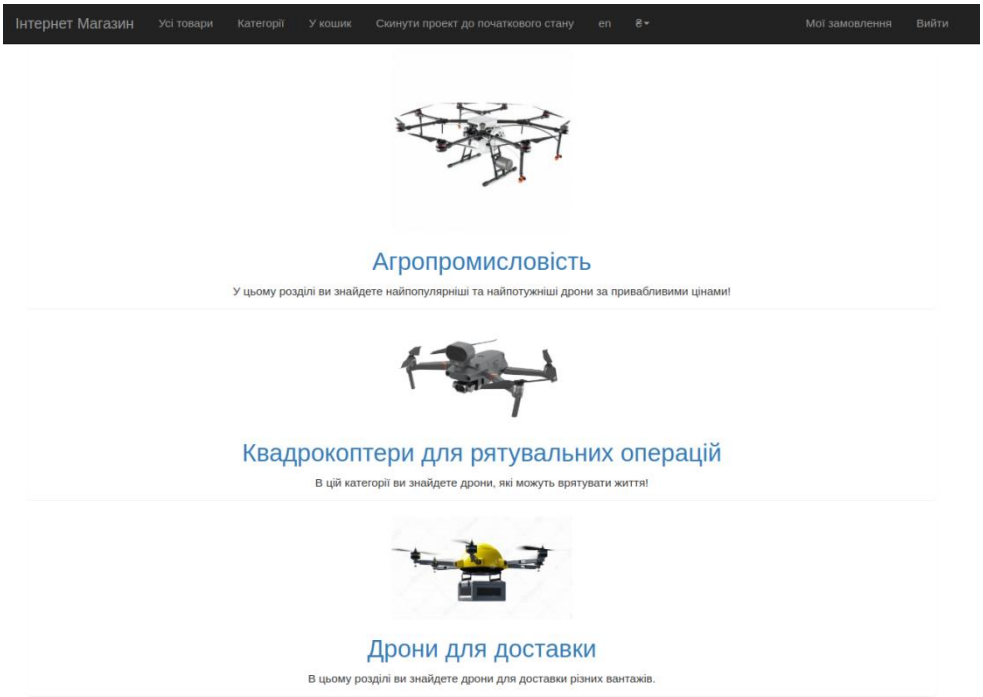

Рис. 3.31. – Сторінка з категорій товарів

Також для користувача є сторінка, де можна зв’язатися із менеджером інтернет-магазину, отримати інформацію про ціль та мету створення та перейти в соціальні мережі (рис. 3.32).

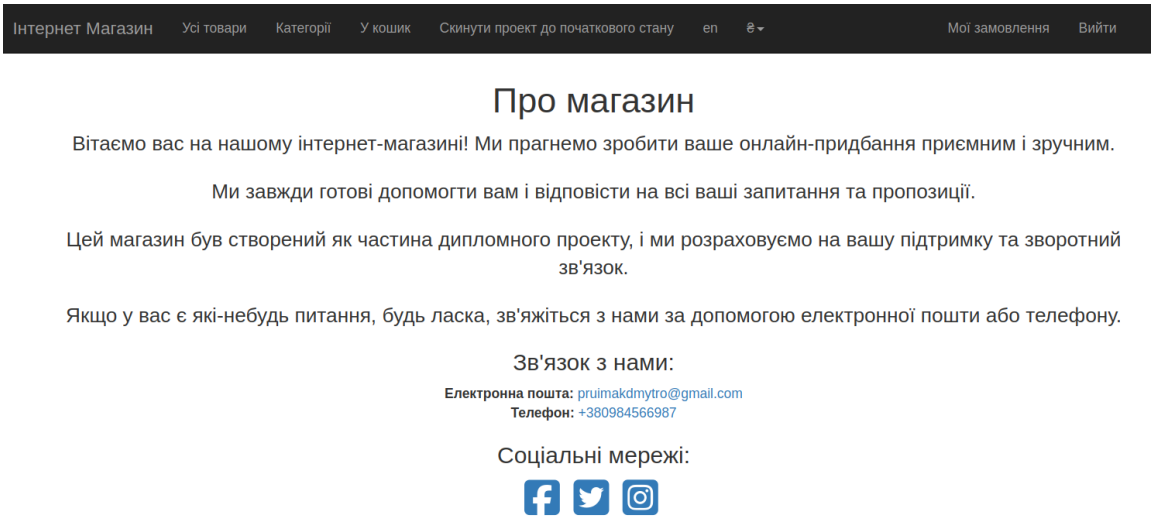

Рис. 3.32. – Інформація про інтернет-магазин

Але, крім звичайних користувачів інтернет-магазину, є ще адміністратор інтернет-магазину. В нього дуже важлива роль - це контролювати процес торгівлі, добавляти товари, змінювати про них дані, отримувати коректну інформацію про замовлення, кількість товару, оформлювати дані про купони тощо.

Для того, щоб авторизувати під адміністратором, потрібно перейти на звичайну сторінку авторизації, ввести там свої дані. При розробці інтернет-магазина, розробником надається доступ до СУБД замовнику. І замовник вже сам визначає, який користувач має право на адмініструванням інтернет-магазином.

Коли користувач авторизується під обліковим записом адміністратора, то він попадає на сторінку замовлень користувача (рис.3.33). Йому надається інформація про замовлення аналогічна, яку ми бачили раніше в особистому кабінеті користувача. Проте, адміністратор має право переглядати та видаляти будь-яке замовлення будь-якого користувача.

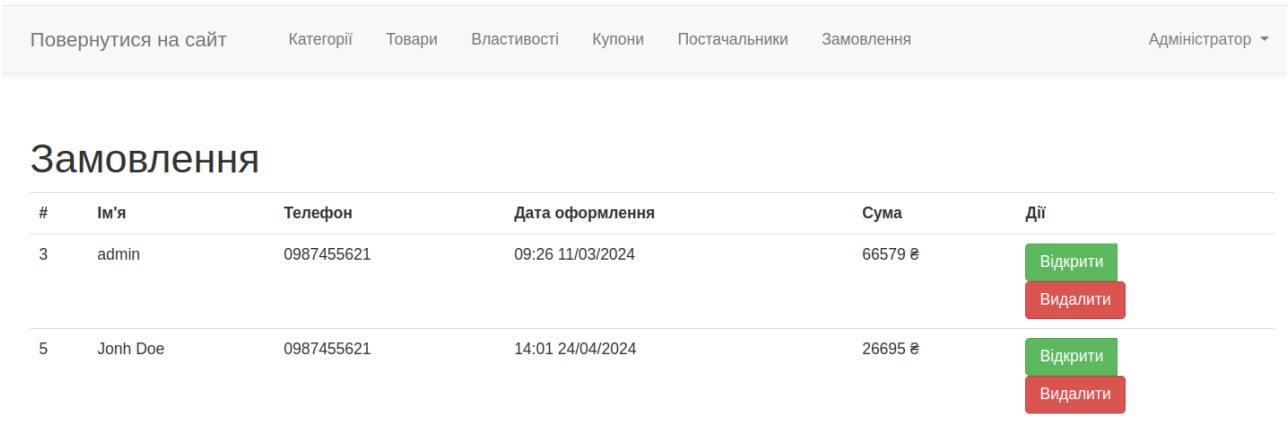

Рис. 3.33. - Сторінка замовлень адміністратора

Як ми можемо вже побачити, вгорі у адміністратора є меню, яке складається із: категорії, товари, властивості, купони, постачальники, замовлення. Розглянемо детально кожне із них.

Категорії - адміністратор бачить назву, код категорії та може виконувати наступні дії: відкрити детальну інформацію про певну категорії, редагувати інформацію або добавити та видалити цю категорію (рис.3.34).

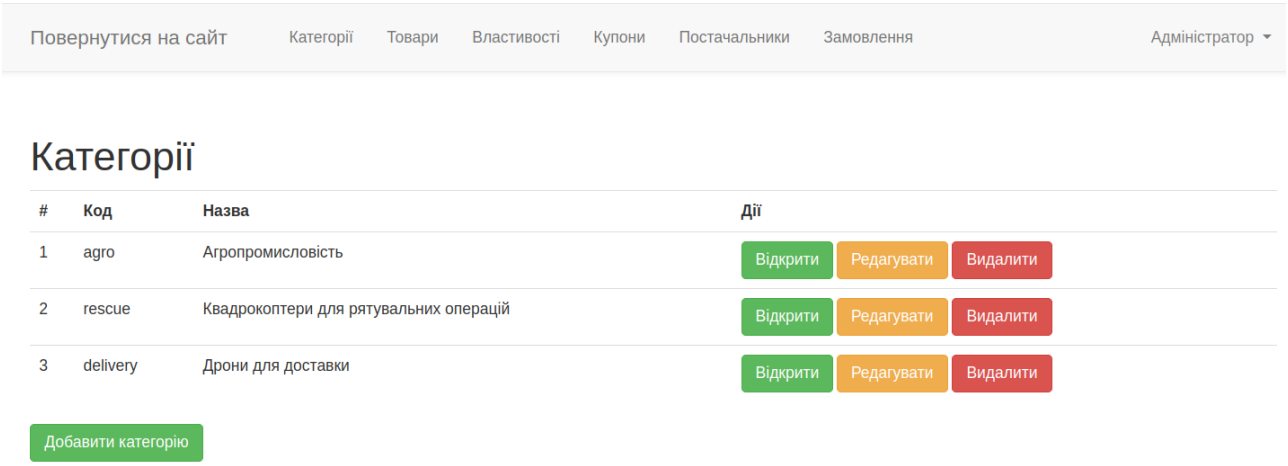

Рис. 3.34. - Сторінка категорії адміністратора

Коли ми натиснемо на кнопку “Відкрити”, то побачимо детальний опис цієї категорії та фото заставки (рис.3.35).

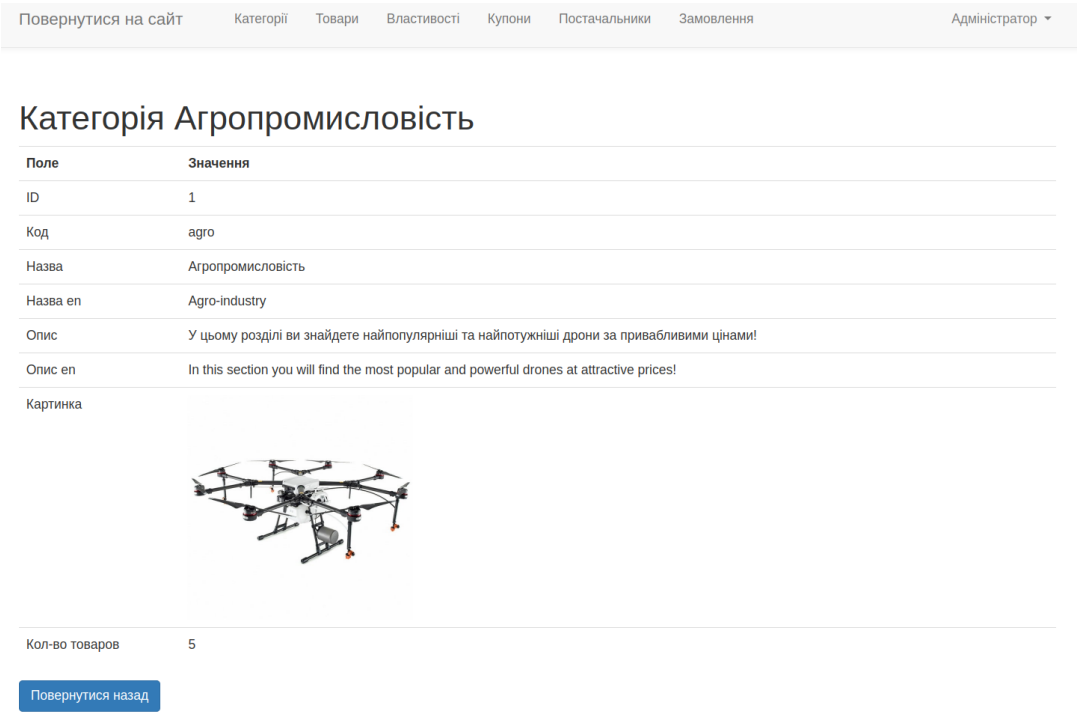

Рис. 3.35. - Сторінка про певну категорію адміністратора

А якщо натиснемо на кнопку “Редагувати”, то будемо мати змогу змінити інформацію про кожне поле цієї категорії.

Наступна кнопка на головному меню адміністратора - товари. Коли ми перейдемо на цю сторінку, то отримаємо інформацію про ідентифікатор, код, назву товару, категорію, до якої відноситься товар, кількість товарних пропозицій та перелік дій, які можемо виконувати із цим товаром (рис.3.36).

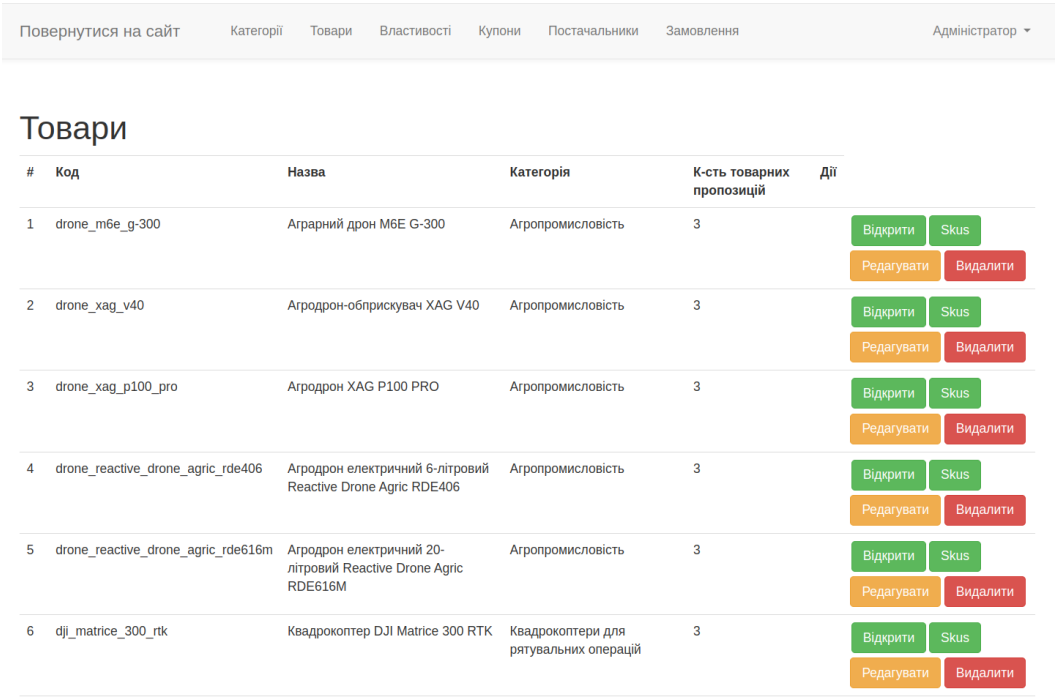

Рис. 3.36. - Сторінка товари адміністратора

Кнопки “Відкрити” та “Редагувати” виконують ті самі дії, що у меню “Категорії”. Розглянемо кнопку “SKUS”. Як ми вже знаємо SKU - це товарна пропозиція, тобто, скільки варіантів товару ми можемо надати для продажу. Якщо перейти по цій кнопці, то адміністратор перед собою побачить інформацію про SKU певного товару та матиме змогу виконати ті самі дії (рис. 3.37).

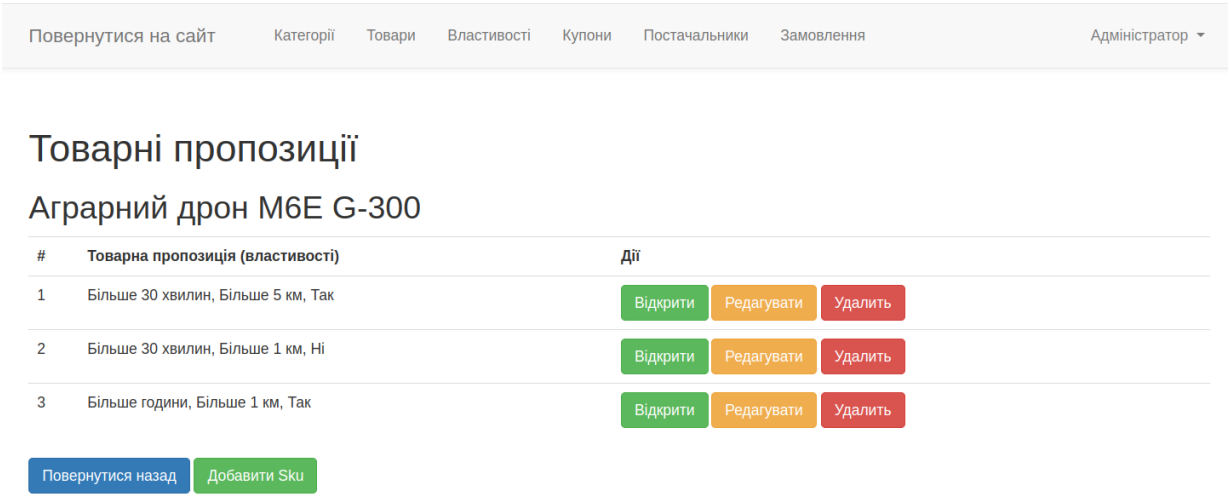

Рис. 3.37. - Сторінка SKU адміністратора

Кнопки “Відкрити” та “Редагувати” виконують аналогічні дії.

У кожного товару мають бути властивості. І за це відповідає наступна кнопку на головному меню. При натисканні на неї, адміністратор переходить на схожу сторінку, де має аналогічні функції при роботі (рис.3.38).

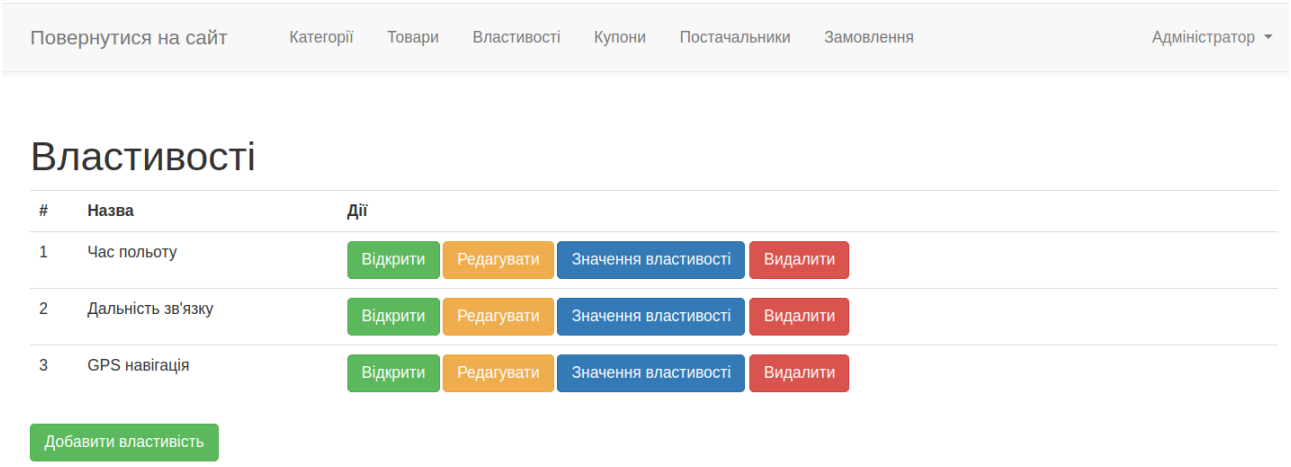

Рис. 3.38. - Сторінка властивості адміністратора

Перейдемо до кнопки “Купони”. Знову ж таки, бачимо схожу сторінку та схожий функціонал при роботі. Але при додаванні нового купона, працюємо із різними типами даних (рис.3.39). Адміністратор повинен ввести там такі дані:

- Код - власне той текст, по якому буде ідентифікуватися та застосовуватися купон;
- Номінал - якщо купон застосовується на певну суму тільки, то вводиться сума знижки;
- Валюта - із списку вибирається валюта для знижки;
- Абсолютне значення - поле, яке має тільки два значення: так або ні. Якщо так, то від номіналу вираховується відсоток знижки.
- Кількість використання купона - також поле має тільки два значення: так або ні. Якщо так, то купон може бути використаний лише один раз.
- Використати до - вибирається із календаря дата до якого числа дійсний купон.
- Опис - адміністратор в цьому полі може написати текст для клієнтів, який допоможе зрозуміти про дію купона.

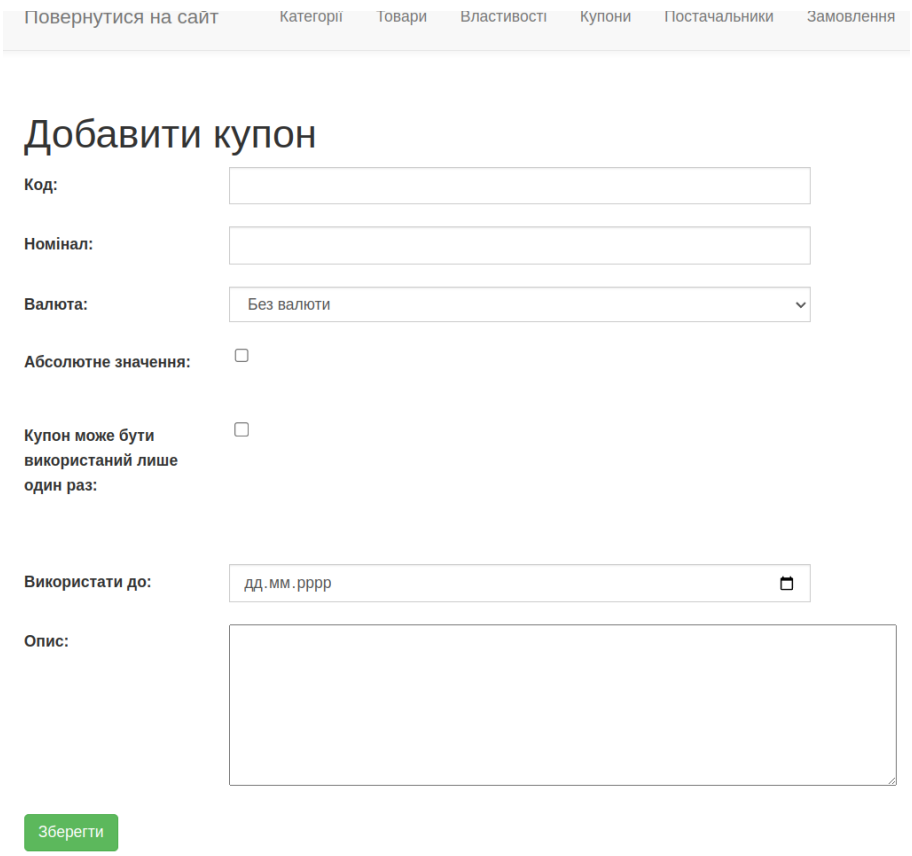

Рис. 3.39. - Сторінка додавання купона адміністратора

Постачальники - адміністратор має змогу вносити всіх постачальників у БД, щоб легко було працювати та знати, який товар від якого постачальника. В цьому меню бачимо ідентифікатор постачальника, ім’я та електронну пошту. Дії виконуються аналогічні (рис.3.40).

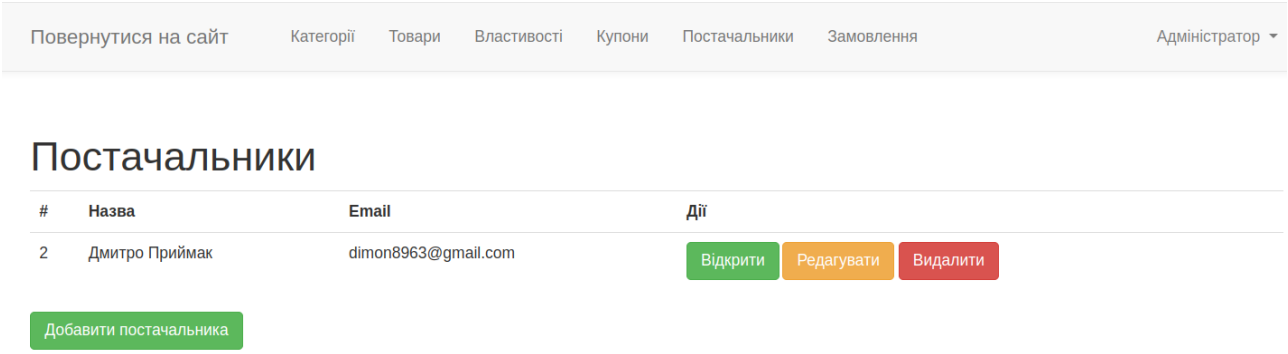

Рис. 3.40. - Сторінка постачальники адміністратора

Особистий кабінет адміністратора простий у використанні та виконує самі основні та необхідні завдання для функціонування інтернет-магазину.

ВИСНОВКИ

У даній дипломній роботі була здійснена розробка онлайн-магазину дронів з використанням фреймворку Laravel.

Ця робота включає в себе аналіз існуючих аналогів магазину дронів, використання діаграм UML для моделювання системи та впровадження функцій користувача та адміністратора.

Під час виконання цієї роботи, одним із першим етапів було проведення детального аналізу існуючих інтернет-магазинів дронів для виявлення сильних і слабких сторін їх роботи. Цей аналіз став основою для розробки нового магазину, який враховує найкращі практики та уникає недоліків існуючих рішень.

Також в процесі розробки активно використовувалися UML-діаграми, особливо діаграми класів і варіантів використання. Ці діаграми дозволили структурувати систему та визначити основні класи та взаємодії між ними. Поведінкові діаграми допомогли візуалізувати поведінку системи на рівні використання.

У процесі розробки PHP використовувався як основна мова програмування, фреймворк Laravel використовувався для швидкого створення веб-додатків, а MySQL використовувався, як база даних для зберігання важливих даних. Для керування та роботи з базою даних використовувався PhpMyAdmin, що значно спростило процес розробки та тестування.

Для користувачів реалізовано широкий функціонал, що дозволяє їм легко шукати дрони за параметрами, додавати їх у кошик і робити замовлення. Також було реалізовано зручний інтерфейс для адміністраторів для керування продуктами, замовленнями, користувачами та звітами.

Тому розроблений на базі Laravel онлайн-магазин дронів є сучасним та функціональним рішенням, яке враховує кращі практики та забезпечує зручний інтерфейс як для користувачів, так і для адміністраторів.

У процесі розробки використовувалися передові технології та аналітичні методи, що дозволило нам створювати ефективні та практичні продукти.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Бази даних у схемах (на основі фундаменталізованого підходу): навч. посіб. / І.О.Бардус, М.І.Лазарєв, А.О.Ніценко. – Харків: Вид-во «Діса-плюс», 2017. – 133 с. ISBN 978-617-7384-77-8.

2. Болотинюк І.М., Зайцев О.О. Електронний бізнес: навч. посіб. Івано-Франківськ: «Лілея-НВ», 2015. 264с.

3. Дакет, Д., М.:Ексмо HTML и CSS. Розробка та дизайн веб-сайтів, 2015. 480с.

4. Довідник по Bootstrap. URL: https://www.w3schools.com/bootstrap4/default.asp (дата звернення: 12.02.2024).

5. Довідник по HTML. URL: https://www.w3schools.com/html/default.asp (дата звернення: 12.02.2024).

6. Довідник по JavaScript. URL https://uk.javascript.info/ (дата звернення: 17.02.2024).

7. Довідник по Laravel. URL https://laravel.com/ (дата звернення: 17.02.2024).

8. Довідник по PHP. URL https://www.php.net/ (дата звернення: 17.02.2024).

9. Duckett J. PHP & MySQL: Server-side Web Development. Wiley, 2022. 672c.

10. Енди Харрис PHP/MySQL для початківців, Одеса, 2005. 384 с.

11. Етапи створення інтернет-магазину [Електронний ресурс]. – 2014. – Режим доступу до ресурсу: https://essuir.sumdu.edu.ua/bitstream-download/123456789/38235/1/Ivanova.pdf.

12. Леонтьєв Б. PHP 5.0 для початківців, або як створити динамічний веб-сайт. Одеса, 2006. 176 с.

13. Matt Stauffer Laravel: Up & Running. 2023. 55c.

14. Мельник Р.А. Програмування веб-застосувань (фронт-енд та бек-енд): навч. посіб. Львів: Видавництво Львівська політехніка, 2018. 248с.

15. Nixon R. Learning PHP, MySQL & JavaScript: With jQuery, CSS & HTML5. CA: «O'Reilly Media, Inc.», 2018. 834c.

16. Шалева О. І. Електронна комерція: навч. посіб. Київ: Центр учбової літератури, 2011. 216с.
17. UML Use Case Diagram. URL: https://www.uml-diagrams.org/use-casediagrams.html (дата звернення: 10.03.2024).
18. HTTP Session [Електронний ресурс] – Режим доступу до ресурсу: https://laravel.com/docs/9.x/session.
19. Zandstra M. PHP 8 Objects, Patterns, and Practice: Mastering OO Enhancements, Design Patterns, and Essential Development Tools. New York City : Apress, 2021. 858 c.
20. ХМЕЛЕВСЬКИЙ І. Інтернет-магазин: організаційні моменти [Електронний ресурс] / Ігор Хмелевський // «Фактор». – 2018. – Режим доступу до ресурсу: 53098.html.

Додатки

Додаток А

Лістинг програми, BasketController

```
<?php

namespace App\Http\Controllers;

use App\Classes\Basket;
use App\Http\Requests\AddCouponRequest;
use App\Models\Coupon;
use App\Models\Sku;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Auth;

class BasketController extends Controller
{
    public function basket()
    {
        $order = (new Basket ())->getOrder();
        return view('basket', compact('order'));
    }

    public function basketConfirm(Request $request) // Функція для
підтвердження замовлення
    {
        $basket = new Basket ();
        if($basket->getOrder()->hasCoupon() && !$basket->getOrder()->coupon-
>availableForUse()){
            $basket->clearCoupon();
            session()->flash('warning', __('basket.coupon.not_available'));
            return redirect()->route('basket');
        }

        $email = Auth::check() ? Auth::user()->email : $request->email;
        if ($basket->saveOrder($request->name, $request->phone, $email)) {
            session()->flash('success', __('basket.you_order_confirmed'));
        } else {
            session()->flash('warning', __('basket.you_cant_order_more'));
        }

        return redirect()->route('index');
    }

    public function basketPlace() // Функція для відображення сторінки
замовлення
    {
        $basket = new Basket ();
        $order = $basket->getOrder();

        if(!$basket->countAvailable()){
            session()->flash('warning', __('basket.you_cant_order_more'));
            return redirect()->route('basket');
        }
        return view('order', compact('order'));
    }

    public function basketAdd(Sku $skus) // Функція для додавання товару в
корзину
    {
        $result = (new Basket (true))->addSku($skus);

        if($result){
```

```
            session()->flash('success', __('basket.added') .$skus->product-
>__('name'));
        }else{
            session()->flash('warning', $skus->product-
>__('name').__('basket.not_available_more'));
        }

        return redirect()->route('basket');
    }

    public function basketRemove(Sku $skus) // Функція для видалення товару з
корзини
    {
        (new Basket ())->removeSku($skus);

        session()->flash('warning', __('basket.removed').$skus->product-
>__('name'));

        return redirect()->route('basket');
    }

    public function setCoupon(AddCouponRequest $request) // Функція для
застосування купона
    {
        $coupon = Coupon::where('code', $request->coupon)->first();

        if ($coupon->availableForUse()) {
            (new Basket())->setCoupon($coupon);
            session()->flash('success', __('basket.coupon.coupon_added'));
        } else {
            session()->flash('warning', __('basket.coupon.not_available'));
        }

        return redirect()->route('basket');
    }

}
```

Лістинг програми, **CheckIsAdmin**

```
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Auth;
use Symfony\Component\HttpFoundation\Response;

class CheckIsAdmin
{
    /**
     * Handle an incoming request.
     *
     * @param  \Closure(\Illuminate\Http\Request):
(\Symfony\Component\HttpFoundation\Response)  $next
     */
    public function handle(Request $request, Closure $next): Response
    {
        $user = Auth::user();
        if (!$user->isAdmin()) {
            session()->flash('warning', 'У вас немає прав адміністратора');
            return redirect()->route('index');
        }
        return $next($request);
    }
}
```

Лістинг програми, **Product.blade.php**

```
category->name }}</h2>
    <p>@lang('product.price'): <b>{{ $skus->price }} {{ $currencySymbol
}}</b></p>

    @isset($skus->product->properties)
        @foreach ($skus->propertyOptions as $propertyOption)
            <h4>{{ $propertyOption->property->__('name') }}: {{
$propertyOption->__('name') }}</h4>
        @endforeach
    @endisset

    <img class="product-image" src="{{ Storage::url($skus->product->image)
}}">
    <p>{{ $skus->product->__('description') }}</p>

    @if($skus->isAvailable())
        <form action="{{ route('basket-add', $skus->product) }}"
method="POST">
            <button type="submit" class="btn btn-success"
role="button">@lang('product.add_to_cart')</button>

            @csrf
        </form>
    @else

        <span>@lang('product.not_available')</span> {{--Товар не доступний
для замовлення--}}
        <br>
        <span>@lang('product.tell_me'):</span> {{--Скажіть мені, коли товар
з'явиться в наявності--}}
        <div class="warning">
            @if($errors->get('email'))      {{--Поле email обов'язкове для
заповнення--}}
                {!! $errors->get('email')[0] !!}
            @endif
        </div>
        <form method="POST" action="{{ route('subscription', $skus) }}">
            @csrf
            <input type="text" name="email"></input>
            <button type="submit">@lang('product.subscribe')</button>
        </form>
    @endif
@endsection
```

Лістинг програми, **Web.php**

```
<?php

use App\Http\Controllers\AboutController;
use App\Http\Controllers\Admin\CouponController;
use App\Http\Controllers\Admin\MerchantController;
use App\Http\Controllers\Admin\OrderController;
use App\Http\Controllers\Admin\ProductController;
use App\Http\Controllers\Admin\PropertyController;
use App\Http\Controllers\Admin\PropertyOptionController;
use App\Http\Controllers\Admin\SkuController;
use App\Http\Controllers\Auth\LoginController;
use App\Http\Controllers\BasketController;
use App\Http\Controllers\MainController;
```

```
use App\Http\Controllers\ResetController;
use Illuminate\Support\Facades\Auth;
use Illuminate\Support\Facades\Route;
use App\Http\Controllers\Admin\CategoryController;

/*
|--------------------------------------------------------------------------
| Web Routes
|--------------------------------------------------------------------------
|
| Here is where you can register web routes for your application. These
| routes are loaded by the RouteServiceProvider and all of them will
| be assigned to the "web" middleware group. Make something great!
|
*/

Auth::routes([
    'reset' => false,
    'confirm' => false,
    'verify' => false,
]);

Route::get('locale/{locale}', [MainController::class, 'changeLocale'])-
>name('locale');
Route::get('currency/{currencyCode}', [MainController::class,
'changeCurrency'])->name('currency');
Route::get('/logout', [LoginController::class, 'logout'])->name('get-
logout');


Route::middleware(['set_locale'])->group(function() {
    Route::get('reset', [ResetController::class, 'reset'])->name('reset');


    Route::middleware(['auth'])->group (function(){
        Route::group([
            'prefix' => 'person',
            'namespace' => 'Person',
            'as' => 'person.'
        ], function (){
            Route::get('/orders',
[\App\Http\Controllers\Person\OrderController::class, 'index'])-
>name('orders.index');
            Route::get('/orders/{order}',
[\App\Http\Controllers\Person\OrderController::class, 'show'])-
>name('orders.show');
            Route::delete('/orders/{order}',
[\App\Http\Controllers\Person\OrderController::class, 'destroy'])-
>name('orders.destroy');
        });

        Route::group([
            /*'namespace' => 'Admin',*/
            'prefix' => 'admin',
        ], function () {
            Route::group(['middleware' => 'is_admin'], function () {
                Route::get('/orders', [OrderController::class, 'index'])-
>name('home');
                Route::get('/orders/{order}', [OrderController::class,
'show'])->name('orders.show');
                Route::delete('/orders/{order}', [OrderController::class,
'destroy'])->name('orders.destroy');

            });
```

```
            Route::resource('categories', CategoryController::class);
            Route::resource('products', ProductController::class);


            Route::get('products/{product}/skus', [SkuController::class,
'index'])->name('skus.index');
            Route::get('products/{product}/skus/create',
[SkuController::class, 'create'])->name('skus.create');
            Route::get('products/{product}/skus/{sku}',
[SkuController::class, 'show'])->name('skus.show');
            Route::post('products/{product}/skus', [SkuController::class,
'store'])->name('skus.store');
            Route::get('products/{product}/skus/{sku}/edit',
[SkuController::class, 'edit'])->name('skus.edit');
            Route::put('products/{product}/skus/{sku}',
[SkuController::class, 'update'])->name('skus.update');
            Route::delete('products/{product}/skus/{sku}',
[SkuController::class, 'destroy'])->name('skus.destroy');
            /*Route::resource('products/{product}/skus',
SkuController::class);*/

            Route::resource('properties', PropertyController::class);
            Route::resource('merchants', MerchantController::class);
            Route::get('merchant/{merchant}/update_token',
[MerchantController::class, 'updateToken'])->name('merchants.update_token');
            Route::resource('coupons', CouponController::class);
            Route::resource('properties/{property}/property-options',
PropertyOptionController::class);


        });
    });

    Route::get('/', [MainController::class, 'index'])->name('index');
    Route::get('/about', [AboutController::class, 'index'])->name('about');
    Route::get('/categories', [MainController::class, 'categories'])-
>name('categories');
    Route::post('subscription/{skus}', [MainController::class, 'subscribe'])-
>name('subscription');

    Route::group(['prefix' => 'basket'], function (){ /*Щоб не писати
постійно в роуті basket*/
        Route::post('/add/{skus}', [BasketController::class, 'basketAdd'])-
>name('basket-add');

        Route::group([
            'middleware' => 'basket_not_empty',
        ], function (){
            Route::get('/', [BasketController::class, 'basket'])-
>name('basket');
            Route::get('/place', [BasketController::class, 'basketPlace'])-
>name('basket-place'); /*Підтвердження замовлення*/
            Route::post('/remove/{skus}', [BasketController::class,
'basketRemove'])->name('basket-remove');
            Route::post('/place', [BasketController::class,
'basketConfirm'])->name('basket-confirm');
            Route::post('coupon', [BasketController::class, 'setCoupon'])-
>name('set-coupon');
        });
    });
```

```
    Route::get('/{category}', [MainController::class, 'category'])-
>name('category');
    Route::get('/{category}/{product}/{skus}', [MainController::class,
'sku'])->name('sku');


});
```

Лістинг програми, **Create_products_table**

```
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
    /**
     * Run the migrations.
     */
    public function up(): void
    {
        Schema::create('products', function (Blueprint $table) {
            $table->id();
            $table->integer('category_id'); //ID категорії
            $table->string('name'); //Назва товару
            $table->string('code'); //Код товару
            $table->text('description')->nullable(); //Опис товару
            $table->string('image');
            $table->double('price')->default(0); //Ціна товару
            $table->timestamps();
            $table->softDeletes();
        });
    }

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