

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

ПОЯСНЮВАЛЬНА ЗАПИСКА

до кваліфікаційної роботи освітнього ступеня «бакалавр»
за спеціальністю 121 «Інженерія програмного забезпечення»
(освітня програма «Інженерія програмного забезпечення»)

на тему:

«Інтернет-магазин зоотоварів»

Виконав студент групи ІПЗ-21-5
БОГАЙЧУК Денис Володимирович

Керівник роботи:
ЧИЖМОТРЯ Олена Геннадіївна

Рецензент:
МОРОЗОВ Андрій Васильович

Житомир – 2025

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

«ЗАТВЕРДЖУЮ»

Зав. кафедри інженерії програмного
забезпечення

Тетяна Вакалюк

«14» лютого 2025 р.

ЗАВДАННЯ
на кваліфікаційну роботу

Здобувач вищої освіти: **БОГАЙЧУК Денис Володимирович**

Керівник роботи: **ЧИЖМОТРЯ Олена Геннадіївна**

Тема роботи: **«Інтернет-магазин зоотоварів»,**

затверджена Наказом закладу вищої освіти від «14» лютого 2025 р., №61/с

Вихідні дані для роботи: **об'єктом дослідження є онлайн-платформа інтернет-магазин зоотоварів. Предметом дослідження є програмні рішення та інструменти, що використовуються для створення зручного і функціонального веб-додатку для продажу товарів для тварин.**

Консультанти з бакалаврської кваліфікаційної роботи із зазначенням розділів, що їх стосуються:

Розділ	Консультант	Завдання видав	Завдання прийняв
1	Чижмотря О.Г	27.02.2025р.	27.02.2025р.
2	Чижмотря О.Г	27.02.2025р.	27.02.2025р.
3	Чижмотря О.Г	27.02.2025р.	27.02.2025р.

РЕФЕРАТ

Випускна кваліфікаційна робота бакалавра складається з веб-платформи «Інтернет-магазин зоотоварів» та пояснювальної записки. Пояснювальна записка містить 72 сторінки, 15 ілюстрацій, 16 таблиць, 4 додатки і 32 використаних літературних джерела.

Метою роботи є створення сучасного, надійного та функціонального інтернет-зоомагазину, який дозволяє користувачам зручно здійснювати онлайн-замовлення з можливістю фільтрації товарів, оплати онлайн та інтеграції з поштовими сервісами. Платформа має забезпечувати високий рівень зручності для клієнтів, прозорість процесу покупки, гнучкість у керуванні товарами та замовленнями, а також захист персональних даних користувачів.

Для реалізації проєкту використано Laravel як основний фреймворк для серверної частини, Nuxt.js для побудови адаптивного фронтенду, MySQL для зберігання даних, LiqPay для обробки платежів і Nova Poshta API для доставки. Також застосовано об'єктно-орієнтовану модель побудови архітектури, RESTful API для взаємодії між фронтом і бекендом, та інструменти безпеки, такі як валідація даних, захист CSRF і middleware для доступу до адміністративних функцій.

Платформа орієнтована на власників домашніх тварин, з можливістю легко й швидко знаходити необхідні товари за категоріями, брендами, типом тварини тощо. Вона може бути використана як повноцінний інструмент для ведення онлайн-продажів у сфері зоотоварів із можливістю подальшого масштабування.

КЛЮЧОВІ СЛОВА: ІНТЕРНЕТ-МАГАЗИН, ЗООТОВАРИ, LARAVEL, NUXT.JS, MYSQL, ОПЛАТА, ДОСТАВКА, LIQPAY, NOVA POSHTA.

					ІПЗ.КР.Б – 121 – 25 – ПЗ				
Змн.	Арк.	№ докум.	Підпис	Дата					
Розроб.		Богайчук Д.В.			Інтернет-магазин зоотоварів Пояснювальна записка	Літ.	Арк.	Аркушів	
Керівник		Чижмоторя О.Г.					3	72	
						Житомирська політехніка, група ІПЗ-21-5			
Рецензент		Морозов А.В.							
Зав. каф.		Вакалюк Т.А.							

ABSTRACT

The bachelor's thesis consists of the web platform “Online Pet Shop” and an explanatory note. The explanatory note contains 72 pages, 15 illustrations, 16 tables, 4 appendices and 32 references.

The goal is to create a modern, reliable and functional online pet store that allows users to conveniently place online orders with the ability to filter products, pay online and integrate with postal services. The platform should provide a high level of customer convenience, transparency of the purchase process, flexibility in managing products and orders, and protection of users' personal data.

To implement the project, we used Laravel as the main framework for the server side, Nuxt.js to build an adaptive frontend, MySQL for data storage, LiqPay for payment processing, and Nova Poshta API for delivery. The platform also uses an object-oriented architecture model, RESTful APIs for interaction between the frontend and backend, and security tools such as data validation, CSRF protection, and middleware for access to administrative functions.

The platform is targeted at pet owners, with the ability to easily and quickly find the products they need by category, brand, type of animal, etc. It can be used as a full-fledged tool for conducting online sales in the field of pet products with the possibility of further scaling.

KEYWORDS: ONLINE STORE, PET PRODUCTS, LARAVEL, NUXT.JS, MYSQL, PAYMENT, DELIVERY, LIQPAY, NOVA POSHTA.

					ДУ «Житомирська політехніка».25.121.3.000 - ПЗ	Арк.
						4

ЗМІСТ

РЕФЕРАТ	3
ABSTRACT	4
ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ.....	6
ВСТУП.....	7
РОЗДІЛ 1. АНАЛІЗ НАПРЯМКІВ ВИКОРИСТАННЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ДЛЯ РЕАЛІЗАЦІЇ СИСТЕМИ	9
1.1. Постановка задачі.....	9
1.2. Аналіз аналогів програмного продукту	11
1.3. Вибір архітектури веб-орієнтованої системи	15
1.4. Обґрунтування вибору інструментальних засобів та вимоги до апаратного забезпечення	17
Висновки до першого розділу.....	20
РОЗДІЛ 2. ПРОЄКТУВАННЯ ОНЛАЙН-ПЛАТФОРМИ ІНТЕРНЕТ- ЗООМАГАЗИНУ	21
2.1. Визначення варіантів використання та об'єктно-орієнтованої структури системи.....	21
2.2. Розробка бази даних систем	27
2.3. Проєктування та реалізація алгоритмів роботи платформи	31
2.4. Реалізація функціоналу платформи.....	33
Висновки до другого розділу	38
РОЗДІЛ 3. ОПИС РОЗРОБЛЕНОЇ ІНТЕРНЕТ-ПЛАТФОРМИ ЗООМАГАЗИНУ	40
3.1. Порядок встановлення та налаштування параметрів платформи	40
3.2. Структура інтерфейсу платформи Інтернет-зоомаган	42
3.3. Тестування роботи платформи.....	48
Висновки до третього розділу	50
ВИСНОВКИ.....	51
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ.....	53
ДОДАТКИ.....	56

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

API – Application Programming Interface

БД – база даних

ПЗ – програмне забезпечення

СУБД – система управління базами даних

MVC – Model-View-Controller

CRUD – Create, Read, Update, Delete

ORM – Object-Relational Mapping

CSRF – Cross-Site Request Forgery

					ДУ «Житомирська політехніка».25.121.3.000 - ПЗ	Арк.
						6

ВСТУП

У сучасному цифровому середовищі інтернет-торгівля стала невід'ємною частиною повсякденного життя мільйонів користувачів. Особливо стрімко зростає попит на спеціалізовані онлайн-магазини, що пропонують товари для домашніх улюбленців. Власники собак, котів, гризунів, птахів та інших тварин дедалі частіше обирають зручний формат онлайн-покупок замість традиційних поїздок до зоомагазинів. Це пояснюється як економією часу, так і ширшим асортиментом продукції, доступним через кілька кліків. Таким чином, платформи електронної комерції, орієнтовані на сферу зоотоварів, перетворюються на ефективний інструмент для покращення якості життя як тварин, так і їх господарів.

Проте зі зростанням попиту зростають і вимоги до функціональності таких платформ. Користувачі очікують інтуїтивно зрозумілий інтерфейс, можливість фільтрації товарів за типом тварини, брендом чи ціною, швидку обробку замовлень, надійну платіжну систему та, звісно ж, безпеку персональних даних. Усе це ставить перед розробниками низку важливих викликів: як забезпечити масштабованість платформи, інтегрувати функції відгуків, рейтингу товарів, історії замовлень, а також ефективно взаємодіяти з клієнтами в режимі реального часу. Одним із найважливіших завдань є створення такої системи, яка б поєднувала в собі зручність для користувача та надійні адміністративні інструменти для власника магазину.

Розробка інтернет-зоомагазину, що відповідає всім цим критеріям, потребує застосування сучасного технологічного стеку. Зокрема, для реалізації серверної частини було обрано фреймворк Laravel, який надає широкі можливості з організації архітектури системи, реалізації RESTful API, а також вбудованих механізмів безпеки. Для створення динамічного інтерфейсу користувача обрано Nuxt.js - прогресивний фреймворк, що базується на Vue.js і підтримує серверний рендеринг, що особливо важливо для покращення SEO-

показників. Інтеграція Docker, WebSocket та LiqPay дозволяє побудувати не лише зручний, а й технологічно зрілий продукт.

Метою даного дослідження є проєктування, розробка та впровадження інтернет-магазину зоотоварів, який забезпечує зручну, швидку та безпечну взаємодію між продавцем і покупцем, враховуючи особливості електронної торгівлі у сфері товарів для тварин.

Для досягнення цієї мети **передбачено виконання таких завдань:**

- проаналізувати ринок інтернет-магазинів у сфері зоотоварів, визначити функціональні особливості найуспішніших сервісів;
- розробити концепцію та архітектуру системи з урахуванням потреб кінцевих користувачів і можливостей сучасних веб-технологій;
- визначити оптимальний технологічний стек для реалізації платформи, зокрема засоби реалізації клієнтської та серверної частини;
- реалізувати механізми реєстрації, авторизації, управління профілем користувача та історією замовлень;
- інтегрувати систему коментарів, рейтингову оцінку товарів, платіжний модуль LiqPay та модуль адміністрування контенту;
- провести тестування реалізованого функціоналу та оцінити перспективи масштабування платформи в майбутньому.

Об'єктом дослідження є інтегрована веб-система для управління онлайн-магазином зоотоварів, що дозволяє організовувати повний цикл обслуговування клієнтів - від ознайомлення з товаром до отримання оплати.

Предметом дослідження є застосування веб-технологій і методів об'єктно-орієнтованого проєктування (зокрема Laravel, Nuxt.js) для створення зручної, надійної та функціональної платформи електронної комерції у сфері зоотоварів.

РОЗДІЛ 1. АНАЛІЗ НАПРЯМКІВ ВИКОРИСТАННЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ДЛЯ РЕАЛІЗАЦІЇ СИСТЕМИ

1.1. Постановка задачі

У сучасних умовах, коли електронна комерція стає домінуючим каналом реалізації товарів і послуг, особливу актуальність набуває створення спеціалізованих онлайн-платформ, які здатні задовольнити потреби окремих категорій споживачів. Розробка інноваційного інтернет-зоомагазину є не просто технічним викликом, а стратегічним кроком у напрямку модернізації процесів придбання товарів для домашніх тварин. Це завдання охоплює як аспект зручності для кінцевого користувача, так і забезпечення ефективного функціонування всієї системи в умовах динамічного зростання попиту. Проєкт передбачає створення інтерактивної платформи, яка б поєднувала функціональність, безпеку, адаптивність і гнучкість в управлінні контентом, пропонуючи клієнтам комфортний досвід купівлі товарів для своїх улюбленців.

Першочерговим напрямком роботи є реалізація повноцінної системи ідентифікації користувачів. Вона має включати можливість реєстрації нових облікових записів із підтримкою як стандартної форми входу, так і авторизації через соціальні мережі. Завдяки персоналізованому профілю користувач отримає доступ до історії своїх замовлень, можливість зберігати обрані товари, налаштовувати особисті дані та здійснювати комунікацію з адміністрацією. Такий підхід дозволить не лише підвищити зручність користування платформою, а й забезпечити належний рівень персоналізації взаємодії.

Ще одним важливим аспектом є організація логічної структури асортименту товарів. Для цього необхідно розробити адаптивну базу даних, яка дозволяє класифікувати продукцію за типами тварин, брендами, характеристиками, ціновими категоріями та рейтингами. Інтерфейс користувача повинен підтримувати гнучку систему фільтрації, яка дасть змогу максимально швидко знаходити потрібний товар навіть у разі великої кількості позицій у

каталозі. Окрім цього, варто передбачити можливість порівняння декількох товарів між собою для підвищення інформованості покупців.

Значну увагу в межах реалізації даного проєкту приділено оптимізації користувацького досвіду. Інтерфейс має бути інтуїтивно зрозумілим, адаптивним до різних розмірів екранів і сумісним з усіма сучасними браузерами. Візуальне оформлення повинно залишатися естетичним і функціональним незалежно від пристрою, з якого користувач заходить на платформу - чи це мобільний телефон, планшет, ноутбук чи настільний комп'ютер. Плавна анімація, швидка реакція на дії користувача та логічна структура сторінок стануть основою позитивного враження від сервісу.

Ще одним ключовим модулем є система відгуків та оцінювання товарів. Кожен покупець після здійснення покупки має мати змогу залишити свою думку про продукт, а за потреби - додати фото або опис. Це створює основу для прозорої оцінки якості товарів, підвищує рівень довіри до платформи та мотивує продавців покращувати свою продукцію. Окрім того, модерація таких відгуків дозволяє уникати спаму чи неприйняттого контенту, зберігаючи репутацію магазину.

Окрему роль у реалізації платформи відіграє питання безпеки. Необхідно забезпечити захист особистих даних користувачів шляхом використання сучасних методів шифрування, двофакторної автентифікації, перевірки введених даних, а також запобігання основним видам атак, таким як SQL-ін'єкції, XSS та CSRF. Безпека даних є не лише технічним, а й етичним зобов'язанням розробника перед користувачами.

Для повноцінного функціонування проєкту також передбачено розробку адміністративної панелі. Вона повинна надати інструменти для керування контентом сайту, перегляду статистики, модерації відгуків, перегляду дій користувачів та аналітики продажів. Завдяки цим можливостям адміністратор зможе ефективно реагувати на зміни в поведінці користувачів, своєчасно оновлювати асортимент та контролювати актуальність інформації.

					ДУ «Житомирська політехніка».25.121.3.000 - ПЗ	Арк.
						10

Реалізація такого масштабного проєкту потребує правильної організації серверного середовища. Йдеться не лише про вибір надійного хостинг-провайдера та оренду доменного імені, але й про впровадження автоматизованих процесів розгортання та оновлення коду за допомогою CI/CD-практик. Надійність серверної інфраструктури безпосередньо впливає на стабільність роботи платформи, її продуктивність та здатність обслуговувати велику кількість одночасних користувачів.

У підсумку, створення сучасного інтернет-зоомагазину потребує не лише глибоких технічних знань, а й аналітичного підходу до розуміння потреб ринку, грамотного UX-дизайну, продуманого механізму роботи із замовленнями та широкого спектру інструментів для управління і масштабування системи. Це має бути комплексне рішення, що поєднує у собі зручність користувача, гнучкість для адміністратора і високі стандарти безпеки. Успішна реалізація проєкту дозволить створити надійний інструмент для електронної комерції у сфері зоотоварів, який стане актуальним і затребуваним на ринку.

1.2. Аналіз аналогів програмного продукту

Для покращення процесу розробки веб-додатку у сфері зоотоварів доцільно здійснити попередній аналіз наявних аналогічних рішень на українському ринку. Це дозволяє виявити сильні та слабкі сторони існуючих проєктів, зрозуміти потреби цільової аудиторії та створити рішення, що буде більш адаптованим до сучасних очікувань користувачів.

Одним із прикладів таких платформ є **Zvirshop** - сайт, що вже певний час працює в Україні як інтернет-магазин товарів для тварин (рис. 1.1).

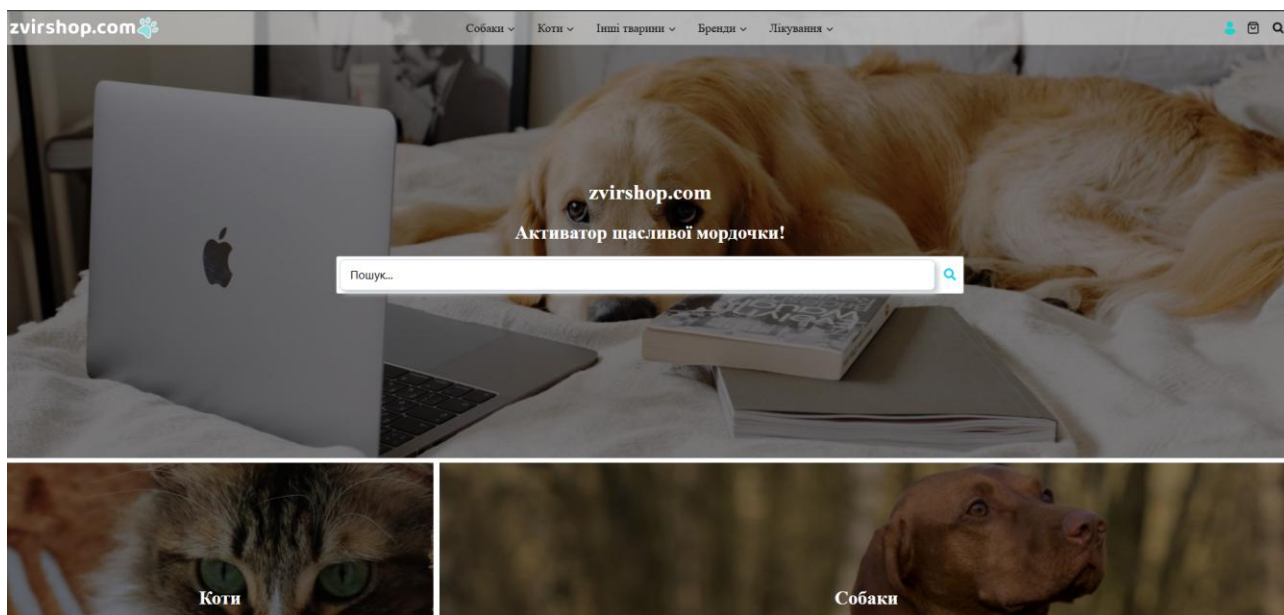


Рис. 1.1 – Вигляд головної сторінки веб-сайту Zvirshop

Zvirshop пропонує базовий набір функцій: каталог товарів, систему кошика та можливість оформлення замовлення. Інтерфейс доволі простий, що на перший погляд може здатися перевагою, проте у процесі взаємодії стає помітним, що навігація є неінтуїтивною, фільтрація відсутня або працює некоректно, а сторінки мають перевантаження елементів. Крім того, мобільна версія не забезпечує повного повторення функціональності, що ускладнює використання сайту з телефонів або планшетів. Контент часто виглядає застарілим, а деякі розділи виглядають як тимчасові, недоопрацьовані або взагалі неактивні. Серед переваг варто відзначити доступ до основних товарів - кормів, аксесуарів, засобів догляду, однак подача не супроводжується зручними візуальними рішеннями або підтримкою користувача.

Іншим прикладом є платформа Тоба.ua - ще один, що пропонує товари для тварин у форматі онлайн-магазину (рис. 1.2).

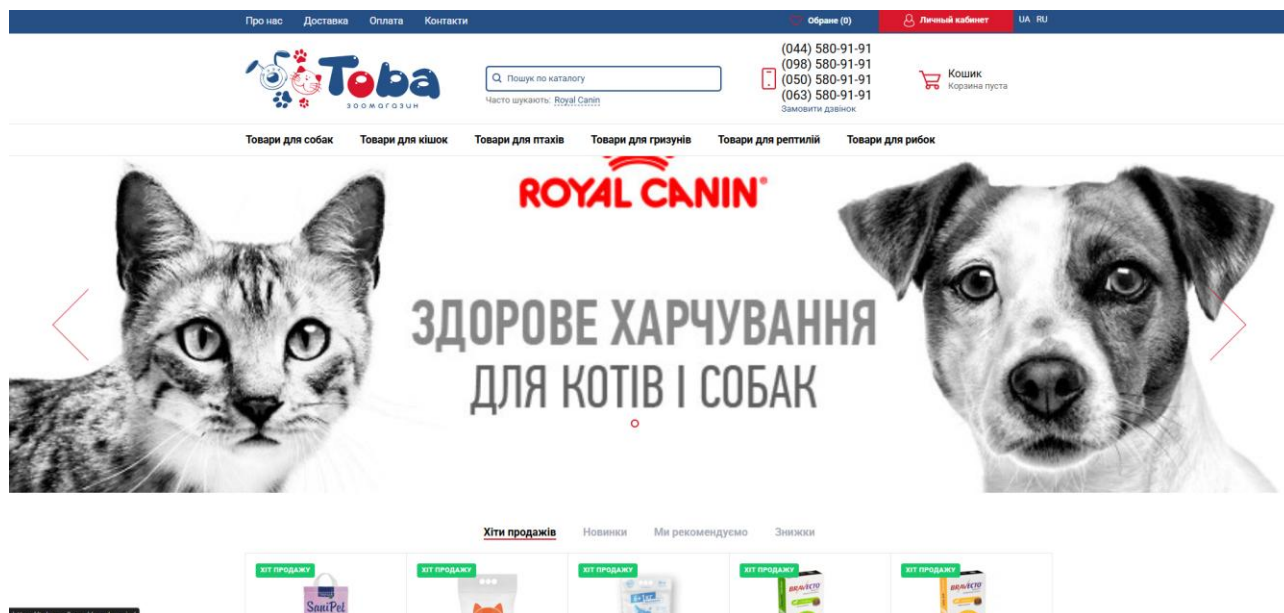


Рис. 1.2 – Вигляд головної сторінки веб-сайту Toba.ua

На перший погляд Toba.ua виглядає функціональніше - є категорії, банери, знижки, відгуки. Проте у більш глибокій взаємодії відчувається перевантаження візуальними блоками, відсутність чіткого структурування контенту та неузгодженість логіки навігації. Сторінки товарів мають занадто багато непотрібної інформації, у той час як важливі елементи (ціна, кнопка «Купити», наявність) розташовані неочевидно. Окрім цього, платформа не має мобільного додатку, що у 2020-х роках є помітною втратою конкурентоспроможності. Хоча оплата та доставка реалізовані, процес не виглядає прозорим - користувач не отримує достатньо інформації про кроки замовлення до його завершення.

На фоні цих аналогів розроблювана платформа вирізняється новим баченням, простотою, легкістю у користуванні та сучасним підходом до базових функцій. Вона не намагається перенасити користувача зайвим, натомість робить ставку на зрозумілий процес купівлі: доступ до товарів, чітке сортування за категоріями, зручне додавання до кошика, просте оформлення замовлення. Усе це - без перевантажених банерів, зайвих реєстрацій і довгих форм. Крім того, особливий акцент зроблено на швидкодії сайту - сторінки відкриваються швидко, а адаптивна верстка дозволяє працювати з платформою з будь-якого пристрою без втрати зручності.

Таблиця Порівняння аналогів

Функція / Сервіс	Zvirshop	Toba.ua	Розроблювана платформа
Актуальність дизайну	—	+/-	+
Зручна мобільна версія	—	—	+
Інтуїтивна структура каталогу	—	+/-	+
Швидкість оформлення замовлення	—	—	+
Якість пошуку та навігації	+/-	+	+
Простота інтерфейсу	—	—	+
Наявність візуальної єдності	—	—	+
Доступ до ключової інформації миттєво	—	+	+
Актуальність товарного наповнення	+/-	+/-	+

Як видно з аналізу, хоча платформи Zvirshop і Toba.ua формально функціонують на ринку зоотоварів, їхнє технічне та візуальне виконання значно поступається сучасним вимогам користувачів. Вони мають обмежену функціональність, слабку адаптивність, перевантажену структуру та відсутність мобільної гнучкості. У той же час, розроблювана система базується на філософії «нічого зайвого» - лише найнеобхідніше, але реалізоване якісно, лаконічно та швидко. Користувач не губиться в категоріях, не чекає на завантаження сторінок і не витрачає час на пошук кнопки «Купити».

Саме завдяки такому підходу платформа має всі шанси стати не просто черговим сайтом з продажу зоотоварів, а дійсно зручним і корисним ресурсом,

що враховує сучасні очікування аудиторії: простоту, швидкість, адаптивність і відсутність бар'єрів при взаємодії.

1.3. Вибір архітектури веб-орієнтованої системи

Вибір оптимальної архітектури - це фундаментальний крок, від якого залежить не лише технічна реалізація проєкту, а й його життєздатність, стійкість до навантажень та сприйняття кінцевим користувачем. Особливо це стосується платформ електронної комерції, де кожна затримка, кожен недолік у роботі інтерфейсу чи функціоналу безпосередньо впливає на довіру, комфорт і бажання користувача повертатися знову. Саме тому архітектура розроблюваного інтернет-зоомагазину була продумана до деталей, з урахуванням як сучасних технічних стандартів, так і звичок користувачів, що формуються в нових умовах цифрової взаємодії.

У рамках реалізації платформи використано концепцію RESTful API, яка дозволяє досягти чіткої розмежованості між серверною та клієнтською частинами застосунку. Це рішення забезпечує стандартизовану взаємодію між різними компонентами системи, спрощує масштабування, дозволяє легко розширювати функціонал у майбутньому та створює гнучку основу для інтеграції з зовнішніми сервісами - зокрема, з платіжними системами, службами доставки та соціальними платформами для авторизації. REST-архітектура є універсальним стандартом сучасної розробки і забезпечує не лише зручність, але й надійність, що особливо важливо в умовах багатокористувацької онлайн-платформи.

Серверна частина реалізована з використанням Laravel - одного з найпотужніших та найпопулярніших PHP-фреймворків, який поєднує простоту, безпеку й масштабованість. Він забезпечує структуровану архітектуру коду, підтримку MVC, інтеграцію засобів аутентифікації, валідації, логування, а також Eloquent ORM - інструмент для взаємодії з базами даних, який дозволяє працювати з інформацією просто, швидко та зрозуміло. Завдяки Laravel розробка стає не лише ефективною, а й контрольованою на кожному етапі. Система

отримує можливість безпечно обробляти дані користувачів, зберігати історію замовлень та ефективно реалізовувати бізнес-логіку магазину.

Особливої уваги заслуговує реалізація авторизації, яка вийшла за рамки стандартного підходу. Окрім класичної форми входу за email та паролем, користувачам доступна авторизація через Google-акаунт, що значно пришвидшує процес доступу до особистого кабінету та знижує бар'єр входу для нових відвідувачів. Така функція не лише підвищує зручність, але й демонструє відповідність актуальним цифровим звичкам користувачів, які прагнуть мінімізувати кількість реєстрацій на нових сайтах.

Клієнтська частина веб-додатку реалізована на базі Nuxt.js - прогресивного фреймворку, який працює поверх Vue.js і дозволяє створювати SEO-оптимізовані, швидкі та адаптивні інтерфейси. Завдяки серверному рендерингу (SSR) забезпечується швидкий рендер сторінок, що не лише покращує взаємодію з користувачем, а й позитивно впливає на індексацію пошуковими системами, що важливо для розвитку проєкту без додаткових витрат на маркетинг. Водночас Nuxt дозволяє ефективно працювати з асинхронними запитами через Axios, забезпечуючи плавну інтеграцію з бекендом і динамічну зміну контенту без перезавантаження сторінки.

Ще однією перевагою платформи є інтеграція з сервісом доставки "Нова Пошта", що надає користувачам не лише стандартну можливість обрати спосіб доставки, але й дозволяє вибрати конкретне відділення або поштомот у будь-якому місті чи навіть невеликому селищі. Така можливість значно підвищує зручність для користувачів, які проживають за межами великих міст. Людина може обрати найближчу точку видачі товару саме там, де їй зручно, не витрачаючи час на додаткові дзвінки чи уточнення адрес. Інтеграція з API Нової Пошти дозволяє автоматично підтягувати список доступних відділень за вибраним населеним пунктом, що економить час і виключає помилки при заповненні форми доставки.

Важливим критерієм при виборі архітектури стало також питання масштабованості та довгострокового розвитку. Комбінація Laravel та Nuxt.js

					ДУ «Житомирська політехніка».25.121.3.000 - ПЗ	Арк.
						16

дозволяє легко реалізовувати нові функції, не переписуючи основну частину коду, що відкриває широкі перспективи для розширення: від додавання нових ролей користувачів до впровадження багатомовності чи аналітики. Такий підхід дає змогу ефективно реагувати на зростання аудиторії, не втрачаючи при цьому стабільності в роботі.

Таким чином, обрана архітектура - це не просто набір інструментів, а продумана екосистема, в якій кожен компонент доповнює інший. Laravel забезпечує надійний фундамент і чітку логіку, Nuxt.js відповідає за зручність та швидкість інтерфейсу, а RESTful API дозволяє легко масштабувати систему та взаємодіяти з іншими сервісами. У поєднанні з такими рішеннями, як авторизація через Google та інтеграція з Новою Поштою, система не лише відповідає базовим технічним вимогам, а й дійсно орієнтована на потреби живої, реальної аудиторії - з її повсякденними запитами, звичками та очікуваннями.

1.4. Обґрунтування вибору інструментальних засобів та вимоги до апаратного забезпечення

Для розробки інтернет-зоомагазину обрано перевірений і водночас сучасний набір інструментальних засобів, який забезпечує не лише технічну стійкість та надійність системи, але й високий рівень гнучкості, що особливо важливо в умовах постійно зростаючих вимог з боку користувачів. Платформа повинна бути не лише функціональною, а й зручною, швидкою, адаптивною, а головне - готовою до подальшого розширення. Саме тому вибір технологій було зроблено на користь перевірених рішень, які мають активну спільноту, постійно оновлюються та підтримують сучасні стандарти безпеки і розробки.

Серверна частина реалізована за допомогою PHP-фреймворку Laravel, який поєднує в собі високу продуктивність та елегантність синтаксису. Однією з головних причин вибору Laravel стала його модульна архітектура, що дозволяє зручно структурувати код та масштабувати систему при розширенні функціональності. Крім того, Laravel має вбудовану ORM-бібліотеку Eloquent, яка забезпечує гнучку взаємодію з базою даних, дозволяє швидко формувати

запити, працювати з міграціями та будувати логічні зв'язки між сутностями. Це важливо для побудови, наприклад, структури замовлень, користувачів і товарів.

Також Laravel надає зручні засоби для реалізації авторизації, включаючи вбудовану підтримку авторизації через Google. Це дає змогу користувачам проходити швидку реєстрацію або вхід у систему, що особливо зручно в умовах мобільного використання або коли немає бажання вводити паролі вручну. Такий підхід підвищує конверсію та спрощує взаємодію з сайтом.

Окрім класичних функцій, Laravel також дозволяє легко реалізовувати інтеграцію з зовнішніми API, що було використано для підключення до сервісу доставки "Нова Пошта". Це дає змогу користувачу не просто обрати доставку, а й вибрати конкретне відділення або поштомат у потрібному місті чи селищі. Інформація про населені пункти та точки видачі підтягується динамічно, залежно від введеного значення, що значно знижує кількість помилок і покращує зручність процесу оформлення замовлення.

Клієнтська частина розробляється за допомогою Nuxt.js - прогресивного фреймворку на базі Vue.js. Nuxt дозволяє реалізувати серверний рендеринг (SSR), що позитивно впливає на швидкість завантаження сторінок, а також значно покращує SEO-оптимізацію. Такий підхід дозволяє залучати органічний трафік з пошукових систем без додаткових витрат на маркетинг. Компонентна структура Nuxt сприяє повторному використанню частин інтерфейсу, що пришвидшує розробку і дозволяє оперативно впроваджувати нові функції.

Для зберігання даних використовується MySQL - надійна реляційна СУБД, яка чудово підходить для обробки транзакцій, пошуку, сортування та фільтрації товарів за категоріями, цінами чи іншими критеріями. У взаємодії з Eloquent ORM, база даних працює швидко й ефективно навіть при великому обсязі записів.

З метою спрощення розгортання проєкту на сервері та підтримки стабільного середовища, використовується Docker - технологія контейнеризації, яка дозволяє запускати серверну, клієнтську частину, базу даних, кеш-систему та інші сервіси в ізольованих середовищах. Це усуває проблему «працює в мене,

					ДУ «Житомирська політехніка».25.121.3.000 - ПЗ	Арк.
						18

але не працює в іншого» та гарантує однакову поведінку додатку у всіх середовищах. Для керування всіма контейнерами використовується docker-compose, що дозволяє запускати всі частини проєкту однією командою.

Проєкт також активно використовує Git для контролю версій. Репозиторій зберігається на GitHub, що забезпечує зручну співпрацю над кодом, ведення історії змін, контроль над релізами та можливість відновлення після помилок. Організація роботи відбувається за методологією GitFlow, що дозволяє паралельно працювати над новими функціями, тестуванням і гарячими виправленнями.

Для повноцінного функціонування платформи на віддаленому сервері необхідна базова, але стабільна інфраструктура. Рекомендовані системні вимоги включають:

- оперативну пам'ять: не менше 8 ГБ для коректної роботи Laravel, бази даних та контейнерів одночасно.
- дисковий простір: від 10 ГБ для зберігання коду, зображень, логів та резервних копій.
- процесор: багатоядерний CPU з тактовою частотою від 1.5 GHz для обробки запитів і компіляції фронтенду.
- швидке й стабільне інтернет-з'єднання для взаємодії з клієнтами, платіжними системами та службою доставки.
- операційна система: Linux (рекомендовано Ubuntu 22.04 LTS або вище) як стабільне серверне середовище з активною спільнотою підтримки.

Таким чином, обраний набір інструментів - Laravel, Nuxt.js, MySQL, Docker та Git - у поєднанні з інтеграцією зовнішніх сервісів, таких як Google Auth та Нова Пошта, створює надійну, зручну та масштабовану архітектуру. Вона не лише задовольняє поточні вимоги проєкту, а й дозволяє впевнено розширювати систему в майбутньому, зберігаючи при цьому високу швидкість, безпеку та гнучкість у розробці.

Висновки до першого розділу

У першому підрозділі було окреслено основні елементи, що є ключовими для розробки інтернет зоомагазину, а також визначено основні технічні вимоги до системи. Це дозволило сформулювати базове розуміння необхідних компонентів та ресурсів для реалізації проєкту.

У другому підрозділі проведено аналіз аналогічних продуктів на ринку, де було порівняно основний функціонал, переваги та недоліки розглянутих сервісів. Результати цього аналізу дозволили визначити потенційні напрямки вдосконалення власного рішення, а також окреслити можливості впровадження інноваційних функцій для підвищення конкурентоспроможності.

У третьому підрозділі здійснено вибір архітектури веб-системи. Обрано оптимальний варіант, який забезпечує високу продуктивність, масштабованість та гнучкість подальшого розвитку проєкту. В результаті серверна частина реалізована на базі Laravel, а клієнтська – з використанням Nuxt.js, що дозволяє забезпечити ефективну взаємодію між компонентами через RESTful API та сприяє зручному управлінню контентом і даними.

У четвертому підрозділі визначено інструментальні засоби та вимоги до апаратного забезпечення. Для реалізації клієнтської частини обрано Nuxt.js, для серверної – Laravel, а для роботи з базами даних обрано MySQL, яка розміщується на віддаленому сервері з ОС Ubuntu. Крім того, для оптимізації роботи застосовано Redis для кешування, а процес розгортання та підтримки забезпечується через контейнеризацію за допомогою Docker з використанням docker-compose.

Ці рішення дозволяють створити стабільну, безпечну та масштабовану систему, що відповідає сучасним вимогам ринку електронної комерції та забезпечує ефективну взаємодію з користувачами навіть при високих навантаженнях.

РОЗДІЛ 2. ПРОЄКТУВАННЯ ОНЛАЙН-ПЛАТФОРМИ ІНТЕРНЕТ-ЗООМАГАЗИНУ

У цьому розділі розглянуто етапи проєктування інтернет-зоомагазину, включаючи визначення варіантів використання, об'єктно-орієнтованої структури, розробку бази даних, алгоритмів і реалізацію функціоналу. Мета - створення зручної, безпечної та масштабованої платформи для продажу товарів для домашніх тварин.

2.1. Визначення варіантів використання та об'єктно-орієнтованої структури системи

Сучасна платформа інтернет-зоомагазину покликана забезпечити зручний, швидкий та інтуїтивно зрозумілий процес взаємодії користувача з товаром, починаючи з моменту першого відвідування сайту і завершуючи успішною покупкою з доставкою. Саме тому вже на етапі проєктування необхідно чітко визначити основні сценарії використання, які охоплюють типові дії користувачів: перегляд товарів, реєстрація, керування замовленнями, взаємодія з кошиком і можливість залишити відгук.

У системі передбачено декілька ролей: неавторизовані користувачі, зареєстровані користувачі та адміністратори. Кожна з них має різний рівень доступу до функціоналу, що дозволяє забезпечити баланс між відкритістю платформи та контролем за її наповненням. Наприклад, неавторизовані користувачі можуть лише переглядати асортимент товарів та скористатись пошуком, тоді як зареєстровані отримують повний доступ до особистого кабінету, мають можливість залишити відгук, оформити замовлення або змінити свої дані. Адміністратори, в свою чергу, виконують технічну та змістову модерацію: додають нові товари, редагують інформацію, переглядають замовлення та аналізують активність користувачів.

Реєстрація або авторизація є відправною точкою для персоналізованої взаємодії користувача з платформою. Користувач може пройти реєстрацію за допомогою стандартної форми з email та паролем або авторизуватись через

Google - для зручності та швидкого входу. Після цього користувач отримує доступ до всіх сервісів: особистого кабінету, історії замовлень, управління даними, перегляду відгуків та взаємодії з кошиком.

Таблиця 2.1

Таблиця прецеденту «Реєстрація / Авторизація»

Назва варіанту використання	Реєстрація / Авторизація
Основні актори	Користувач
Короткий опис	Користувач реєструється або входить у систему через email або Google для подальших дій на сайті

Після входу до системи користувач отримує можливість керувати особистим кабінетом. Тут зосереджені всі його замовлення, контактні дані, завантажений аватар, а також можливість змінити номер телефону, email чи пароль. Це дозволяє забезпечити актуальність даних, персоналізувати інтерфейс і швидше оформлювати повторні покупки.

Таблиця 2.2

Таблиця прецеденту «Керування профілем»

Назва варіанту використання	Керування профілем
Основні актори	Користувач
Короткий опис	Користувач редагує свої контактні дані, аватар, переглядає історію замовлень та активність

Інтернет-магазин побудований навколо концепції «товару» - кожен користувач повинен мати можливість зручно переглядати асортимент, шукати товари за назвою, фільтрувати за категоріями, ціновим діапазоном або наявністю знижок. Для цього реалізовано механізм фільтрації, сортування та переходу до детальної сторінки товару.

Таблиця прецеденту «Перегляд та пошук товарів»

Назва варіанту використання	Перегляд та пошук товарів
Основні актори	Гість, Користувач
Короткий опис	Користувач переглядає товари за категоріями, здійснює пошук, фільтрує по назві та ціні

Формування та підтвердження замовлення - ключовий процес платформи. Користувач обирає потрібні товари, додає їх до кошика, вводить свої контактні дані та обирає спосіб доставки. Особливістю нашої платформи є гнучка інтеграція з сервісом «Нова Пошта», що дозволяє обрати точку доставки - як відділення, так і поштомот - у будь-якому населеному пункті.

Таблиця прецеденту «Оформлення замовлення»

Назва варіанту використання	Оформлення замовлення
Основні актори	Користувач
Короткий опис	Користувач додає товар до кошика, обирає відділення Нової Пошти та здійснює оплату

Функція залишення коментаря є важливим елементом зворотного зв'язку, що дозволяє створити атмосферу довіри між покупцями та платформою. Завдяки цій можливості користувач може детально описати свої враження від придбаного товару, звернути увагу інших на його якість, зручність у використанні, відповідність до опису або навіть на швидкість доставки. Така інформація нерідко стає вирішальним чинником для інших потенційних покупців. У системі впроваджена інтуїтивно зрозуміла та доступна форма для додавання текстових коментарів, що дозволяє з легкістю поділитися власною думкою. Водночас усі коментарі проходять обов'язкову модерацію, що гарантує відсутність спаму,

образливих висловлювань або неправдивої інформації, забезпечуючи тим самим безпечне та чесне середовище для взаємодії..

Таблиця 2.5

Таблиця прецеденту «Залишення коментаря»

Назва варіанту використання	Залишення коментаря
Основні актори	Користувач
Короткий опис	Користувач пише текстовий коментар до товару, який публікується після модерації

Адміністратор — це користувач із розширеними правами, який має повний доступ до керування системою. Він створює та оновлює товари, модерує відгуки, переглядає список замовлень, керує категоріями й фільтрами. Саме адміністратор забезпечує підтримку стабільності, безпеки й актуальності контенту.

Таблиця 2.6

Таблиця прецеденту «Адміністративне управління»

Назва варіанту використання	Адміністративне управління
Основні актори	Адміністратор
Короткий опис	Адміністратор керує товарами, переглядає замовлення, додає категорії та модерує відгуки

У процесі розробки інтернет-зоомагазину було обрано об'єктно-орієнтований підхід до моделювання всієї системи. Це рішення дозволило створити структуру, в якій кожен окремий функціональний блок представлений як незалежний логічний об'єкт зі своїми властивостями та методами. Завдяки цьому система стає більш гнучкою, зручною для розширення та підтримки, а також дозволяє повторно використовувати код при розробці нових модулів. Уся архітектура побудована навколо сутностей, які безпосередньо відповідають реальним процесам, що відбуваються в межах онлайн-покупок: реєстрація, пошук, замовлення, перегляд, доставка, адміністрування тощо.

Наприклад, центральну роль у системі відіграє клас користувача. Саме він відповідає за базові атрибути — такі як унікальний ідентифікатор, електронна пошта, пароль, ім'я та роль у системі. Завдяки реалізованим методам користувач має змогу пройти реєстрацію, авторизуватись, редагувати профіль або переглядати історію своїх замовлень. Поруч із ним функціонує клас товару, який включає всі основні характеристики одиниці продукції — назву, опис, ціну, зображення, кількість у наявності та категорію. Товари можна фільтрувати за категорією, додавати до кошика чи шукати за заданими параметрами.

Також важливим елементом є структура категорій — вона дозволяє користувачеві орієнтуватися у великому асортименті зоотоварів, структуруючи їх за групами. Кожна категорія може містити підкатегорії та мати свої зв'язки з відповідними товарами, що особливо зручно при навігації. Коли покупець формує замовлення, система створює об'єкт класу замовлення, до якого додаються окремі позиції — товари з кошика. Ці позиції відображаються через дочірній об'єкт, який фіксує назву товару, його кількість та вартість. Після цього формується загальна сума й статус замовлення.

Особливу увагу приділено також функціоналу відгуків. Коментарі дозволяють іншим покупцям ознайомитися з досвідом попередніх клієнтів, а також формують загальну репутацію товару. Всі відгуки модеруються, аби запобігти публікації спаму чи неприйняттого контенту. У свою чергу, адміністратор має доступ до всіх аспектів керування платформою, включаючи

модерацію, оновлення товарів, перегляд замовлень і контроль структури категорій.

На основі таких ролей і дій сформульовано як функціональні, так і нефункціональні вимоги до системи. У функціональному контексті передбачено, що користувач повинен мати змогу зареєструватися або увійти до системи — як традиційно через електронну пошту, так і через обліковий запис Google. Далі він має переглядати каталог, користуватися фільтрами, додавати товари до кошика, обирати спосіб доставки через інтеграцію з сервісом «Нова Пошта», сплачувати замовлення та, за бажанням, залишати коментарі. Адміністратор при цьому виконує ключову роль у підтриманні цілісності платформи — стежить за наявністю актуального контенту, обробляє замовлення, відповідає за модерування відгуків і змінює статуси доставки.

Щодо нефункціональних аспектів, то система повинна працювати швидко — сервер має обробляти запити не довше ніж за 500 мілісекунд навіть при великому навантаженні. Безпека — ще один критично важливий момент. Усі дані передаються через зашифроване з'єднання HTTPS, паролі зберігаються в хешованому вигляді за допомогою bcrypt, а вхідні запити перевіряються на наявність загроз (SQL-ін'єкції, XSS, CSRF). Також система повинна масштабуватись, аби витримати збільшення кількості користувачів без втрати стабільності, бути адаптованою для перегляду з мобільних пристроїв і мати резервне копіювання, яке гарантує збереження всіх даних навіть у разі аварійних ситуацій.

Візуально вся логіка та взаємозв'язки між сутностями можуть бути представлені на діаграмі класів, що демонструє, як між собою пов'язані об'єкти системи: користувачі, замовлення, товари, категорії та коментарі. Такий підхід до побудови архітектури є надзвичайно важливим, адже дозволяє системі зростати разом із потребами користувачів і не втрачати стабільність у процесі масштабування.

2.2. Розробка бази даних систем

Для забезпечення надійного зберігання, швидкої обробки даних та підтримки різноманітних зв'язків між об'єктами, система інтернет-зоомагазину використовує реляційну базу даних MySQL. Вона є однією з найпоширеніших систем керування базами даних і забезпечує ефективну взаємодію між компонентами бекенду, реалізованого на Laravel, та клієнтською частиною, що працює на Nuxt.js.

Структура бази даних була ретельно продумана і спроектована з урахуванням усіх аспектів діяльності магазину. Особливу увагу приділено гнучкості та розширюваності, що дозволяє безперешкодно масштабувати функціонал у майбутньому без необхідності повної перебудови основи системи. Нижче описані ключові таблиці, на яких базується робота ресурсу.

Однією з найважливіших складових бази даних є таблиця users. Саме вона зберігає інформацію про клієнтів, які можуть переглядати товари, оформлювати замовлення, проходити реєстрацію та авторизацію — як класично за email, так і через Google-акаунт. Окрім основних ідентифікаційних полів, тут також зберігається аватар користувача та телефон, а для безпеки всі паролі зберігаються в хешованому вигляді.

Таблиця 2.7

Таблиця Users

Поле	Тип	Опис
id	bigint(20) UNSIGNED NOT NULL	Ідентифікатор користувача
name	varchar(255) COLLATE utf8mb4_unicode_ci NOT NULL	Ім'я користувача
surname	varchar(255) COLLATE utf8mb4_unicode_ci DEFAULT NULL	Прізвище користувача

Продовження таблиці 2.7

email	varchar(255) COLLATE utf8mb4_unicode_ci NOT NULL	Email користувача
google_id	varchar(255) COLLATE utf8mb4_unicode_ci DEFAULT NULL	Google ідентифікатор користувача
email_verified_at	timestamp NULL DEFAULT NULL	email_verified_at
password	varchar(255) COLLATE utf8mb4_unicode_ci DEFAULT NULL	password
remember_token	varchar(100) COLLATE utf8mb4_unicode_ci DEFAULT NULL	remember_token
created_at	timestamp NULL DEFAULT NULL	created_at
updated_at	timestamp NULL DEFAULT NULL	updated_at
phone	varchar(255) COLLATE utf8mb4_unicode_ci DEFAULT NULL	phone
avatar	varchar(255) COLLATE utf8mb4_unicode_ci DEFAULT NULL	avatar

Таблиця products є центральною складовою бази даних будь-якого електронного магазину, адже саме вона містить найважливіше — перелік усіх товарів, які потенційно можуть бути придбані користувачами. Ця таблиця виступає своєрідним «вітринним каталогом» системи, де зберігаються всі ключові атрибути, що дозволяють не лише коректно відображати товар у користувацькому інтерфейсі, а й оперативно реагувати на зміну попиту, залишків чи цін.

Таблиця 2.8

Таблиця Products

Поле	Тип	Опис
id	bigint(20) UNSIGNED NOT NULL	Ідентифікатор товару

barcode	varchar(255) COLLATE utf8mb4_unicode_ci DEFAULT NULL COMMENT 'Штрихкод'	Код товару
name	varchar(255) COLLATE utf8mb4_unicode_ci NOT NULL COMMENT 'Назва товару'	Назва товару
description	text COLLATE utf8mb4_unicode_ci COMMENT 'Опис товару'	Опис товару
price	bigint(20) DEFAULT '0' COMMENT 'Ціна товару'	Ціна товару
discount	int(10) UNSIGNED NOT NULL DEFAULT '0' COMMENT 'Знижка у відсотках'	Знижка товару
status_id	bigint(20) UNSIGNED NOT NULL	Статус товару
created_at	timestamp NULL DEFAULT NULL	Дата створення товару
updated_at	timestamp NULL DEFAULT NULL	Дата оновлення товару

Коли користувач оформлює покупку, усі дані автоматично зберігаються в таблиці orders, що виступає центральним джерелом інформації про всі замовлення на платформі. У цій таблиці фіксуються не лише основні атрибути замовлення, як-от контактні дані покупця чи загальна сума, а й деталізовано вказується вибраний спосіб доставки — наприклад, через мережу відділень або поштоматів Нової Пошти. Крім того, в таблиці зберігається актуальний статус оплати, що дозволяє адміністраторам легко та оперативно відслідковувати успішність транзакцій, обробляти нові замовлення або виявляти потенційні

проблеми. Такий підхід значно підвищує керованість процесами та забезпечує високий рівень обслуговування клієнтів.

Таблиця 2.9

Таблиця orders

Поле	Тип	Опис
id	bigint(20) UNSIGNED NOT NULL	
order_number	varchar(255) COLLATE utf8mb4_unicode_ci NOT NULL	
email	varchar(255) COLLATE utf8mb4_unicode_ci NOT NULL	
name	varchar(255) COLLATE utf8mb4_unicond NOT NULL	
surname	varchar(255) COLLATE utf8mb4_unicode_ci NOT NULL	
phone	varchar(255) COLLATE utf8mb4_unicode_ci NOT NULL	
city_ref	varchar(255) COLLATE utf8mb4_unicode_ci NOT NULL	
warehouse_ref	varchar(255) COLLATE utf8mb4_unicode_ci NOT NULL	
delivery_method_id	bigint(20) UNSIGNED NOT NULL	
total_amount	bigint(20) NOT NULL DEFAULT '0'	
payment_status	varchar(255) COLLATE utf8mb4_unicode_ci NOT NULL DEFAULT 'pending'	
transaction_id	varchar(255) COLLATE utf8mb4_unicode_ci DEFAULT NULL	transaction_id
created_at	timestamp NULL DEFAULT NULL	created_at

2.3. Проектування та реалізація алгоритмів роботи платформи

Для того щоб інтернет-зоомагазин працював плавно, логічно й при цьому залишався інтуїтивно зрозумілим для користувача, у процесі розробки було реалізовано кілька ключових механізмів, які забезпечують коректну взаємодію між елементами системи. Ці механізми дозволяють користувачам не тільки зручно шукати товари, оформлювати замовлення, але й дають адміністрації змогу керувати платформою, а також регулювати процеси, пов'язані з відгуками. Весь внутрішній функціонал реалізовано у вигляді окремих логічних блоків, які відповідають за певну операцію - фільтрацію, створення замовлення, перевірку залишків на складі, а також оновлення статусів чи взаємодію з користувачами.

Один із найважливіших процесів, що реалізований у системі, - це механізм фільтрації товарів. Коли користувач заходить на сайт і починає шукати потрібний товар, йому хочеться якнайшвидше звузити вибір і побачити тільки ті позиції, які його дійсно цікавлять. Щоб це забезпечити, система дозволяє відбирати товари за цілою низкою параметрів - наприклад, за категорією, вартістю, брендом або навіть типом домашньої тварини. Користувач вибирає необхідні опції, і система формує спеціальний запит до бази даних, який повертає саме ті результати, які відповідають зазначеним умовам. Усе це відбувається у фоновому режимі, миттєво і без оновлення сторінки, що забезпечує приємний досвід використання. Такий підхід дозволяє значно покращити ефективність пошуку, а також зменшує навантаження на інтерфейс.

Ще одним важливим процесом, який реалізовано у межах роботи зоомагазину, є створення замовлення. Цей процес хоч і виглядає зовні доволі простим для звичайного користувача — додати товар у кошик і натиснути «Оформити» — але за ним стоїть чітко прописана логіка, яка відповідає за фіксацію замовлення в системі. Коли покупець вирішує замовити певні товари, система формує запис у базі даних, в якому зберігаються всі основні дані: хто зробив замовлення, які саме товари було обрано, їх кількість, загальна сума та час створення замовлення. Оскільки платформа ще не передбачає складного механізму перевірки залишків або динамічного резервування товару, весь процес

зосереджений на тому, щоб максимально просто й швидко зберегти ключову інформацію для подальшої обробки адміністратором.

Після створення замовлення йому присвоюється стандартний статус — наприклад, «нове» або «очікує підтвердження». Адміністратор або менеджер магазину надалі бачить усі активні замовлення, переглядає деталі й приймає рішення: підтвердити, зателефонувати клієнту, уточнити щось або оновити статус вручну. Таким чином, основна логіка зводиться до збереження інформації та її подальшої обробки — усе інше наразі виконується вручну, що є природним на етапі старту невеликого інтернет-магазину. Цей підхід дозволяє уникнути помилок, які можуть виникнути при автоматизації на початкових етапах, та залишає простір для подальшого вдосконалення. У майбутньому цю частину можна легко розширити — додати перевірку залишків на складі, сповіщення для користувача або навіть інтеграцію з CRM-системами для автоматичної обробки.

Не менш важливою частиною є перевірка відгуків. Щоб забезпечити якість і чесність на платформі, кожен відгук, який залишає користувач, проходить перевірку модератором. Це зроблено для того, щоб уникнути спаму, неправдивої інформації або нецензурної лексики. Як тільки модератор бачить новий відгук, він може прийняти рішення — схвалити його, якщо все відповідає правилам, або відхилити, якщо в ньому є порушення (рис. 2.1)

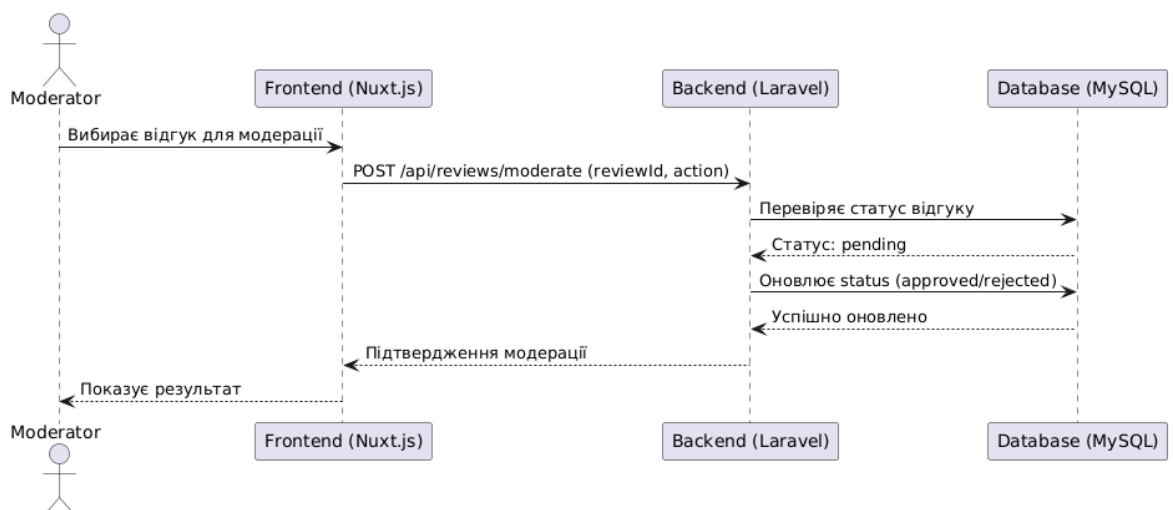


Рис. 2.1 – Діаграма послідовності

Усі ці процеси — фільтрація, обробка замовлень, модерація — працюють в рамках єдиної архітектури системи, де кожен компонент має свою чітку роль. На рисунку нижче зображено один із таких процесів — модерацію відгуку. На діаграмі показано, як система реагує на дії модератора: від моменту, коли він відкриває список нових відгуків, до моменту, коли користувач отримує лист про статус свого коментаря. Така наочна схема дозволяє краще зрозуміти, як працює внутрішня логіка застосунку.

2.4. Реалізація функціоналу платформи

Розробка інтернет-зоомагазину передбачала створення платформи, яка б поєднувала зручний інтерфейс для покупців, гнучкий бекенд для обробки замовлень, а також інтеграцію з актуальними сервісами — такими як оплата, доставка, авторизація тощо. Щоб це реалізувати, було обрано архітектуру клієнт-серверного типу, де фронтенд побудовано на базі Nuxt.js — сучасного інструменту, який дозволяє створювати швидкі, динамічні й SEO-дружні сайти. Бекендова частина реалізована на Laravel — одному з найпопулярніших РНР-фреймворків, який чудово справляється із завданнями обробки запитів, валідації, збереження даних і взаємодії з базою.

Однією з перших реалізованих функцій стала авторизація, адже без неї будь-яка взаємодія з платформою залишалася б поверхневою. Для цього була створена система авторизації через email, а також з можливістю входу за допомогою облікового запису Google. Після введення електронної пошти та пароля, користувач проходить верифікацію, і якщо дані збігаються — система генерує спеціальний токен, який дозволяє зберегти авторизований стан користувача при переходах по сайту. Цей токен передається клієнтській частині й використовується для ідентифікації на подальших етапах, таких як оформлення замовлення чи збереження профілю.

```
public function handleGoogleAuth(Request $request):
\Illuminate\Http\JsonResponse
{
    $payload = $this->verifyGoogleToken($request-
>header('GISToken'));
    $user = User::where('google_id', $payload['sub'])->first();
```

```

    if ($user) {
        return $this->makeTokens($user);
    }

    $existingUser = User::where('email', $payload['email'])-
>first();
    if ($existingUser) {
        throw new HttpException(422, 'Email already exists');
    }

    $user = User::create([
        'google_id' => $payload['sub'],
        'email' => $payload['email'],
        'name' => $payload['given_name'],
        'surname' => $payload['family_name'] ?? null,
    ]);

    $user->roles()->attach(2);
    return $this->makeTokens($user);
}

```

Ще одним фундаментальним компонентом стала реалізація кошика покупок — функції, яка дозволяє зберігати вибрані товари, додавати або видаляти позиції, а також контролювати їх кількість. Кошик працює як для неавторизованих користувачів (зберігаючись у пам'яті браузера), так і для зареєстрованих, де його можна синхронізувати з сервером. Усе це реалізовано через Pinia — централізоване сховище стану у Nuxt.js, що дозволяє гарантувати збереження даних навіть при оновленні сторінки. Коли користувач додає товар до кошика, система автоматично перевіряє, чи він там уже є, і збільшує кількість, або ж додає нову позицію — усе це без перезавантаження й з плавною взаємодією.

```

// store/modals/favorite.js
import { defineStore } from 'pinia';
export const useProductBasketStore = defineStore('modals.basket',
{
    state: () => ({
        isOpen: false,
        productBasket: [],
    }),
    actions: {
        toggleModal() {
            this.isOpen = !this.isOpen;
        },
        getData() {
            this.productBasket =
JSON.parse(window.localStorage.getItem('productBasket'));

```

```

    },
    setData(products) {
        window.localStorage.setItem('productBasket',
JSON.stringify(products.value));
    }
}
});

```

Не менш важливим кроком стала інтеграція платіжної системи LiqPay, яка дозволяє провести оплату безпосередньо після підтвердження замовлення. Це значно підвищує зручність користувача, оскільки немає потреби контактувати з менеджером або вносити оплату вручну. Після того як користувач натискає кнопку «Оплатити», система формує спеціальну форму оплати, яка вже містить необхідні поля: суму, опис замовлення, ідентифікатор — усе це передається до LiqPay через захищене API. Користувач бачить інтерфейс оплати й може вибрати зручний спосіб: картка, Apple Pay, Google Pay тощо. Після цього результат повертається назад у систему, і замовлення автоматично змінює свій статус.

```

class PaymentController extends Controller
{
    protected $liqpay;

    public function __construct()
    {
        $this->liqpay = new LiqPay(
            config('services.liqpay.public_key'),
            config('services.liqpay.private_key')
        );
    }

    public function checkout($orderId)
    {
        $order = Orders::findOrFail($orderId);

        $params = [
            'version' => 3,
            'action' => 'pay',
            'amount' => $order->totalAmountInCurrency, // Assume
total_amount is the order total
            'currency' => config('services.liqpay.currency'),
            'description' => "Payment for Order #{$order->id}",
            'order_id' => $order->id,
            'language' => config('services.liqpay.language'),
            'result_url' => env('FRONTEND_URL') .
'/checkout/success', // Frontend success page
            'server_url' => route('api.liqpay.callback'), //

```

```

Callback URL
];

$form = $this->liqpay->cnb_form_raw($params);
return response()->json([
    'action' => 'https://www.liqpay.ua/api/3/checkout',
    'data' => $form['data'],
    'signature' => $form['signature'],
]);
}

public function callback(Request $request)
{
    $data = $request->input('data');
    $signature = $request->input('signature');

    // Verify signature
    $expectedSignature = base64_encode(sha1(
        config('services.liqpay.private_key') . $data .
        config('services.liqpay.private_key'),
        true
    ));

    if ($signature !== $expectedSignature) {
        Log::error('Invalid LiqPay callback signature',
            ['data' => $data]);
        return response()->json(['error' => 'Invalid
signature'], 400);
    }

    // Decode callback data
    $callbackData = $this->liqpay->decode_params($data);

    $orderId = $callbackData['order_id'];
    $status = $callbackData['status'];

    $order = Orders::findOrFail($orderId);

    if ($status === 'success') {
        $order->update([
            'payment_status' => 'paid',
            'transaction_id' =>
            $callbackData['transaction_id'] ?? null,
        ]);
    } elseif ($status === 'error' || $status === 'failure') {
        $order->update(['payment_status' => 'failed']);
    }

    Log::info('LiqPay callback processed', ['order_id' =>
        $orderId, 'status' => $status]);

    return response()->json(['status' => 'ok']);
}

```

```

    }
}

```

Також у рамках адміністративної частини реалізовано модерацію відгуків, яка дозволяє уникати публікації некоректного або образливого контенту. Коли користувач залишає відгук до товару, він спершу потрапляє у статус «на розгляді». Модератор, увійшовши в адмінпанель, бачить список усіх нових відгуків і може або схвалити, або відхилити кожен окремо. Після затвердження відгук стає видимим для інших відвідувачів, а в разі відмови — зберігається в базі, але не відображається на сайті. Це забезпечує не лише чистоту контенту, а й підвищує довіру до платформи.

```

/**
 * Змінити статус коментаря на один із allowed | disallowed |
 pending.
 *
 * @param Request $request
 * @param int      $commentId
 * @return array
 * @throws HttpException
 */
public function changeStatus(Request $request, int $commentId):
array
{
    $data = $request->validate([
        'status' =>
'required|string|in:pending,approved,disallowed',
    ]);

    /** @var Comment $comment */
    $comment = Comment::find($commentId);
    if (!$comment) {
        throw new HttpException(404, 'Comment not found');
    }

    $comment->status = $data['status'];
    $comment->save();

    return [
        'success' => true,
        'comment' => [
            'id'          => $comment->id,
            'status'      => $comment->status,
            'updated_at' => $comment->updated_at-
>toDateTimeString(),
        ],
    ];
}

```

Також для зручності архітектурного аналізу проєкту було побудовано діаграму компонентів, яка демонструє зв'язки між основними модулями: клієнтська частина (Nuxt), серверна логіка (Laravel), база даних, сервіси оплати, доставки та email-повідомлень. Завдяки такій структурі кожен компонент залишається ізольованим і самостійним, що дозволяє легко вносити зміни, оновлювати або масштабувати систему в майбутньому.

Висновки до другого розділу

У другому розділі було закладено основи архітектури інтернет-зоомагазину, що охоплюють як логіку користувацької взаємодії, так і технічну реалізацію основних компонентів системи. Розпочато з аналізу можливих сценаріїв використання платформи, де особливу увагу приділено ролям користувачів — зокрема, введено роль модератора, яка відіграє важливу роль у підтриманні порядку, перевірці контенту та модерації відгуків. Такий підхід дозволяє створити контрольоване, безпечне середовище для взаємодії покупців і адміністрації.

На основі визначених функціональних вимог було розроблено об'єктно-орієнтовану модель, що охоплює основні сутності, як-от користувачі, товари, замовлення, категорії, відгуки тощо. Було спроектовано реляційну базу даних, що дотримується принципів нормалізації та дозволяє зберігати всю ключову інформацію в структурованому вигляді.

Окрему увагу приділено реалізації основних алгоритмів, зокрема фільтрації товарів, створення замовлень, модерації відгуків і формування рекомендацій. Усі ці процеси описані не лише як логічна послідовність дій, але й реалізовані на практиці через програмний код. Таким чином, забезпечується як функціональність системи, так і її стабільність у роботі з великими обсягами даних.

Функціональні модулі, включаючи авторизацію, обробку кошика, оплату через LiqPay та взаємодію з RESTful API, реалізовано за допомогою Laravel (на стороні сервера) та Nuxt.js (на стороні клієнта). Застосування сучасного стеку дало змогу досягти продуктивної, масштабованої та легко підтримуваної

архітектури. У межах цього етапу система вже має готову базову логіку та структуру, а наступним кроком стане реалізація повного інтерфейсу користувача і тестування функціоналу в реальних умовах.

					ДУ «Житомирська політехніка».25.121.3.000 - ПЗ	Арк.
						39

РОЗДІЛ 3. ОПИС РОЗРОБЛЕНОЇ ІНТЕРНЕТ-ПЛАТФОРМИ ЗООМАГАЗИНУ

3.1. Порядок встановлення та налаштування параметрів платформи

Інтерфейс користувача є одним із ключових елементів будь-якого сучасного вебзастосунку, оскільки саме через нього відбувається взаємодія кінцевого користувача з усією функціональністю системи. Для інтернет-зоомагазину, який орієнтований на широку аудиторію, включаючи як досвідчених онлайн-покупців, так і тих, хто вперше оформлює замовлення через інтернет, простота, доступність та інтуїтивність інтерфейсу відіграють критично важливу роль. Було розроблено інтерфейс, що відповідає сучасним стандартам UX/UI-дизайну, забезпечуючи максимальну зручність для кожного відвідувача сайту.

Початковою точкою взаємодії користувача з платформою є головна сторінка сайту. Тут представлений короткий огляд товарного асортименту із категоріями, акційними позиціями та найпопулярнішими товарами. Особливу увагу було приділено візуальній складовій — великі зображення товарів, зручна навігаційна панель, логічна структура сторінки — усе це сприяє приємному досвіду користування.

Гість, тобто неавторизований користувач, має можливість без обмежень переглядати товари, шукати продукцію за допомогою фільтрів (наприклад, за категорією, брендом, типом тварини тощо), а також додавати товари до кошика. Це дозволяє навіть новачкам у сфері онлайн-шопінгу оцінити можливості системи ще до створення облікового запису.

Сторінка реєстрації є мінімалістичною, з мінімально необхідними полями: електронна пошта, пароль, підтвердження пароля та ім'я користувача. Реєстрація можлива як класичним способом, так і через інтеграцію з Google-акаунтом. Це дозволяє значно пришвидшити процес входу та покращити користувацький досвід.

Після входу до системи інтерфейс користувача зазнає певних змін: з'являється меню профілю, стає доступним перегляд історії замовлень, можливість редагування особистих даних та управління своїми відгуками. Користувач також може бачити статус своїх замовлень у реальному часі — від «Очікує обробки» до «Доставлено». Інтерфейс побудований таким чином, щоб навіть користувач із мінімальним досвідом користування подібними платформами міг інтуїтивно зрозуміти, куди натискати та як взаємодіяти із сайтом.

Особливу увагу приділено адаптивності інтерфейсу. Завдяки використанню фреймворку Nuxt.js із підтримкою SSR (server-side rendering) сайт коректно працює як на десктопних пристроях, так і на планшетах та мобільних телефонах. Широкі панелі автоматично перетворюються на «бургер-меню», зображення масштабуються відповідно до роздільної здатності екрана, а шрифти та кнопки зберігають читабельність і натискання.

Кожен товар представлений на окремій сторінці, яка містить розширену інформацію: детальний опис, характеристики, зображення, рейтинг та відгуки користувачів. Сюди ж інтегровано функцію «залишити відгук», яка доступна лише після покупки товару, що гарантує автентичність коментарів.

Інтерфейс також включає кошик, реалізований через Pinia, де користувач може змінювати кількість товару, видаляти позиції, бачити загальну вартість та переходити до оформлення замовлення. Оформлення замовлення розбито на кілька простих кроків — вибір способу доставки (автоматичне підтягування відділень Нової Пошти), введення контактних даних, вибір способу оплати (наприклад, LiqPay), підтвердження — усе побудовано логічно та зрозуміло.

Загалом, інтерфейс користувача у розробленій системі демонструє прагнення зробити процес онлайн-покупок максимально прозорим, простим і приємним. Навіть незважаючи на значний обсяг функціональності, структура інтерфейсу залишилась чистою та не перевантаженою, що позитивно впливає на загальне враження від взаємодії з платформою.

3.2. Структура інтерфейсу платформи Інтернет-зоомаган

Інтерфейс користувача платформи інтернет-магазину зоотоварів розроблено відповідно до сучасних тенденцій у сфері веб-дизайну на фреймворку Nuxt. Головний акцент зроблено на простоту, зручність та доступність, що забезпечує комфортне користування як для новачків, так і для досвідчених користувачів. Система побудована з урахуванням вимог адаптивності, що означає коректне відображення усіх елементів сайту незалежно від типу пристрою — чи це настільний комп'ютер із широким екраном (рис 3.1) або мобільний телефон (рис 3.2).

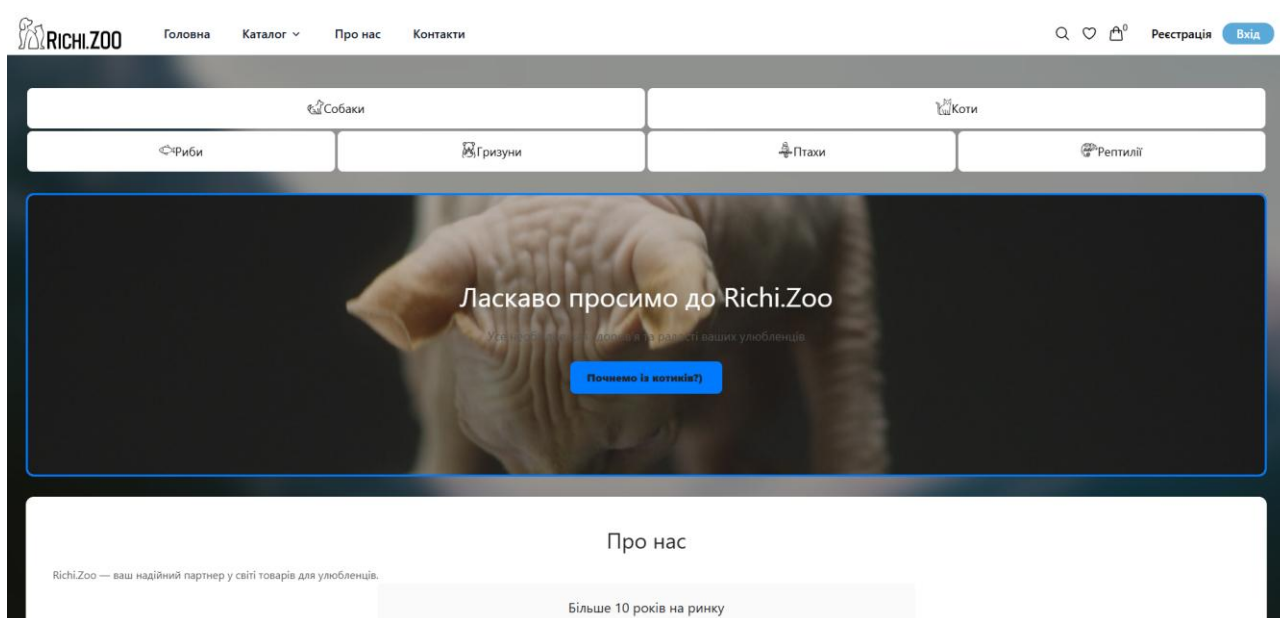


Рис. 3.1. Головна сторінка платформи комп'ютерна версія

Головна сторінка платформи виконує не лише інформативну, а й навігаційну функцію. У верхній її частині розміщено зручну панель із навігаційними посиланнями до основних розділів магазину: каталог товарів, акційні пропозиції, реєстрація та вхід, особистий кабінет користувача, а також сторінка контактів. Особливе значення має поле пошуку, яке дозволяє здійснювати швидке й точне знаходження потрібного товару за ключовими словами. Це значно підвищує ефективність користувацького досвіду, особливо для тих, хто шукає конкретний продукт. Крім того, головна сторінка містить динамічний рекламний банер, що оновлюється залежно від поточних акцій і

сезонних пропозицій — це сприяє залученню уваги та підвищенню конверсії користувачів у покупців.

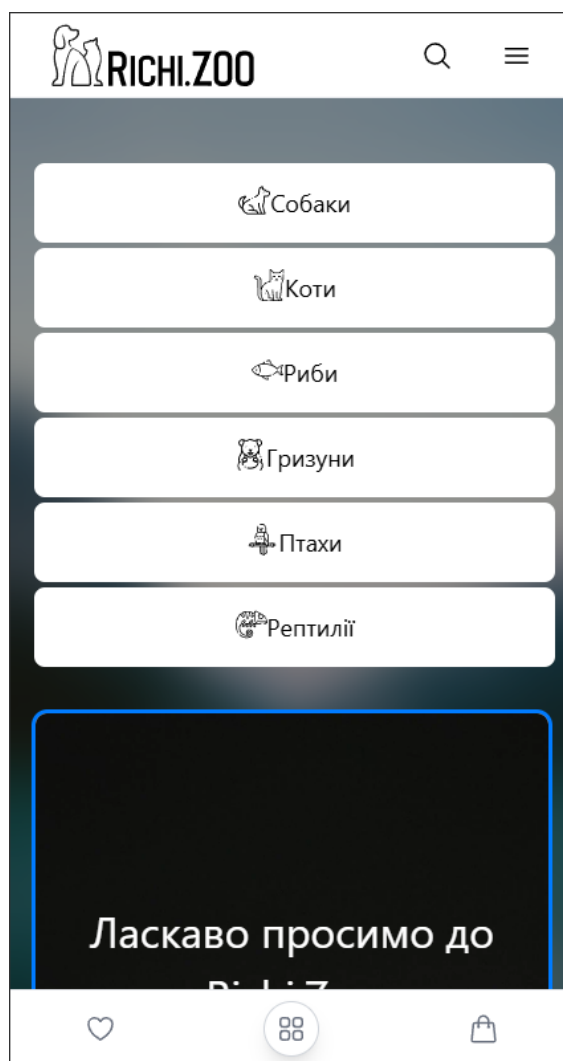


Рис. 3.2. Головна сторінка платформи мобільна версія

Сторінка конкретного товару детально і всебічно представляє інформацію, яка необхідна для здійснення зваженої покупки. Користувач може ознайомитися з детальним описом товару, його характеристиками, актуальною ціною, а також переглянути галерею зображень, яка демонструє товар із різних ракурсів(рис. 3.3).

Окремої уваги заслуговує можливість залишення коментарів до товарів. Цей інструмент не тільки формує зворотний зв'язок, але й створює певну спільноту навколо платформи. Користувач може оцінити товар за допомогою зіркового рейтингу, поділитися враженнями або порадити іншим покупцям, на що варто звернути увагу. Така функція є важливою не лише для кінцевих

користувачів, а й для адміністраторів платформи, адже дає змогу оперативно реагувати на негативні сигнали, удосконалюючи сервіс (рис. 3.4).

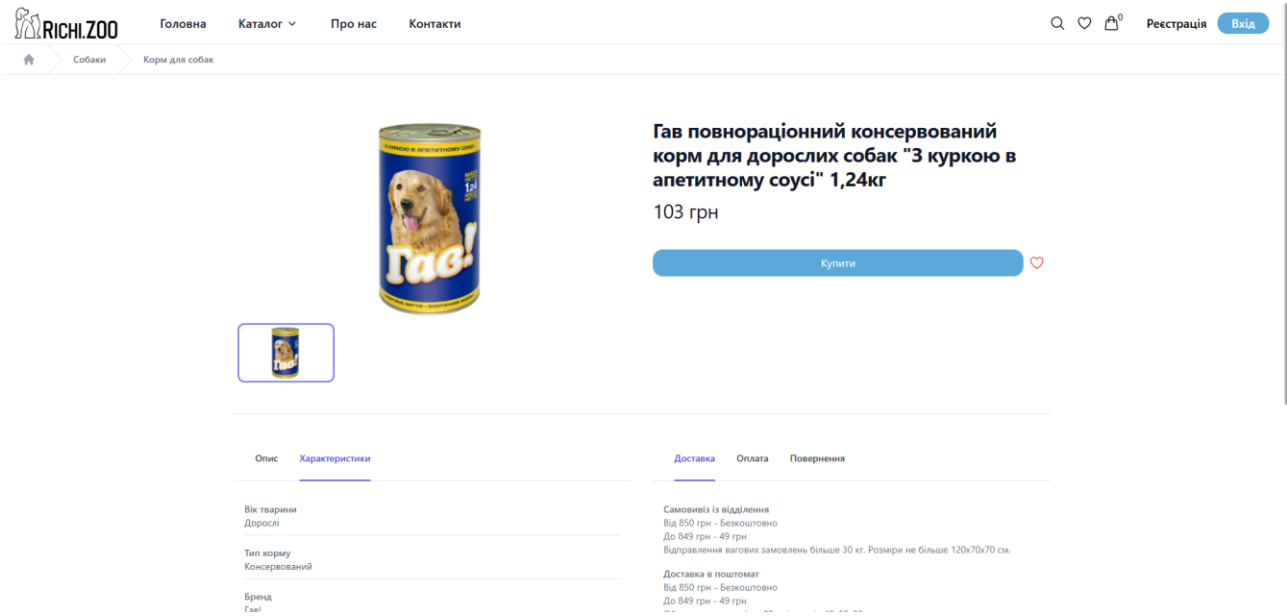


Рис. 3.3. Сторінка детального опису товару

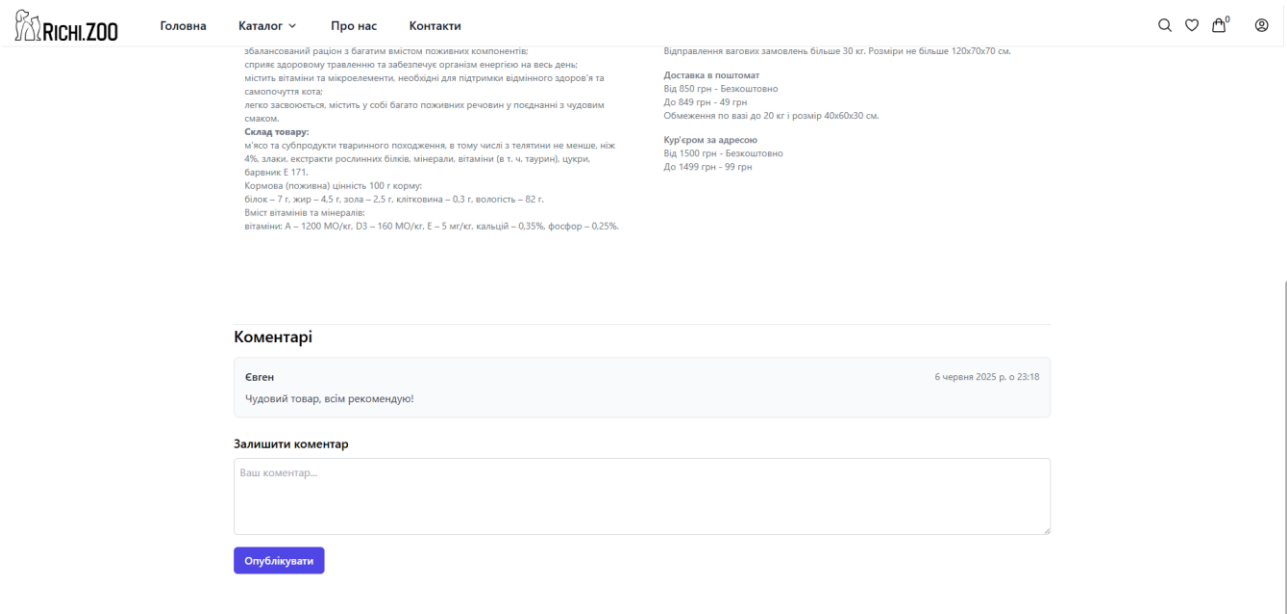


Рис. 3.4. Блок коментарів під товаром для авторизованих користувачів

У процесі взаємодії з інтернет-платформою користувачі можуть перебувати в різних ролях, які суттєво впливають на доступний функціонал і зручність користування сервісом. Одним із фундаментальних принципів побудови інтерфейсу системи стало чітке розмежування між можливостями авторизованого та неавторизованого користувача. Такий підхід дозволяє забезпечити як безпеку персональних даних, так і належну гнучкість у взаємодії

із сервісом на всіх рівнях — від простого ознайомлення до активної участі в процесі купівлі товарів чи залишення зворотного зв'язку.

Користувач, який не має облікового запису або ще не пройшов процедуру входу в систему, все одно може вільно переглядати каталог продукції. Йому доступні фільтрація товарів за категоріями, пошук за ключовими словами, а також перегляд детальної інформації про обраний товар. Таким чином, навіть гість отримує повноцінну картину асортименту, що дозволяє сформуванню перше враження про платформу, її інтерфейс, структуру товарів і загальну зручність використання.

Після реєстрації користувач отримує можливість увійти до особистого кабінету. Це окремий простір, що відкриває новий рівень взаємодії з платформою. Саме тут він може змінювати свої персональні дані, наприклад, оновити контактний номер, змінити електронну адресу або завантажити фотографію профілю. Крім цього, авторизований користувач бачить історію своїх замовлень — це не лише зручно з точки зору обліку, але й дозволяє швидко знайти інформацію про попередні покупки, повторити замовлення або відстежити статус поточних транзакцій (рис 3.5).

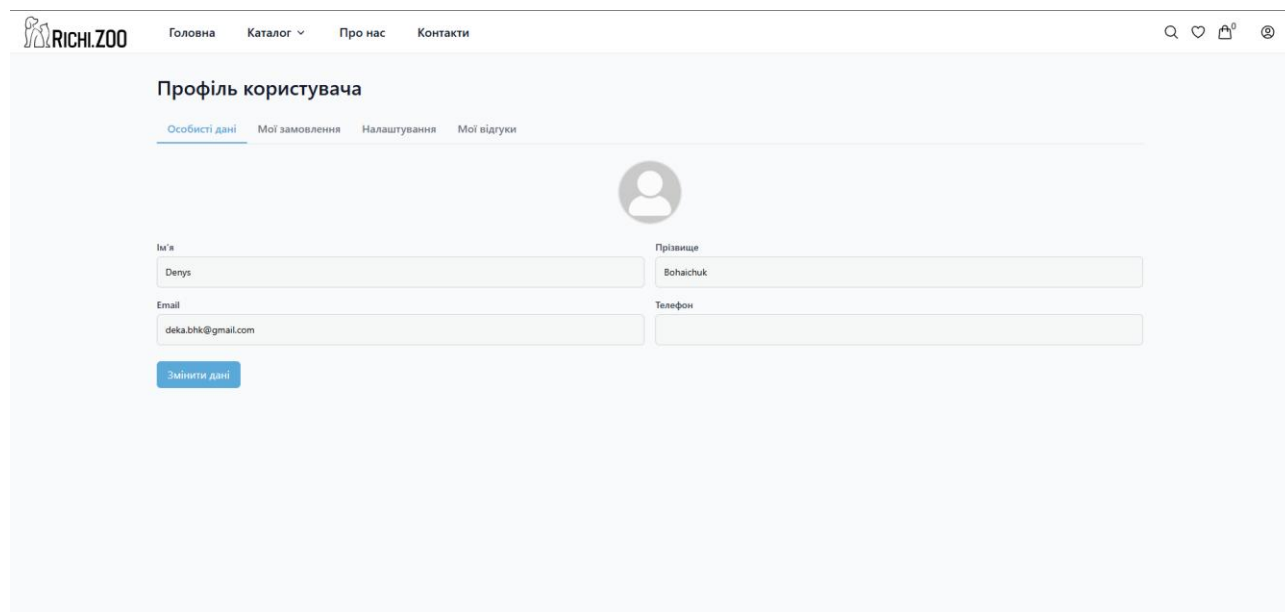


Рис. 3.5. Профіль користувача

Особистий кабінет також дозволяє переглядати список написаних коментарів або історію замовлень. Це створює додатковий комфорт, оскільки

користувач має простий спосіб отримати дані, які він виконував на цьому ресурсі що стали внеском у наповнення платформи. Редагування профілю також реалізоване з максимальною зручністю — без необхідності повторного входу або додаткових підтверджень на кожному кроці (рис 3.6).

Рис. 3.6. Профіль користувача

Адміністративна панель платформи, реалізована з використанням фреймворку Laravel, є важливим інструментом, що дозволяє адміністратору повністю керувати всіма аспектами функціонування магазину. Інтерфейс панелі побудований таким чином, щоб навіть недосвідчені адміністратори могли легко орієнтуватися у її функціоналі. Основним розділом панелі є «Товари», де адміністратор може додавати нові продукти, редагувати існуючі та, за необхідності, видаляти їх. Форма додавання товарів містить всі необхідні поля для зручного управління інформацією про товари: назву, детальний опис, ціну, кількість на складі та можливість завантаження зображень. Після збереження інформації новий товар автоматично стає доступним для перегляду на сайті (рис. 3.7).

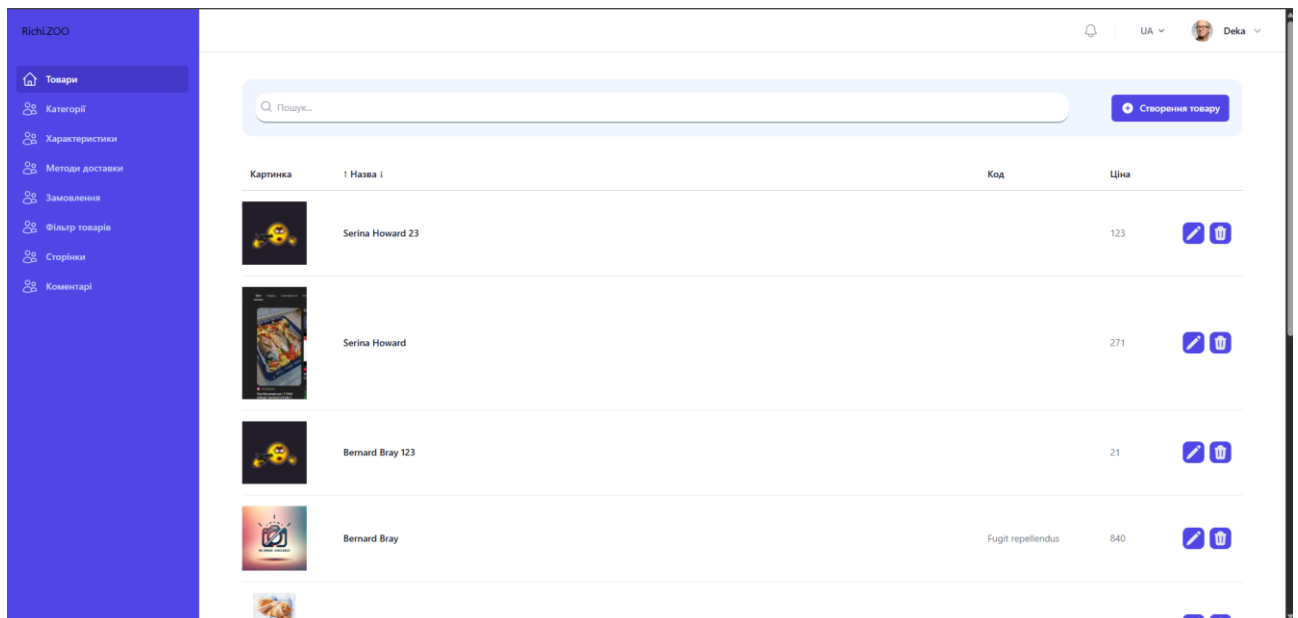


Рис. 3.7. Сторінка управління товарами

Розділ «Замовлення» в адміністративній панелі відображає всі необхідні деталі кожного замовлення, включаючи статус, список замовлених товарів, контактні дані користувача, спосіб оплати та доставки. Це дає можливість адміністратору оперативно реагувати на нові замовлення, швидко змінювати статуси та здійснювати контроль за процесом виконання замовлень (рис. 3.8).

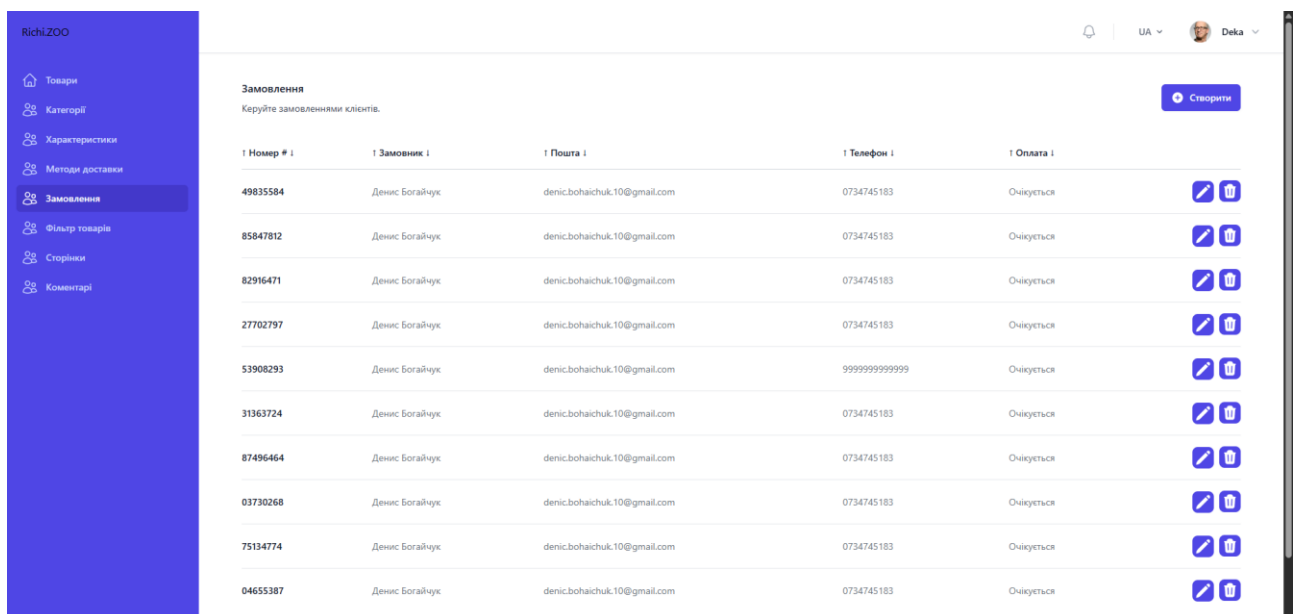


Рис. 3.8. Сторінка управління замовленнями

Для керування характеристиками товарів у адміністративній панелі призначена вкладка «Характеристики». Цей функціонал дозволяє адміністратору додавати нові характеристики, наприклад колір, розмір або матеріал, які пізніше

можна призначати конкретним товарам. Адміністратор має можливість редагувати вже існуючі характеристики, змінювати їх опис та назву або видаляти непотрібні. Це спрощує процес управління каталогом та підвищує гнучкість системи (рис. 3.9).

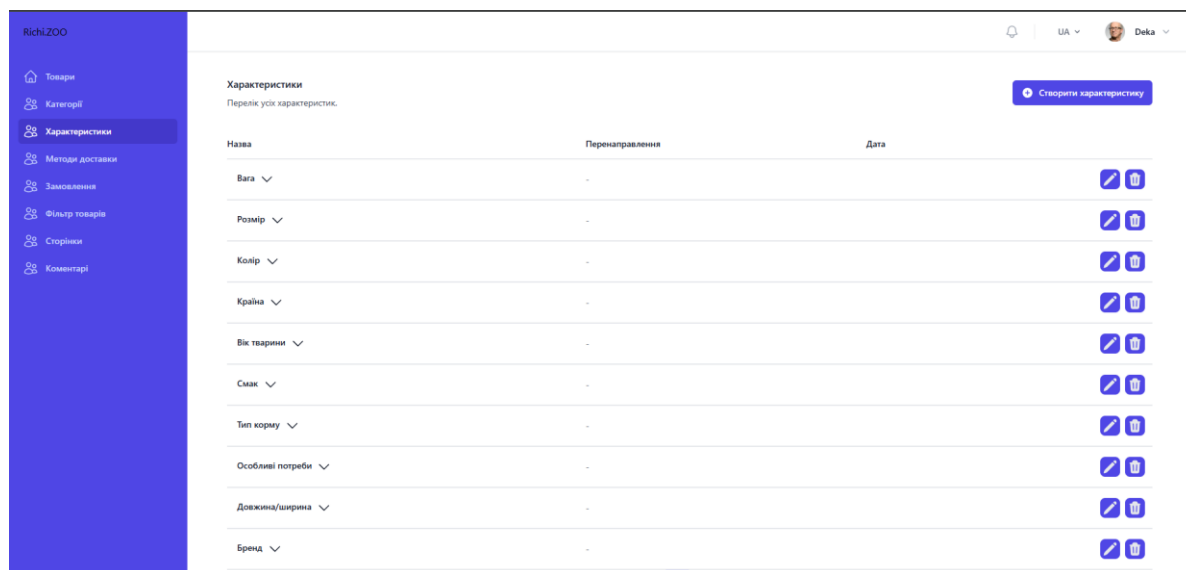


Рис. 3.9. Сторінка управління характеристиками

3.3. Тестування роботи платформи

Для перевірки роботи розробленої платформи проведено функціональне та користувацьке тестування, що дозволило оцінити ефективність та зручність її використання. Тестування охопило основні сценарії користування платформою, такі як реєстрація користувачів, перегляд та пошук товарів, оформлення замовлення та взаємодію з адміністративною панеллю.

Користувач заходить на сайт, переглядає список товарів та обирає продукт для детального перегляду. Додає товар до кошика та переходить до оформлення замовлення. Заповнює необхідні дані та здійснює оплату через інтегровану платіжну систему LiqPay (рис. 3.10).

Рис. 3.10. Сторінка оформлення замовлення

Рис. 3.10. Сторінка оформлення замовлення

Після здійснення покупки інформація про замовлення відображається в адміністративній панелі з відповідним статусом, що дозволяє адміністратору оперативно обробляти замовлення (рис. 3.11).

Рис. 3.11. Відображення замовлення в адміністративній панелі

Рис. 3.11. Відображення замовлення в адміністративній панелі

Також проведено тестування реєстрації нового користувача. Після заповнення форми реєстрації система створює профіль, який доступний користувачу для подальшої взаємодії з платформою (рис. 3.8).

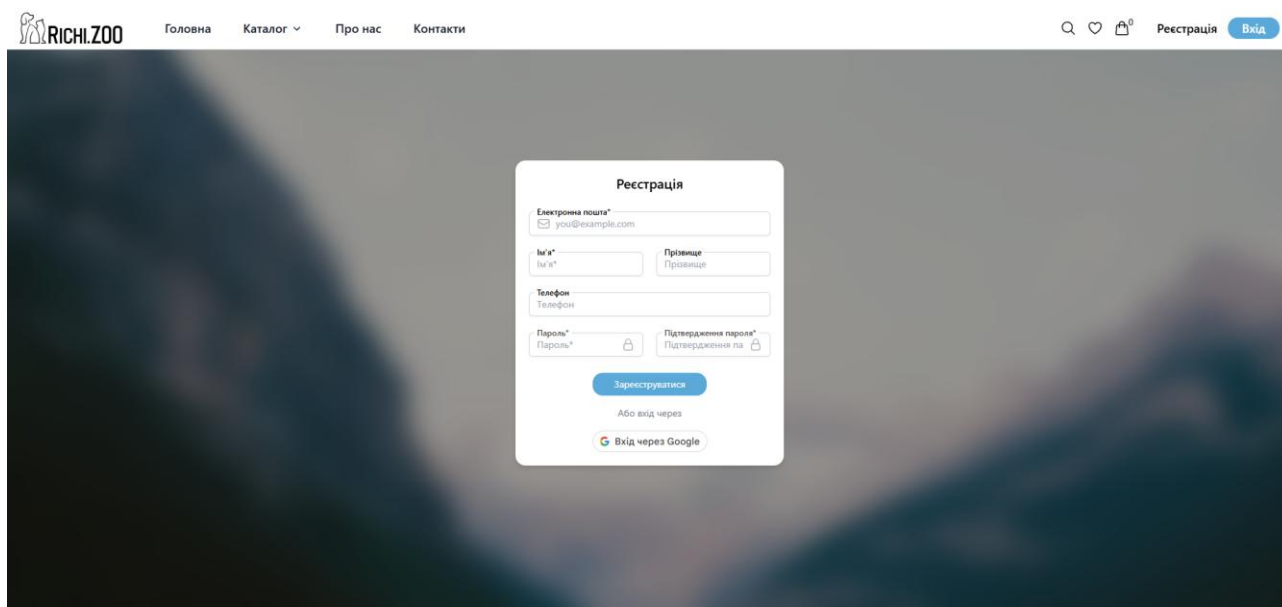


Рис. 3.12 Форма реєстрації нового користувача

Висновки до третього розділу

У третьому розділі детально описано інтерфейс користувача та адміністративної панелі, які спроектовані з урахуванням комфорту та інтуїтивності взаємодії. Завдяки цьому користувачі легко орієнтуються на сайті, швидко знаходять потрібні товари та без зайвих зусиль здійснюють покупки. Адміністративна панель, своєю чергою, дозволяє ефективно управляти всім процесом продажів, завдяки простому й зрозумілому інтерфейсу. Проведені тестування підтвердили коректність роботи таких критичних функцій, як реєстрація користувачів, оформлення замовлень та адміністрування платформи, що вказує на повну готовність інтернет-магазину до практичного використання в реальних умовах.

ВИСНОВКИ

У процесі виконання кваліфікаційної роботи бакалавра було розроблено повноцінну веб-платформу для інтернет-зоомагазину, яка дозволяє реалізовувати сучасний онлайн-торгівельний процес із продажу товарів для домашніх тварин. На етапі підготовки було проаналізовано існуючі аналоги, зокрема такі сервіси, як **ZvirShop**, **Toba.ua** та інші подібні ресурси. Результати аналізу показали наявність значних недоліків у дизайні, швидкості роботи та зручності інтерфейсу в конкурентів, що надихнуло створити більш сучасне, функціональне та інтуїтивно зрозуміле рішення.

Було визначено основні вимоги до майбутньої системи — простота для кінцевого користувача, підтримка авторизації через email та Google, гнучка система доставки через Нову Пошту, можливість залишати коментарі, бачити історію замовлень та забезпечення зручного адміністрування. У процесі проектування було спроектовано об'єктно-орієнтовану архітектуру, що включає окремі класи для користувачів, товарів, замовлень, категорій та відгуків, і розроблено реляційну базу даних на основі MySQL, яка повністю відповідає функціональним потребам системи.

Було створено набір алгоритмів, що покривають ключові етапи взаємодії з платформою: від фільтрації товарів і оформлення замовлення — до модерації відгуків. Усі ці процеси ретельно продумані та протестовані, забезпечуючи коректну роботу системи навіть при значному навантаженні. Сам застосунок розроблено із використанням зв'язки **Laravel** для бекенду та **Nuxt.js** для фронтенду, що забезпечило сучасний та гнучкий підхід до архітектури.

Функціонал авторизації реалізовано з урахуванням безпеки — із застосуванням Sanctum та JWT, а оформлення замовлень передбачає інтеграцію з LiqPay, що дозволяє здійснювати безпечні онлайн-платежі. Важливим аспектом стало також впровадження адаптивного інтерфейсу, який однаково добре працює як на ПК, так і на мобільних пристроях. Особлива увага приділялась інтерфейсу особистого кабінету користувача — він дозволяє переглядати попередні покупки, змінювати свої дані, залишати або переглядати коментарі.

					ДУ «Житомирська політехніка».25.121.3.000 - ПЗ	Арк.
						51

Усі модулі системи були протестовані — як функціонально, так і в контексті інтеграції між клієнтською та серверною частинами. Тестування підтвердило стабільність платформи, коректність обробки замовлень, надійність авторизації та обробки платежів. Було також реалізовано базові механізми логування та обробки помилок.

Таким чином, результати роботи демонструють ефективність використання сучасного стеку технологій для реалізації онлайн-торгівельного майданчика. Система дозволяє забезпечити повний цикл роботи інтернет-магазину — від перегляду товару до фінального оформлення замовлення та його оплати. Практична цінність роботи полягає в можливості розгортання повноцінного комерційного магазину з гнучким керуванням контентом та зручною клієнтською частиною.

Подальші вдосконалення системи можуть включати впровадження системи рейтингу продавців, розширення панелі адміністратора з аналітикою, впровадження push-сповіщень та підтримку багатомовності, що дозволить вивести проєкт на новий рівень з точки зору зручності та комерційного потенціалу.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Офіційний сайт «ZvirShop» [Електронний ресурс] – Режим доступу до ресурсу: <https://zvirshop.com/>
2. Офіційний сайт «Toba.ua» [Електронний ресурс] – Режим доступу до ресурсу: <https://toba.ua/>
3. Документація Laravel Framework [Електронний ресурс] – Режим доступу до ресурсу: <https://laravel.com/docs>
4. Документація Laravel Eloquent ORM [Електронний ресурс] – Режим доступу до ресурсу: <https://laravel.com/docs/eloquent>
5. Документація Laravel Blade [Електронний ресурс] – Режим доступу до ресурсу: <https://laravel.com/docs/blade>
6. Документація Nuxt.js [Електронний ресурс] – Режим доступу до ресурсу: <https://nuxt.com/docs>
7. Документація Nuxt Axios [Електронний ресурс] – Режим доступу до ресурсу: <https://axios.nuxtjs.org/>
8. Документація Vue.js 3 [Електронний ресурс] – Режим доступу до ресурсу: <https://vuejs.org/guide/introduction.html>
9. Документація Tailwind CSS [Електронний ресурс] – Режим доступу до ресурсу: <https://tailwindcss.com/docs>
10. Документація jQuery [Електронний ресурс] – Режим доступу до ресурсу: <https://jquery.com/>
11. Документація jsTree [Електронний ресурс] – Режим доступу до ресурсу: <https://www.jstree.com/docs/>
12. Документація MySQL [Електронний ресурс] – Режим доступу до ресурсу: <https://dev.mysql.com/doc/>
13. Документація MySQL Foreign Keys [Електронний ресурс] – Режим доступу до ресурсу: <https://dev.mysql.com/doc/refman/8.0/en/create-table-foreign-keys.html>
14. Документація PHP [Електронний ресурс] – Режим доступу до ресурсу: <https://www.php.net/manual/>

15. Документація PHP Secure Password Hashing [Електронний ресурс] – Режим доступу до ресурсу: <https://www.php.net/manual/en/function.password-hash.php>

16. Документація Laravel Sanctum [Електронний ресурс] – Режим доступу до ресурсу: <https://laravel.com/docs/sanctum>

17. Документація Laravel Validation [Електронний ресурс] – Режим доступу до ресурсу: <https://laravel.com/docs/validation>

18. Документація HTML5 (W3C) [Електронний ресурс] – Режим доступу до ресурсу: <https://www.w3.org/TR/html52/>

19. Документація CSS (W3C) [Електронний ресурс] – Режим доступу до ресурсу: <https://www.w3.org/Style/CSS/>

20. Документація Bootstrap 5 [Електронний ресурс] – Режим доступу до ресурсу: <https://getbootstrap.com/docs/5.3/getting-started/introduction/>

21. Документація Bootstrap Forms [Електронний ресурс] – Режим доступу до ресурсу: <https://getbootstrap.com/docs/5.3/forms/overview/>

22. Документація Bootstrap Modal Components [Електронний ресурс] – Режим доступу до ресурсу: <https://getbootstrap.com/docs/5.3/components/modal/>

23. Документація JavaScript (MDN) [Електронний ресурс] – Режим доступу до ресурсу: <https://developer.mozilla.org/en-US/docs/Web/JavaScript>

24. Документація DOM API (MDN) [Електронний ресурс] – Режим доступу до ресурсу: https://developer.mozilla.org/en-US/docs/Web/API/Document_Object_Model

25. Документація Docker [Електронний ресурс] – Режим доступу до ресурсу: <https://docs.docker.com/>

26. Документація Docker Compose [Електронний ресурс] – Режим доступу до ресурсу: <https://docs.docker.com/compose/>

27. Офіційний сайт LiqPay [Електронний ресурс] – Режим доступу до ресурсу: <https://www.liqpay.ua/>

28. Документація LiqPay API [Електронний ресурс] – Режим доступу до ресурсу: <https://www.liqpay.ua/documentation/api>

					ДУ «Житомирська політехніка».25.121.3.000 - ПЗ	Арк.
						54

29. Документація REST API [Електронний ресурс] – Режим доступу до ресурсу: <https://restfulapi.net/>

30. Документація GitHub [Електронний ресурс] – Режим доступу до ресурсу: <https://docs.github.com/>

31. Документація OpenServer [Електронний ресурс] – Режим доступу до ресурсу: <https://ospanel.io/>

32. Документація Web Accessibility Standards (WAI) [Електронний ресурс] – Режим доступу до ресурсу: <https://www.w3.org/WAI/standards-guidelines/>

ДОДАТКИ

					ДУ «Житомирська політехніка».25.121.3.000 - ПЗ	Арк.
						56

Фільтрація Товарів

```

class Service
{
    public function store(array $data): array
    {
        try {
            DB::beginTransaction();

            $filter = CategoryFilterFeature::create([
                'feature_id' => $data['feature_id'],
                'category_id' => $data['category_id'],
                'filter_type' => $data['filter_type'],
            ]);

            // Додавання вибраних існуючих значень
            if (!empty($data['existing_values'])) {
                foreach ($data['existing_values'] as $valueId) {
                    CategoryFilterFeatureValue::create([
                        'category_filter_feature_id' => $filter-
>id,
                        'feature_value_id' => $valueId,
                    ]);
                }
            }

            // Додавання нових значень
            if (!empty($data['new_values'])) {
                foreach ($data['new_values'] as $newValue) {
                    $featureValue = FeatureValue::create([
                        'feature_id' => $data['feature_id'],
                        'value' => $newValue,
                    ]);
                    CategoryFilterFeatureValue::create([
                        'category_filter_feature_id' => $filter-
>id,
                        'feature_value_id' => $featureValue->id,
                    ]);
                }
            }

            DB::commit();
            return ['status' => true, 'message' => 'Фільтр
створено успішно.'];
        } catch (\Exception $e) {
            DB::rollBack();
            return ['status' => false, 'message' => 'Помилка при
створенні фільтра: ' . $e->getMessage()];
        }
    }
}

```

```

    }
    public function update(int $filterId, array $data): array
    {
        try {
            DB::beginTransaction();

            $filter =
CategoryFilterFeature::findOrFail($filterId);
            $filter->update([
                'feature_id' => $data['feature_id'],
                'category_id' => $data['category_id'],
                'filter_type' => $data['filter_type'],
            ]);

            // Видаляємо старі зв'язки значень
            $filter->filterValues()->delete();

            // Додаємо вибрані існуючі значення
            if (!empty($data['existing_values'])) {
                foreach ($data['existing_values'] as $valueId) {
                    CategoryFilterFeatureValue::create([
                        'category_filter_feature_id' => $filter-
>id,
                        'feature_value_id' => $valueId,
                    ]);
                }
            }

            // Додаємо нові значення
            if (!empty($data['new_values'])) {
                foreach ($data['new_values'] as $newValue) {
                    $featureValue = FeatureValue::create([
                        'feature_id' => $data['feature_id'],
                        'value' => $newValue,
                    ]);
                    CategoryFilterFeatureValue::create([
                        'category_filter_feature_id' => $filter-
>id,
                        'feature_value_id' => $featureValue->id,
                    ]);
                }
            }

            DB::commit();
            return ['status' => true, 'message' => 'Фільтр
оновлено успішно.'];
        } catch (\Exception $e) {
            DB::rollBack();
            return ['status' => false, 'message' => 'Помилка при
оновленні фільтра: ' . $e->getMessage()];
        }
    }

```

```

public function destroy(CategoryFilterFeature $filter)
{
    try {
        $filter->delete();
        return response()->json([
            'success' => "Filter for category \"{$filter-
>category_id}\" was successfully deleted"
        ]);
    } catch (\Exception $e) {
        return response()->json([
            'error' => "Failed to delete filter: " . $e-
>getMessage()
        ], 400);
    }
}
}

```

Управління товарами

```

class Service
{
    protected ImageManager $imageManager;

    public function __construct(ImageManager $imageManager)
    {
        $this->imageManager = $imageManager;
    }

    // Метод для отримання всіх товарів з пагінацією
    public function getAllProductsPaginate($request)
    {
        $perPage = $request->input('perPage', 10);

        $sortBy = $request->query('sort_by', 'created_at');
        $sortOrder = $request->query('sort_order', 'desc');

        $search = $request->query('search');

        $data = Products::when($search, function ($query, $search)
        {
            return $query->where(function ($q) use ($search) {
                $q->where('name', 'LIKE', '%' . $search . '%')
                    ->orWhere('barcode', 'LIKE', '%' . $search .
                    '%');
            });
        })
        ->when($sortBy, function ($query, $sortBy) use
        ($sortOrder) {
            return $query->orderBy($sortBy, $sortOrder);
        })
        ->with('productImages')
        ->paginate($perPage);

        return $data->appends([
            'perPage' => $perPage,
            'sort_by' => $sortBy,
            'sort_order' => $sortOrder,
            'search' => $search
        ]);
    }

    // Метод для створення нового продукту
    public function store($productData)
    {
        DB::beginTransaction();
    }

```

					ДУ «Житомирська політехніка».25.121.3.000 - ПЗ	Арк.
						60

```

try {
    $productData = reset($productData);
    $issetImages = isset($productData['image']);

    // Upload images if present
    if ($issetImages) {
        $images = collect($productData['image'])-
>filter(fn($image) => $image && $image->isValid());
        $imagesPaths = $this->imageManager-
>uploadImages($images, 'paths.products_image_storage');
    }

    // Remove images from the data array
    $productData = Arr::except($productData, ['image']);

    // Create the product
    $product = Products::create($productData);

    // Create a new product group and associate it with
the product
    $groupId = $this-
>extractGroupIdFromExistingProduct($product->id);
    if (is_null($groupId)) {
        $groupId = $this->generateNewGroupId();
    }
    $this->ensureProductGroupExists($product->id,
$groupId);

    // Attach categories and features
    $product->categories()-
>attach($productData['categories']);
    $product->productFeatures()-
>attach($productData['features']);

    // Associate images with the product if any
    if ($issetImages) {
        $this->associateImagesWithProduct($product,
$imagesPaths);
    }

    DB::commit();

    return [
        'status' => true,
        'message' => 'Товар успішно створено',
        'product_id' => $product->id,
    ];
} catch (\Exception $e) {
    DB::rollBack();

    return [
        'status' => false,
        'message' => $e->getMessage(),
    ];
}

```

```

        ];
    }
}

// Метод для оновлення продукту
public function update($productId, $productsData)
{
    DB::beginTransaction();

    try {
        // Extract categories and group_id from the first form
        $categories = $this-
>extractDataFromFirstForm($productsData, 'categories');
        $description = $this-
>extractDataFromFirstForm($productsData, 'description');
        $selected_features = $this-
>extractDataFromFirstForm($productsData, 'selected_features');
        $groupId = $this-
>extractGroupIdFromExistingProduct($productId);

        // Отримати всі продукти, які належать до поточної
групи
        $allProductsInGroup = ProductGroup::where('group_id',
$groupId)->get();

        // Видалити всі старі асоціації продуктів з групою
        ProductGroup::where('group_id', $groupId)->delete();

        // Створюємо новий group_id для переданих товарів
        $newGroupIdForPassedProducts = $this-
>generateNewGroupId();

        foreach ($productsData as $formKey => $productForm) {
            // Ensure each productForm has the categories from
the first form
            if (is_array($productForm)) {
                $productForm['categories'] = $categories;
                $productForm['description'] = $description;
                $productForm['selected_features'] =
$selected_features;
                $this-
>updateOrCreateSingleProduct($productForm,
$newGroupIdForPassedProducts);
            }
        }

        // Створити новий group_id для товарів, які не були
передані
        $newGroupIdForOtherProducts = $this-
>generateNewGroupId();

        // Оновити всі інші продукти, які були в старій групі,

```

					ДУ «Житомирська політехніка».25.121.3.000 - ПЗ	Арк.
						62

```

але не передані в update
$productIdsInData = [];

foreach ($productsData as $formKey => $productForm) {
    if (isset($productForm['id'])) {
        $productIdsInData[] = $productForm['id'];
    }
}

// Далі перевіряємо продукти з групи
foreach ($allProductsInGroup as $product) {
    if (!in_array($product->product_id,
$productIdsInData)) {
        // Оновити продукт новим group_id, відмінним
від того, що було передано
        $product->group_id =
$newGroupIdForOtherProducts;

        // Оновлення продукту через update або save()
        if (!$product->save()) {
            throw new \Exception("Не вдалося зберегти
товар з ID: {$product->id}");
        }
    }
}
DB::commit();
//dd('end');

return [
    'status' => true,
    'message' => 'Товар успішно оновлено',
];
} catch (\Exception $e) {
    DB::rollBack();

    return [
        'status' => false,
        'message' => $e->getMessage(),
    ];
}

}

private function extractDataFromFirstForm(array $data, $key):
array|string
{
    $firstFormKey = array_key_first($data);
    return $data[$firstFormKey][$key] ?? '';
}

private function
extractGroupIdFromExistingProduct($productId): ?int
{
    $productGroup = ProductGroup::where('product_id',

```

```

$productId)->first();
    return $productGroup ? $productGroup->group_id : null;
}

private function ensureProductGroupExists($productId,
$groupId)
{
    // Check if a product group exists for this product and
group ID
    $productGroup = ProductGroup::where('product_id',
$productId)->where('group_id', $groupId)->first();

    // If no product group exists, create one
    if (!$productGroup) {
        $productGroup = ProductGroup::create([
            'product_id' => $productId,
            'group_id' => $groupId,
            'created_at' => now(),
            'updated_at' => now(),
        ]);
    }

    return $productGroup;
}

private function generateNewGroupId(): int
{
    do {
        $groupId = random_int(100000, 999999); // Generate a
random group ID within a range
    } while (ProductGroup::where('group_id', $groupId)-
>exists());

    return $groupId;
}

private function updateOrCreateSingleProduct(array
$productData, $productGroup): void
{
    $product = Products::find($productData['id'] ?? null);

    if ($product) {
        // Update existing product
        $this->updateProduct($product, $productData,
$productGroup);
    } else {
        // Create a new product
        $this->createProduct($productData, $productGroup);
    }
}

private function updateProduct(Products $product, array
$productData, $groupId): void

```

```

{
    // Handle image updates
    $this->handleImageUpdates($product, $productData);

    // Exclude 'image' from the product data to avoid issues
    with mass assignment
    $productData = Arr::except($productData, ['image']);

    // Update the product with new data
    $product->update($productData);

    // Sync categories and features
    $this->syncRelations($product, $productData);

    // Ensure the product group is correctly associated
    $this->associateProductWithGroup($product->id, $groupId);
}

private function createProduct(array $productData, $groupId):
void
{
    // Create a new product instance
    $product = new Products($productData);

    // Save the new product
    $product->save();

    // Handle image uploads if there are any
    if (isset($productData['image'])) {
        $this->handleImageUpdates($product, $productData);
    }

    // Sync categories and features
    $this->syncRelations($product, $productData);

    // Ensure the product group is correctly associated
    $this->associateProductWithGroup($product->id, $groupId);
}

private function handleImageUpdates(Products $product, array
&$productData): void
{
    $existingImagePaths = $product->images->pluck('image')->
toArray();

    if (isset($productData['image'])) {
        $newImages = collect($productData['image'])->
filter(fn($image) => $image && $image->isValid());
        $updatedImagePaths = $this->imageManager->
updateImages($existingImagePaths, $newImages,
'paths.products_image_storage');

        $product->images()->delete();
        $this->associateImagesWithProduct($product,

```

```

$updatedImagePaths);
    } else {
        $this->imageManager->
>deleteImages($existingImagePaths);
        $product->images()->delete();
    }
}

private function syncRelations(Products $product, array
$productData): void
{
    if (isset($productData['categories'])) {
        $product->categories()-
>sync($productData['categories']);
    }

    if (isset($productData['features'])) {
        $product->productFeatures()-
>sync($productData['features']);

        if (isset($productData['selected_features'])) {
//            dump($productData['selected_features']);

            // Перетворюємо рядок з feature_id у масив
            if (!empty($productData['selected_features']) &&
isset($productData['selected_features'][0])) {
                $selectedFeatureIds = explode(',',
$productData['selected_features'][0]);
            } else {
                // Обробка випадку, коли дані відсутні або
порожні
                $selectedFeatureIds = [];
            }

            // Отримуємо всі feature_value_id для вибраних
feature_id
            $selectedFeatureValueIds =
DB::table('feature_values')
                ->whereIn('feature_id', $selectedFeatureIds)
                ->pluck('id')
                ->toArray();

            // dump($selectedFeatureValueIds);

            // Перетворюємо масив в строку для SQL запиту
            $selectedFeatureValueIdsStr = implode(',',
$selectedFeatureValueIds);
            // dump($selectedFeatureValueIdsStr);

            // Виконуємо запит для оновлення is_selected для
відповідних feature_value_id
            if (!empty($selectedFeatureValueIds)) {
                $selectedFeatureValueIdsStr = implode(',',

```

```

$selectedFeatureValueIds);

        // Виконуємо запит для оновлення is_selected
для відповідних feature_value_id
        DB::statement("
            UPDATE product_features
            SET is_selected = CASE
                WHEN feature_value_id IN ($selectedFeatureValueIdsStr)
THEN 1
                ELSE 0
            END
            WHERE product_id = ?
        ", [$product->id]);
        } else {
            // Можливо, скинути is_selected для всіх
значень або інші дії
            DB::statement("
                UPDATE product_features
                SET is_selected = 0
                WHERE product_id = ?
            ", [$product->id]);
        }
    }
}

private function associateProductWithGroup($productId,
$groupId): void
{
    ProductGroup::updateOrCreate(
        ['product_id' => $productId], // Умова для пошуку
наявного запису
        [
            'group_id' => $groupId,
            'created_at' => now(),
            'updated_at' => now(),
        ]
    );
}

// Метод для асоціації зображень з продуктом
protected function associateImagesWithProduct($product,
$imagesPaths)
{
    foreach ($imagesPaths as $path) {
        $product->images()->create(['image' => $path]);
    }
}
}

```

Управління замовленнями

```

/**
 * Get all orders with pagination.
 *
 * @param \Illuminate\Http\Request $request
 * @param int $perPage
 * @return \Illuminate\Contracts\Pagination\LengthAwarePaginator
 */
public function getAllOrdersPaginate($request, $perPage = 10)
{
    return Orders::paginate($perPage);
}

/**
 * Store a new order and its related items.
 *
 * @param array $orderData
 * @return array
 */
public function store($orderData)
{
    DB::beginTransaction();

    $totalAmount = 0;
    foreach ($orderData['order_items'] as $item) {
        $product = Products::findOrFail($item['product_id']);
        $totalAmount += $product->price * $item['quantity'];
    }

    try {
        // Create the order
        $order = Orders::create([
            'order_number' => $orderData['order_number'],
            'email' => $orderData['email'],
            'name' => $orderData['name'],
            'surname' => $orderData['surname'],
            'phone' => $orderData['phone'],
            'city_ref' => $orderData['city_ref'],
            'warehouse_ref' => $orderData['warehouse_ref'],
            'delivery_method_id' =>
                $orderData['delivery_method_id'],
            'total_amount' => $totalAmount,
            'payment_status' => 'Очікується',
        ]);

        // Prepare the order items data
        $orderItems = array_map(function ($item) use ($order) {
            return [
                'order id' => $order->id,

```

```

        'product_id' => $item['product_id'],
        'quantity' => $item['quantity'],
    ];
}, $orderData['order_items']);

// Insert the order items into the database
OrderItems::insert($orderItems);

DB::commit();
return [
    'status' => true,
    'message' => 'Order is successfully created',
    'order_id' => $order->id,
];
} catch (\Exception $e) {
    DB::rollBack();

    return [
        'status' => false,
        'message' => 'Failed to create order: ' . $e-
>getMessage(),
    ];
}
}

/**
 * Update an existing order and its related items.
 *
 * @param \App\Models\Admin\Orders $order
 * @param array $orderData
 * @return array
 */
public function update($order, $orderData)
{
    DB::beginTransaction();

    try {
        $totalAmount = 0;
        foreach ($orderData['products'] as $item) {
            $product = Products::findOrFail($item['product_id']);
            $totalAmount += $product->price * $item['quantity'];
        }

        // Update the order
        $order->update([
            'email' => $orderData['email'],
            'name' => $orderData['name'],
            'surname' => $orderData['surname'],
            'phone' => $orderData['phone'],
            'city_ref' => $orderData['cityRef'],
            'warehouse_ref' => $orderData['warehouseRef'],
            'delivery_method_id' =>
$orderData['deliveryMethodId'],

```

```

        'total_amount' => $totalAmount,
    ]);

    // Delete old order items
    $order->orderItems()->delete();

    // Prepare new order items data
    $orderItems = array_map(function ($item) use ($order) {
        return [
            'order_id' => $order->id,
            'product_id' => $item['product_id'],
            'quantity' => $item['quantity'],
        ];
    }, $orderData['products']);

    // Insert the new order items into the database
    OrderItems::insert($orderItems);

    DB::commit();
    return [
        'status' => true,
        'message' => 'Order is successfully updated',
    ];
} catch (\Exception $e) {
    DB::rollBack();

    return [
        'status' => false,
        'message' => 'Failed to update order: ' . $e-
>getMessage(),
    ];
}
}

```

Конфігурація API маршрутів

```

Route::name('api.')->group(function () {

    Route::prefix('auth')->group(function () {
        Route::post('login', [AuthController::class, 'login'])->
>name('auth.login');
        Route::post('signup', [AuthController::class, 'signup'])->
>name('auth.signup');
        Route::post('google', [AuthController::class,
'googleAuth'])->>name('auth.google');
        Route::post('google', [AuthController::class,
'googleAuth'])->>name('auth.google');
        Route::get('me', [AuthController::class, 'me'])->
>name('auth.me');
        Route::post('/token/refresh', [AuthController::class,
'refreshToken']);
        Route::post('logout', [AuthController::class, 'logout'])->
>name('auth.logout');
    });

    Route::prefix('categories')->group(function () {
        Route::get('get-list-categories',
[CategoriesController::class, 'getListCategories'])->
>name('index');
        Route::get('get-category/{slug}',
[CategoriesController::class, 'getCategory'])->>where('slug',
'.*')->name('get-category');
    });

    Route::prefix('products')->group(function () {
        Route::get('get-product/{id}', [ProductsController::class,
'getProduct'])->>name('get-product');
        Route::get('search-product', [ProductsController::class,
'searchProduct'])->>name('search-product');
    });

    Route::prefix('delivery-methods')->group(function () {
        Route::get('/', [DeliveryMethodController::class,
'index'])->>name('delivery-methods');
    });

    Route::prefix('create-order')->group(function () {
        Route::post('/', [OrdersController::class, 'store'])->
>name('create-order');
    });

    Route::prefix('product-filter')->group(function () {

```

```

        Route::get('{categoryId}',
[ProductFilterController::class, 'getFilterFeaturesByCategory'])->
name('get-product-filter');
    });

    Route::prefix('pages')->group(function () {
        Route::get('{slug}', [PageController::class, 'show']);
    });

    Route::get('products/{product}/comments',
[CommentController::class, 'index'])
        ->name('products.comments.index');

    Route::get('payment/checkout/{orderId}',
[PaymentController::class, 'checkout'])->name('liqpay.checkout');
    Route::post('payment/callback', [PaymentController::class,
'callback'])->name('liqpay.callback');

    Route::middleware('onlyHttp.jwt.auth')->group(function () {
        // Створити новий коментар (status='pending')
        Route::post('products/{product}/comments',
[CommentController::class, 'store'])
            ->name('products.comments.store');

        /* Оновити власний коментар (тільки status='pending')
        Route::put('comments/{comment}',
[CommentController::class, 'update'])
            ->name('comments.update');

        // Видалити власний коментар (тільки status='pending')
        Route::delete('comments/{comment}',
[CommentController::class, 'destroy'])
            ->name('comments.destroy');*/
        Route::get('/user/comments', [CommentController::class,
'userComments'])
            ->name('user.comments');

        Route::get('/user', [UserController::class,
'getProfile']);
        Route::put('/user', [UserController::class,
'updateProfile']);
        Route::post('/user/avatar', [UserController::class,
'uploadAvatar']);
        Route::put('/user/password', [UserController::class,
'changePassword']);
    });

});

```