МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Чорноморський національний університет імені Петра Могили

Факультет комп'ютерних наук

Кафедра інженерії програмного забезпечення

ДОПУЩЕНО ДО ЗАХИСТУ

Завідувач кафедри інженерії
програмного забезпечення

_____ Євген ДАВИДЕНКО

«_19_» __червня___ 2026 р.

КВАЛІФІКАЦІЙНА РОБОТА
НА ЗДОБУТТЯ ОСВІТНЬОГО СТУПЕНЯ БАКАЛАВРА

ВЕБЗАСТОСУНОК ЗООМАГАЗИНУ

Спеціальність 121 Інженерія програмного забезпечення
Освітня програма «Інженерія програмного забезпечення»

Здобувач _____ **Катерина КИРПОТЕНКО**

«_19_» __червня__ 20__ р.

Керівник роботи
PhD, старший
викладач _____ **Ігор КАНДИБА**

«_19_» __червня__ 20__ р.

Миколаїв – 2026

Завдання на виконання кваліфікаційної роботи

Чорноморський національний університет імені Петра Могили

Факультет	Комп'ютерних наук
Кафедра	Інженерії програмного забезпечення
Рівень вищої освіти	Перший (бакалаврський)
Освітній ступінь	Бакалавр
Спеціальність	121 Інженерія програмного забезпечення
Освітня програма	Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ
Завідувач кафедри інженерії
програмного забезпечення
_____ Євген ДАВИДЕНКО
«_1_» ___лютого___ 2026 р.

ЗАВДАННЯ

на кваліфікаційну бакалаврську роботу здобувача

Кирпотенко Катерина

1. Тема кваліфікаційної роботи Вебзастосунок зоомагазину затверджена наказом ректора ЧНУ ім. Петра Могили № _349__ від «_26_» _грудня_ 2026 р.
2. Строк представлення кваліфікаційної роботи «___» _____ 2026 р.
3. Результатом роботи є створений вебзастосунок зоомагазину, який може бути використаний у зоомагазинах для організації роботи продажів, надання онлайн-консультацій, пошуку потрібних товарів та підвищення рівня обслуговування клієнтів.
4. Перелік питань, що підлягають розробці:

- Проаналізувати існуючі веб-ресурси зоомагазинів для виявлення їх переваг та недоліків.
- Визначити функціональні вимоги до сайту та необхідні модулі.
- Розробити структуру сайту.
- Створити прототип інтерфейсу користувача.
- Реалізувати основний функціонал сайту.

5. Перелік графічних матеріалів: Презентація
6. Консультанти:

Консультант	**Кафедра (організація)**	**Частина роботи**

Дата видачі завдання «____» _____ 20___ р.

КАЛЕНДАРНИЙ ПЛАН
виконання кваліфікаційної роботи

Тема: **Вебзастосунок зоомагазину**

№	Найменування роботи	Початок	Закінчення	Примітки
	Розробка та затвердження завдання на виконання КБР	08.02.26	14.02.26	Виконано
	Огляд літератури за темою роботи	14.02.26	20.02.26	Виконано
	Складання календарного плану КБР	20.02.26	23.02.26	Виконано
	Аналіз предметної області	23.02.26	25.02.26	Виконано
	Розробка проєктних рішень	25.03.26	05.04.26	Виконано
	Моделювання та конструювання ПЗ	05.04.26	01.05.26	Виконано
	Кодування, тестування та апробація розробленого ПЗ, аналіз результатів тестування, розробка керівництва користувача	01.05.26	25.06.26	Виконано
	Відгук керівника КБР	20.05.26	21.05.26	Виконано
	Оформлення КБР та презентації	20.05.26	24.05.26	Виконано
	Попередній захист	25.05.26	26.05.26	Виконано
	Рецензування	11.06.26	12.06.26	Виконано
	Завершення оформлення КБР та презентації	12.06.26	18.06.26	Виконано
	Захист кваліфікаційної роботи			

Здобувач _____ **Катерина КИРПОТЕНКО**

«__» _____ 20__ р.

Керівник роботи

Старший викладач,

PhD _____ **Ігор КАНДИБА**

«__» _____ 20__ р.

АНОТАЦІЯ

до кваліфікаційної бакалаврської роботи

Вебзастосунок зоомагазину

Здобувачка 408 гр.: Кирпотенко Катерина

Керівник: PhD, старший викладач Кандиба Ігор

Актуальність роботи полягає в тому, що у сучасних умовах розвитку електронної комерції вебсайти є одним із важливих відгалужень для продажу та комунікації з клієнтами. Для зоомагазинів, які пропонують широкий асортимент товарів для домашніх тварин, наявність сайту дозволяє забезпечити онлайн-продаж, ознайомлення з каталогом товарів, консультації та зручний пошук продукції.

Об’єктом кваліфікаційної роботи є процес обслуговування клієнтів зоомагазину.

Предметом кваліфікаційної роботи є методи та засоби створення вебзастосунку для автоматизації продажів та управління каталогом товарів зоомагазину.

Метою роботи є розробка вебзастосунку зоомагазину для покращення процесу обслуговування клієнтів.

Кваліфікаційна робота складається із вступу, 4 розділів, висновків та переліку джерел посилання.

У вступі детально описано актуальність теми, визначено об’єкт та предмет дослідження, а також сформовано мету.

У першому розділі було проведено аналіз предметної області та існуючих аналогів вебсайтів зоомагазинів, розібрано їх плюси та мінуси, сформовано подальші кроки розробки вебзастосунку .

У другому розділі проведено аналіз сучасного стану, обґрунтування вибору технологій та специфікація вимог до вебзастосунку .

Третій розділ присвячено проєктуванню вебзастосунку , зокрема розробці структури сайту та макапів, ER-діаграми бази даних, UML-діаграм та описано вибір технологій реалізації.

У четвертому розділі описано процес реалізації вебсайту, програмну частину, інтерфейс користувача, а також тестування розробленого застосунку.

У висновках підведено підсумки виконаної роботи, визначено досягнення поставленої мети та окреслено можливі напрями подальшого розвитку системи.

Кваліфікаційна робота викладена на _86_ сторінках машинописного тексту, складається із вступу, _4_ розділів, загальних висновків, переліку джерел посилання з _18_ найменувань та _1_ додатків. Праця містить _3_ таблиць та _23_ рисунків.

Ключові слова: зоомагазин, веброзробка, PHP, Laravel, Microsoft SQL Server.

ABSTRACT

to the qualifying bachelor's thesis

Pet store web aplication

Student of 408 group: Kyrpotenko Kateryna

Supervisor: PhD, senior lecturer Kandyba Ihor

Relevance essence of this work lies in the fact that, in today’s e-commerce landscape, websites are a vital channel for sales and customer communication. For pet stores offering a wide range of pet products, having a website enables online sales, access to the product catalog, consultations, and convenient product search

Object of this thesis is the customer service process at a pet store.

Subject of this thesis is the methods and tools for developing a web application to automate sales and manage the product catalog of a pet store.

Purpose of this project is to develop a web application for a pet store to improve the customer service process.

The qualification work consists of an introduction, 4 sections, conclusions and a list of references.

In the introduction In the introduction, a detailed description of the relevance of the topic is provided, the object and subject of the study are defined, and the objective is outlined.

The first chapter analyzed the subject area and existing similar pet store websites, examined their pros and cons, and outlined the next steps in the development of the web application.

The second chapter analyzes the current state of the field, justifies the choice of technologies, and specifies the requirements for the web application.

The third chapter is devoted to the design of the web application, specifically the development of the site structure and mockups, the database ER diagram, UML diagrams, and describes the choice of implementation technologies.

The fourth chapter describes the website implementation process, the software component, the user interface, and the testing of the developed application.

In the conclusions the results of the completed work are summarized, the achievement of the set objective is determined, and possible directions for further system development are outlined.

The qualification work is presented on _86_ pages of typewritten text, consists of an introduction, __ sections, general conclusions, a list of references with _18_ titles and _1_ appendices. The work contains _3_ tables and _23_ figures.

Keywords: Pet store, web development, PHP, Laravel, Microsoft SQL Server.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

ПЗ – Програмне забезпечення

HTML – Hyper Text Markup Language

CSS – Cascading Style Sheets

UML – Unified Modeling Language

MVC – Model-View-Controller

ВСТУП

У сучасних умовах розвитку електронної комерції вебсайти є важливим каналом продажу та комунікації з клієнтами. Для зоомагазинів, які пропонують широкий асортимент товарів для домашніх тварин, наявність сайту дозволяє забезпечити онлайн-продаж, ознайомлення з каталогом, консультації та зручний пошук продукції.

Науково-практичне значення розробки полягає у створенні ефективного інструменту для автоматизації продажів та взаємодії з клієнтами, підвищення рівня обслуговування й оптимізації управління асортиментом.

Метою роботи є розробка вевебзастосунку зоомагазину для підвищення ефективності продажів і забезпечення зручності обслуговування клієнтів.

Відповідно до мети визначено такі завдання:

1. Проаналізувати предметну область та існуючі сайти зоомагазинів для виявлення переваг і недоліків;
2. Обґрунтувати вибір технологій (HTML, CSS, JavaScript, PHP, PHPMailer, LiqPay) та сформувати специфікацію вимог;
3. Визначити функціональні вимоги до сайту (каталог товарів, корзина, онлайн-оплата, контактна інформація);
4. Спроєктувати структуру сайту, інформаційну модель сторінок, ER-діаграму бази даних та UML-діаграми;
5. Створити прототип інтерфейсу користувача з урахуванням ергономіки;
6. Реалізувати основний функціонал (каталог, кошик, оформлення замовлень, адміністративну панель);
7. Провести тестування сайту на відповідність вимогам та зручність для клієнтів і персоналу.

Об’єкт дослідження: процес продажу товарів у зоомагазині та інформаційні технології, що його підтримують.

Предмет дослідження: методи та засоби створення веб-сайту для автоматизації продажів і управління каталогом товарів зоомагазину.

Обґрунтування необхідності розробки

Аналіз сучасного стану показує, що більшість зоомагазинів мають обмежені або застарілі вебсайти, які не забезпечують зручний онлайн-продаж і повний каталог товарів. Існує потреба у створенні сайту, що поєднує каталог, кошик, онлайн-оплату та контактну інформацію. Провідні розробники та фахівці з електронної комерції рекомендують інтеграцію сучасних технологій для забезпечення зручності користувачів та ефективності продажів. Світові тенденції свідчать про зростання попиту на інтернет-магазини та онлайн-сервіси для зоотоварів.

Розробка сайту передбачає використання адаптивних, інтерактивних технологій та інтеграцію з платіжними системами. Структурований каталог і кошик забезпечать ефективне управління товарами й полегшать процес покупки.

Результати розробки можуть бути використані в зоомагазинах для організації онлайн-продажів, управління асортиментом, ведення бази клієнтів та підвищення рівня обслуговування. Високий рівень безпеки транзакцій сприяє формуванню довіри між учасниками ринку.

Особливої цінності набуває створення вебзастосунків, які дозволяють ефективно поєднувати продавців і покупців товарів для домашніх тварин у єдиному віртуальному середовищі. Завдяки сучасним веб-технологіям такі платформи забезпечують зручний інтерфейс, високу швидкість роботи та можливості для масштабування функціоналу. Це відкриває нові шляхи для розвитку електронної комерції в сегменті зоотоварів, а також сприяє цифровій трансформації традиційних зоомагазинів.

Актуальність розробки також зумовлена підвищеною потребою малого та середнього бізнесу в онлайн-середовищі через глобалізацію та розвиток логістичних сервісів. Такі застосунки дають змогу адміністраторам керувати асортиментом, а споживачам – швидко порівнювати товари, переглядати відгуки та відстежувати замовлення.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ІСНУЮЧИХ АНАЛОГІВ ВЕБСАЙТІВ ВЕТЕРИНАРНИХ КЛІНІК

1.1 Аналіз предметної області

Сьогодні важко уявити будь-яку сферу діяльності без використання цифрових інструментів. Торгівля товарами для тварин не є винятком – звичайні зоомагазини дедалі частіше переходять в онлайн-простір або доповнюють офлайн-роботу сайтами та застосунками. У сучасних умовах розвитку електронної комерції вебсайти є одним із важливих відгалужень для продажу та комунікації з клієнтами. Для зоомагазинів, які пропонують широкий асортимент товарів для домашніх тварин, наявність сайту дозволяє забезпечити онлайн-продаж, ознайомлення з каталогом товарів, консультації та зручний пошук продукції.

Разом з тим кількість товарів для тварин постійно збільшується – корми, аксесуари, ветеринарні препарати, засоби гігієни. У великому асортименті без структурованого підходу легко заплутатись. Тому виникає потреба у створенні такої інформаційної системи, яка б поєднувала в собі одразу кілька напрямків роботи зоомагазину: каталог товарів, оформлення покупок, облік постійних клієнтів, контроль наявності на складі, а також формування звітів для власника. Система повинна бути гнучкою, щоб її можна було з часом розширювати під нові завдання, і водночас достатньо простою в користуванні для персоналу без спеціальної технічної підготовки.

Варто зазначити, що звичайний фізичний магазин, навіть якщо він має досить великий асортимент, стикається з обмеженнями: площа приміщення, кількість персоналу, неможливість одночасно обслужити великий потік покупців. До того ж паперовий чи табличний облік товарних залишків, замовлень та клієнтів часто призводить до помилок, затримок і втрати актуальної інформації. У той же час сучасні веб-системи здатні не тільки показати весь асортимент із зручною фільтрацією, а й автоматично опрацьовувати замовлення, вести історію

покупок, формувати звіти та нагадувати про себе клієнтам через email чи push-сповіщення.

Також не варто забувати про надійність роботи. Якщо система «ляже» в момент оформлення замовлення або дані про залишки товару будуть показані невірно, це одразу призводить до втрати покупців і фінансових збитків. Тому вже на етапі проектування потрібно закласти механізми захисту інформації, перевірки коректності введених даних та швидкого відновлення роботи після збоїв. Усе це допоможе викликати довіру до магазину з боку відвідувачів.

Окремо слід сказати про те, що ринок зоотоварів постійно розширюється, з'являються нові бренди, спеціалізовані лікувальні корми, різноманітні аксесуари. Тому звичайному покупцеві все важче зорієнтуватися у всьому цьому різноманітті. Звідси випливає потреба в такому вебзастосунку , де б людина могла за лічені хвилини підібрати потрібний товар за категорією, типом тварини, ціною чи брендом, прочитати відгуки інших покупців і одразу ж оформити замовлення. Для власника магазину ж на перший план виходить можливість керувати товарними позиціями, обробляти заявки, відслідковувати фінансові показники та формувати лояльність клієнтів через систему знижок.

Крім суто практичного значення, розробка подібної системи має й науково-практичну цінність. Під час її створення доводиться вирішувати типові для сучасної веб-інженерії завдання: проєктування реляційної бази даних, організація безпечної авторизації, інтеграція з платіжними сервісами, побудова адаптивного інтерфейсу. Таким чином, отримані результати можуть бути використані як шаблон для аналогічних інтернет-магазинів невеликого та середнього бізнесу.

Таким чином, актуальність теми обумовлена необхідністю підвищення ефективності обслуговування клієнтів зоомагазину шляхом впровадження сучасного вебзастосунку , який забезпечує автоматизацію продажів, зручне управління каталогом товарів та онлайн-взаємодію з покупцями.

Об’єктом дослідження є процес обслуговування клієнтів зоомагазину.

Предметом дослідження є методи та засоби створення вебзастосунку для автоматизації продажів та управління каталогом товарів зоомагазину.

Метою роботи є розробка вебзастосунку зоомагазину для покращення процесу обслуговування клієнтів.

Науково-практичне значення роботи полягає в тому, що розроблений вебзастосунок може бути безпосередньо впроваджений у діяльність зоомагазинів для організації онлайн-продажів, а також використаний у навчальному процесі як приклад сучасного веб-проєкту з повним циклом розробки – від аналізу вимог до тестування та документування.

1.2 Аналіз існуючих аналогічних рішень

Перш ніж розпочати проєктування власної системи, було вивчено кілька чинних вебсайтів, що представляють собою готові рішення для торгівлі зоотоварами в українському сегменті інтернету. Це дозволило виявити їхні сильні сторони, а також недоліки, які варто усунути у власній розробці. Розглянемо найбільш показові приклади.

Таблиця 1.1 – Програмний аналог ZooBonus [1]

Розробник	зоомагазин/мережа ZooBonus (Україна)
Архітектура	3-tier web application (веб-магазин з сервером, базою даних та клієнтським UI)
Мови реалізації	HTML/CSS, JavaScript, PHP
Перелік функцій	1. Каталог товарів із категоріями (для собак, котів, гризунів тощо) 2. Пошук і фільтрація товарів 3. Онлайн-замовлення товарів 4. Можливість доставки по Україні 5. Консультації консультантів (онлайн/офлайн) 6. Знижки, акції та бонусна система

Кінець таблиці 1.1

Переваги	1. Великий асортимент зоотоварів 2. Можливість замовлення онлайн з доставкою 3. Інформація з детальними описами товарів 4. Стабільний імідж бренду з мережею фізичних магазинів
Недоліки	1. Обмежений функціонал щодо персональних профілів та рекомендацій 2. Немає розширеної аналітики купівельної поведінки 3. Відсутність сучасних персоналізованих пропозицій

Як видно з наведених даних, ZooBonus пропонує доволі зручний каталог із розбивкою за видами тварин, дає змогу оформити доставку по країні та має базову програму лояльності. Водночас ця система майже не використовує інформацію про поведінку користувачів для формування персональних рекомендацій, а звітність для управління бізнесом реалізована скоріше на базовому рівні. Крім того, профіль покупця обмежений мінімальним набором даних, що не дозволяє повноцінно сегментувати клієнтську базу.

Таблиця 1.2 – Програмний аналог leyven.com.ua [2]

Розробник	leyven.com.ua
Архітектура	3-tier web application
Мови реалізації	HTML/CSS, JavaScript, серверна технологія
Перелік функцій	1. Онлайн-каталог з товарами для тварин 2. Доставка по Україні 3. Профіль користувача з історією замовлень 4. Система знижок та новинки 5. Оцінки та відгуки клієнтів

Кінець таблиці 1.2

Переваги	1. Зручний каталог товарів із фільтрами 2. Блог із додатковою інформацією для клієнтів 3. Можливість перегляду відгуків та рейтингів
Недоліки	1. Обмежена інтеграція з CRM та маркетинговою аналітикою 2. Деякі функції оформлення замовлення не дуже інтуїтивні

Цей сайт вигідно відрізняється наявністю блогу з корисними порадами, а також можливістю залишати відгуки, що позитивно впливає на довіру відвідувачів. Профіль користувача тут дещо багатший, ніж у попереднього аналога, адже зберігається історія замовлень. Однак і тут спостерігається недостатня робота з аналітикою та маркетинговими інструментами: немає тісної інтеграції з зовнішніми системами управління взаємовідносинами з клієнтами, а деякі кроки оформлення покупки, за відгуками користувачів, викликають труднощі.

1.3 Аналіз потреб користувачів та постановка задач

Користувачі сучасних вебзастосунків для зоомагазинів висувають підвищені вимоги до швидкодії, безпеки та зручності користування платформою. Ці чинники не лише впливають на загальне враження від сервісу, але й відіграють ключову роль у забезпеченні лояльності клієнтів, а також у залученні нових користувачів.

Швидкодія платформи є одним із вирішальних аспектів для ефективного функціонування онлайн-зоомагазину. Користувачі очікують миттєвого завантаження сторінок каталогу, оперативного пошуку товарів за назвою, категорією або типом тварини, а також швидкого оформлення замовлення. Будь-які затримки або перебої в роботі системи можуть негативно позначитися на користувацькому досвіді й призвести до втрати довіри з боку клієнтів, особливо в

моменти пікового навантаження (наприклад, під час сезонних розпродажів або акцій). Отже, платформи повинні бути технічно оптимізованими для стабільної роботи навіть при значному навантаженні, а пошук і фільтрація мають повертати результати за 1–2 секунди.

Безпека – ще один пріоритетний параметр для користувачів онлайн-магазинів. Власники домашніх тварин, купуючи корми, аксесуари або ветеринарні препарати, довіряють платформі свої персональні дані, адресу доставки та платіжну інформацію. Забезпечення захисту цих даних, конфіденційності інформації та безпеки фінансових транзакцій є обов'язковою умовою функціонування будь-якого сучасного вебзастосунку для зоомагазину. Використання сучасних методів шифрування (HTTPS, TLS), дотримання стандартів безпеки платіжних систем (PCI DSS), а також створення механізмів захисту від SQL-ін'єкцій, XSS-атак та CSRF формує довіру до платформи. Особливо критично це для онлайн-оплат через платіжні шлюзи, такі як LiqPay, де паролі мають зберігатися в зашифрованому вигляді, а всі дані передаватися через захищене з'єднання.

Зручність використання визначається не лише якістю графічного інтерфейсу, але й функціональною наповненістю системи. Для зоомагазину важливими елементами є ефективна система пошуку і фільтрації (за ціною, брендом, типом тварини, категорією товару), можливість додавання товарів до списку обраних, легкий доступ до детальних описів товарів, відгуків і рейтингів, а також простота процедури оформлення замовлення з вибором способу доставки та оплати. Наявність персонального кабінету з історією замовлень та можливістю відстеження статусу сприяє підвищенню рівня задоволеності користувачів. Для адміністратора ж важливою є наявність зручної панелі керування для управління товарами, категоріями, замовленнями та перегляду аналітичних звітів.

Сам процес розробки вебзастосунку для зоомагазину можна поділити на кілька ключових етапів, кожен з яких є критичним для успішної реалізації проєкту.

Перший етап – це попереднє планування та аналітика, на якому здійснюється аналіз вимог до проєкту, визначення основних функцій (каталог товарів, кошик, оформлення замовлень, адміністративна панель, система оплати, відгуки, звіти) та технологічних аспектів, що забезпечать успішну реалізацію вебзастосунку . Під час цього етапу важливо сформулювати чіткі цілі, а також створити технічне завдання, яке визначить вимоги до всіх частин системи – від функціональних (реєстрація, фільтрація, управління товарами) до нефункціональних (продуктивність, надійність, безпека). Також на цьому етапі проводиться аналіз існуючих аналогів для виявлення їх переваг та недоліків .

Другим етапом є проєктування системи – розробка архітектури вебзастосунку , яка включає в себе вибір технологій для фронтенду (HTML, CSS, JavaScript) та бекенду (PHP, Laravel), розробку структури бази даних (ER-діаграма, Microsoft SQL Server). Це також включає в себе створення UML-діаграм (варіантів використання, класів, послідовності, діяльності) та розробку макетів (прототипів) інтерфейсу користувача, які повинні бути інтуїтивно зрозумілими і зручними для всіх категорій користувачів: гостей, зареєстрованих клієнтів, менеджерів та адміністраторів.

Третій етап – розробка прототипу та первісна реалізація. На цьому етапі відбувається створення основних компонентів вебзастосунку зоомагазину, таких як система реєстрації та аутентифікації користувачів, каталог товарів з категоріями, система фільтрації та пошуку, кошик, інтерфейс оформлення замовлень, адміністративна панель для управління товарами та замовленнями. Важливо, щоб прототип був повноцінним з точки зору основних функцій, навіть якщо дизайн інтерфейсу ще знаходиться на стадії доопрацювання. На цьому етапі також реалізується інтеграція з платіжним шлюзом LiqPay за допомогою PHP SDK.

Четвертим етапом є інтеграція та тестування. Після того, як основні компоненти розроблені, їх необхідно інтегрувати в єдину систему. Проводяться тести на функціональність (чи працюють усі заявлені функції), сумісність

(коректне відображення на різних пристроях та в різних браузерах), безпеку (перевірка захисту від SQL-ін'єкцій, XSS, CSRF) та продуктивність. Це етап, на якому важливо виявити всі потенційні проблеми і помилки, а також оцінити ефективність роботи вебзастосунку під навантаженням (імітація одночасної роботи кількох десятків користувачів).

Після інтеграції та тестування наступає етап оптимізації та підготовки до запуску. Після виявлення та усунення всіх помилок, система оптимізується для підвищення продуктивності – оптимізуються запити до бази даних, зменшується час завантаження сторінок, налаштовується кешування. На цьому етапі проводяться останні перевірки, створюється керівництво користувача з покроковими скриншотами, і вебзастосунок готовий до запуску на реальному середовищі.

Останній етап – підтримка та оновлення. Після запуску вебзастосунок не повинен залишатися незмінним. Регулярне оновлення функціоналу (додавання нових категорій товарів, акцій, знижок), виправлення помилок, а також поліпшення ефективності через додавання нових можливостей (наприклад, системи рекомендацій товарів на основі історії покупок, push-повідомлень про зміну статусу замовлення) і покращення зручності користування є важливими для підтримки конкурентоспроможності платформи в майбутньому.

Висновки до розділу 1

У першому розділі кваліфікаційної роботи було проведено комплексний аналіз предметної області інтернет-торгівлі зоотоварами, визначено актуальність теми, досліджено існуючі програмні рішення, а також сформовано основні задачі, які необхідно вирішити для створення вебзастосунку зоомагазину.

Актуальність теми обґрунтовано стрімким розвитком електронної комерції в Україні, зростанням попиту на онлайн-покупки товарів для домашніх тварин та необхідністю автоматизації бізнес-процесів зоомагазинів (управління каталогом, обробка замовлень, облік клієнтів, формування звітів). Визначено, що сучасні

вебсистеми здатні не лише демонструвати асортимент, а й забезпечувати зручну фільтрацію, онлайн-оплату, автоматичні сповіщення та аналітику, що підвищує ефективність роботи магазину та якість обслуговування клієнтів. Сформульовано об’єкт (процес обслуговування клієнтів зоомагазину), предмет (методи та засоби створення вебзастосунку для автоматизації продажів) та мету роботи (розробка вебзастосунку для покращення обслуговування клієнтів).

Аналіз існуючих аналогічних рішень (ZooBonus, leyven.com.ua) дозволив виявити їхні сильні сторони (зручний каталог, доставка, відгуки, бонусні системи) та суттєві недоліки: обмежений функціонал персональних профілів, відсутність розширеної аналітики купівельної поведінки, слабка інтеграція з CRM-системами, недостатньо інтуїтивне оформлення замовлень. Ці недоліки підтвердили доцільність створення власного вебзастосунку , який би їх усував.

Аналіз потреб користувачів показав, що ключовими вимогами до сучасного вебзастосунку зоомагазину є висока швидкодія, надійний захист персональних даних і фінансових транзакцій, а також зручність використання (фільтрація, персональний кабінет, історія замовлень, відгуки). На основі цього визначено етапи розробки (планування, проєктування, реалізація, тестування, оптимізація, підтримка) та сформульовано конкретні задачі, які необхідно вирішити для досягнення мети: розробка структури бази даних; реалізація реєстрації та розмежування ролей; створення каталогу з фільтрацією та пошуком; реалізація кошика, оформлення замовлень та інтеграція з платіжним шлюзом LiqPay; розробка особистого кабінету з історією замовлень та відгуками; створення адміністративної панелі для управління товарами, замовленнями та формування звітів з візуалізацією; організація автоматичних email-сповіщень; тестування та створення керівництва користувача.

Отже, результати виконаного в першому розділі аналізу створюють необхідне підґрунтя для подальшого вибору технологій, специфікації вимог до програмного забезпечення та безпосереднього проєктування і реалізації вебзастосунку .

2 АНАЛІЗ СУЧАСНОГО СТАНУ, ОБҐРУНТУВАННЯ ВИБОРУ ТЕХНОЛОГІЙ ТА СПЕЦИФІКАЦІЯ ВИМОГ ДО ВЕБЗАСТОСУНКУ ВЕТЕРИНАРНОЇ КЛІНІКИ

2.1 Аналіз сучасного стану розробки вебзастосунків для зоомагазинів

За останні роки інтернет-торгівля в Україні демонструє стрімке зростання. Все більше підприємців, зокрема власників зоомагазинів, переходять від традиційних фізичних точок продажу до гібридних бізнес-моделей, що включають онлайн-компонент. Це значно впливає на розвиток ринку платформ для створення інтернет-магазинів, формуючи певні стандарти у їхній архітектурі.

Сучасні веб-додатки для електронної комерції здебільшого ґрунтуються на трирівневій архітектурі. Така модель забезпечує чітке розмежування між рівнями представлення даних (front-end), бізнес-логіки (back-end) та зберігання інформації (база даних). Завдяки цьому можна окремо оновлювати інтерфейс користувача і серверну частину, а також легко масштабувати систему при збільшенні користувачів. Для реалізації front-end часто використовуються технології HTML, CSS і JavaScript. Зокрема, JavaScript дозволяє динамічно оновлювати веб-сторінки без необхідності повного перезавантаження, що забезпечує більш комфортну взаємодію користувачів із продуктами, наприклад, каталогами та кошиками.

Щодо back-end, PHP залишається однією з найпоширеніших мов програмування для малих і середніх проєктів. Її популярність пояснюється доступністю для новачків, широким набором бібліотек і можливістю інтеграції з численними базами даних. Важливо й те, що PHP відповідає вимогам недорогих хостингових послуг, що робить її привабливою для малого бізнесу. У комбінації з JavaScript на стороні клієнта ця технологія надає змогу реалізувати майже будь-який потрібний функціонал для онлайн-зоомагазинів.

Ще одним важливим напрямком розвитку веб-додатків є обробка та візуалізація даних. Підприємцям потрібно не лише зберігати інформацію про продажі, але й швидко отримувати аналітичні звіти. У таких випадках

адміністративні панелі все частіше інтегрують JavaScript-бібліотеки для створення графіків і діаграм у реальному часі без необхідності додаткового програмного забезпечення. Одним із популярних рішень є бібліотека Chart.js, яка дозволяє легко візуалізувати дані через HTML та Canvas у вигляді різних діаграм. Її основні переваги – малий розмір, простота налаштування та підтримка анімацій, що сприяють зручному аналізу інформації для адміністраторів системи.

Таким чином, сучасний ринок пропонує безліч технологічних рішень для створення ефективних інтернет-магазинів як для зоорітейлу, так і для інших сфер. Вибір інструментів має бути ретельно продуманим і враховувати вимоги до функціоналу, рівня безпеки та довгострокового обслуговування системи.

2.2 Обґрунтування вибору технологій реалізації вебзастосунку зоомагазину

Фронтенд (HTML, CSS, JavaScript). Якість інтерфейсу безпосередньо впливає на конверсію інтернет-магазину. Сучасні стандарти HTML5 та CSS3 дозволяють створювати адаптивні сайти, що критично для мобільних покупців. JavaScript забезпечує динамічне оновлення кошика та асинхронне завантаження даних без перезавантаження сторінки. Як зазначається в журналі London Journal of Research in Computer Science and Technology [3], використання сучасних фронтенд-технологій є основою для побудови ефективних e-commerce рішень.

Серверна логіка (PHP) [6]. Для малих та середніх проєктів, таких як зоомагазин, PHP залишається однією з найдоцільніших мов завдяки доступності хостингу, великій кількості готових бібліотек та легкому старту. У збірнику матеріалів конференції FOSS-2025 [5] на прикладі розробки онлайн-кінотеатру показано, що вибір серверної мови та фреймворку має визначатися масштабом проєкту та вимогами до продуктивності. Для зоомагазину, який не потребує надскладних обчислень, PHP є оптимальним рішенням. Крім того, згадана робота [5] окремо підкреслює важливість інтеграції платіжних систем (Stripe, PayPal, LiqPay) для забезпечення зручної оплати, що ми й реалізуємо.

Візуалізація даних (Chart.js) [7]. Адміністратору магазину необхідно аналізувати продажі та динаміку замовлень. Легковага бібліотека Chart.js дозволяє створювати інтерактивні графіки та діаграми безпосередньо у веб-інтерфейсі. У ґрунтовному огляді Addepalli L. et al., 2023 [5] проведено порівняльний аналіз сучасних інструментів візуалізації даних. Автори відзначають, що Chart.js вирізняється простотою налаштування, малим розміром та підтримкою анімації, що робить його ідеальним вибором для адміністративних панелей e-commerce проєктів, де не потрібна надмірна складність, але важлива швидкість роботи.

Електронні сповіщення (PHPMailer). Для автоматичного інформування клієнта про статус замовлення необхідний надійний засіб надсилання електронних листів. Використання стандартної функції mail() у PHP є небезпечним через відсутність вбудованої SMTP-автентифікації та шифрування. PHPMailer, навпаки, підтримує SMTP та TLS, що забезпечує захист від перехоплення листування. У статті зі збірника наукових праць ХНУРЕ [4] розглядаються сучасні підходи до розробки вебзастосунків, зокрема наголошується на необхідності використання спеціалізованих бібліотек для роботи з електронною поштою як стандарту безпеки.

Платіжний шлюз (LiqPay). Приймання онлайн-платежів є ключовою функцією інтернет-магазину. LiqPay – це перевірений український платіжний сервіс, який надає офіційний PHP SDK, що спрощує інтеграцію та гарантує безпеку транзакцій. У роботі Булатов О.С., Гайдаєнко О.В. [5] прямо зазначено, що «інтеграція платіжних систем (Stripe, PayPal, LiqPay) дозволяє реалізувати зручну оплату». Це підтверджує актуальність вибору LiqPay для українського e-commerce. Використання готового SDK дозволяє уникнути типових помилок при роботі з фінансовими даними та забезпечити відповідність вимогам PCI DSS.

Таким чином, обраний стек технологій, обґрунтований у проаналізованих наукових джерелах, дозволяє створити повнофункціональний, безпечний та

зручний вебзастосунок зоомагазину з урахуванням сучасних вимог до e-commerce проєктів..

2.3 Специфікація вимог до програмного забезпечення

Моделювання предметної області є важливим етапом розробки програмного забезпечення, оскільки дозволяє формалізувати основні бізнес-процеси, визначити структуру системи та встановити взаємозв’язки між її компонентами. У рамках даної кваліфікаційної роботи моделювання здійснюється для процесів управління каталогом товарів, замовленнями, кошиком, оплатою, користувачами та відгуками у вебзастосунку зоомагазину.

Основними акторами системи є гість, клієнт (зареєстрований покупець), менеджер, адміністратор та партнер (обмежений доступ до управління товарами). Гість переглядає каталог, виконує пошук та фільтрацію товарів. Клієнт додає товари до кошика, оформлює замовлення, здійснює онлайн-оплату, переглядає історію покупок, залишає відгуки та редагує свій профіль. Менеджер працює з адміністративною панеллю: керує замовленнями (змінює статуси), переглядає звіти. Адміністратор має повний доступ: управління товарами, категоріями, користувачами, блогом, знижками, генерацію звітів.

У межах моделювання визначаються основні сутності системи, серед яких користувачі (з ролями), профілі клієнтів, товари, категорії, кошик, замовлення, статуси замовлень, способи доставки, платежі, відгуки, блог, знижки та акції. Для кожної сутності визначаються атрибути та зв’язки, що дозволяє сформувати логічну модель бази даних.

1) Призначення та межі проєкту

1.1) Призначення системи

Вебзастосунок призначений для автоматизації роботи зоомагазину: надання користувачам можливості перегляду каталогу товарів, їх пошуку, фільтрації, додавання до кошика, оформлення замовлень з онлайн-оплатою,

відстеження статусу замовлень, написання відгуків, а також для адміністрування товарів, категорій, замовлень та генерації звітів.

1.2) Погодження, що ухвалені в програмній документації

Узгоджено, що система має працювати в середовищі веббраузера, використовувати клієнт-серверну архітектуру, забезпечувати розмежування ролей (гість, клієнт, менеджер, адміністратор, партнер) та інтеграцію з платіжним шлюзом LiqPay.

1.3) Межі проєкту ПЗ

Система не включає мобільний додаток, розрахована на роботу через вебінтерфейс. Не передбачено синхронізацію із зовнішніми складами або ERP-системами. Управління доставкою обмежується вибором способу доставки, без інтеграції з логістичними службами.

2) Загальний опис

2.1) Сфера застосування

Вебзастосунок використовується для продажу товарів для тварин (корми, іграшки, аксесуари, ліки), інформування власників тварин через блог, накопичення відгуків та оцінок, обробки замовлень в онлайн-режимі.

2.2) Характеристики користувачів

Гість – неавторизований відвідувач, має доступ до перегляду каталогу, пошуку, фільтрації, блогу, акцій.

Клієнт – зареєстрований користувач, додатково може редагувати профіль, додавати товари до кошика, оформлювати замовлення, оплачувати, писати відгуки, переглядати історію покупок та статус замовлень.

Менеджер – відстежує статуси замовлень, може їх змінювати; має доступ до адміністративної панелі (обмежено).

Адміністратор – повний доступ до управління товарами, категоріями, користувачами, замовленнями, генерації звітів.

Партнер – обмежене управління товарами (додавання, редагування, видалення власних товарів).

2.3) Загальна структура і склад системи

Система складається з: фронтенд-частини (адаптивний вебінтерфейс), бекенд-частини (серверна логіка, обробка запитів, робота з БД), бази даних (користувачі, товари, категорії, замовлення, відгуки), платіжного шлюзу LiqPay, сервісу надсилання email (PHPMailer).

2.4) Загальні обмеження

Час завантаження сторінок – до 4 секунд (до 2 секунд для пошуку/фільтрації).

Використання HTTPS, шифрування паролів, захист від SQL-ін’єкцій, XSS, CSRF.

Підтримка одночасної роботи декількох десятків користувачів.

Адаптивний дизайн для мобільних пристроїв.

3) Функції системи

3.1) Реєстрація та авторизація користувачів

Опис функції – забезпечення створення облікового запису та входу в систему для гостей, клієнтів, менеджерів, адміністраторів.

Вхідна інформація – логін/email, пароль, реєстраційні дані (ім’я, контакти).

Вихідна інформація – підтвердження реєстрації, сесійний токен, перенаправлення на особистий кабінет.

3.2) Профіль користувача з можливістю редагування даних

Опис – клієнт може переглядати та змінювати свої персональні дані (контакти, адреса, пароль).

Вхідна інформація – нові значення полів профілю.

Вихідна інформація – оновлений профіль, повідомлення про успішне збереження.

3.3) Перегляд каталогу товарів із категоріями

Опис – відображення списку товарів, згрупованих за категоріями.

Вхідна інформація – вибрана категорія, параметри пагінації.

Вихідна інформація – набір товарів (з фото, назвою, ціною).

3.4) Фільтрація та пошук товарів за назвою, ціною, категорією

Опис – надання користувачеві інструменту пошуку та фільтрації товарів.

Вхідна інформація – пошуковий запит, діапазон цін, вибрані категорії.

Вихідна інформація – відфільтрований список товарів.

3.5) Перегляд детальної інформації про товар

Опис – відображення повної інформації про товар (опис, фото, ціна, наявність).

Вхідна інформація – ідентифікатор товару.

Вихідна інформація – детальна картка товару.

3.6) Додавання товарів у корзину

Опис – клієнт може додавати товари у віртуальний кошик перед оформленням замовлення.

Вхідна інформація – ID товару, кількість.

Вихідна інформація – оновлений вміст корзини.

3.7) Оформлення замовлення з вибором способу доставки

Опис – формування замовлення на основі товарів у корзині з вибором доставки.

Вхідна інформація – адреса доставки, вибраний спосіб доставки, контактні дані.

Вихідна інформація – створене замовлення з унікальним номером.

3.8) Система оплати онлайн через LiqPay

Опис – інтеграція з платіжним шлюзом для проведення оплати замовлення.

Вхідна інформація – сума замовлення, платіжні дані клієнта (через LiqPay).

Вихідна інформація – статус оплати (успішно/неуспішно).

3.9) Підтвердження замовлення та відправлення сповіщення на email

Опис – після оформлення замовлення система генерує лист-підтвердження.

Вхідна інформація – email клієнта, дані замовлення.

Вихідна інформація – сповіщення на email через PHPMailer.

3.10) Відстеження статусу замовлення

Опис – клієнт та менеджер можуть переглядати поточний статус замовлення.

Вхідна інформація – номер замовлення або ID.

Вихідна інформація – статус (наприклад, «оплачено», «в обробці», «відправлено»).

3.11) Історія покупок

Опис – клієнт може переглядати всі свої попередні замовлення.

Вхідна інформація – ID клієнта.

Вихідна інформація – список замовлень з деталями.

3.12) Блог з порадами для власників тварин

Опис – розділ із статтями, доступний усім ролям.

Вхідна інформація – вибір статті.

Вихідна інформація – текст статті, дата публікації, автор.

3.13) Система оцінок та відгуків про товари

Опис – клієнт може залишити оцінку (зірочки) та текстовий відгук на придбаний товар.

Вхідна інформація – ID товару, оцінка, текст відгуку.

Вихідна інформація – опублікований відгук, оновлений рейтинг товару.

3.14) Знижки та акції (відображення на сторінках товарів)

Опис – система відображає спеціальні ціни та позначки акційних товарів.

Вхідна інформація – розмір знижки, термін дії акції (з БД).

Вихідна інформація – знижена ціна, мітка «Акція».

3.15) Поштові повідомлення через PHPMailer

Опис – система використовує PHPMailer для надсилання email-сповіщень.

Вхідна інформація – налаштування SMTP, тема, тіло листа, адресат.

Вихідна інформація – статус відправки (успішно/помилка).

3.16) Адміністративна панель

Опис – інтерфейс для адміністратора та менеджера з керування системою.

Вхідна інформація – облікові дані адміністратора/менеджера.

Вихідна інформація – доступ до функцій управління (товари, категорії, замовлення, звіти).

3.17) Управління товарами (додавання, редагування, видалення)

Опис – адміністратор та партнер (обмежено) можуть керувати товарами.

Вхідна інформація – атрибути товару (назва, ціна, опис, фото, категорія, кількість).

Вихідна інформація – оновлений каталог товарів.

3.18) Управління категоріями товарів

Опис – адміністратор створює, редагує або видаляє категорії.

Вхідна інформація – назва категорії, опис, батьківська категорія.

Вихідна інформація – оновлене дерево категорій.

3.19) Генерація звітів із візуалізацією через Chart.js

Опис – адміністратор може отримувати звіти (продажі, популярні товари) у вигляді графіків.

Вхідна інформація – період, тип звіту.

Вихідна інформація – діаграма (Chart.js) та таблиця даних.

3.20) Інтеграція з платіжним шлюзом LiqPay

Опис – система взаємодіє з LiqPay для обробки платежів.

Вхідна інформація – ключі мерчанта, дані замовлення, сума.

Вихідна інформація – підтвердження платежу, callback-повідомлення.

4) Вимоги до інформаційного забезпечення

4.1) Джерела і зміст вхідної інформації (даних)

Реєстраційні дані користувачів.

Довідники товарів, категорій, акцій.

Замовлення, кошик, платіжні транзакції.

Відгуки, оцінки, статті блогу.

4.2) Нормативно-довідкова інформація

Класифікатори категорій товарів.

Довідник статусів замовлень.

Список способів доставки.

4.3) Вимоги до способів організації, збереження та ведення інформації

Резервне копіювання даних (щоденне).

Забезпечення цілісності (зовнішні ключі, транзакції).

Шифрування паролів (bcrypt/Argon2).

5) Вимоги до технічного забезпечення

Сервер: будь-яка платформа з підтримкою PHP ≥7.4 та вебсервера (Apache/Nginx).

Операційна система сервера: Linux (Ubuntu/CentOS) або Windows Server.

Дисковий простір: мінімум 20 ГБ (для БД, фото товарів).

Оперативна пам’ять: від 2 ГБ (з можливістю масштабування до 8 ГБ).

6) Вимоги до програмного забезпечення

6.1) Архітектура програмної системи

Трирівнева архітектура: клієнтська частина (HTML/CSS/JS), серверна частина (PHP/модель-контролер), рівень даних (БД).

Використання патерну MVC.

6.2) Системне програмне забезпечення

Вебсервер: Apache 2.4+ або Nginx 1.18+.

Інтерпретатор PHP 7.4/8.x.

Система керування БД: MSSQL.

6.3) Мережне програмне забезпечення

Протоколи HTTP/HTTPS, SMTP (для PHPMailer).

Наявність SSL-сертифіката.

6.4) Мова і технологія розробки ПЗ

Бекенд: PHP.

Фронтенд: HTML, CSS (адаптивний), JavaScript (ES6), Chart.js для графіків.

БД: MSSQL.

Бібліотеки: PHPMailer, LiqPay SDK.

7) Вимоги до зовнішніх інтерфейсів

7.1) Інтерфейс користувача

Адаптивний вебінтерфейс під десктопні, планшетні та мобільні екрани.

Інтуїтивна навігація: головна сторінка, каталог, кошик, особистий кабінет, блог, адмінпанель.

7.2) Апаратний інтерфейс

Не передбачено прямого підключення апаратних пристроїв (окрім стандартних: клавіатура, миша, сенсорний екран).

7.3) Програмний інтерфейс

REST-like API для взаємодії з фронтендом.

API LiqPay для платежів.

SMTP-інтерфейс через PHPMailer.

7.4) Комунікаційний протокол

HTTPS (TLS 1.2/1.3).

HTTP/2 за можливості.

8) Властивості програмного забезпечення

8.1) Доступність

Система цілодобово доступна, крім часу планового технічного обслуговування (повідомлення адміністратора).

8.2) Супроводжуваність

Модульна структура коду, коментування, дотримання стандартів PSR для PHP.

8.3) Переносимість

Кросплатформна: може працювати на будь-якій ОС, що підтримує PHP та вебсервер.

8.4) Продуктивність

Час завантаження сторінок ≤4 сек, пошук та фільтрація – 1-2 сек.

Підтримка до 50 одночасних користувачів без деградації.

8.5) Надійність

Коректна обробка помилок (перехоплення винятків, логування).

Відновлення після збоїв (автоматичний перезапуск сервісів).

Цілісність даних забезпечується транзакціями БД.

8.6) Безпека

Використання HTTPS, шифрування паролів (bcrypt).

Захист від SQL-ін’єкцій (підготовлені запити), XSS (екранування виводу), CSRF (токени).

Безпечна передача платіжних даних через LiqPay (без зберігання на сервері).

9) Інші вимоги

Відповідність законодавству про захист персональних даних (ЗУ «Про захист персональних даних»).

Документування API (для подальшої інтеграції).

Висновки до розділу 2

У другому розділі кваліфікаційної роботи було проведено аналіз сучасного стану розробки вебзастосунків для зоомагазинів, обґрунтовано вибір технологічного стеку для реалізації програмного продукту, а також сформовано специфікацію вимог до програмного забезпечення.

Аналіз сучасного стану показав, що ринок електронної комерції в Україні, зокрема сегмент зоотоварів, демонструє стрімке зростання. Сучасні вебзастосунки для інтернет-магазинів базуються на трирівневій архітектурі (front-end, back-end, база даних), що забезпечує гнучкість, масштабованість та зручність супроводу. Визначено, що для реалізації клієнтської частини домінуючими технологіями є HTML, CSS та JavaScript, а для серверної – PHP, який залишається популярним завдяки доступності, великій екосистемі та сумісності з недорогими хостинговими послугами. Окрему увагу приділено питанням безпеки (захист від SQL-ін'єкцій, XSS, CSRF) та візуалізації даних (бібліотека Chart.js), що є невід'ємними складовими сучасних інтернет-магазинів.

Обґрунтування вибору технологій дозволило сформувати наступний стек: HTML5/CSS3 для створення адаптивного інтерфейсу, JavaScript для забезпечення інтерактивності (динамічне оновлення кошика, фільтрація, валідація форм), PHP та фреймворк Laravel для реалізації серверної логіки та бізнес-процесів, Chart.js для візуалізації аналітичних звітів, PHPMailer для автоматичних email-сповіщень, а також LiqPay для приймання онлайн-платежів. Кожен із обраних засобів має наукове обґрунтування, що підтверджує його ефективність, надійність та доцільність використання саме для вебзастосунку зоомагазину.

Специфікація вимог до програмного забезпечення, які охоплюють усі ключові сценарії роботи системи: реєстрацію та авторизацію, управління профілем, каталог з фільтрацією та пошуком, кошик, оформлення замовлень, онлайн-оплату, відстеження статусу, історію покупок, відгуки, адміністративну панель (управління товарами, категоріями, генерація звітів) та інтеграцію з платіжним шлюзом. Також сформульовано нефункціональні вимоги: продуктивність (завантаження сторінок до 4 секунд, пошук до 2 секунд), надійність (коректна обробка помилок, відновлення після збоїв), масштабованість (підтримка десятків одночасних користувачів) та зручність використання.

3 ПРОЄКТУВАННЯ ТА РЕАЛІЗАЦІЯ

3.1 Вибір архітектури та технологій

У процесі створення сучасного вебзастосунку зоомагазину надзвичайно важливим є правильний вибір архітектури програмного забезпечення, оскільки саме від нього залежить масштабованість, розширюваність, надійність та зручність супроводу системи. Ретельне архітектурне планування дозволяє уникнути помилок на пізніших етапах життєвого циклу програмного продукту, забезпечити розділення обов'язків між підсистемами та оптимізувати взаємодію між модулями. Проєктування архітектури вебзастосунку зоомагазину має на меті забезпечити надійну й адаптивну структуру, що відповідає сучасним вимогам до електронної комерції.

У ході виконання роботи було прийнято рішення реалізувати застосунок на основі клієнт-серверної архітектури [8] з поділом на фронтенд та бекенд частини. Як архітектурний патерн для серверної логіки було обрано модель MVC, яка є основою фреймворку Laravel, а на клієнтській стороні застосовується класичний підхід із генерацією HTML на стороні сервера за допомогою динамічної взаємодії через JavaScript.

Клієнт-серверна архітектура є класичною і однією з найбільш поширених моделей побудови вебзастосунків.[9] Її суть полягає у розділенні системи на два автономні компоненти:

– клієнтська частина (frontend) – інтерфейс користувача, що функціонує у браузері (HTML, CSS, JavaScript);

– серверна частина (backend) – обробка бізнес-логіки, управління даними та взаємодія з базою даних (PHP, Laravel).

Цей підхід дозволяє забезпечити незалежність клієнтської та серверної частин, спрощує їх паралельну розробку та тестування, а також дозволяє масштабувати застосунок у разі збільшення навантаження. Крім того, обрана архітектурна модель дозволяє ефективно використовувати кешування, забезпечує

централізовану обробку запитів та полегшує реалізацію механізмів автентифікації й авторизації.

У даному проєкті використовується Laravel як основний фреймворк для реалізації серверної логіки [10]. Він формує HTML-сторінки за допомогою шаблонізатора Blade, обробляє HTTP-запити від браузера, реалізує бізнес-правила (розрахунок кошика, оформлення замовлення, обробка платежів), виконує авторизацію та автентифікацію користувачів. Сервер також здійснює валідацію введених даних, формує JSON-відповіді для AJAX-запитів та управляє доступом до бази даних. Для динамічної взаємодії на стороні клієнта використовується JavaScript [11], який дозволяє оновлювати кошик та фільтрувати товари без перезавантаження сторінки.

Система керування базами даних Microsoft SQL Server відповідає за збереження [12], запит та обробку інформації про товари, категорії, замовлення, користувачів, відгуки, кошики та інші сутності зоомагазину. Всі дані структуровано у вигляді взаємозв’язаних реляційних таблиць.

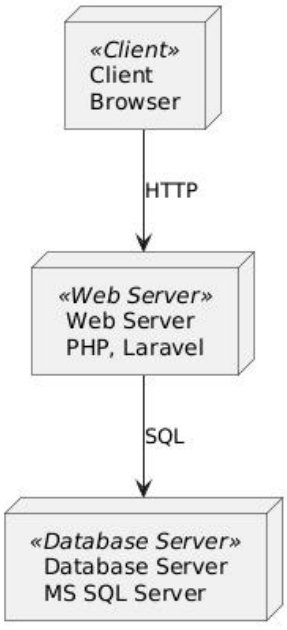

Рисунок 3.1 – Діаграма розгортання застосунку

Як проілюстровано діаграмою, браузерна частина надсилає запити через HTTP/HTTPS до веб-сервера. PHP-застосунок на базі Laravel обробляє запити,

перевіряє права доступу, звертається до бази даних через Eloquent ORM, обробляє результат і надсилає його клієнту (у вигляді HTML-сторінки або JSON). SQL Server отримує SQL-запити з сервера, виконує їх і повертає результати у вигляді наборів даних.

Архітектурний шаблон MVC в свою чергу дозволяє логічно розділити програму на три основні компоненти [13]. Модель (Model) відображає структуру даних, взаємодіє з базою даних та реалізує бізнес-логіку. Представлення (View) відповідає за відображення даних користувачеві (шаблони Blade). Контролер (Controller) приймає вхідні запити від клієнта, взаємодіє з моделлю та передає дані у представлення. Завдяки такому розділенню кожен компонент застосунку може розвиватися окремо, що позитивно впливає на підтримуваність та гнучкість проєкту. Також MVC забезпечує інтеграцію із зовнішніми сервісами (наприклад, платіжним шлюзом LiqPay), дозволяє зручно реалізовувати RESTful API [14]та підтримує тестування окремих частин застосунку..

3.2 Моделювання системи

Для забезпечення зручності та безпеки роботи вебзастосунку зоомагазину, у системі виділено кілька категорій користувачів (акторів), кожна з яких отримує лише ті можливості, що необхідні для виконання її завдань. Такий підхід дозволяє не перевантажувати інтерфейс зайвими елементами та захищає службову інформацію від несанкціонованого доступу.

На основі аналізу бізнес-процесів та побудованої діаграми варіантів використання (рис. 3.2) визначено наступні ролі:

1. Адміністратор – має найширші повноваження. Саме він додає нові товари та категорії, керує обліковими записами працівників, переглядає статистику продажів, формує звіти.

2. Зареєстрований користувач (клієнт) – основна аудиторія магазину. Може вільно переглядати каталог, додавати товари в корзину, оформлювати

замовлення, відстежувати їх виконання та переглядати історію своїх покупок, залишати відгуки.

3. Гість –незареєстрований відвідувач. Йому доступний лише перегляд товарів, проте для здійснення покупки потрібно створити обліковий запис.

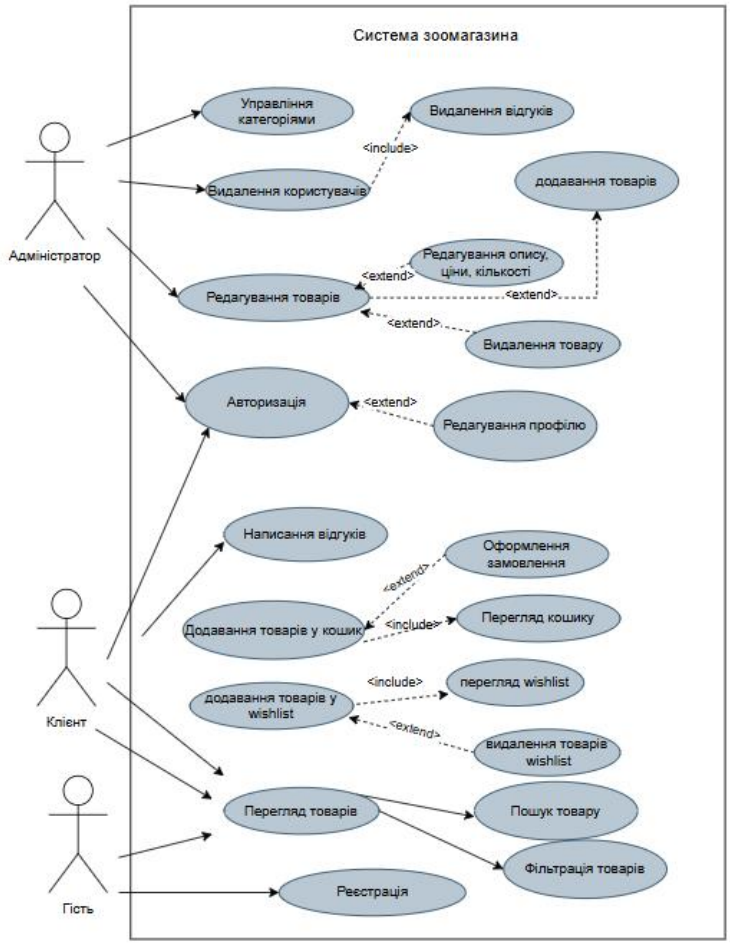

Рисунок 3.2 – Діаграма Use-case

На діаграмі відображено основні прецеденти та зв’язки між ними. Зокрема, актор «Адміністратор» виконує модерацію відгуків (що обов’язково включає видалення некоректних), модерацію товарів (включає додавання нових), видалення користувачів. Актор «Система» забезпечує авторизацію (включає перевірку наявності електронної пошти), додавання фото при створенні товару, додавання товару до кошика (з урахуванням відсутності авторизації), заповнення опису товару (опціонально) та розрахунок профілю (як розширення).

Нижче подано детальний опис ключових варіантів використання, узгоджений із діаграмою. Use Cases, які охоплюють основні сценарії роботи із системою.

Use Case 1: Реєстрація нового користувача

Use Case Name: Реєстрація нового користувача

Scope: Вебсистема інтернет-Зоомагазину

Level: User-goal

Primary Actor: Користувач (новий клієнт)

Stakeholders and Interests:

1. Користувач – хоче швидко зареєструватися та отримати доступ до магазину;
2. Адміністратор – контролює правильність реєстрацій та облік користувачів;

Preconditions:

1. Користувач відкрив вебсайт;
2. Має доступ до мережі Інтернет;
3. Вебсайт працює у звичайному режимі.

Success Guarantee (Postconditions):

1. Користувач успішно зареєстрований;
2. Дані користувача збережено у базі;
3. Користувач отримав доступ до особистого кабінету.

Main Success Scenario:

1. Користувач переходить на сторінку «Реєстрація»;
2. Вводить необхідні дані (ім’я, e-mail, пароль);
3. Система перевіряє правильність введених даних;
4. Користувач підтверджує реєстрацію (через e-mail чи SMS);
5. Система створює обліковий запис у базі;
6. Користувач отримує повідомлення про успішну реєстрацію.

Extensions (Alternative Scenarios):

1. Користувач ввів некоректний e-mail або пароль: система показує повідомлення про помилку;

2. Користувач вже зареєстрований: система пропонує увійти у свій обліковий запис.

Special Requirements:

1. Дані передавати через захищене з’єднання (HTTPS);

2. Паролі зберігати в зашифрованому вигляді;

3. Можливість одночасної реєстрації кількох користувачів.

Technology and Data Variations List:

1. Реєстрація може виконуватися через веббраузер або мобільний додаток;

2. Підтвердження може надсилатися e-mail або SMS.

Frequency of Occurrence: Часто – кожен новий користувач проходить реєстрацію.

Use Case 2: Перегляд каталогу та пошук товару

Use Case Name: Перегляд каталогу та пошук товару

Scope: Вебсистема інтернет-Зоомагазину

Level: User-goal

Primary Actor: Користувач (зареєстрований або гість)

Stakeholders and Interests:

1. Користувач – хоче швидко знайти потрібний товар;

2. Адміністратор – забезпечити зручну навігацію по каталогу;

3. Розробник – гарантувати стабільну роботу каталогу та пошуку.

Preconditions:

1. Вебсайт працює;

2. Каталог товарів заповнений;

3. Користувач має доступ до Інтернету.

Success Guarantee (Postconditions):

1. Користувач переглянув товари;

2. Можливість додати товар у корзину;

3. Дані про переглянуті товари збережено для статистики.

Main Success Scenario:

1. Користувач відкриває каталог товарів;

2. Вибирає категорію товару;

3. Використовує пошук або фільтри (ціна, бренд, тип тварини);

4. Переглядає деталі товару;

5. Додає обраний товар у корзину або список бажань.

Extensions:

1. Немає товарів за заданими критеріями: система повідомляє «Товари не знайдено»;

2. Некоректний запит пошуку : система пропонує альтернативні варіанти.

Special Requirements:

1. Пошук повинен повертати результати за 1-2 секунди;

2. Підтримка одночасної роботи декількох користувачів.

Use Case 3: Оформлення замовлення з оплатою

Use Case Name: Оформлення замовлення з оплатою

Scope: Вебсистема інтернет-Зоомагазину

Level: User-goal

Primary Actor: Користувач (зареєстрований)

Stakeholders and Interests:

1. Користувач – хоче швидко оплатити та отримати товар;

2. Адміністратор – контролює оплату та доставку;

3. Розробник – гарантує безпеку платежів та стабільність системи.

Preconditions:

1. Користувач додав товари до корзини;

2. Користувач авторизований;

3. Доступ до платіжного шлюзу.

Success Guarantee (Postconditions):

1. Замовлення оформлено;
2. Оплата пройшла успішно;
3. Створено запис замовлення у базі;
4. Користувач отримав підтвердження замовлення.

Main Success Scenario:

1. Користувач відкриває корзину;
2. Обирає спосіб доставки;
3. Обирає спосіб оплати (онлайн/готівка);
4. Підтверджує замовлення;
5. Система перевіряє платіж;
6. Замовлення створено та відправлено у обробку;
7. Користувач отримує підтвердження (e-mail або SMS).

Extensions:

1. Платіж не пройшов: система повідомляє про помилку і пропонує повторити;
2. Товар закінчився: система пропонує альтернативу або видаляє з корзини.

Use Case 4: Перегляд історії замовлень

Use Case Name: Перегляд історії замовлень

Scope: Вебсистема інтернет-Зоомагазину

Level: User-goal

Primary Actor: Користувач (зареєстрований)

Stakeholders and Interests:

1. Користувач – хоче перевірити попередні покупки;
2. Адміністратор – контролює стан замовлень;
3. Розробник – забезпечує швидкий доступ до історії замовлень.

Preconditions:

1. Користувач авторизований;
2. Є попередні замовлення у системі.

Success Guarantee (Postconditions):

1. Користувач переглянув замовлення;
2. Можливість повторного замовлення або скарги;
3. Дані збережені для статистики.

Main Success Scenario:

1. Користувач переходить до розділу «Мої замовлення»;
2. Система відображає список замовлень;
3. Користувач обирає конкретне замовлення;
4. Переглядає деталі (товари, статус, дату, суму);
5. Може повторити замовлення або залишити відгук.

Extensions:

Історія замовлень порожня – система повідомляє «Замовлень не знайдено».

Use Case 5: Залишити відгук про товар

Use Case Name: Залишити відгук про товар

Scope: Вебсистема інтернет-Зоомагазину

Level: User-goal

Primary Actor: Користувач (зареєстрований)

Stakeholders and Interests:

1. Користувач – хоче поділитися враженням про товар;
2. Адміністратор – контролює якість контенту;
3. Розробник – гарантує стабільну роботу функції відгуків.

Preconditions:

1. Користувач авторизований;
2. Товар було придбано користувачем.

Success Guarantee (Postconditions):

1. Відгук додано до бази;
2. Інші користувачі можуть переглядати відгук;
3. Адміністратор отримує повідомлення про новий відгук.

Main Success Scenario:

1. Користувач відкриває сторінку придбаного товару;
2. Обирає «Залишити відгук»;
3. Вводить оцінку та коментар;
4. Натискає «Надіслати»;
5. Система зберігає відгук та відображає підтвердження.

Extensions:

1. Відгук не відповідає правилам: адміністратор може відхилити;
2. Система тимчасово недоступна: повідомляє користувачу про помилку.

Важливим компонентом моделювання є діаграми активності, які відображають покрокову логіку виконання процесів у системі. Для зоомагазину такі діаграми мають особливу цінність, оскільки вони наочно ілюструють сценарії, що охоплюють декілька підсистем – наприклад, процес оформлення замовлення (рис. 3.3). У межах цього сценарію клієнт обирає товар, додає його до кошика та переходить до оформлення. На цьому етапі виконується перевірка: якщо кошик виявляється порожнім, клієнт повертається до каталогу товарів для його наповнення. Після успішної перевірки генерується замовлення та надсилається підтвердження. Такі діаграми не лише допомагають аналізувати логіку роботи системи, а й дозволяють вчасно виявити потенційні проблеми, зокрема на етапах перевірки або валідації даних. Вони також слугують основою для реалізації обробки виняткових ситуацій і забезпечення зручного користувацького досвіду.

Рисунок 3.3 – Діаграма активності замовлення товару

Також продемонстровано процес модерації контенту адміністратором системи (рис 3.4), ця діаграма активності відображає процес роботи адміністратора в системі зоомагазину. На початку користувач входить у систему. Якщо його роль не є "Admin", доступ забороняється. У разі успішної перевірки відображається панель керування адміністратора, де він обирає розділ для модерації. Діаграма демонструє дві паралельні гілки модерації:

Модерація товарів – адміністратор перевіряє, чи потребує товар оновлення. Якщо так – виконує оновлення, якщо ні – пропускає цей крок.

Модерація замовлень – адміністратор переглядає список замовлень. Якщо потрібне втручання (наприклад, зміна статусу, проблеми з доставкою), він змінює статус замовлення та зв'язується з клієнтом або службою доставки. Якщо втручання не потрібне – замовлення залишається без змін.

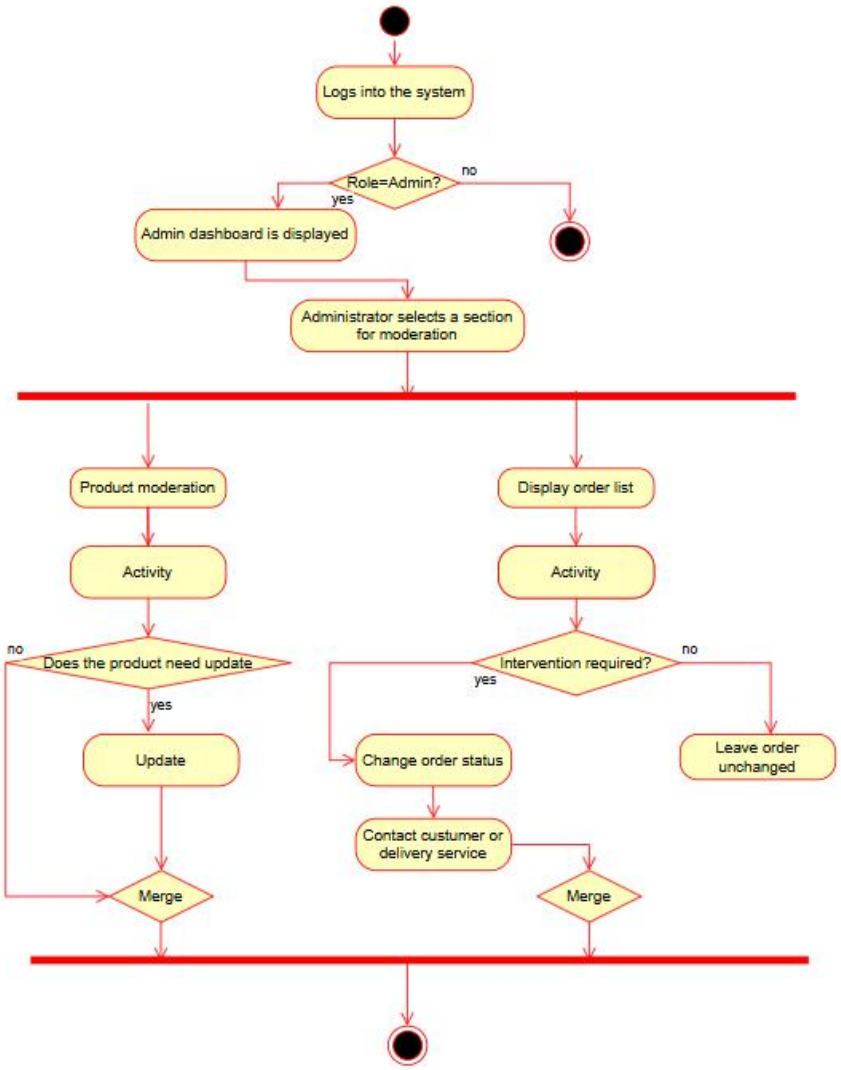

Рисунок 3.4 – Діаграма активності модерації контенту

Обидві гілки завершуються об'єднанням (merge) потоків, після чого адміністратор може повернутися до вибору наступного розділу або завершити роботу. Діаграма наочно ілюструє розгалужену логіку модерації з двома типами контенту та різними сценаріями прийняття рішень

3.3 Проєктування бази даних

Проєктування бази даних є одним із ключових етапів розробки вебзастосунку зоомагазину, оскільки саме база даних забезпечує збереження, обробку та взаємозв’язок усієї інформації системи. Для реалізації вебзастосунку було спроєктовано реляційну базу даних, яка дозволяє ефективно працювати з

товарами, користувачами, замовленнями, відгуками та іншими сутностями системи.

Під час проєктування бази даних було враховано необхідність:

- зберігання інформації про товари та їх категорії;
- підтримки реєстрації та авторизації користувачів;
- оформлення та збереження замовлень;
- реалізації кошика та списку бажань;
- збереження відгуків і рейтингів товарів;
- забезпечення цілісності та уникнення дублювання даних.

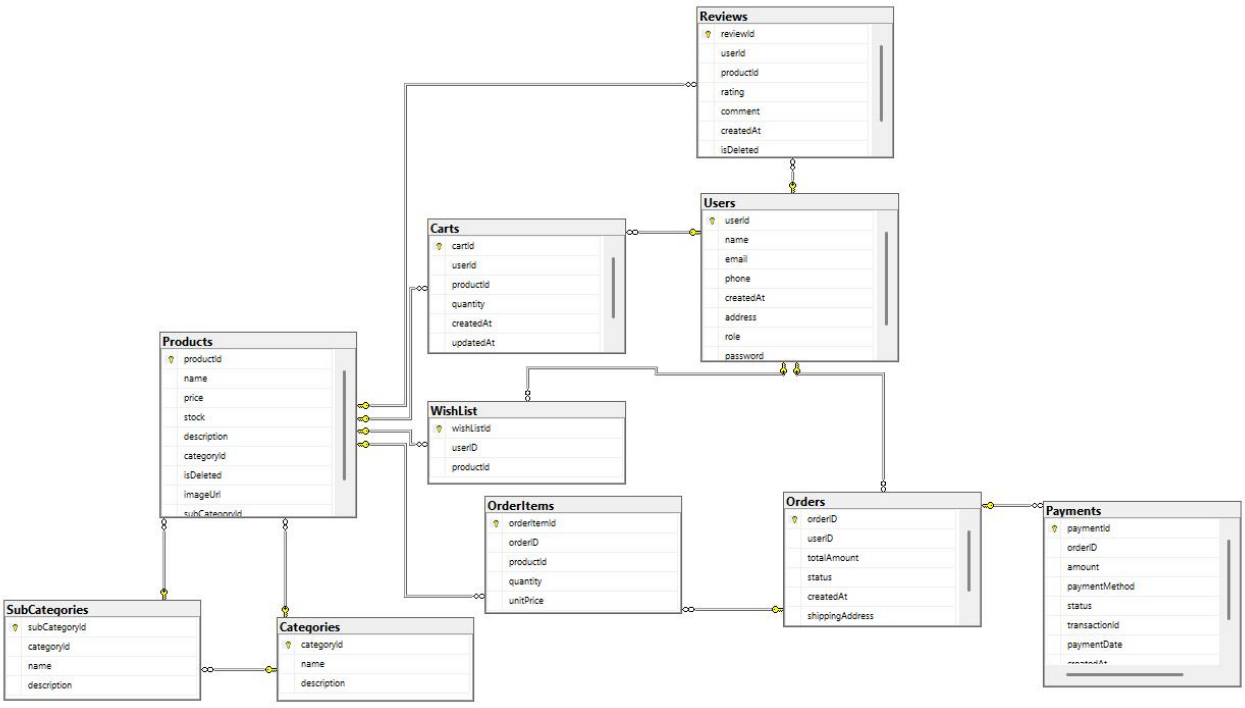

Рисунок 3.5 – ER-діаграма

Основою структури бази даних є сутність «Користувач», яка зберігає інформацію про всіх зареєстрованих користувачів системи. Її ключовими атрибутами є ідентифікатор, ім'я, електронна пошта, телефон, адреса, дата реєстрації, роль (клієнт, адміністратор, менеджер) та хеш пароля. Поле ролі дозволяє реалізувати розмежування доступу до функціоналу системи залежно від прав користувача.

Для організації ієрархічної структури каталогу товарів введено сутності «Категорія» та «Підкатегорія». Сутність «Категорія» містить базові категорії, наприклад «Собаки», «Коти», «Гризуни», а сутність «Підкатегорія» забезпечує деталізацію, наприклад «Корми», «Іграшки», «Аксесуари». Зв'язок між категорією та підкатегорією є типом «один-до-багатьох», тобто одна категорія може містити багато підкатегорій.

Центральною сутністю системи є «Товар», яка описує одиницю товару в асортименті магазину. Її атрибути включають назву, ціну, кількість на складі, опис, URL-зображення, посилання на категорію та підкатегорію, а також прапорець м'якого видалення. Поле «isDeleted» дозволяє приховувати товар від перегляду покупцями без фізичного видалення запису з бази даних, що є корисною практикою для збереження історії замовлень.

Сутність «Список бажань» або «Кошик» зберігає зв'язки між користувачем та товарами, які він додав для подальшого придбання. Кожен запис у цій таблиці відповідає одному товару, доданому користувачем. Варто зазначити, що для повноцінного функціоналу кошика доцільно додати атрибут кількості товару, щоб користувач міг додати кілька одиниць одного продукту.

Сутності «Замовлення» та «Склад замовлення» призначені для обробки оформлених покупок. Сутність «Замовлення» фіксує загальну суму, статус (наприклад, «очікує оплати», «оплачено», «відправлено», «доставлено», «скасовано»), дату створення та адресу доставки. Сутність «Склад замовлення» є деталізуючою – вона зберігає кожну окрему позицію замовлення з кількістю та ціною на момент покупки. Такий підхід забезпечує історичну точність даних: навіть якщо згодом ціна товару в каталозі зміниться, в архіві замовлення залишиться правильна сума.

Сутність «Платіж» фіксує транзакції з оплати замовлень. Вона включає суму платежу, метод оплати (банківська картка, PayPal, післяплата при отриманні), статус платежу, унікальний ідентифікатор транзакції з платіжної

системи та дату виконання платежу. Кожне замовлення має один платіж, тобто зв'язок між цими сутностями є типом «один-до-одного».

Сутність «Відгук» дозволяє користувачам залишати оцінки (рейтинг) та текстові коментарі до придбаних товарів. Прапорець «isDeleted» реалізує механізм м'якого видалення для модерації некоректних або образливих відгуків адміністратором, при цьому сам запис залишається в базі даних для історії.

У межах розробленої діаграми можна виділити наступні важливі взаємозв'язки між класами. Один користувач може мати багато замовлень, багато відгуків та багато товарів у кошику – усі ці зв'язки є типом «один-до-багатьох». Аналогічно, одна категорія містить багато підкатегорій, а категорія та підкатегорія можуть включати багато товарів. Один товар може бути присутнім у багатьох позиціях різних замовлень та мати багато відгуків від різних користувачів. Одне замовлення містить багато позицій, але має лише один платіж. Така структура зв'язків забезпечує цілісність даних та дозволяє ефективно виконувати запити до бази даних.

Розроблена діаграма класів дозволяє реалізувати наступний ключовий функціонал інтернет-магазину. По-перше, це реєстрація та автентифікація користувачів з розмежуванням доступу за ролями, що забезпечує безпеку системи. По-друге, це перегляд каталогу товарів з ієрархічною навігацією за категоріями та підкатегоріями, що робить пошук товарів зручним для покупця. По-третє, це керування кошиком, тобто додавання та видалення товарів перед оформленням покупки. По-четверте, це оформлення замовлення з фіксацією складу кошика, цін на момент покупки та адреси доставки. По-п'яте, це інтеграція з платіжними системами та відстеження статусу оплати замовлення. Крім того, система підтримує функціонал відгуків та рейтингів товарів, що допомагає іншим покупцям приймати рішення. І нарешті, передбачено адміністративну панель для керування категоріями, товарами, замовленнями та модерації відгуків, що забезпечує підтримку системи в актуальному стані.

3.4 Розробка мокапів інтерфейсу

Мокап системи - це імітація вигляду та функціональності програмного забезпечення, що створюється. Він служить інструментом для візуалізації та концептуалізації інтерфейсу користувача, функцій та потоків роботи системи. Під час проєктування вебзастосунку онлайн-клініки розроблено наступні мокапи сторінок:

1. головна сторінка застосунку;
2. сторінка особистого кабінету;
3. форма реєстрації;
4. сторінка wishlist;
5. сторінка кошику.

Мокап головної сторінки застосунку, представлено на рисунку 3.6

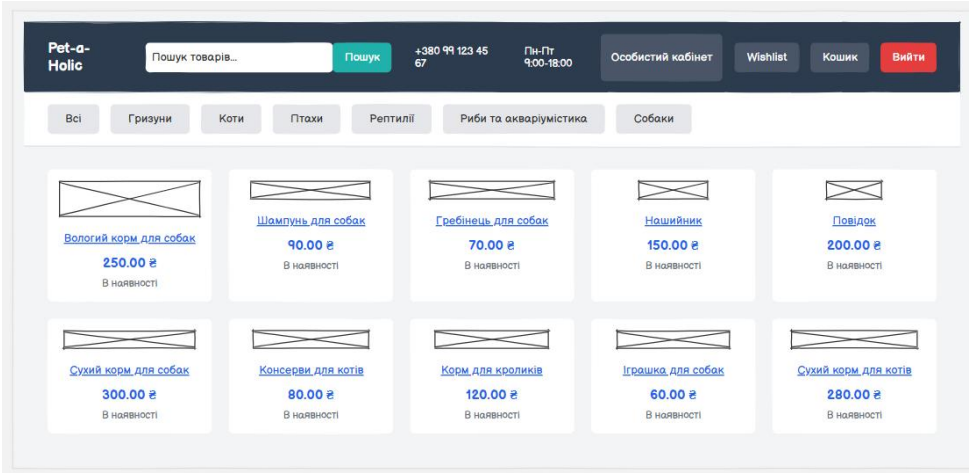

Рисунок 3.6 – Головна сторінка застосунку

У верхній частині екрана розташований рядок пошуку товарів. Ліворуч знаходиться бічна панель з розділами каталогу, серед яких «Всі», «Гризуни», «Коти», «Особистий кабінет», «Wishlist» та «Кошик». Основний простір займають картки товарів: для кожного товару зазначено назву, ціну та статус «В наявності».

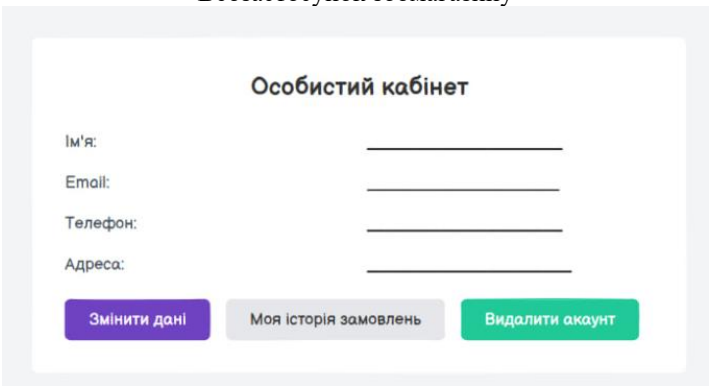

Рисунок 3.7 – Сторінка особистого кабінету

На цьому мокапі зображено особистий кабінет користувача в інтернет-магазині Pet-a-Holic. У верхній частині екрана розташована панель із ключовими елементами навігації.

Центральну частину мокапу займає форма особистого кабінету, де користувач може переглядати та редагувати свої дані. Тут передбачені поля для заповнення імені, електронної пошти, номера телефону та адреси. Під формою розташовані три кнопки: «Змінити дані», «Моя історія замовлень» та «Видалити акаунт».

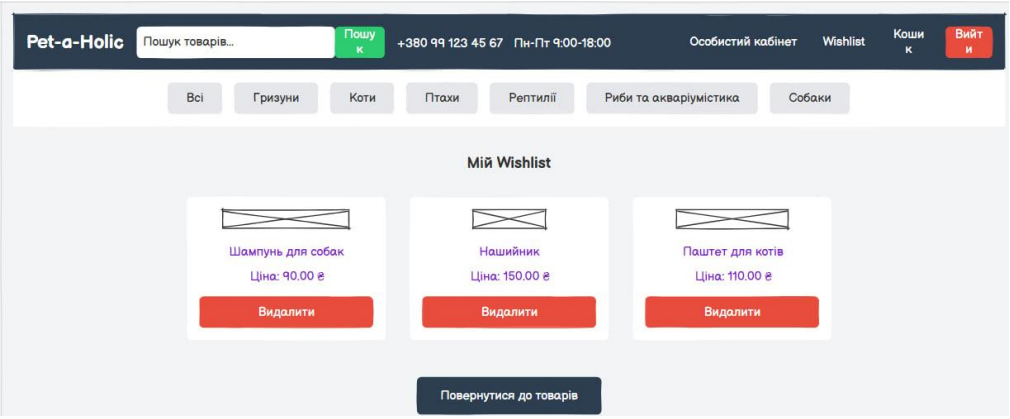

Рисунок 3.8 – сторінка wishlist

Центральну частину мокапу займає розділ «Мій Wishlist», де представлені товари, які користувач зберіг для подальшої покупки. Поруч із кожним товаром є кнопка «Видалити». Внизу сторінки розміщена кнопка «Повернутися до товарів» для навігації назад до каталогу.

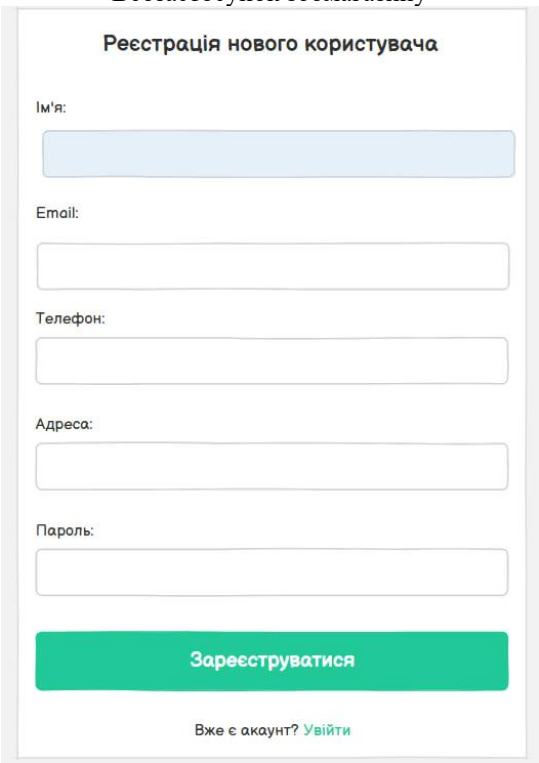

Рисунок 3.9 – форма реєстрації

Цей мокап є сторінкою реєстрації нового користувача. Він має мінімалістичну форму, у центрі якої розташовані поля для введення особистих даних: ім’я, електронна пошта, номер телефону, адреса та пароль. Під формою знаходиться кнопка «Зареєструватися» для завершення створення облікового запису, а під нею – текстовий лінк «Вже є акаунт? Увійти», який веде на сторінку входу для зареєстрованих користувачів.

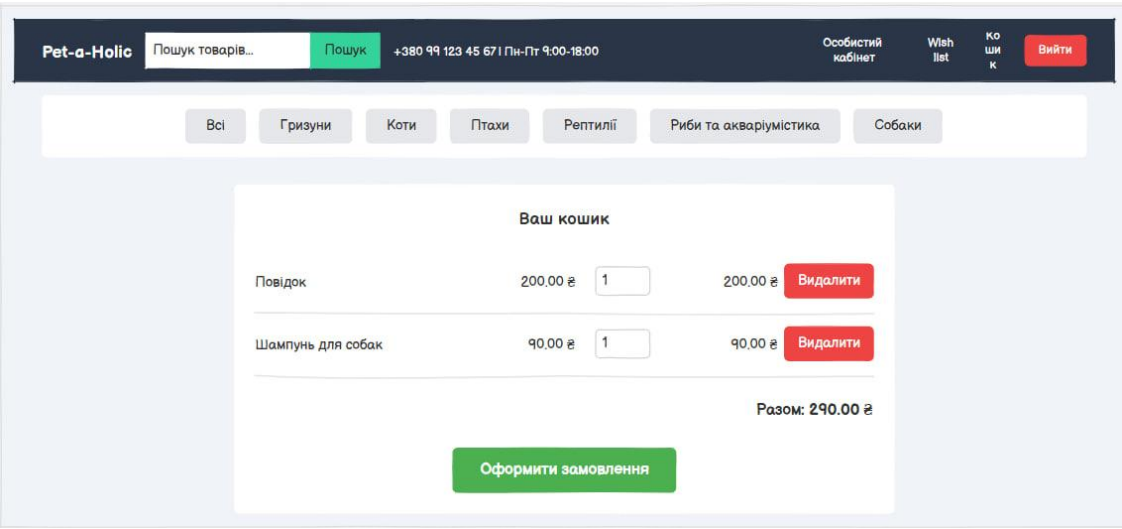

Рисунок 3.10 – сторінка кошику

Цей мокап відображає сторінку кошика. У центральній частині розташована таблиця з товарами, доданими до кошика. Для кожної позиції зазначена окрема вартість, а також кнопка «Видалити» для видалення товару з кошика. У нижній частині сторінки підведено підсумок: загальна сума кошика. Під нею розміщена велика кнопка «Оформити замовлення», яка веде до наступного етапу оформлення покупки.

Дані мокапи допоможуть розробити більш структурований та детальний вебзастосунок

Висновки до розділу 3

У третьому розділі кваліфікаційної роботи було виконано комплексне проєктування та розробку ключових архітектурних і функціональних компонентів вебзастосунку зоомагазину, що забезпечило надійну основу для його подальшої програмної реалізації.

На етапі вибору архітектури та технологій було обґрунтовано рішення використовувати клієнт-серверну модель із чітким розподілом на фронтенд і бекенд. Застосування архітектурного патерну MVC у поєднанні з фреймворком Laravel дозволило логічно розділити бізнес-логіку, інтерфейс користувача та управління даними. Такий підхід, проілюстрований діаграмою, гарантує масштабованість, гнучкість у супроводі та безпеку системи завдяки централізованій обробці запитів до бази даних під управлінням Microsoft SQL Server.

У процесі моделювання системи було визначено ролі користувачів (адміністратор, зареєстрований клієнт, гість) та детально описано ключові варіанти використання, що охоплюють повний цикл взаємодії з магазином: від реєстрації до оформлення замовлення та модерації контенту. Побудовані діаграми активності наочно візуалізували логіку виконання критично важливих бізнес-процесів, зокрема оформлення замовлення з обробкою виняткових ситуацій та

паралельні гілки модерації товарів і замовлень адміністратором, що дозволило передбачити логіку обробки помилок ще до написання коду.

Результатом проєктування бази даних стала розроблена ER-діаграма, яка відображає цілісну реляційну структуру з усіма необхідними сутностями. Спроєктована схема забезпечує ефективне збереження даних про користувачів, ієрархічний каталог товарів, кошик, замовлення, платежі та відгуки. Важливими архітектурними рішеннями стали використання механізму «м'якого видалення» для товарів і відгуків, а також фіксація ціни товару безпосередньо в деталях замовлення, що гарантує історичну точність фінансової інформації.

Також, розроблено мокапи користувацького інтерфейсу, які дозволяють заздалегідь візуалізувати та узгодити структуру основних сторінок застосунку (головна, особистий кабінет, реєстрація, список бажань, кошик) до початку їх верстки та програмування. Таким чином, сукупність виконаних на етапі проєктування робіт створила чіткий технічний план, який мінімізує ризики виникнення помилок на подальших стадіях розробки та закладає фундамент для створення надійного, зручного та функціонального вебзастосунку для електронної комерції.

4 РЕАЛІЗАЦІЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ВЕБЗАСТОСУНКУ

4.1 Створення основних сторінок застосунку та їх функціонал клієнтської частини

Для реалізації клієнтської та адміністративної частин зоомагазину було розроблено наступний набір основних сторінок, об’єднаних спільною системою навігації та базою даних.

4.1.1 Серверна частина: PHP та MS SQL Server

Клієнтська частина вебзастосунку зоомагазину реалізована за архітектурним патерном MVC [15] з використанням власної ORM-системи. Такий підхід забезпечує чітке розділення логіки програми: моделі відповідають за роботу з даними, представлення – за відображення інтерфейсу, а контролери – за обробку запитів користувача та координацію між моделями та представленнями.

Ядром системи є базовий клас ORM (файл core/ORM_init.php), який реалізує основні операції з базою даних: створення, читання, оновлення та м’яке видалення записів (поле isDeleted). Клас використовує PDO для підключення до реляційної бази даних (у проєкті – Microsoft SQL Server, але через PDO може працювати з будь-якою СУБД). Для кожної сутності (товар, категорія, замовлення тощо) створюється окрема модель-наслідник, яка визначає назву таблиці (protected static $table) та первинний ключ (protected static $primaryKey).

Клас ORM надає такі методи:

- where($column, $value) – додає умову фільтрації;
- get() – виконує запит і повертає масив об’єктів моделі;
- first() – повертає перший знайдений об’єкт;
- save() – автоматично визначає, чи потрібно виконати INSERT або UPDATE;
- update() – оновлює існуючий запис;
- delete() – виконує м’яке видалення (встановлює isDeleted = 1);

– all($pdo) – повертає всі невидалені записи;
– find($pdo, $id) – знаходить запис за первинним ключем.

core > ORM_init.php

```php
<?php

class ORM {
    protected static $table;
    protected static $primaryKey;
    protected $pdo;
    protected $attributes = [];
    protected $wheres = [];

    public function __construct($pdo, $attributes = []) {
        $this->pdo = $pdo;
        $this->attributes = $attributes;
        $this->wheres = [];
    }

    public function __get($key) {
        return $this->attributes[$key] ?? null;
    }

    public function __set($key, $value) {
        $this->attributes[$key] = $value;

        if(property_exists($this, $key)) {
        $this->$key = $value;
        }
    }

    public function where($column, $value) {
        $this->wheres[] = [$column, $value];
        return $this;
    }
```

Рисунок 4.1 – Приклад з коду ORM

Підключення до бази даних здійснюється через файл конфігурації config/db.php [16], який створює об’єкт PDO та зберігає його в глобальній змінній $GLOBALS['pdo']. Це дозволяє будь-якій моделі отримати доступ до з’єднання.

Маршрутизація в застосунку реалізована через параметри URL. Всі запити спрямовуються на єдиний вхідний файл index.php, який аналізує параметри controller та action. Наприклад, запит index.php?controller=products&action=list викликає метод list() контролера ProductsController. Такий підхід спрощує навігацію та дозволяє легко додавати нові сторінки.

Для роботи з користувачами використовуються сесії PHP. Після успішної авторизації в сесії зберігається user_id та role, що дозволяє розмежовувати права

доступу (наприклад, адміністратор бачить товари з нульовим залишком, а звичайний клієнт – ні).

Файлова структура клієнтської частини має наступний вигляд:

- controllers/контролери (AuthController, CartController, ProductsController та ін.);
- models/ – моделі, успадковані від ORM;
- views/ – шаблони представлень (HTML з вкрапленнями PHP);
- assets/css/ – каскадні таблиці стилів;
- uploads/ – каталог для завантажених зображень товарів;
- core/ – ядро системи (ORM, допоміжні функції).

Така організація коду забезпечує легку підтримуваність, розширюваність та тестування окремих компонентів.

4.1.2 Головна сторінка

Головна сторінка вебзастосунку являє собою каталог товарів з можливістю фільтрації за назвою, категорією та підкатегорією.

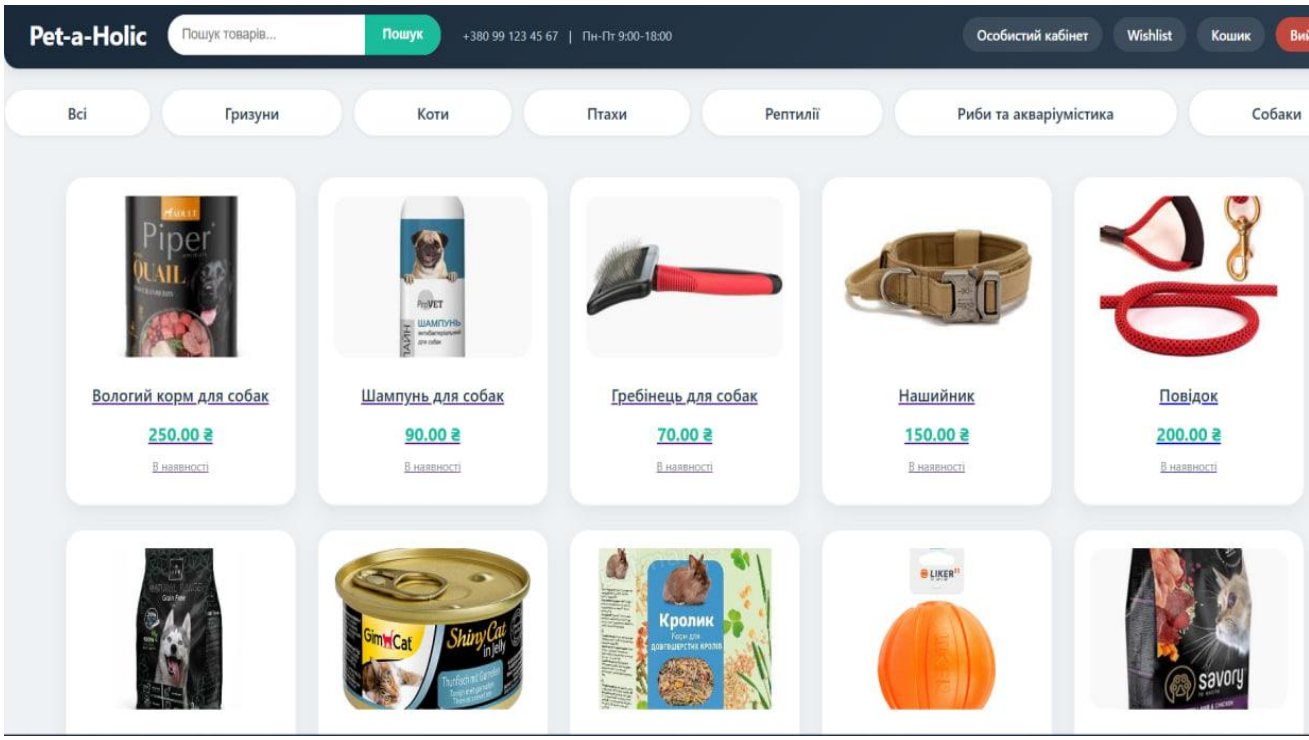

Рисунок 4.2 – Головна сторінка

Реалізація каталогу виконується контролером ProductsController (файл controllers/ProductsController.php) та моделлю Product.

Логіка роботи методу list(). При зверненні до сторінки каталогу викликається метод list() контролера. Він виконує наступні кроки:

1. Отримує параметри фільтрації з масиву $_GET: name (назва товару), categoryId (ідентифікатор категорії), subCategoryId (ідентифікатор підкатегорії).

2. Визначає роль поточного користувача з сесії ($_SESSION['role']). Якщо роль дорівнює 'admin', встановлюється прапорець $showOutOfStock = true, що дозволяє адміністратору бачити товари з нульовим залишком на складі.

3. Викликає метод getAll() моделі Product, передаючи йому отримані фільтри та прапорець.

4. Завантажує список усіх категорій за допомогою моделі Category для відображення у випадаючому списку фільтра.

5. Якщо вибрано конкретну категорію, додатково отримує список підкатегорій цієї категорії (через модель SubCategory).

6. Передає отримані дані у представлення views/products/list.php.

Метод getAll() моделі Product. Цей метод формує динамічний SQL-запит до таблиці Products з урахуванням фільтрів та прав доступу. Важливою особливістю є те, що для звичайних користувачів запит включає умову p.stock > 0, тобто приховує товари, яких немає в наявності. Для адміністратора ця умова пропускається. Крім того, запит враховує м’яке видалення (isDeleted = 0) та виконує LEFT JOIN з таблицею Categories для отримання назви категорії. Фрагмент коду методу getAll() наведено на рисунку 4.3:

```php
public function getAll($filterName = '', $categoryId = null, $subCategoryId = null, $showOutOfStock = false) {
    $sql = "
        SELECT p.*, c.name AS categoryName
        FROM Products p
        LEFT JOIN Categories c ON p.categoryID = c.categoryId
        WHERE p.isDeleted = 0
    ";
    $params = [];

    // Фільтр: не показувати товари з stock = 0, якщо не дозволено
    if (!$showOutOfStock) {
        $sql .= " AND p.stock > 0";
    }

    if ($filterName) {
        $sql .= " AND p.name LIKE ?";
        $params[] = "%$filterName%";
    }

    if ($categoryId) {
        $sql .= " AND p.categoryID = ?";
        $params[] = $categoryId;
```

Рисунок 4.3 – Функція get all

Також було розроблено динамічне оновлення фільтрів. При зміні категорії у випадаючому списку за допомогою JavaScript надсилається AJAX-запит до методу getSubcategoriesByCategory() контролера ProductsController, який повертає JSON-список підкатегорій для обраної категорії. Це дозволяє динамічно оновлювати другий випадаючий список без перезавантаження сторінки.

Взаємодія з кошиком та списком бажань. Кнопки «До кошика» та «Wishlist» викликають JavaScript-функції, які надсилають асинхронні POST-запити до відповідних контролерів (CartController та WishlistController). Це забезпечує зручний користувацький досвід – товар додається без перезавантаження сторінки, а кількість товарів у кошику оновлюється миттєво.

Таким чином, головна сторінка каталогу реалізує всі необхідні функції для перегляду, пошуку та фільтрації товарів, а також надає інтуїтивно зрозумілий інтерфейс для додавання товарів у кошик та список бажань.

4.1.3 Реєстрація та логін

Для забезпечення персоніфікації користувачів та розмежування доступу до функціоналу вебзастосунку реалізовано механізми реєстрації нового облікового

запису та авторизації (входу) зареєстрованих користувачів. Ці механізми включають сторінки реєстрації:

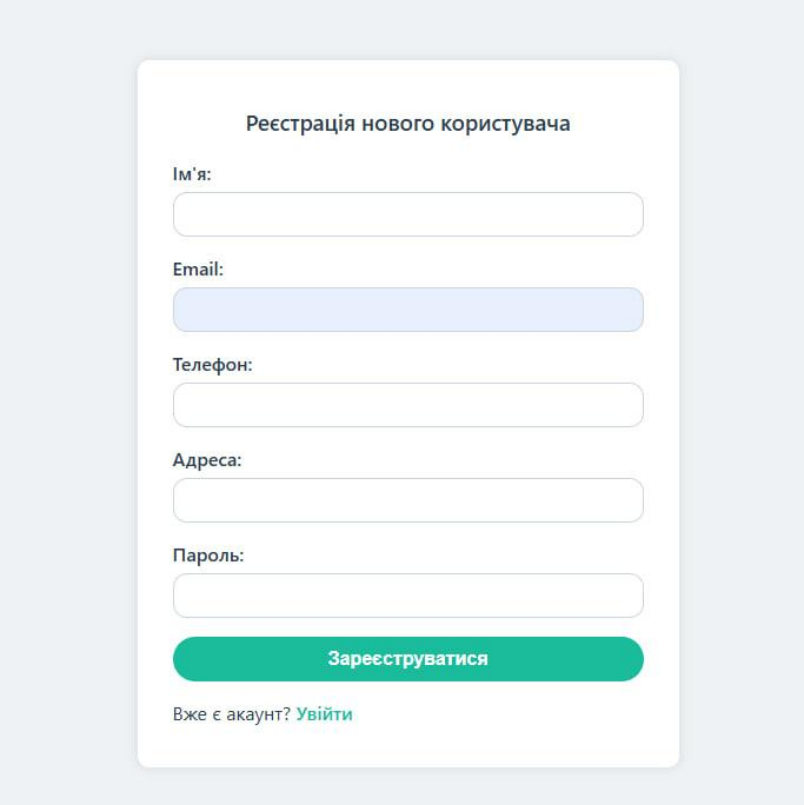

Рисунок 4.4 – Реєстрація

Та входу (малюнок 4.5), відповідні контролери, модель User для роботи з базою даних, а також управління сесіями

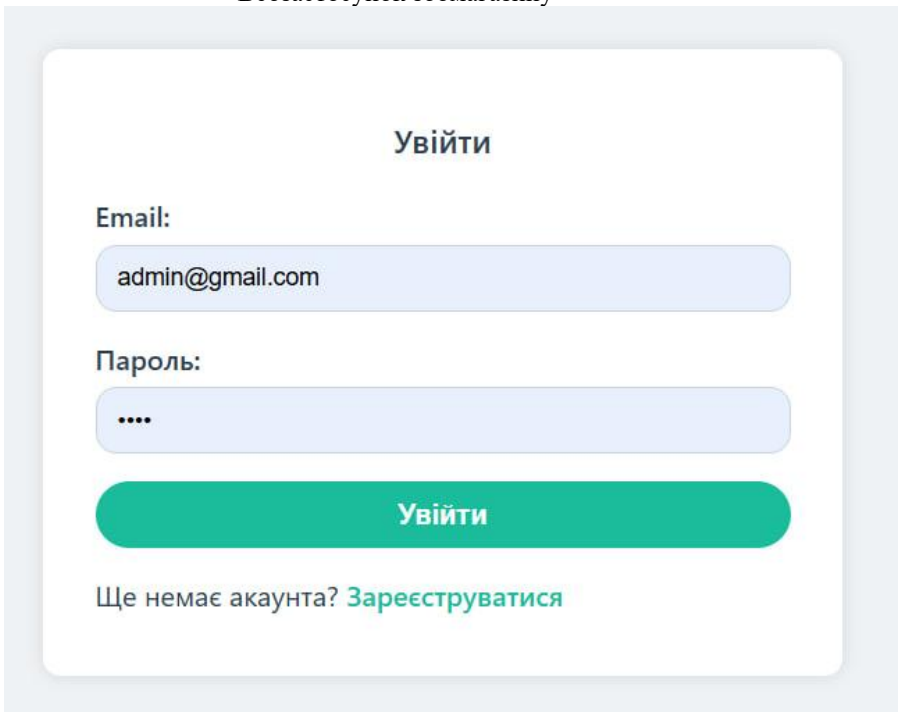

Рисунок 4.5 – Форма входу

Сторінка реєстрації (файл views/auth/signup.php) містить HTML-форму для введення особистих даних нового користувача: ім’я, електронна пошта, номер телефону, адреса та пароль. Форма надсилає дані методом POST на URL index.php?controller=users&action=signup. Поля форми обов’язкові для заповнення (атрибут required). Для зручності використання сторінка має адаптивний дизайн з центрованим контейнером, тіньовим оформленням та зеленою кнопкою відправки. У разі виникнення помилки (наприклад, спроба реєстрації з вже існуючим email) виводиться повідомлення про помилку через змінну $error. Фрагмент форми реєстрації наведено нижче на рисунку 4.6:

```
<form method="post" action="index.php?controller=users&action=signup">
    <label>Ім'я:</label>
    <input type="text" name="name" required>

    <label>Email:</label>
    <input type="email" name="email" required>

    <label>Телефон:</label>
    <input type="text" name="phone" required>

    <label>Адреса:</label>
    <input type="text" name="address" required>

    <label>Пароль:</label>
    <input type="password" name="password" required>

    <button type="submit">Зареєструватися</button>
</form>

<p>Вже є акаунт? <a href="index.php?controller=users&action=login">Увійти</a></p>
```

Рисунок 4.6 – Фрагмент коду

Також для зберігання стану автентифікації використовуються сесії PHP. Після успішного входу в сесії зберігаються ідентифікатор користувача, ім’я та роль:

```
public function logout() {
    session_destroy();
    header('Location: index.php');
    exit;
}
```

Рисунок 4.7 – Метод logout

Це дозволяє на всіх сторінках перевіряти, чи користувач авторизований, та відповідно до його ролі надавати доступ до адміністративних функцій (наприклад, редагування товарів). Метод logout() просто знищує сесію (session_destroy()) і перенаправляє на головну сторінку.

4.1.4 Особистий кабінет

Особистий кабінет є центральним місцем для керування обліковим записом зареєстрованого клієнта. Він реалізований у вигляді окремої сторінки, доступ до

якої надається після авторизації. Сторінка дозволяє переглядати та редагувати персональні дані, переходити до історії замовлень (для звичайних користувачів) або видаляти акаунт. Інтерфейс особистого кабінету складається з PHP-представлення views/users/profile.php та методу edit() контролера (у наведеній реалізації логіка редагування інтегрована безпосередньо в модель User).

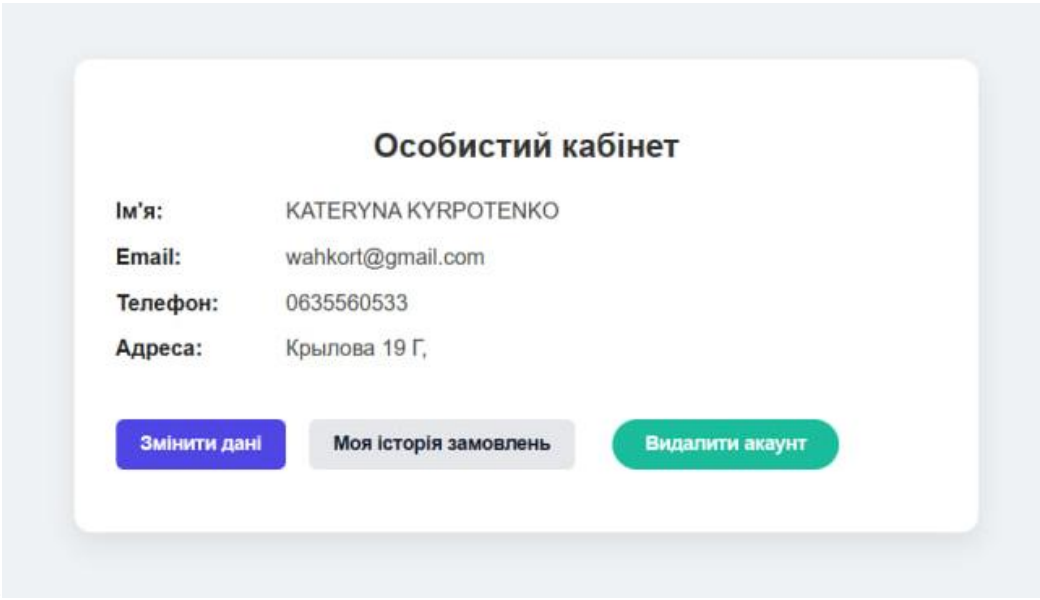

Рисунок 4.7 – Особистий кабінет

Сторінка отримує дані користувача з масиву $user (заповнюється контролером) та параметр $editing з рядка запиту, який визначає, чи перебуває користувач у режимі редагування. За замовчуванням відображається перегляд профілю. Оформлення виконано за допомогою CSS: білий контейнер з округлими кутами, тінню, зручним відступом та адаптивними кнопками. Для покращення взаємодії реалізовано автоматичне зникнення повідомлень про успіх або помилку через 4–5 секунд за допомогою JavaScript.

Коли користувач натискає «Змінити дані», сторінка переходить у режим редагування ($editing = true). Замість текстового відображення полів з’являється форма з уже заповненими полями (ім’я, email, телефон, адреса). Після внесення змін і натискання кнопки «Зберегти» дані надсилаються методом POST на той самий маршрут index.php?controller=users&action=edit. Метод edit() моделі User обробляє оновлення:

```
if($_SERVER['REQUEST_METHOD'] === 'POST') {
    $name = $_POST['name'] ?? '';
    $email = $_POST['email'] ?? '';
    $phone = $_POST['phone'] ?? '';
    $address = $_POST['address'] ?? '';
    $userModel->update($userId, $name, $email, $phone, $address);
    // оновлення даних у сесії
    $_SESSION['user']['name'] = $name;
    $_SESSION['user']['email'] = $email;
    $_SESSION['user']['phone'] = $phone;
    $_SESSION['user']['address'] = $address;
    $successMessage = "Дані оновлено успішно.";
}
```

Після успішного оновлення сторінка повертається в режим перегляду, а користувач бачить повідомлення про успіх, яке через кілька секунд зникає. Якщо виникла помилка (наприклад, некоректний email), виводиться червоне повідомлення про помилку.

4.1.5 Wishlist

Wishlist (список бажань) є корисним доповненням інтернет-магазину, що дозволяє зареєстрованим користувачам зберігати обрані товари для подальшого придбання. Функціонал реалізовано у вигляді окремого контролера WishlistController, моделі Wishlist (яка успадковує базовий клас ORM) та сторінки перегляду views/wishlist/view.php.

Модель Wishlist (файл models/Wishlist.php) визначає таблицю Wishlist без первинного ключа (оскільки це зв’язкова таблиця між Users та Products). Вона містить методи для додавання, видалення, перевірки існування та отримання всього списку бажань користувача.

```
class Wishlist extends ORM {
    protected static $table = 'Wishlist';
```

```
    protected static $primaryKey = null;
    public function add($userId, $productId) {
        if ($this->exists($userId, $productId)) return false;
        $sql = "INSERT INTO Wishlist (userId, productId) VALUES (:userId, :productId)";
        $stmt = $this->pdo->prepare($sql);
        return $stmt->execute([':userId' => $userId, ':productId' => $productId]);
    }
    public function remove($userId, $productId) {
        $sql = "DELETE FROM Wishlist WHERE userId = :userId AND productId = :productId";
        $stmt = $this->pdo->prepare($sql);
        return $stmt->execute([':userId' => $userId, ':productId' => $productId]);
    }
    public function getUserWishlist($userId) {
        $sql = "SELECT w.productId, p.name, p.price, p.imageUrl
                FROM Wishlist w JOIN Products p ON w.productId = p.productId
                WHERE w.userId = :userId";
        $stmt = $this->pdo->prepare($sql);
        $stmt->execute([':userId' => $userId]);
        return $stmt->fetchAll(PDO::FETCH_OBJ);
    }
}
```

Метод exists() перевіряє, чи товар вже додано до списку бажань, що дозволяє уникнути дублювання.

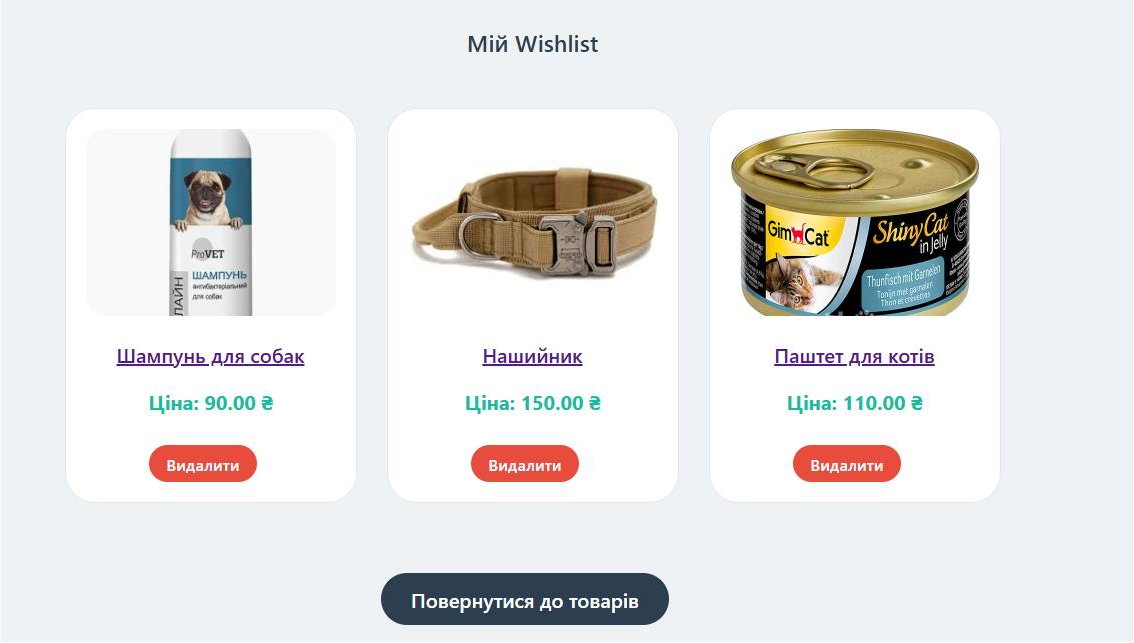

Рисунок 4.7 – Вигляд сторінки

Сторінка перегляду списку бажань виводить картки товарів з можливістю видалення кожного елемента або додавання в кошик. На головній сторінці каталогу та на сторінці товару передбачені кнопки для додавання до списку бажань, які відправляють AJAX-запити до WishlistController::addAjax(). Після успішного додавання зовнішній вигляд кнопки змінюється (наприклад, стає активною або з’являється повідомлення). Користувач може будь-коли перейти до /index.php?controller=wishlist&action=view та переглянути всі збережені товари.

Таким чином, список бажань реалізує зручний механізм відкладеного придбання, повністю інтегрований з каталогом та кошиком.

4.1.6 Кошик

Кошик є ключовим компонентом інтернет-магазину, що відповідає за тимчасове зберігання вибраних товарів перед оформленням замовлення. У розробленому вебзастосунку кошик реалізовано з підтримкою як авторизованих користувачів (зберігання в базі даних), так і гостей (зберігання в сесії). Після входу в систему кошик гостя автоматично зливається з кошиком користувача в базі даних.

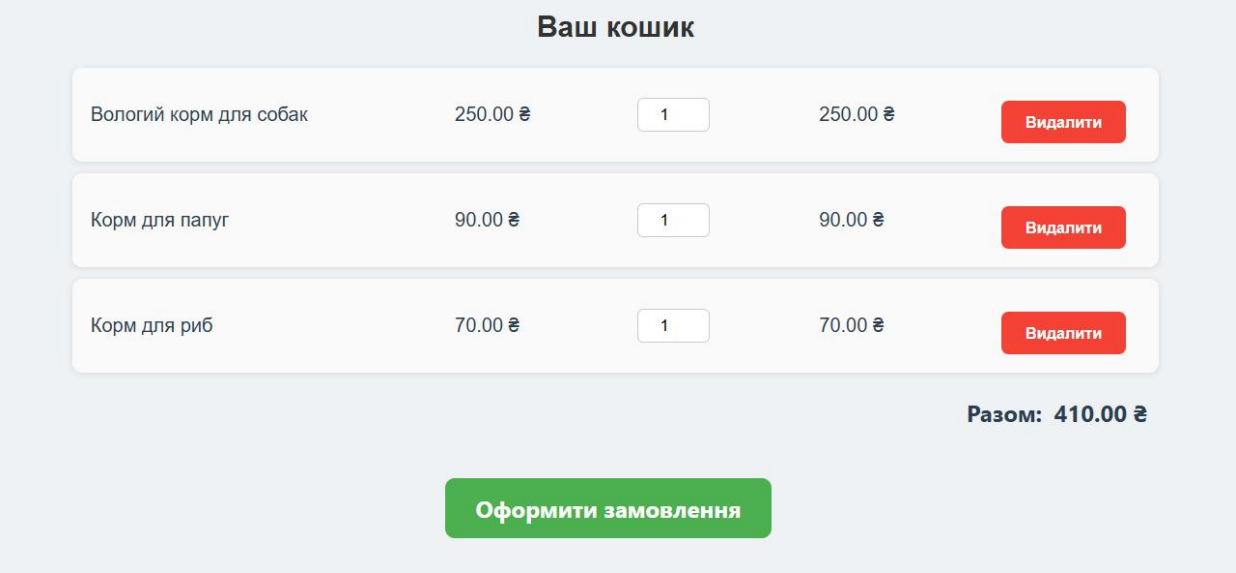

Рисунок 4.8 – Кошик

Для забезпечення безпечних онлайн-платежів у вебзастосунку використано платіжний шлюз LiqPay[17]. LiqPay надає зручний PHP SDK, який дозволяє інтегрувати форму оплати без необхідності зберігати платіжні дані на сервері магазину.

У загальному випадку користувача перекидає на форму оплати, де потрібно ввести дани картки Visa або Mastercard та свою елекрону пошту для отримання повідомлення про оплату.

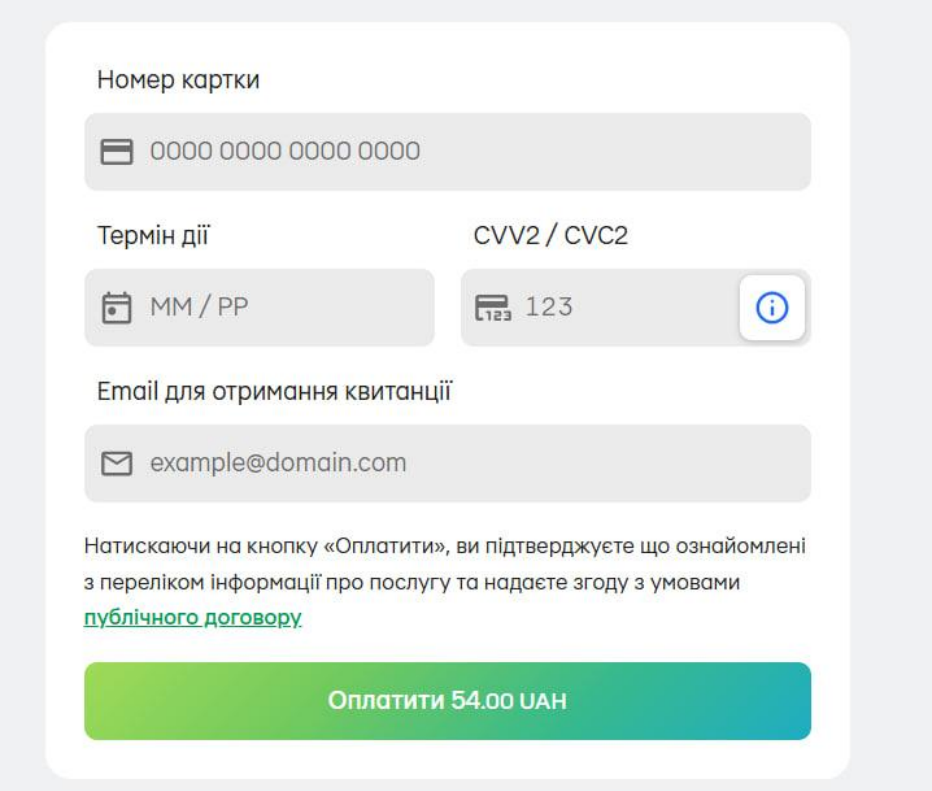

Рисунок 4.9 – Форма оплати

Після успішної оплати LiqPay надсилає POST-запит на server_url. У методі callback необхідно перевірити підпис, оновити статус замовлення та очистити кошик.

```
public function callback() {
    $data = json_decode(base64_decode($_POST['data']), true);
    $signature = $_POST['signature'];
    $liqpay = new LiqPay($public_key, $private_key);
    $expected = $liqpay->str_to_sign($private_key . $data . $private_key);
    if ($signature === $expected && $data['status'] == 'success') {
        $orderId = $data['order_id'];
        $this->orderModel->updateStatus($orderId, 'paid');
    }
    echo 'OK';
}
```

Таким чином, інтеграція з LiqPay дозволяє приймати онлайн-платежі без зберігання конфіденційної інформації на сервері, забезпечуючи високий рівень безпеки.

4.2 Основні сторінки панелі адміністратора

Адміністративна панель вебзастосунку призначена для керування всіма аспектами роботи інтернет-магазину. Доступ до неї мають лише користувачі з роллю admin. Панель включає головну сторінку (дашборд) з ключовими показниками, модуль управління товарами (CRUD), модуль управління замовленнями, а також можливість перегляду списку користувачів (окремі контролери). У даному розділі розглянуто реалізацію дашборду (рисунок 4.10) та повного циклу роботи з товарами.

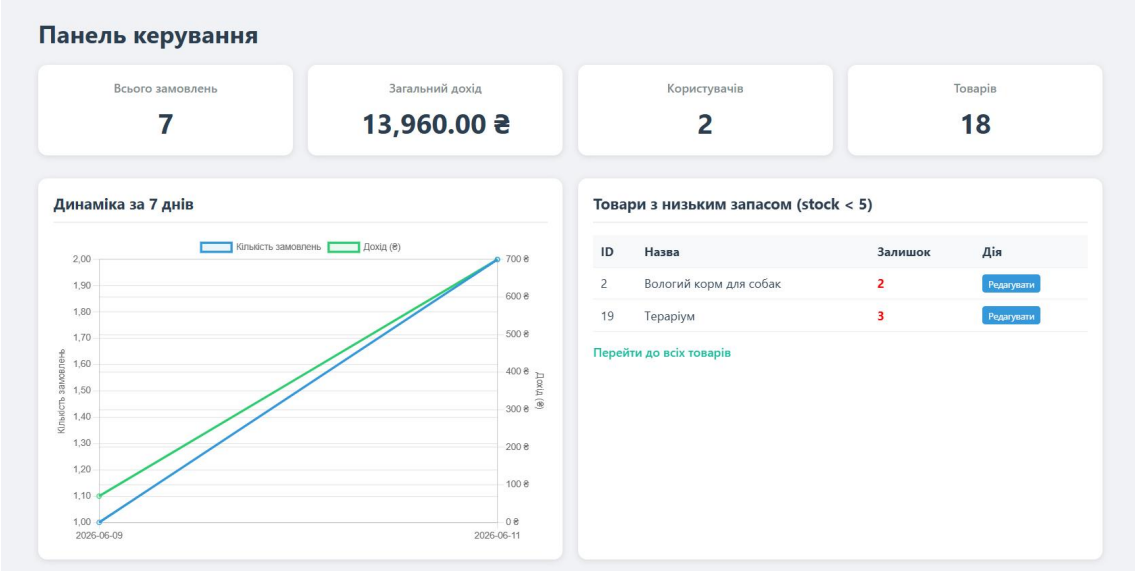

Рисунок 4.10 – Панель керування

Головна сторінка адміністративної панелі (файл controllers/AdminDashboardController.php) надає зведену інформацію про стан магазину: загальну кількість замовлень, дохід, кількість користувачів, активних товарів, товари з низьким залишком, останні замовлення, топ-5 товарів за продажами, а також динаміку за останні 7 днів у вигляді графіка. Доступ до цієї сторінки захищено перевіркою ролі ($_SESSION['role'] === 'admin').

Метод index() контролера виконує такі запити до бази даних:

1. Загальна кількість замовлень (COUNT(*) FROM Orders).
2. Загальний дохід (сума totalAmount для замовлень, крім скасованих).
3. Кількість користувачів (COUNT(*) FROM Users).
4. Кількість активних товарів (COUNT(*) FROM Products WHERE isDeleted = 0).
5. Товари із залишком менше 5 одиниць для швидкого реагування.
6. Останні 5 замовлень з іменами користувачів (через JOIN Users).
7. Топ-5 товарів за кількістю проданих одиниць (SUM(oi.quantity) з групуванням).
8. Статистика за останні 7 днів: кількість замовлень та дохід за кожен день (з використанням DATEADD для SQL Server).

Для візуалізації динаміки використано бібліотеку Chart.js [18]. Дані для графіка формуються у вигляді масивів $chartLabels, $chartOrders, $chartRevenue, які передаються у представлення та перетворюються у JSON для ініціалізації графіка.

Представлення views/admin/dashboard.php містить адаптивну HTML-структуру з сіткою статистичних карток, двоколонковим компонуванням, таблицями та Canvas-елементом для графіка. Для різних статусів замовлень визначено класи CSS (status-new, status-processing, status-shipped, status-delivered, status-cancelled), що дозволяє візуально виділяти їх кольором. Код JavaScript створює лінійний графік з двома осями Y (кількість замовлень та дохід).

Приклад коду формування графіка (фрагмент представлення):

```
new Chart(ctx, {
    type: 'line',
    data: {
        labels: <?= json_encode($chartLabels) ?>,
        datasets: [
            {
```

```
            label: 'Кількість замовлень',
            data: <?= json_encode($chartOrders) ?>,
            borderColor: '#3498db',
            yAxisID: 'y'
        },
        {
            label: 'Дохід (₴)',
            data: <?= json_encode($chartRevenue) ?>,
            borderColor: '#2ecc71',
            yAxisID: 'y1'
        }
    ]
  },
  options: {
    scales: { y1: { position: 'right', title: { text: 'Дохід (₴)' } } }
  }
});
```

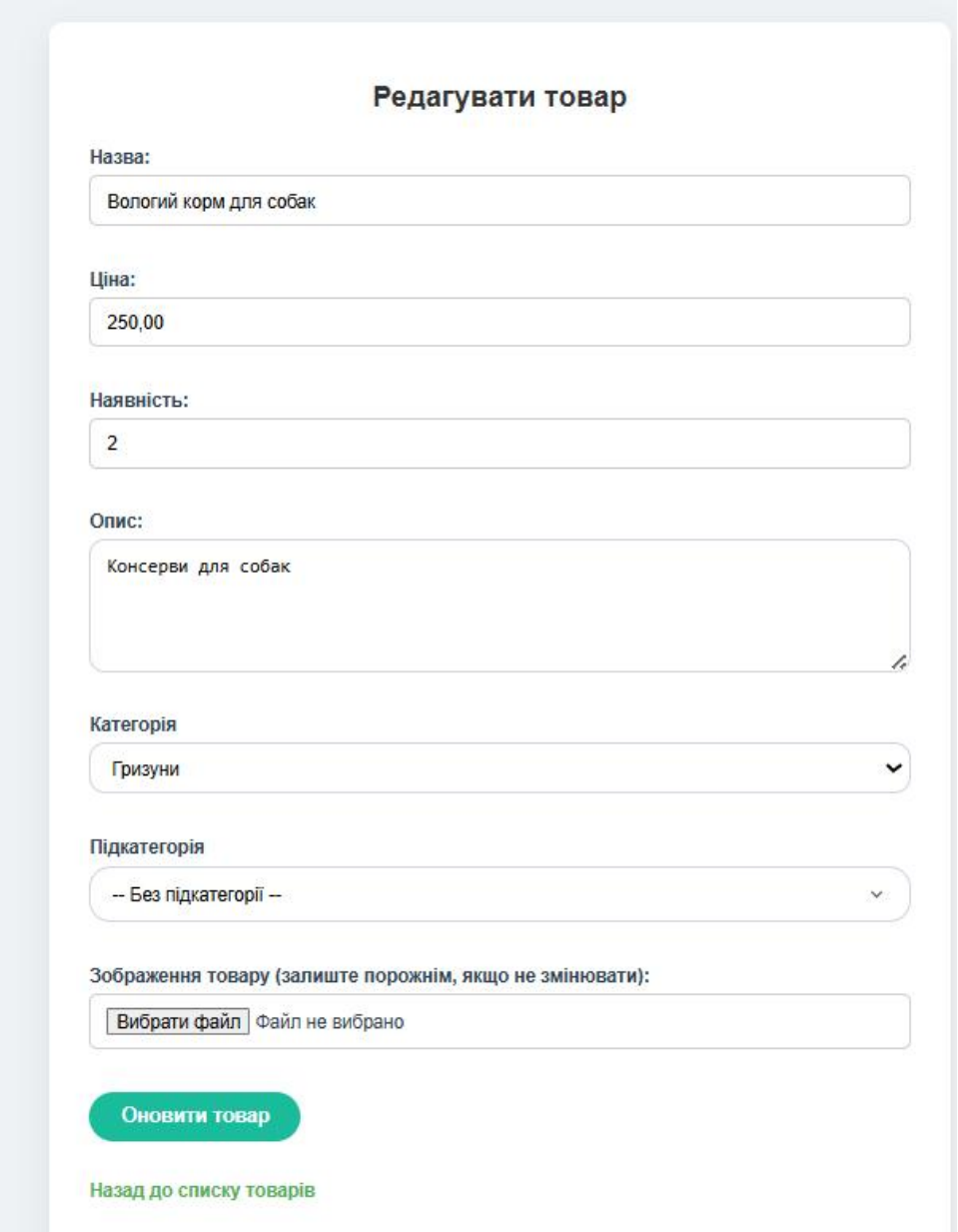

Рисунок 4.11 – редагування товару

Модуль управління товарами реалізовано в контролері ProductsController (файл controllers/ProductsController.php), який надає повний набір операцій: перелік товарів з фільтрацією, додавання нового товару, редагування існуючого, м’яке видалення та перегляд детальної інформації. Для роботи з базою даних використовується модель Product, успадкована від базового класу ORM.

Метод list() відповідає за відображення каталогу товарів в адміністративному режимі (хоча той самий метод використовується і для клієнтської частини, але з різними правами). Він приймає параметри фільтрації

(name, categoryId, subCategoryId) з $_GET. Залежно від ролі користувача встановлюється прапорець $showOutOfStock: для адміністратора він дорівнює true, що дозволяє бачити товари з нульовим залишком. Далі викликається метод Product::getAll() з відповідними параметрами. Отриманий список товарів разом з даними про категорії передається у представлення views/products/list.php. Додатково, якщо вибрано категорію, завантажуються підкатегорії для фільтра.

Метод add() обробляє як GET-запит (відображення форми), так і POST-запит (збереження нового товару). Форма містить поля: назва, ціна, кількість на складі, опис, категорія, підкатегорія та завантаження зображення. Особливу увагу приділено обробці підкатегорії: якщо передано порожній рядок, 'null' або 0, значення встановлюється в NULL (оскільки підкатегорія може бути не обов’язковою). Додатково виконується перевірка існування підкатегорії в базі даних – це запобігає помилкам зовнішнього ключа. Зображення товару зберігається в каталозі uploads/products/ з унікальним ім’ям (на основі часу). Якщо зображення не завантажено, встановлюється шлях до файлу default.jpg. Після створення об’єкта Product, заповнення його атрибутів та виклику методу save() (успадкованого від ORM) відбувається вставка запису в таблицю Products. Потім виконується перенаправлення на сторінку зі списком товарів.

Приклад коду обробки підкатегорії у методі add():

```
if ($subCategoryId === '' || $subCategoryId === 'null' || $subCategoryId === 0) {
    $subCategoryId = null;
}
if ($subCategoryId !== null) {
    $subCategoryId = (int)$subCategoryId;
    $stmt = $this->pdo->prepare("SELECT SubCategoryID FROM SubCategories WHERE SubCategoryID = ?");
    $stmt->execute([$subCategoryId]);
    if (!$stmt->fetch()) {
        $subCategoryId = null;
```

```
    }
}
```

У представленні views/products/add.php реалізовано динамічне керування списком підкатегорій за допомогою JavaScript: при зміні категорії підкатегорії фільтруються за атрибутом data-category, що забезпечує зручність заповнення форми.

Метод edit() отримує ідентифікатор товару з параметра id. За допомогою Product::where('productId', $id)->first() завантажується об’єкт товару. Для заповнення форми редагування використовуються ті самі допоміжні моделі (Category, SubCategory). При POST-запиті атрибути товару оновлюються безпосередньо в об’єкті (наприклад, $product->name = $_POST['name']), після чого викликається метод update() (який формує SQL UPDATE автоматично). Особливістю є оновлення зображення: якщо завантажено новий файл, старе зображення (крім default.jpg) видаляється з файлової системи, а нове зберігається в тому самому каталозі.

Метод delete() реалізує м’яке видалення: він встановлює атрибут isDeleted = 1 для вказаного товару (через $product->isDeleted = 1; $product->save();). Це дозволяє зберігати історію замовлень і не порушувати цілісність даних. Товар більше не відображається в каталозі для звичайних користувачів, але адміністратор може бачити його та відновити за потреби.

4.3 Тестування застосунку

Етап тестування програмного забезпечення є критичним для розробки маркетплейсу, адже саме він визначає якість, стабільність і захищеність системи під час реальної експлуатації. Зважаючи на специфіку проєкту, основну увагу зосереджено на ручному функціональному тестуванні ключових сценаріїв, оскільки такий підхід дає змогу оцінити, як інтерфейс реагує на дії користувача. На початковій стадії тестування було підготовлено набір тест-кейсів, що охоплюють усі критичні бізнес-функції: реєстрацію та вхід, перегляд і фільтрацію

каталогу, роботу з кошиком і бонусами, оформлення замовлень, порівняння товарів, а також написання коментарів і відгуків. Кожен тест-кейс містить передумови, чітку послідовність кроків і очікуваний результат, що дозволяє відтворити його незалежно від зовнішнього середовища. Його результати наведено в табл. 4.1

Таблиця 4.1 – Тест-кейси вебзастосунку

№	Назва тесту	Попередні умови	Кроки тестування	Очікуваний результат
1	Реєстрація нового користувача	Сторінка реєстрації відкрита	Ввести ім’я, email, телефон, адресу, пароль; 2) Натиснути «Зареєструватися»	Новий запис у таблиці Users; повідомлення «Реєстрація успішна»; перенаправлення на сторінку каталогу
2	Вхід існуючого користувача	Є зареєстрований користувач	Ввести email та пароль; 2) Натиснути «Увійти»	Отримання сесії; перенаправлення на каталог; у навігації з’являються «Особистий кабінет» та «Вихід»
3	Перегляд каталогу з фільтрацією за категорією та підкатегорією	Користувач на головній сторінці	Обрати категорію «Собаки»; 2) Обрати підкатегорію «Корми»	Відображено лише товари, що належать до «Собаки» → «Корми»; кожна картка містить назву, ціну, зображення та позначку наявності
4	Фільтрація за ціною	Відкрито каталог (будь-яка категорія)	Ввести діапазон цін «100 – 500»; 2) Натиснути «Фільтрувати»	Показано лише товари з ціною від 100 до 500 грн включно

Кінець таблиці 4.1

5	Пошук товару за назвою	Користувач на сторінці каталогу	Ввести в пошуковий рядок «Royal Canin»; 2) Натиснути кнопку пошуку	Відображаються всі товари, назва яких містить «Royal Canin»
6	Додавання товару до кошика з вибором кількості	Користувач на сторінці товару; товар у наявності (stock ≥ 3)	Вибрати кількість «2»; 2) Натиснути «До кошика»	Сповіщення «Товар додано до кошика»; лічильник кошика в навігації збільшується на 2; у БД створено/оновлено запис CartItem з кількістю 2
7	Додавання товару до списку бажань (Wishlist)	Користувач авторизований; відкрита картка товару	Натиснути кнопку «Wishlist» (іконка серця)	Кнопка змінює стан на активний; товар з’являється на сторінці «Список бажань»; у БД створено запис у таблиці Wishlist
8	Оформлення замовлення з вибором способу доставки та онлайн-оплатою (LiqPay)	Користувач авторизований; кошик не порожній; усі товари в наявності	Відкрити кошик; 2) Натиснути «Оформити замовлення»; 3) Підтвердити адресу доставки, обрати спосіб доставки; 4) Натиснути «Оплатити»; 5) На	Після успішної оплати — повідомлення «Замовлення успішно створено»; у БД створено записи Orders та OrderItems; кошик очищено; залишки товарів зменшено; на

			формі LiqPay ввести тестові дані картки та підтвердити платіж	email надіслано підтвердження

Кінець таблиці 4.1

9	Перегляд списку бажань	Користувач додав ≥1 товар до Wishlist	Перейти до розділу «Wishlist» через бічне меню	Відображається список збережених товарів із назвою, ціною, зображенням; для кожного є кнопка «Видалити»
10	Перегляд історії замовлень	Користувач авторизований; має ≥1 завершене замовлення	В особистому кабінеті натиснути «Моя історія замовлень»	Відображається список замовлень із датою, статусом, сумою; є можливість переглянути деталі конкретного замовлення
11	Залишення відгуку про придбаний товар	Користувач авторизований; має завершене замовлення з товаром, на який ще не залишав відгук	Перейти на сторінку товару; 2) У блоці «Відгуки» обрати рейтинг 5 зірок; 3) Ввести текст «Чудовий корм»; 4) Натиснути «Надіслати»	Новий запис у таблиці Reviews; відгук з іменем користувача та рейтингом з’являється на сторінці товару під блоком «Відгуки»
12	Перегляд детальної інформації про товар	Каталог завантажено; товар присутній у списку	Клікнути на назву або зображення будь-якого товару	Відкривається сторінка товару з повним описом, ціною, кількістю на складі, категорією, зображенням, а також кнопками «До кошика», «Wishlist» та блоком із відгуками

Після виконання зазначених тестових випадків було підтверджено, що вебзастосунок відповідає поставленим цілям і має необхідні функції. Ручна перевірка вебсайту проводилася за допомогою браузера «Opera GX», при введенні даних суворо дотримувалися умов, необхідних для успішного проходження тестових випадків. Спроби виконати тестові випадки без дотримання попередніх умов призвели до невдачі тестування.

Висновки до розділу 4

У четвертому розділі кваліфікаційної роботи описано процес практичної реалізації вебзастосунку зоомагазину, розглянуто архітектурні рішення, реалізовані програмні компоненти, інтерфейс користувача та результати тестування.

На основі вибраного технологічного стеку (PHP, власна ORM, Microsoft SQL Server, HTML/CSS/JavaScript, Chart.js) було створено повнофункціональний вебзастосунок, який повністю відповідає сформульованим вимогам. Ключовим елементом серверної частини стала розроблена ORM-система, що забезпечує абстракцію доступу до бази даних, підтримку м'якого видалення та автоматичне формування SQL-запитів. Маршрутизація на основі параметрів URL та управління сесіями дозволили організувати чітку структуру застосунку та розмежувати права доступу між гостем, зареєстрованим клієнтом та адміністратором.

У клієнтській частині реалізовано всі необхідні сторінки:

- головна сторінка каталогу з фільтрацією за назвою, категорією та підкатегорією, адаптивним дизайном та підтримкою AJAX для оновлення кошика та списку бажань;
- сторінки реєстрації та авторизації з валідацією даних та керуванням сесіями;

− особистий кабінет, що дозволяє переглядати та редагувати профіль, переглядати історію замовлень та видаляти акаунт (з каскадним видаленням залежних записів у транзакції);

− сторінка списку бажань (Wishlist) з можливістю додавання/видалення товарів через AJAX;

− динамічний кошик з підтримкою як авторизованих користувачів (зберігання в БД), так і гостей (зберігання в сесії) з автоматичним злитям після входу;

− оформлення замовлення з перевіркою залишків товарів, оновленням складських запасів у межах транзакції та інтеграцією з платіжним шлюзом LiqPay через PHP SDK.

Адміністративна панель включає:

− дашборд з візуалізацією ключових показників (загальна кількість замовлень, дохід, кількість користувачів, товарів) та динаміки продажів за останні 7 днів за допомогою бібліотеки Chart.js;

− повноцінний CRUD для управління товарами (додавання, редагування з можливістю оновлення зображення, м'яке видалення) з підтримкою категорій та підкатегорій;

− управління замовленнями (перегляд, фільтрація за статусом та користувачем, зміна статусу, видалення).

Особливу увагу приділено безпеці: використання підготовлених запитів PDO для запобігання SQL-ін'єкціям, екранування виводу через htmlspecialchars() для захисту від XSS, перевірка ролі користувача перед доступом до адміністративних функцій. Платіжна інтеграція з LiqPay здійснюється без зберігання конфіденційних даних на сервері, з використанням callback-запитів для підтвердження оплати.

Також було проведено ручне тестування вебзастосунку, адже відсутність синтаксичних помилок у програмному коді не гарантує вірної роботи програми. Тестування програми було призначено для виявлення помилок, та виправлення їх.

Таким чином, у розділі 4 повністю підтверджено працездатність розробленого вебзастосунку, його відповідність вимогам специфікації та готовність до впровадження в реальну експлуатацію. Отримані результати демонструють ефективність обраних технологій та архітектурних рішень для створення сучасних e-commerce систем.

ВИСНОВКИ

У результаті виконання кваліфікаційної роботи всі сформульовані завдання, виконано в повному обсязі, а поставлену мету досягнуто – розроблено повнофункціональний вебзастосунок зоомагазину, який забезпечує автоматизацію продажів, зручне управління каталогом товарів та ефективну онлайн-взаємодію з клієнтами. Робота охопила повний цикл створення програмного продукту: від аналізу предметної області до тестування та документування.

У першому розділі виконано аналіз предметної області та існуючих аналогів, виявлено їхні переваги й недоліки, на основі чого сформульовано конкретні задачі розробки та вимоги.

У другому розділі обґрунтовано вибір технологічного стеку та сформовано специфікацію вимог із повним переліком функціональних і нефункціональних вимог до програмного забезпечення.

Третій розділ присвячено проєктуванню: розроблено структуру сайту, інформаційну модель сторінок, ER-діаграму бази даних, UML-діаграми, а також створено прототипи інтерфейсу користувача, що дозволило попередньо візуалізувати ключові сторінки.

У четвертому розділі реалізовано основний функціонал: клієнтську частину з адаптивним дизайном, інтеграцію з платіжним шлюзом LiqPay, адміністративну панель із візуалізацією звітів та управлінням товарами й замовленнями. Проведене тестування підтвердило відповідність застосунку вимогам.

Таким чином, створений вебзастосунок є готовим до впровадження рішенням для автоматизації роботи зоомагазину. Він надає клієнтам зручний інструмент для пошуку та купівлі товарів, а адміністраторам – ефективні засоби для управління асортиментом, обробки замовлень та аналізу продажів. Науково-практичне значення роботи полягає в тому, що отримані результати можуть бути використані як основа для подальших досліджень у сфері електронної комерції та впроваджені в діяльність малих і середніх зоомагазинів, сприяючи їхній цифровій трансформації.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Zoobonus: інтернет-магазин зоотоварів. URL: https://zoobonus.ua/ (дата звернення: 21.04.2026).

2. Leyven: інтернет-магазин зоотоварів. URL: https://www.leyven.com.ua/ (дата звернення: 21.04.2026).

3. Sharma H., Tripathi K. The Importance of Website Usability in Digital Marketing: A Review. International Journal of Innovative Research in Computer Science and Technology. 2023. DOI: 10.55524/ijircst.2023.11.4.10. URL: https://journalspress.uk/index.php/LJRCST/article/view/194 (date of access: 21.04.2026).

4. Сотнік С., Манаков В., Ляшенко В. Overview: PHP and MySQL Features for Creating Modern Web Projects. International Journal of Academic Information Systems Research (IJAISR). 2023. Vol. 7, Issue 1. P. 11-17. URL: https://openarchive.nure.ua/server/api/core/bitstreams/99c2a8e9-aefd-47d4-a06f-838650b760db/content (дата звернення: 21.04.2026).

5. Булатов О. С., Гайдаєнко О. В. Інтеграція платіжних систем в сучасні вебзастосунки. Вільне ПЗ у освіті, науці та бізнесі (FOSS 2025): збірник тез доповідей Міжнародної науково-практичної конференції. Харків: ХНЕУ ім. С. Кузнеця, 2025. С. 117-119. URL: https://repository.hneu.edu.ua/bitstream/123456789/35624/3/foss-2025-theses.pdf#page=117 (дата звернення: 21.04.2026).

6. JetBrains. The State of PHP 2024 – Expert review. JetBrains Blog. 2025. URL: https://blog.jetbrains.com/phpstorm/2025/10/the-state-of-php-2024/ (date of access: 15.05.2026).

7. Chart.js. Simple yet flexible JavaScript charting library. Офіційна документація. URL: https://www.chartjs.org/ (date of access: 26.05.2026).

8. Oluwatosin H. S. Client-Server model. IOSR Journal of Computer Engineering. 2014. Vol. 16, no. 1. P. 57–71. URL: https://www.iosrjournals.org/ (date of access: 30.04.2026).

9. MDN Web Docs. Client-Server Overview. MDN Web Docs. URL: https://developer.mozilla.org/en-US/docs/Learn/Server-side/First_steps/Client-Server_overview (date of access: 12.05.2026).

10. Laravel. Laravel Documentation. The PHP Framework for Web Artisans. URL: https://laravel.com/docs (date of access: 12.05.2026).

11. MDN Web Docs. JavaScript Guide: Introduction. MDN Web Docs. URL: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Introduction (date of access: 18.05.2026).

12. IBM. Relational Database Overview. IBM Think. URL: https://www.ibm.com/topics/relational-databases (date of access: 12.06.2026).

13. Microsoft Learn. Adding a View (MVC Pattern). ASP.NET MVC Documentation. URL: https://learn.microsoft.com/en-us/aspnet/mvc/overview/getting-started/introduction/adding-a-view (date of access: 20.05.2026).

14. Red Hat. What is a REST API? Red Hat Topics. URL: https://www.redhat.com/en/topics/api/what-is-a-rest-api (date of access: 22.05.2026).

15. Fowler M. MVC Architecture Explanation. Martin Fowler’s Website. URL: https://martinfowler.com/eaaDev/uiArchs.html (date of access: 22.05.2026).

16. The PHP Group. PHP Data Objects (PDO) Documentation. PHP Manual. URL: https://www.php.net/manual/en/book.pdo.php (date of access: 23.05.2026).

17. LiqPay. SDK for seamless integration with LiqPay payment gateway. GitHub Repository. URL: https://github.com/liqpay/liqpay-php-sdk (date of access: 25.05.2026).

18. Silas K. Create a Dynamic Dashboard With Chart.js and PHP // Envato Tuts+. – 2020. URL: https://code.tutsplus.com/create-a-dynamic-dashboard-with-chartjs-and-php--cms-34587t (date of access: 28.05.2026).

ДОДАТОК А ЛІСТИНГ КОДУ

```
<?php
if (session_status() === PHP_SESSION_NONE) session_start();
class AdminDashboardController {
    private $pdo;
    public function __construct($pdo) {
        $this->pdo = $pdo;
    }
    public function list() {
        $this->index();
    }
    public function index() {
        if (!isset($_SESSION['role']) || $_SESSION['role'] !== 'admin') {
            header('Location: index.php?controller=auth&action=login');
            exit;
        }
        // 1. Загальна кількість замовлень
        $stmt = $this->pdo->query("SELECT COUNT(*) as cnt FROM Orders");
        $totalOrders = $stmt->fetch(PDO::FETCH_ASSOC)['cnt'];
        // 2. Загальний дохід (без скасованих)
        $stmt = $this->pdo->query("SELECT SUM(totalAmount) as rev FROM
Orders WHERE status != 'cancelled'");
        $revenue = $stmt->fetch(PDO::FETCH_ASSOC)['rev'] ?? 0;
        // 3. Кількість користувачів
        $stmt = $this->pdo->query("SELECT COUNT(*) as cnt FROM Users");
        $totalUsers = $stmt->fetch(PDO::FETCH_ASSOC)['cnt'];
        // 4. Кількість активних товарів
```

```
$stmt = $this->pdo->query("SELECT COUNT(*) as cnt FROM Products WHERE isDeleted = 0");
$totalProducts = $stmt->fetch(PDO::FETCH_ASSOC)['cnt'];
// 5. Товари з залишком < 5
$stmt = $this->pdo->prepare("SELECT productId, name, stock FROM Products WHERE stock < 5 AND isDeleted = 0 ORDER BY stock");
$stmt->execute();
$lowStock = $stmt->fetchAll(PDO::FETCH_ASSOC);
// 6. Останні 5 замовлень
$stmt = $this->pdo->prepare("
    SELECT TOP 5 o.orderId, o.totalAmount, o.status, o.createdAt, u.name as userName
    FROM Orders o
    JOIN Users u ON o.userId = u.userId
    ORDER BY o.createdAt DESC
");
$stmt->execute();
$recentOrders = $stmt->fetchAll(PDO::FETCH_ASSOC);
// 7. Топ-5 товарів за кількістю продажів
$stmt = $this->pdo->prepare("
    SELECT TOP 5 p.name, SUM(oi.quantity) as totalSold
    FROM OrderItems oi
    JOIN Products p ON oi.productID = p.productId
    GROUP BY p.name
    ORDER BY totalSold DESC
");
$stmt->execute();
$topProducts = $stmt->fetchAll(PDO::FETCH_ASSOC);
```

```
        // 8. Статистика за останні 7 днів (для графіка)
        $stmt = $this->pdo->prepare("
            SELECT
                CAST(createdAt AS DATE) as orderDate,
                COUNT(*) as ordersCount,
                SUM(totalAmount) as dailyRevenue
            FROM Orders
            WHERE createdAt >= DATEADD(day, -7, GETDATE())
            GROUP BY CAST(createdAt AS DATE)
            ORDER BY orderDate
        ");
        $stmt->execute();
        $weekly = $stmt->fetchAll(PDO::FETCH_ASSOC);
        $chartLabels = [];
        $chartOrders = [];
        $chartRevenue = [];
        foreach ($weekly as $day) {
            $chartLabels[] = $day['orderDate'];
            $chartOrders[] = $day['ordersCount'];
            $chartRevenue[] = (float)$day['dailyRevenue'];
        }
        require_once __DIR__ . '/../views/admin/dashboard.php';
    }
}
```