

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

ПОЯСНЮВАЛЬНА ЗАПИСКА

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

на тему:

«Веб-сайт з продажу зоотоварів»

Виконала студент групи ІІЗ-19-2
ФЕДОРЕНКО Евеліна Олегівна

Керівник роботи:
КРАВЧЕНКО Світлана Миколаївна

Рецензент:
ЧИЖМОТРЯ Олексій Володимирович

Житомир – 2023

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

«ЗАТВЕРДЖУЮ»

В.о. зав. кафедри інженерії
програмного забезпечення
Андрій МОРОЗОВ
«14» лютого 2023 р.

ЗАВДАННЯ
на кваліфікаційну роботу

Здобувач вищої освіти: **ФЕДОРЕНКО Евеліна Олегівна**

Керівник роботи: **КРАВЧЕНКО Світлана Миколаївна**

Тема роботи: **«Веб-сайт з продажу зоотоварів»,**

затверджена Наказом закладу вищої освіти від «14» лютого 2023 р., №70с

Вихідні дані для роботи: **Предмет дослідження – розробка об'єктно-орієнтованої системи.**

Об'єктом дослідження – методи та засоби проектування та розробки веб-сайту.

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

Розділ	Консультант	Завдання видав	Завдання прийняв
1	Кравченко С.М.,	16.02.2023р.	16.02.2023р.
2	Кравченко С.М.,	16.02.2023р.	16.02.2023р.
3	Кравченко С.М.,	16.02.2023р.	16.02.2023р.

Календарний план

№ з/п	Назва етапів випускної кваліфікаційної роботи	Термін виконання етапів роботи	Примітка
1	Аналізування вимог до програмного продукту	14 лютого 2023 – 25 лютого 2023	Виконано
2	Пошук, огляд та аналіз аналогічних веб-сайтів	26 лютого 2023 – 13 березня 2023	Виконано
3	Вивчення матеріалу по конкретній тематиці	14 березня 2023 – 22 березня 2023	Виконано
4	Планування структури програми	23 березня 2023 – 31 березня 2023	Виконано
5	Проектування та реалізація архітектури веб-системи	1 квітня 2023 – 9 квітня 2023	Виконано
6	Написання програмного коду	10 квітня 2023 – 22 травня 2023	Виконано
7	Тестування роботи програми	23 травня 2023 – 25 травня 2023	Виконано
8	Аналіз одержаних результатів	26 травня 2023 – 28 травня 2023	Виконано
9	Оформлення пояснювальної записки	29 травня 2023 – 11 червня 2023	Виконано
10	Попередній захист	4 червня 2023	Виконано

РЕФЕРАТ

Завданням кваліфікаційної роботи було створення веб-сайту об'єктно-орієнтованої системи, зі застосуванням PHP фреймворку Laravel, проектування проекту та алгоритмів системи.

Пояснювальна записка до кваліфікаційної роботи на тему «Веб-сайт з продажу зоотоварів» складається зі вступу, трьох розділів, висновків, літератури та додатків.

Пояснювальна записка має 58 сторінку тексту. Список використаних джерел містить 18 найменувань та займає 2 сторінки. В роботі наведено 48 рисунків. Загальний обсяг роботи – 74 сторінок.

Ключові поняття: БАЗА ДАНИХ, LARAVEL, PHP, ІНТЕРНЕТ-МАГАЗИН, ФРЕЙМВОРК.

					ІПЗ.КР.Б – 121-23 – ПЗ			
Змн.	Арк.	№ докум.	Підпис	Дата				
Розроб.		Е.О Федоренко.			Веб-сайт з продажу зоотоварів	Літ.	Арк.	Аркушів
Керівник		С.М. Кравченко					4	74
Рецензент		О.В. Чижмотря			Пояснювальна записка	Житомирська політехніка, група ІПЗ-19-2		
Зав. каф.		А.В. Морозов						

ABSTRACT

The task of qualification work was to create object-oriented website with PHP framework Laravel, design project and his algorithms.

Explanatory note to qualification work «Zoo website» consists of introduction, three chapters, conclusion, list of used sources and additions.

Explanatory note has 58 pages of text. The list of used sources has 18 names and occupies 2 pages. The work contains 48 illustrations. The total amount – 74 pages.

Key words: DATA BASE, LARAVEL, PHP, ONLINE-SHOP, FRAMEWORK.

					ИПЗ.КР.Б – 121-23 – ИПЗ	Арк.
						5

ЗМІСТ

ВСТУП	7
РОЗДІЛ 1. АНАЛІЗ ЗАДАЧІ ТА МЕТОДИ ВИРІШЕННЯ	8
1.1 Постановка задачі	8
1.2 Аналіз існуючих аналогів за темою кваліфікаційної роботи	9
1.3 Архітектура веб-сайту з продажу зоотоварів.....	11
1.4 Обґрунтування вибору інструментальних засобів розробки та вимоги.....	13
Висновки до розділу 1	18
РОЗДІЛ 2. ПРОЕКТУВАННЯ ВЕБ-САЙТУ	19
2.1 Визначення варіантів використання та об'єктно-орієнтованої системи....	19
2.2 Розробка бази даних проекту.....	23
2.3 Проектування та реалізація алгоритмів роботи системи	25
2.4 Реалізація веб-сайту з продажу зоотоварів	29
Висновки до розділу 2	38
3. ІНТЕРФЕЙС ТА РОЗРОБКА ВЕБ-САЙТУ	39
3.1 Порядок налаштування та встановлення системи та визначення прав доступу до системи	39
3.2 Структура інтерфейсу веб-сайту з продажу зоотоварів.....	41
3.3. Тестування роботи програмного продукту	55
Висновки до розділу 3	57
ВИСНОВКИ.....	58
ЛІТЕРАТУРА	59
ДОДАТКИ.....	62

ВСТУП

Під час роботи над проектом «Веб-сайт з продажу зоотоварів» були розглянуті основні методи моделювання та аналізу програмного забезпечення, які були використані для написання пояснювальної записки до кваліфікаційної роботи.

Тема роботи – веб-сайт з продажу зоотоварів.

Актуальність теми – в наш час онлайн-магазини є дуже популярними, особливо в час дистанційної роботи або навчання, покупки в інтернеті стали частиною рутини більшості людей. Тому веб-сайт з продажу зоотоварів є доволі актуальним в наш час.

За мету роботи було поставлено проектування та розробку веб-додатку зі застосуванням РНР фреймворку Laravel, об'єктно-орієнтованої системи, реалізація алгоритмів роботи.

Завданням роботи є проведення аналізу задач веб-сайту, визначення архітектури та узагальнення структури теми, проектування проекту та алгоритмів системи, реалізація програмного продукту.

Предметом дослідження є розробка об'єктно-орієнтованої системи, створення бази даних з використанням патернів, використання веб-орієнтованих технологій.

Об'єктом дослідження даної кваліфікаційної роботи є методи та засоби проектування та розробки веб-сайту.

Методи дослідження – в роботі були використанні різні методи системного аналізу, веб-проектування, об'єктно-орієнтованого проектування.

Також в даному проекті був використаний РНР фреймворк Laravel, з відкритим кодом, створений Тейлором Отвелом і призначений для розробки веб додатків відповідно до шаблону model–view–controller (MVC).

РОЗДІЛ 1. АНАЛІЗ ЗАДАЧІ ТА МЕТОДИ ВИРІШЕННЯ

1.1 Постановка задачі

1. Назва системи:

Повне найменування системи: Веб-сайт з продажу зоотоварів.

Коротке найменування системи: Програмний комплекс, Програма, Програмне забезпечення (ПЗ), Система.

2. Призначення системи:

Програма призначена для надання максимальної інформації про продукцію, що реалізується, з метою скорочення часу на її придбання, сукупність веб-вітрин і торгової системи, зв'язок між якими забезпечує програмне забезпечення.

3. Вимоги до системи:

Програмний продукт повинен володіти наступними функціональними характеристиками:

- Доступ до системи за допомогою web-інтерфейсів;
- Експорт звітів та аналітики до документів pdf, docx, csv;
- Експорт списку клієнтів, замовлень, товарів, користувачів системи, та окремих записів до pdf, docx, csv;
- Можливість одночасної роботи із системою 25 і більше користувачам;
- Наявність двох типів облікових записів (клієнт та адміністратор).

4. Вимоги до програмного забезпечення:

- Використання веб-технологій без додаткового ліцензування та компонентів, встановлених на ПК;
- Генерація сторінок за стандартом HTML5;
- Коректна робота та підтримка новими версіями браузерів (Opera не нижче 36.0, Firefox не нижче 41.0, Google Chrome не нижче 50.0).

5. Вимоги до інтерфейсу:

					ІПЗ.КР.Б – 121-23 – ІПЗ	Арк.
						8

- Мінімально можливий час відгуку системи;
- Оптимізація інтерфейсу системи з точки зору максимальної роботи із програмним комплексом;
- Ґрунтування на використанні загальноприйнятих методів і засобів представлення інформації;
- Підтвердження виконання критичних дій в системі.

6. Етапи розробки системи:

- Створення UML-моделі системи;
- Проектування інтерфейсів та розробка графічного дизайну всіх модулів;
- HTML-верстка інтерфейсів;
- Проектування та створення бази даних системи;
- Розробка програмної частини модулів системи, інтеграція шаблонів, налаштування інтерфейсів.

1.2 Аналіз існуючих аналогів за темою кваліфікаційної роботи

В даному підрозділі будуть наведені аналоги даної роботи та порівняння з системою.

Першим прикладом є Petslike.net. Даний онлайн-магазин має стильний дизайн, зручний інтерфейс та навігацію. Проте, асортимент товару є трохи обмеженим, так як він тільки для двох категорій тварин: котів і собак.

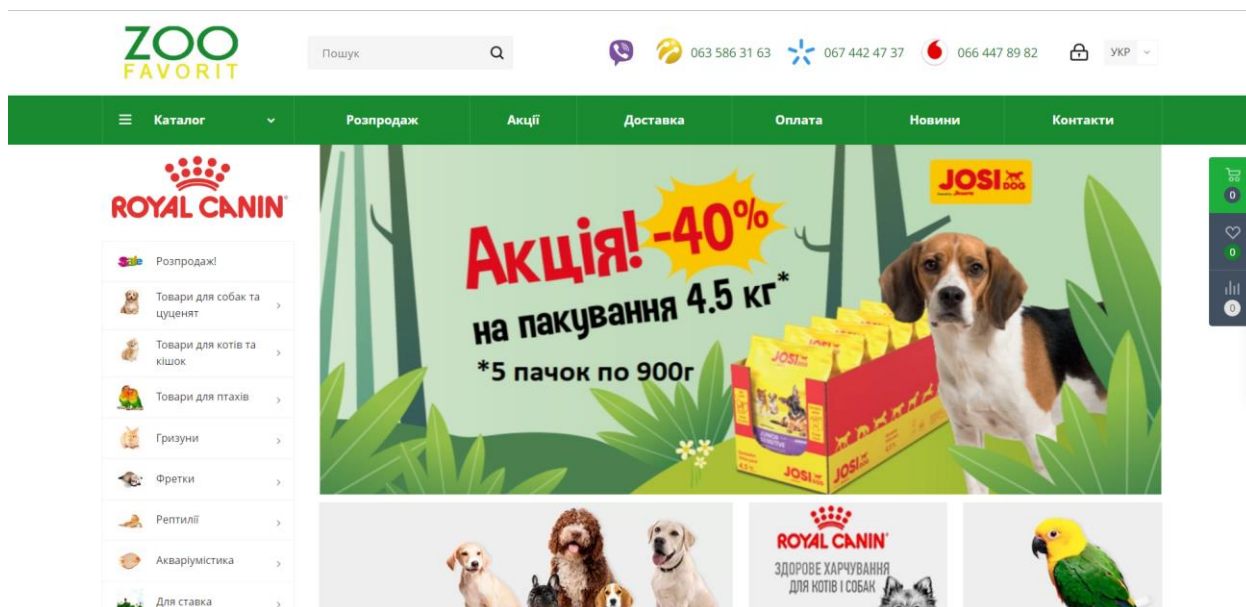


Рисунок 1.3 – Інтерфейс веб-сайту Zoo Favorit

Для порівняння зазначимо основні характеристики засобів у табл.1.1.

Таблиця 1.1 – Порівняння аналогів веб-сайтів з продажів зоотоварів

Назва	Варіативність асортименту	Зручність використання	Інтерфейс
Petslike.net	+-	+	+
MasterZoo	+	-	-
Zoo Favorit	+	+	+

Відповідно до таблиці, найкращим варіантом для аналогу нашого майбутнього проекту є третій веб-сайт Zoo Favorit.

1.3 Архітектура веб-сайту з продажу зоотоварів

Даний веб-сайт розроблений за шаблоном model–view–controller (MVC), має об'єктно-орієнтовану проектування та алгоритми роботи системи [7].

Модель–вигляд–контролер (MVC, Модель–представлення–контролер, англ. Model-view-controller) — архітектурний шаблон, який використовується під час проектування та розробки програмного забезпечення.

Model – база даних та основна логіка програми для роботи з нею, обробки запитів, проведення обчислень тощо;

View – інтерфейс користувача, усі візуальні елементи, з якими буде взаємодіяти користувач;

Controller – прошарок між View і Model, що реагує на зміни.

Цей шаблон передбачає поділ системи на три взаємопов'язані частини: модель даних, вигляд (інтерфейс користувача) та модуль керування. Застосовується для відокремлення даних (моделі) від інтерфейсу користувача (вигляду) так, щоб зміни інтерфейсу користувача мінімально впливали на роботу з даними, а зміни в моделі даних могли здійснюватися без змін інтерфейсу користувача.

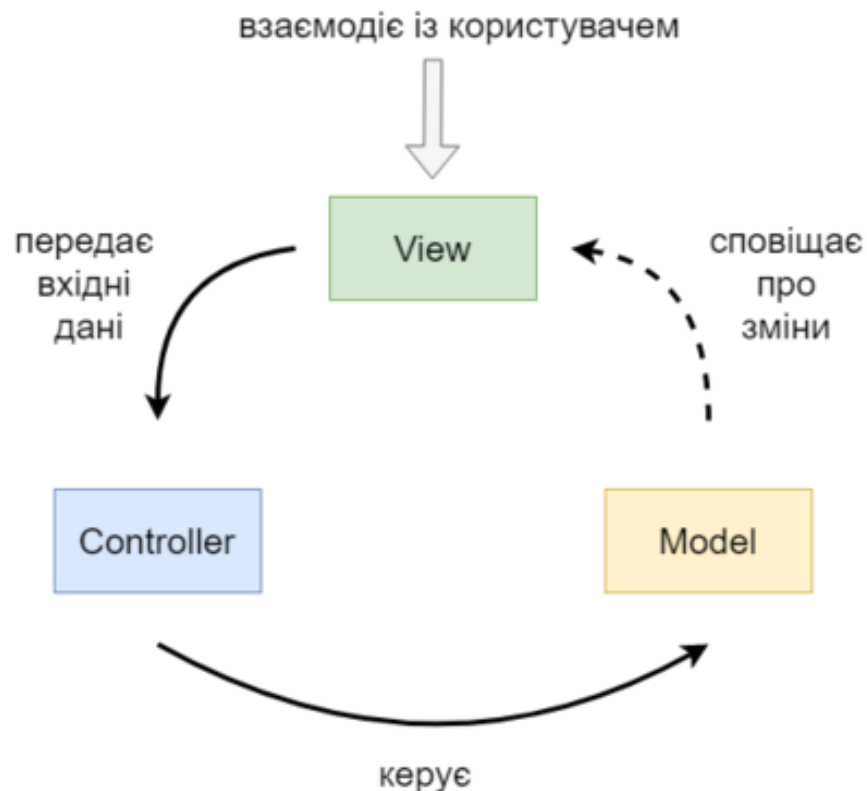


Рисунок 1.4 – Схема архітектури MVC

Мета шаблону — гнучкий дизайн програмного забезпечення, який повинен полегшувати подальші зміни чи розширення програм, а також надавати можливість повторного використання окремих компонентів програми.

Переваги MVC у розробці:

1. Чіткий розподіл: Будь-який шаблон архітектури передбачає створення різних блоків коду та взаємодії між ними. MVC — класичне рішення для поділу конструктивних блоків програми. Його використання підвищує ефективність роботи на етапі розробки й далі в масштабуванні та підтримці. Чіткий поділ робить його добрим вибором для великих

проектів, над якими працюватиме багато фахівців. Особливо, якщо програма має складний графічний інтерфейс.

2. Паралельна розробка та проста реалізація адаптивності: Працювати над розбитим на блоки кодом можуть кілька команд, а готові рішення інтегруватися за необхідності. Водночас MVC заощаджує ресурси на розробку, дозволяючи використовувати ті самі дані для виведення видів на різних типах пристроїв.
3. Спрощене тестування та зручне обслуговування: Цей патерн люблять за те, що він «зручний і красивий», він чудово справляється із завданням поділу, що підвищує ефективність розробки, як на старті, так і під час масштабування проєкту. Навіть глобальні зміни структури даних та логіки інтерфейсу проходитимуть простіше, а пошук та виправлення помилок — ще легше. Використання MVC підвищує керованість та передбачуваність програми, незалежно від його масштабів та складності.

1.4 Обґрунтування вибору інструментальних засобів розробки та вимоги

В цьому підрозділі описані вибір інструментальних засобів, детальні вимоги, правила та показники якості програмного комплексу «Веб-сайт з продажу зоотоварів» та подальший аналіз.

Для проектування та розробки даного веб-сайту були використанні такі інструментальні засоби:

- додаток StarUML та мова UML, які були використанні для моделювання та аналізу даного програмного забезпечення;
- СУБД MySQL;
- мова програмування PHP та відповідний фреймворк Laravel, для створення серверної частини проєкту;
- HTML, CSS, JavaScript, Bootstrap для написання фронтенду даного програмного продукту;
- PHPStorm як середовище розробки сайту.

PHP (англ. PHP: Hypertext Preprocessor — PHP: гіпертекстовий репроцесор) — скриптова мова програмування, була створена для генерації HTML-сторінок на стороні веб сервера. PHP є однією з найпоширеніших мов, що використовуються у сфері веб розробок (разом із Java, .NET, JavaScript, Python, Ruby). PHP підтримується переважною більшістю хостинг-провайдерів. PHP — проект відкритого програмного забезпечення [17].

Саме тому для написання більшої частини системи була використана ця мова програмування. В даному випадку для цього також використовується фреймворк Laravel, який підходить як для написання серверної частини, так і клієнтської частини сайту [8].

Для створення моделі в даній системі було застосовано мову UML.

UML — уніфікована мова моделювання, використовується у парадигмі об'єктно-орієнтованого програмування [1]. Є невід'ємною частиною уніфікованого процесу розробки програмного забезпечення. UML є мовою широкого профілю, це відкритий стандарт, що використовує графічні позначення для створення абстрактної моделі системи, яка називається UML-моделлю. UML був створений для визначення, візуалізації, проектування й документування в основному програмних систем. UML не є мовою програмування, але в засобах виконання UML-моделей як інтерпретованого коду можлива кодогенерація.

Для створення бази даних і роботи з нею було використано MySQL.

MySQL — вільна система керування реляційними базами даних, яка була розроблена компанією «ТсХ» для підвищення швидкодії обробки великих баз даних. Ця система керування базами даних (СКБД) з відкритим кодом була створена як альтернатива комерційним системам. MySQL з самого початку була дуже схожою на mSQL, проте з часом вона все розширювалася і зараз MySQL — одна з найпоширеніших систем керування базами даних. Вона використовується, в першу чергу, для створення динамічних веб сторінок, оскільки має чудову підтримку з боку різноманітних мов програмування [18].

Далі будуть описані вимоги до веб-орієнтованого продукту.

Бізнес-вимоги:

1. Основні цілі: проект створюється з метою покращення обліку клієнтів, товарів та моніторингу продажів продукції компанії;
2. Представлення проекту: проект буде реалізовано у вигляді сайту, що містить структуровану інформацію по даній тематичній області.
3. Розрахунок нормативного часу виконання замовлень: надається можливість для кожного окремого замовлення, на основі введених параметрів, автоматично визначати час, відведений на його виконання.
5. Контроль за виконанням та оперативне коректування ходу робіт: погодинний контроль виконання та можливість швидкого коректування планів при виникненні не стандартних (критичних) ситуацій.

Вимоги користувачів:

- внутрішній користувач – клієнт:
 1. Перегляд інформації про поточні замовлення (дата їх проведення, місце, термін, плата);
 2. Оформлення замовлень товарів;
 3. Реєстрація;
 4. Переглядати товари.
- внутрішній користувач – адміністратор:
 1. Дії над товарами (додавання, редагування, видалення);
 2. Перегляд товарів;
 3. Створення нових облікових записів (Адміністраторів та користувачів)
 4. Перегляд останніх новин системи;
 5. Облік і аналіз статистики руху замовлень.

Характеристика об'єкту комп'ютеризації:

Адміністратор через веб-інтерфейс буде мати можливість редагувати, додавати дані, користуватися всіма функціями системи.

Функціональні вимоги:

					ІПЗ.КР.Б – 121-23 – ІПЗ	Арк.
						15

- В системі повинна бути представлення можливість реєстрації та авторизації користувачів;
- Валідація введених даних;
- Модифікація введених даних;
- Відображення результатів виконаної процедури;
- Можливість збереження інформації: Система повинна зберігати інформацію;

Не функціональні вимоги:

1. Сприйняття:

- Час, потрібний для навчання роботи з програмою для звичайних користувачів — 4 години, для досвідчених — 1 година;
- Час відповіді системи для звичайних запитів не повин перевищувати 3 секунд, а для більш складних запитів — 10 секунд;
- Інтерфейс представлення програми повинен бути інтуїтивно зручним для користувача;

2. Надійність:

- Доступність - час, потрібний для обслуговування системи не повинен перевищувати 10% від загального часу роботи;
- Середній час безперервної роботи — 5 робочих днів;
- Максимальна норма помилок та дефектів в роботі програми - 1 помилка на 1000 запитів користувача;

3. Продуктивність:

Система повинна підтримувати мінімум 25 одночасно працюючих користувачів, пов'язаних з спільною базою даних.

4. Можливість експлуатації:

- Масштабування – система повинна мати можливість збільшувати потужності (продуктивність), зі збільшенням користувачів таким чином, щоб це аж ні як негативно не відобразилося на її роботі;
- Оновлення версій – Оновлення версій повинно здійснюватися в автоматичному режимі.

Системні вимоги:

1. Вимоги до середовища виконання: Система повинна задовільнять вимогам на комп'ютері (клієнті), що знаходиться в наступній мінімальній комплектації:

- 2 Гб оперативної пам'яті;
- Процесор з тактовою частотою не нижче 2.0 GHz;
- Операційна система не нижче Windows 7;
- Браузер (Опера не нижче 36.0, Firefox не нижче 41.0, Google Chrome не нижче 50.0).

2. Вимоги до середовища виконання: Система повинна задовільнять вимогам на комп'ютері (сервер), що знаходиться в наступній мінімальній комплектації:

- 4 Гб оперативної пам'яті;
- 4 ядерний процесор з тактовою частотою не нижче 2.5 GHz;
- Вимоги до операційного середовища (для сервера UNIX, до web-сервера Apache або Nginx);
- СУБД MySQL версії не нижче 5.5;
- PHP версії не нижче 5.4.

Бізнес правила:

Система повинна відповідати всім стандартам інтерфейсу користувача Microsoft Windows.

Антивимоги:

1. Не використовує інформацію, заборонену нормами моралі та законодавством;
2. Не використовує інформацію, що є недійсною або може зашкодити психіці користувачів;
3. Введення стоп-словників для операторів.

Бізнес логіка:

Реклама сайту на інших сайтах. Умова: власні кошти

Висновки до розділу 1

В даному розділі було визначено завдання кваліфікаційної роботи, вимоги до даного програмного забезпечення, вибір та обґрунтування інструментів розробки, архітектури та алгоритмів системи.

В першому підрозділі була описана постановка задачі, в тому числі назва, призначення веб сайту та основні вимоги.

В свою чергу, в другому підрозділі розглянуто було три аналоги даного проекту та зважені всі плюси та мінуси кожного.

В третьому підрозділі була описана архітектура, використана для розробки сайту, а саме MVC шаблон.

В останньому підрозділі було детальніше подано вимоги до проекту, а також описані та обґрунтовані інструментальні засоби, вибрані для даної розробки.

РОЗДІЛ 2. ПРОЕКТУВАННЯ ВЕБ-САЙТУ

2.1 Визначення варіантів використання та об'єктно-орієнтованої системи

Детально проаналізувавши умови, можна побудувати діаграму варіантів використання (Use Case Diagram).

Діаграма прецедентів (Use case diagram, діаграма варіантів використання) – діаграма, на якій відображені відносини, що існують між акторами і прецедентами. Основне завдання – засіб, що дає можливість замовнику, кінцевому користувачеві і розробнику спільно обговорювати функціональність і поведінку системи.

Діаграма прецедентів є графом, що складається з множини акторів, прецедентів (варіантів використання) обмежених межею системи (прямокутник), асоціацій між акторами та прецедентами, відношень серед прецедентів, та відношень узагальнення між акторами. Діаграми прецедентів відображають елементи моделі варіантів використання.

Суть діаграми прецедентів полягає в тому, що проєктована система подається у вигляді множини сутностей чи акторів, що взаємодіють із системою за допомогою так званих варіантів використання. Варіант використання використовують для описання послуг, які система надає актору. Іншими словами, кожен варіант використання визначає деякий набір дій, який виконує система під час діалогу з актором. При цьому нічого не говориться про те, яким чином буде реалізовано взаємодію акторів із системою.

У мові UML є кілька стандартних видів відношень між акторами і варіантами використання:

- асоціації (англ. association relationship);
- включення (англ. include relationship);
- розширення (англ. extend relationship);
- узагальнення (англ. generalization relationship).

Враховуючи аналіз та все вище згадане маємо діаграму варіантів використання (прецедентів), зображену на рис.1.4.

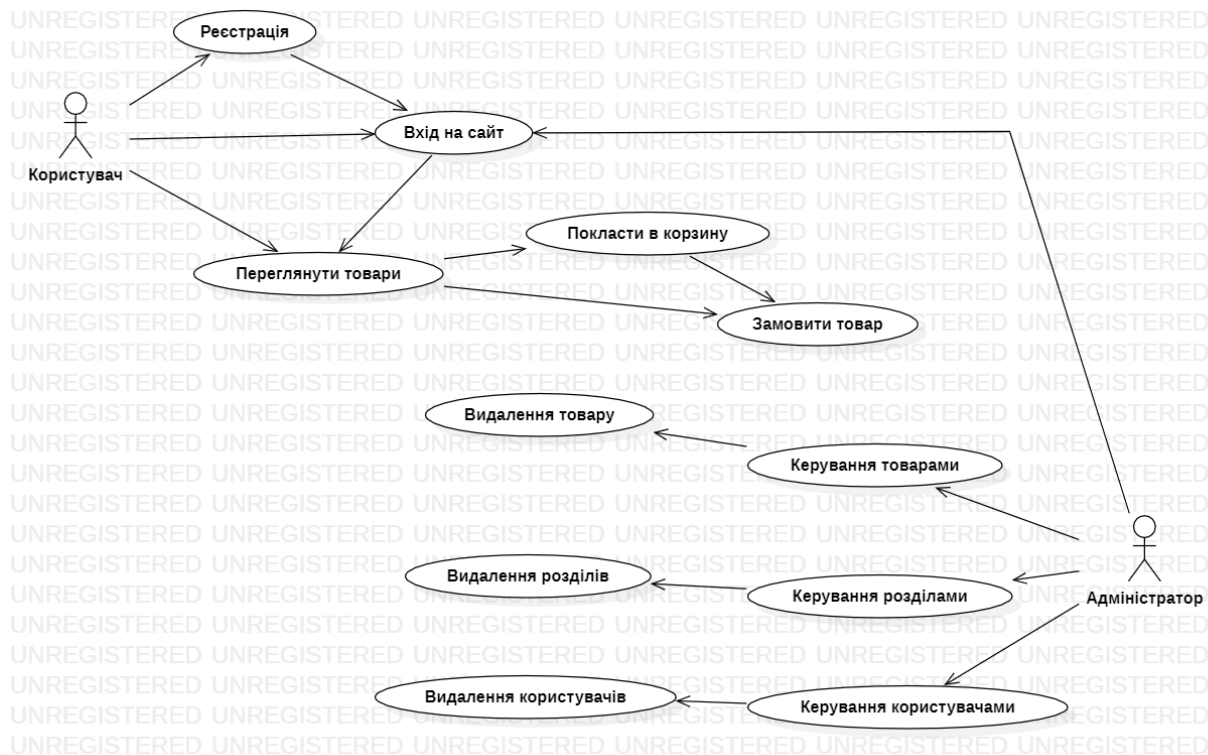


Рисунок 2.1 – Діаграма варіантів використання

Class diagram (діаграма класів) – основна діаграма для створення коду додатка. За допомогою діаграми класів створюється внутрішня структура системи, описується спадкування й взаємне положення класів друг щодо друга. Тут описується логічне представлення системи, тому що класи – це лише заготовки, на основі яких потім будуть визначені фізичні об'єкти.

Клас в мові UML служить для позначення множини об'єктів, які мають однакову структуру, поведінку і відносини з об'єктами інших класів. Графічно клас зображується у вигляді прямокутника, який додатково може бути розділений горизонтальними лініями на розділи або секції. У цих розділах можуть зазначатися ім'я класу, атрибути (змінні) та операції (методи).

Ім'я класу має бути унікальним в межах діаграми, який описується деякою сукупністю діаграм класів або однією діаграмою.

Атрибути класу або властивості записуються у другій зверху секції прямокутника класу.

Квантор видимості може приймати одне із трьох можливих значень і відображається за допомогою відповідних спеціальних символів:

- «+» позначає атрибут з областю видимості типу загальнодоступний (public). Атрибут з цією областю видимості доступний з будь якого іншого класу пакету, в якому визначена діаграма;
- «#» - атрибут із зоною видимості типу захищений (protected). Недоступний для всіх класів, за винятком підкласів даного класу;
- «-» - атрибут із зоною видимості типу закритий «private». Недоступний для всіх класів без винятків;
- «~» - видимість для елементів того ж простору імен (пакета) (package). Квантор видимості може бути опущений. У цьому випадку його відсутність просто означає, що видимість атрибута не вказується.

Ім'я атрибута – рядок тексту, який використовується в якості ідентифікатора відповідного атрибута і тому повинний бути унікальним в межах даного класу.

Методи класу – записуються в третій зверху секції прямокутника.

Спостерігач — це поведінковий патерн проектування, який створює механізм підписки, що дає змогу одним об'єктам стежити й реагувати на події, які відбуваються в інших об'єктах [15].

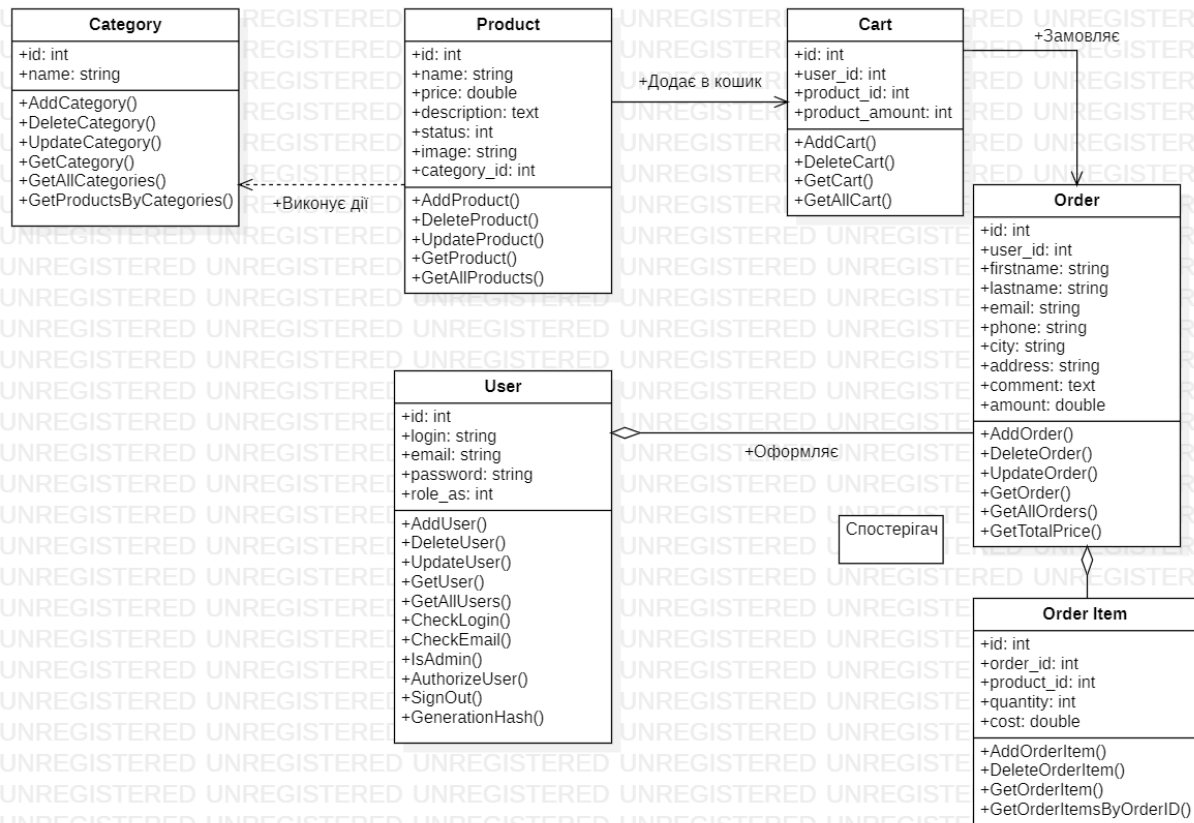


Рисунок 2.2 – Діаграма класів

Опис діаграми класів на рис. 2.2:

Для даної програми було реалізовано 6 класів.

USER, клас, що описує користувачів у системі. Має такі властивості як: email, password – необхідні для авторизації у систему; персональні дані – login, а властивість role_as визначає тип користувача – адміністратор або клієнт. Наявні методи для управління користувачами (AddUser, DeleteUser, UpdateUser). Є методи необхідні для авторизації (CheckLogin, CheckEmail, AuthorizeUser, SignOut) та метод IsAdmin, що необхідний при зверненні до класів PRODUCT, ORDER, ORDER ITEM, CATEGORY, CART, виклається щоб отримати доступ до функціоналу адміністратора.

PRODUCT, клас, що описує товари у системі. Має властивості опису товару (name, price, description, category_id, status). За допомогою category_id має зв'язок з класом CATEGORY. Методи цього класу забезпечують створення нового товару, видалення існуючого, редагування, отримання

списку всіх існуючих(AddProduct, DeleteProduct, UpdateProduct(), GetProduct(), GetAllProducts()).

CATEGORY, клас, що описує категорії товарів системи. Має властивості, що описують дані категорії (name). Методи дозволяють створити, видалити, редагувати, вивести список категорій.

CART, клас, що описує товари, які були додані певним користувачем до кошику. Має властивості, що описують дані товарів у кошику (user_id, product_id, product_amount), які також пов'язують з іншими класами. Методи дозволяють створити, видалити, редагувати, вивести список категорій.

ORDER, клас, що описує замовлення клієнта. Зберігає в собі дані про клієнта та суму (user_id, firstname, lastname, email, phone, city, address, comment, amount). Наявні методи для взаємодії з об'єктами цього класу, такі як створення, видалення, редагування, отримання всіх послуг, отримання загальної вартості (addOrder, deleteOrder, getOrder, updateOrder, getAllOrders, getTotalPrice).

ORDER ITEM, клас, що описує товари, які у замовленні. Зберігає в собі дані товар та замовлення(order_id, product_id, quantity, cost). Наявні методи для взаємодії з об'єктами цього класу, такі як створення, видалення, редагування, отримання всіх послуг (addOrderItem, deleteOrderItem, getOrderItem, getAllOrderItems, getOrderItemsByOrderID).

Також при створенні діаграми класів був використаний патерн спостерігач, згаданий вище. З його використанням було створено збереження товарів з замовлення для кожного окремого замовлення.

2.2 Розробка бази даних проекту

Для даної системи базу даних було створено за допомогою phpMyAdmin, що дозволить розробити БД із використанням графічного інтерфейсу та наочного відображення структури під час змін та редагування.

Для коректної роботи веб сайту потрібно обмежити створення записів у всіх таблицях користувачам, тому буде використано автентифікацію та розподілення ролей.

База даних налічує 6 таблиць, не враховуючи згенерованих додаткових.

Таблиця користувачів містить такі параметри: ім'я, електрона пошта, пароль, роль. Всі поля є обов'язковими, також є додаткові згенеровані поля, як і у інших таблиць.

Наступна таблиця товарів має параметри, як назва, ціна, опис, ідентифікатор категорії, за яким можна визначити категорію, статус, який визначає чи є товар наявності, шлях зображення. Більшість полів мають рядкове значення, окрім опису, ціни, статусу та ідентифікатору категорії. Ціна буде дробовим значенням з двома знаками після коми. Всі поля також не можуть бути порожніми.

Таблиця категорій має основний параметр тільки назву категорії.

Таблиця кошику зберігає всі товари, які додає собі конкретний користувач сайту. Має такі параметри, як ідентифікатор користувача, ідентифікатор продукту та кількість товару. Всі поля мають бути заповненими.

Таблиця замовлень зберігає замовлення кожного користувача, містить такі поля, як ідентифікатор користувача, ім'я, прізвище, електрона пошта, телефон, місто, адресу, коментар до замовлення як єдине не обов'язкове поле, сума замовлення. Всі параметри не можуть бути порожніми окрім коментаря.

Таблиця предметів замовлення зберігає замовленні товари, кожен окремо, для відповідного замовлення, має ідентифікатор замовлення, ідентифікатор самого товару, кількість товару та ціна.

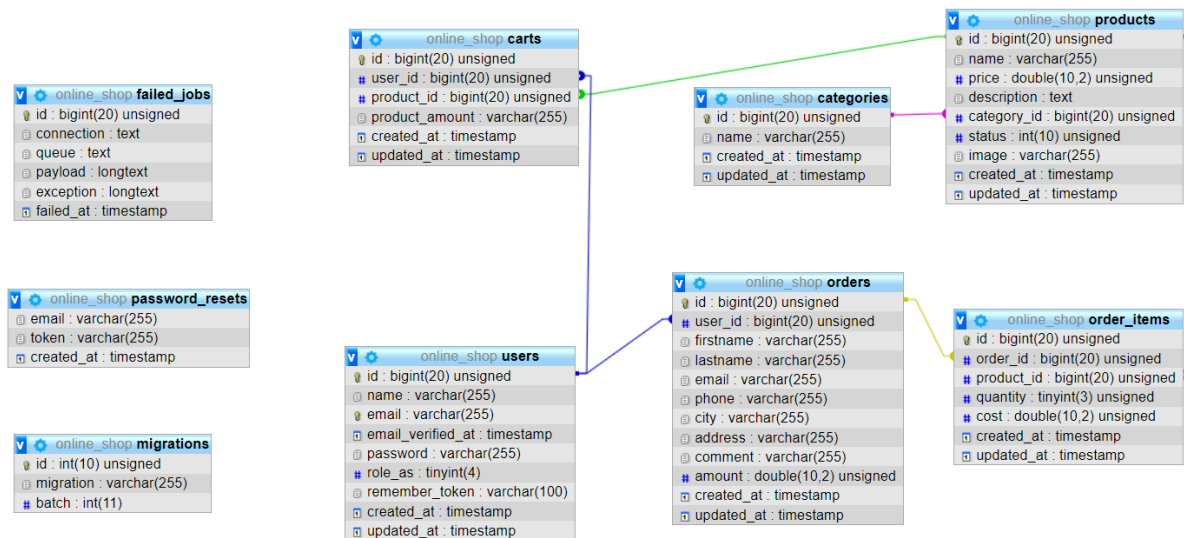


Рис.2.3 – Структура бази даних системи

2.3 Проектування та реалізація алгоритмів роботи системи

Для проектування було також використано діаграми, створенні у Star UML.

Діаграма діяльності (англ. activity diagram) — в UML та SysML, візуальне представлення графу діяльностей. Граф діяльностей є різновидом графу станів скінченного автомату, вершинами якого є певні дії, а переходи відбуваються по завершенню дій.

Дія (англ. action) є фундаментальною одиницею визначення поведінки в специфікації. Дія отримує множину вхідних сигналів, та перетворює їх на множину вихідних сигналів. Одна із цих множин, або обидві водночас, можуть бути порожніми. Виконання дії відповідає виконанню окремої дії. Подібно до цього, виконання діяльності є виконанням окремої діяльності, буквально, включно із виконанням тих дій, що містяться в діяльності. Кожна дія в діяльності може виконуватись один, два, або більше разів під час одного виконання діяльності. Щонайменше, дії мають отримувати дані, перетворювати їх та тестувати, деякі дії можуть вимагати певної послідовності. Специфікація діяльності (на вищих рівнях сумісності) може дозволити виконання декількох (логічних) потоків, та існування механізмів синхронізації для гарантування виконання дій у правильному порядку.

Діаграми активностей будуються з обмеженої кількості фігур, з'єднаних стрілочками. Найважливіші типи фігур:

- округлені прямокутники позначають дії;
- ромби позначають рішення;
- риски позначають початок (розподіл) чи кінець (об'єднання) паралельних активностей;
- чорний кружок позначає старт (початковий стан) процесу;
- чорний кружок в колі позначає кінець (кінцевий стан).

Стрілки ведуть від старту до кінця і позначають порядок в якому відбуваються активності.

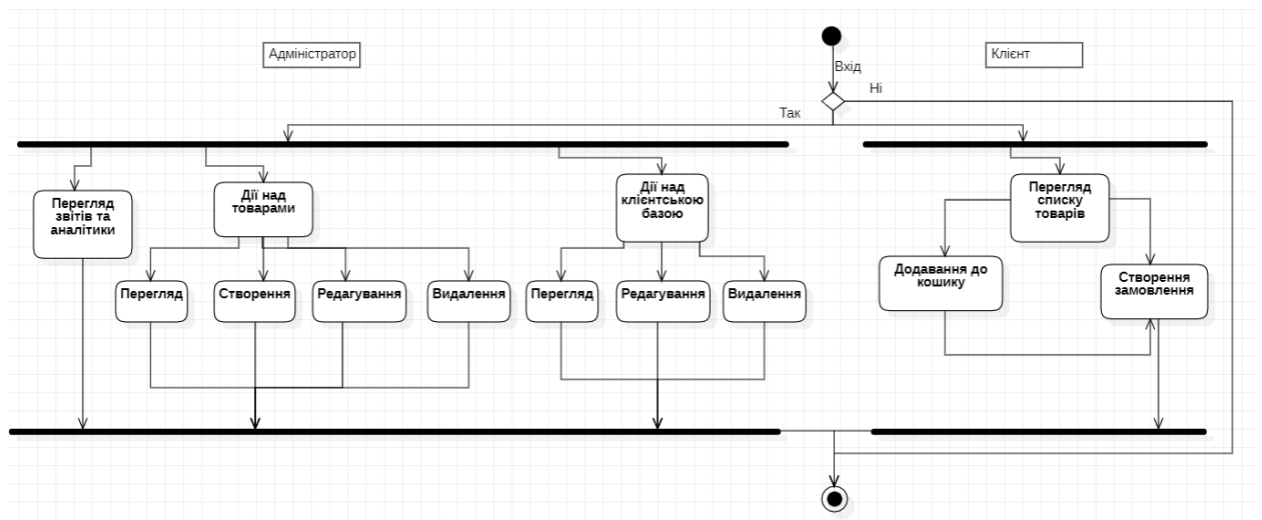


Рисунок 2.4 – Діаграма активності та відповідність станів та дій

Опис діаграми активності на рис. 2.4 (адміністратор):

Опис діаграма активності (адміністратор):

- Вхід (так – перехід до компонентів системи, ні – завершення або перехід до початку);
- Дії над клієнтською базою:
 - Додавання;
 - Видалення;
 - Редагування;
 - Перегляд.
- Дії над товарами:
 - Додавання;

- Видалення;
- Редагування;
- Перегляд.
- Облік і аналіз статистики руху замовлень;
- Перегляд останніх новин системи;
- Завершення роботи.

Опис діаграми активності на рис. 2.4 (клієнт):

- Вхід (так – перехід до компонентів системи, ні – завершення або перехід до початку);
- Дії над замовленнями:
 - Додавання;
 - Видалення;
 - Редагування;
 - Перегляд.
- Дії над товарами:
 - Перегляд.
- Завершення роботи.

Діаграма послідовності — відображає взаємодії об'єктів впорядкованих за часом. Зокрема, такі діаграми відображають задіяні об'єкти та послідовність відправлених повідомлень.

Діаграми послідовностей можна використовувати для уточнення діаграм прецедентів, більш детального опису логіки сценаріїв використання. Це відмінний засіб документування проекту з точки зору сценаріїв використання. Діаграми послідовностей зазвичай містять об'єкти, які взаємодіють у рамках сценарію, повідомлення, якими вони обмінюються, і які повертаються результати, які пов'язані з повідомленнями.



Рисунок 2.5 – Діаграма послідовності користувача

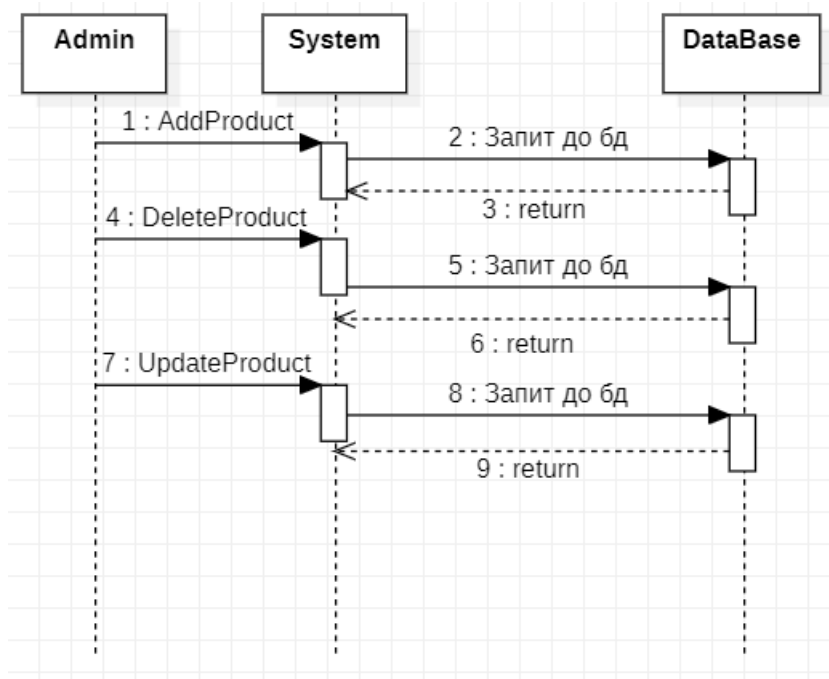


Рисунок 2.6 – Діаграма послідовності адміністратора

На діаграмах на рис. 2.5 та рис. 2.6 продемонстровано як відбуваються різні процеси додавання, видалення та редагування в обох клієнта та адміністратора.

2.4 Реалізація веб-сайту з продажу зоотоварів

Веб-сайт спроектований з використанням MVC (Model-View-Controller). Для цього буде створено основний шаблон проекту з використанням Laravel.

Він складатиметься з декількох сторінок, таких як головна, категорія, товари, кошик, профіль. Також буде створена адміністративна панель.

Ролі визначатимуться в залежності від значення поля `role_as` таблиці `users`. Користувачі з роллю 0 по замовчуванню – звичайні клієнти, а користувач з роллю 1 є адміністратором.

Відповідно до ролей, веб сайт буде розділено на дві частини – фронтенд, та що взаємодіє з користувачами, та бекенд, або панель адміністратора, до якої є доступ тільки у адміністратора, вона дозволяє взаємодіяти з базою даних безпосередньо з сайту.

Основна логіка алгоритму буде написана за допомогою шаблону MVC. Моделі використовуються для керування даними з таблиць баз даних. Контролер, в свою чергу, отримує дані з моделі та перетворює їх на команди, відстежує події і виводить все через вигляд. Вигляд – це представлення будь-якої інформації, їх може бути декілька для однієї моделі та контролера. Вигляди створюються для взаємодії даних з користувачем.

У даному проекті наявні декілька контролерів, для фронтенд частини та адмін частини відповідно: `FrontendController`, `HomeController`, `ProductController`, `CategoryController`, `CartController`, `OrderController` та окремо в папці Admin `ProductController`, `CategoryController`, `FrontendController`.

Розглянемо один з контролерів на прикладі `Admin/ProductController`. Функція з контролера, яка відповідає за створення товару:

```
public function create(Request $request)
{
    $product = new Product();
    if ($request->hasFile('image')) {
        $file = $request->file('image');
        $ext = $file->getClientOriginalExtension();
        $filename = time() . '.' . $ext;
        $file->move('assets/products', $filename);
        $product->image = $filename;
    }
}
```

```

    }
    $product->name = $request->input('name');
    $product->price = $request->input('price');
    $product->description = $request->input('description');
    $product->category_id = $request->input('category_id');
    $product->status = $request->input('status') == TRUE ? '0' : '1';
    $product->save();
    return redirect('admin/products');
}

```

В цьому фрагменті коду показано, що при створенні товару створюється новий об'єкт моделі Product. Якщо отримано файл зображення, шлях до нього зберігається в полі image, при цьому створюється унікальна назва картинки, щоб не створити помилку з однаковими зображеннями. З полів форми функція бере всі інші значення полів та зберігає їх в базу даних. Після цього функція повертає загального списку товарів в панелі адміністратора.

Також за схожим принципом працює функція, яка дозволяє редагувати дані товарів. Фрагмент коду:

```

public function update(Request $request, $id)
{
    $product = Product::find($id);
    if ($request->hasFile('image')) {
        $path = 'assets/products/' . $product->image;
        if (File::exists($path)) {
            File::delete($path);
        }
        $file = $request->file('image');
        $ext = $file->getClientOriginalExtension();
        $filename = time() . '.' . $ext;
        $file->move('assets/products', $filename);
        $product->image = $filename;
    }
    $product->name = $request->input('name');
    $product->price = $request->input('price');
    $product->description = $request->input('description');
    $product->category_id = $request->input('category_id');
    $product->status = $request->input('status') == TRUE ? '0' : '1';
    $product->update();
    return redirect('admin/products');
}

```

Різниця полягає в тому, що дані потрібно переписати, а не створити, то ж використовується існуючий об'єкт моделі Product.

Також є функція для видалення об'єкту моделі.

```
public function delete($id)
{
    $product = Product::find($id);
    if ($product->image) {
        $path = 'assets/products/' . $product->image;
        if (File::exists($path)) {
            File::delete($path);
        }
    }
    $product->delete();
    return redirect('admin/products');
}
```

Вона відправляє запит до бази даних, який видаляє існуючий запис.

Приклади роботи даних функцій наведені нижче.

Додати новий продукт

Назва	Ціна (грн)
тест	150
Опис	
Тестовий опис	
Категорія	Статус (виберіть якщо товар в наявності)
Для собак	☒
Зображення	
Выберите файл1686685813.jpg	
ДОДАТИ	

Рис.2.7 – Форма створення товару

Редагувати

Назва	Ціна (грн)
Royal Canin Sterilised – вологий корм у соусі для стерилізованих котів (від 12 місяці)	41,9
Опис	
Низька активність кішок після стерилізації - міф. Тварина грає, розважається, але не прагне втекти чи помітити територію. Операція продовжує життя вихованця, запобігає розвитку сечостатевої хвороби.	
Категорія	Статус (виберіть якщо товар в наявності)
Для котів	☐



Рис.2.8 – Форма редагування

Також при видаленні об'єкту з'являється попередження:

Подтвердите действие на странице zooshop

Впевнені що хочете видалити?

OK	Отмена
-----------------------------------	---------------------------------------

Рис.2.9 – Підтвердження дії

Аналогічним чином створено CategoryController в папці Admin, та вивід даних на сайті.

Тепер розглянемо контролер, який підтримує весь функціонал кошику в клієнтській частині, CartController.

Фрагмент коду, який відповідає за додавання товарів до кошику користувачем:

```
public function addProduct(Request $request)
{
    $product_id = $request->input('product_id');
    $product_amount = $request->input('product_amount');
    if (Auth::check()) {
        $prod_check = Product::where('id', $product_id)->first();
        if ($prod_check) {
```

```

        if (Cart::where('product_id', $product_id)->where('user_id',
Auth::id()->exists())) {
            return response()->json(['status' => $prod_check->name . "
вже додано в кошик"]);
        } else {
            $cartItem = new Cart();
            $cartItem->product_id = $product_id;
            $cartItem->user_id = Auth::id();
            $cartItem->product_amount = $product_amount;
            $cartItem->save();
            return response()->json(['status' => $prod_check->name . "
додано в кошик"]);
        }
    } else {
        return response()->json(['status' => "Продукту не існує"]);
    }
} else {
    return response()->json(['status' => "Увійдіть в свій аккаунт"]);
}
}
}

```

Для підрахування кількості товару використовується функції інкременту та декременту, написані за допомогою jQuery. Функція передає значення у поле кількості, звідки його зчитує контролер при додаванні товару до кошику.

Приклад функції інкременту:

```

$('.increment-btn').click(function (e) {
    e.preventDefault();

    let increment_value = $(this).closest('.data').find('.amount').val();
    let value = parseInt(increment_value, 10);
    value = isNaN(value) ? 0 : value;
    if(value < 10)
    {
        value++;
        $(this).closest('.data').find('.amount').val(value);
    }
});

```

Схоже написана функція декременту.

Для керування замовлення маємо дві моделі Order та OrderItem до відповідних таблиць у базі даних. В контролері OrderController відбувається звернення до обох моделей. Продемонструємо це на прикладі фрагменту створення нового замовлення:

```
public function createorder(Request $request)
{
    $order = new Order();
    $order->user_id = Auth::id();
    $order->firstname = $request->input('firstname');
    $order->lastname = $request->input('lastname');
    $order->email = $request->input('email');
    $order->phone = $request->input('phone');
    $order->city = $request->input('city');
    $order->address = $request->input('address');
    $order->comment = $request->input('comment');
    $order->amount = $request->input('amount');
    $order->save();

    $cartitems = Cart::where('user_id', Auth::id())->get();
    foreach ($cartitems as $item) {
        OrderItem::create([
            'order_id' => $order->id,
            'product_id' => $item->product_id,
            'quantity' => $item->product_amount,
            'cost' => $item->products->price,
        ]);
    }
    $cartitems = Cart::where('user_id', Auth::id())->get();
    $cartitems->each->delete();

    return redirect('/');
}
```

Для збереження даних до бази даних основна інформація додається до таблиці orders, але для збереження товарів використовується таблиця orderitems, де товари пов'язанні за допомогою ідентифікатора замовлення. Так як у базі даних, створених в phpMyAdmin, не можна зберігати масиви, було вибрано такий спосіб зберігати замовленні товари.

Детальніше лістинг контролерів буде наведено у додатках Б.

					ІПЗ.КР.Б – 121-23 – ІЗ	Арк.
						34

Маршрутизацію веб додатку забезпечує файл web.php. Він відповідає за доступ до всіх сторінок сайту, а також перевіряє чи може даний користувач відкрити конкретну сторінку.

Структура проекту даного інтернет-магазину містить папки з контролерами, моделями, представленнями, стилями, скриптами та папками бібліотек.

Структура папок проекту:

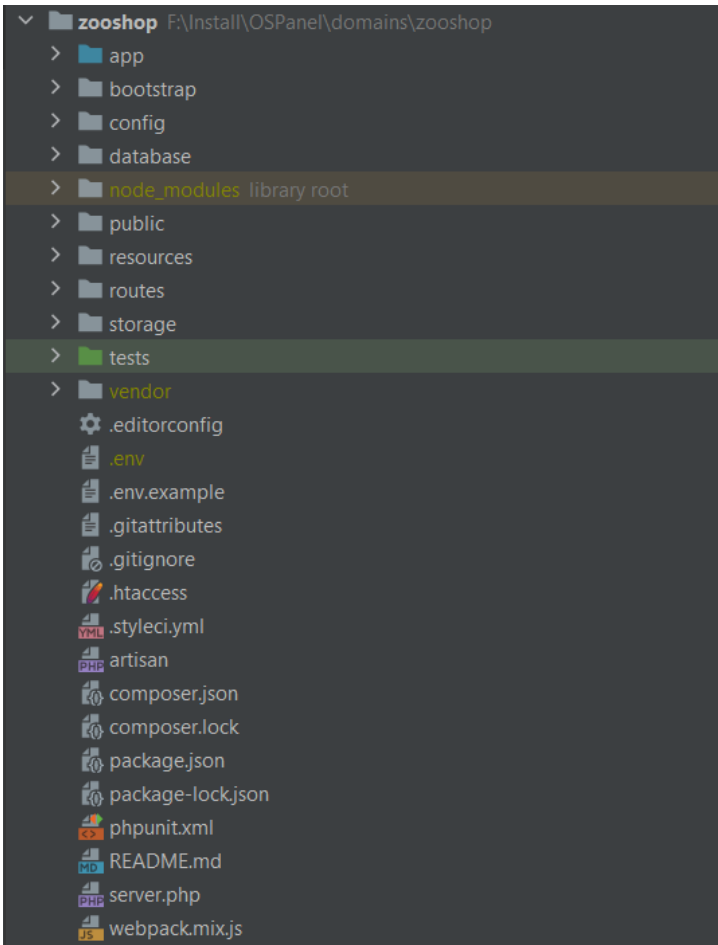


Рис.2.10 – Структура проекту

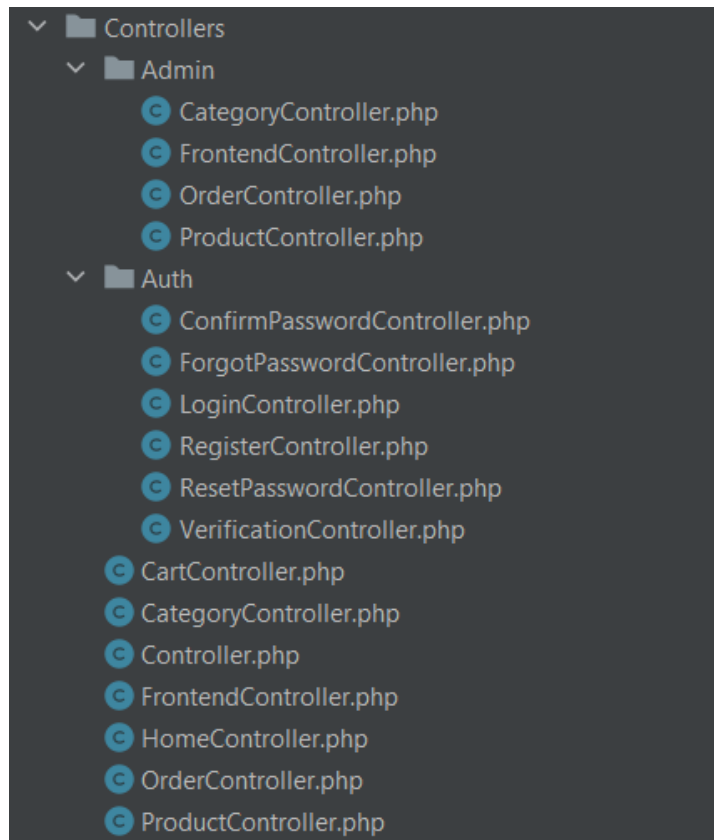


Рис.2.11 – Структура папки контролерів

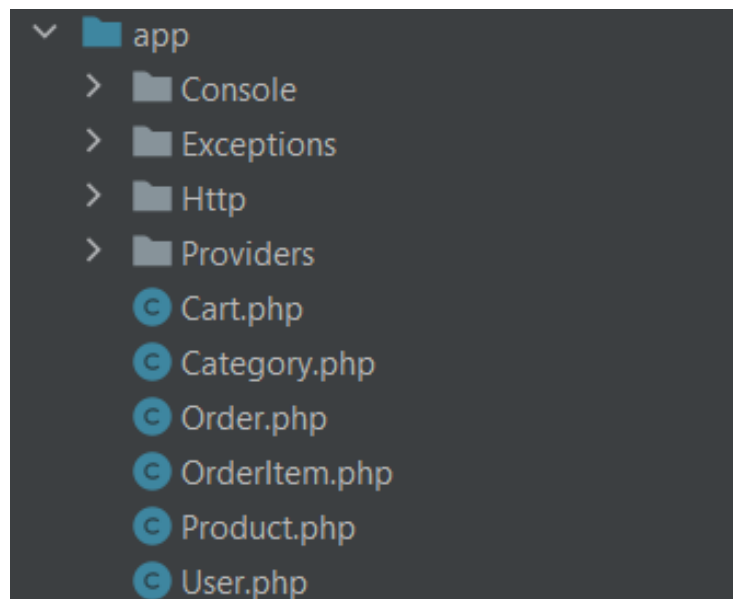


Рис.2.12 – Розміщення моделей

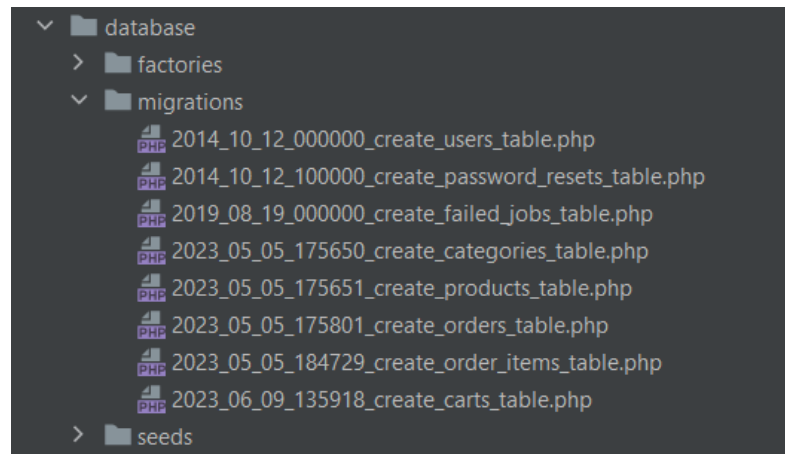


Рис.2.13 – Структура папки database та розміщення файлів міграцій

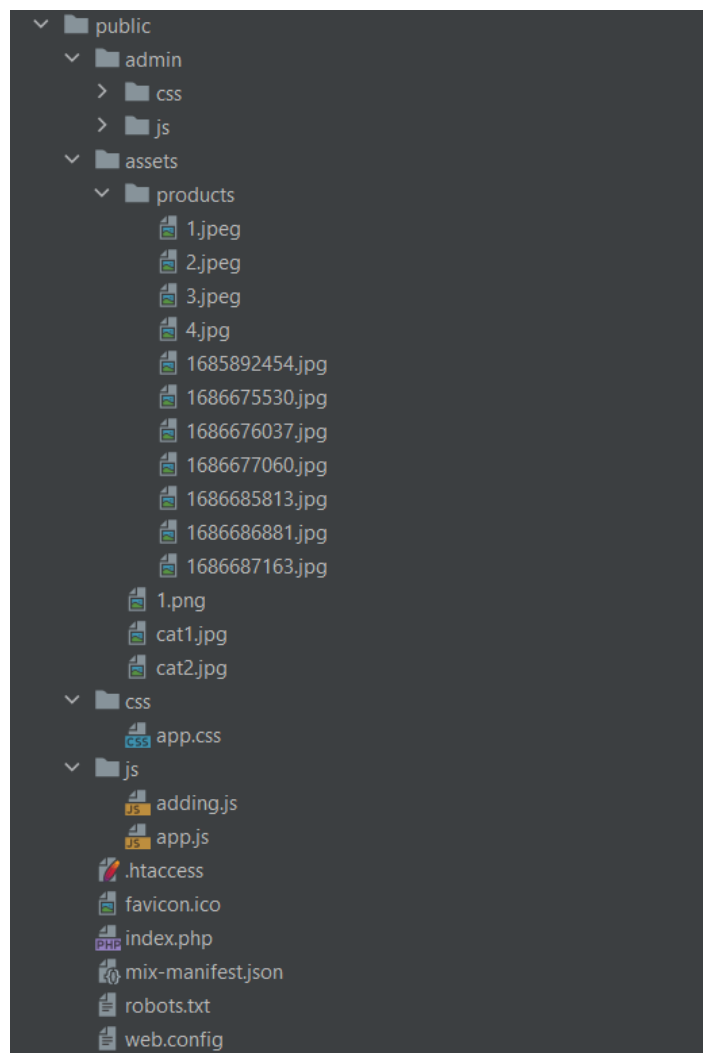


Рис.2.14 – Структура папки public, де знаходяться стилі сайту, скрипти, також картинки, які використовуються на головній або для товарів

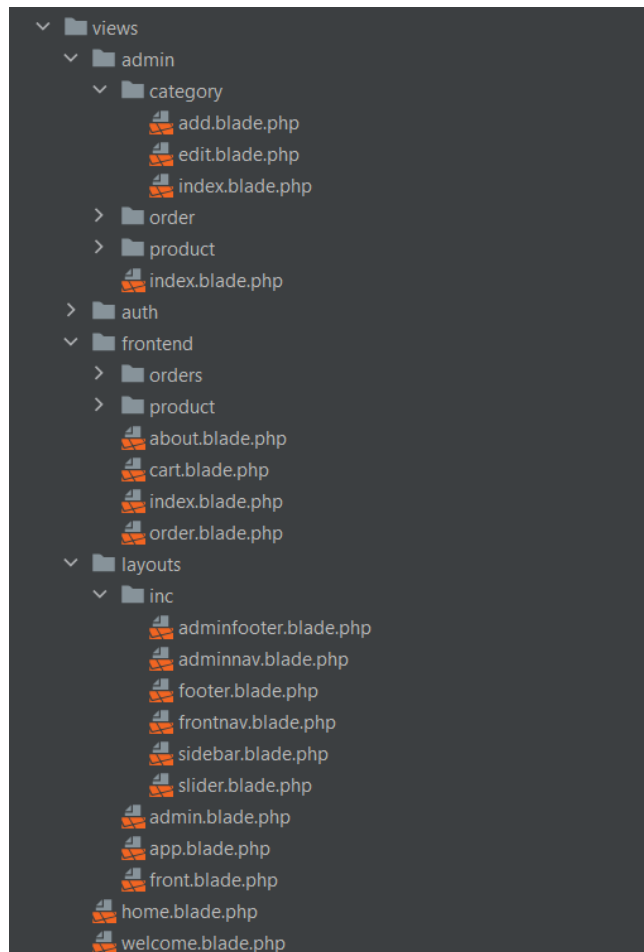


Рис.2.15 – Структура представлень

Висновки до розділу 2

В першому підрозділі були наведенні діаграми варіантів використання та класів з використанням об'єктно-орієнтованого підходу. Усі діаграми продемонстровані на рисунках та описані.

В другому підрозділі було описано базу даних, створення, проектування та роботу з нею.

У третьому підрозділі були описані проектування та алгоритми роботи проекту, а саме показані та описані схеми діяльності і послідовності.

На останок, у четвертому підрозділі було описану реалізацію даної системи. Детальніше розібрані фрагменти коду, які викликають найбільше складнощів при написанні коду.

3. ІНТЕРФЕЙС ТА РОЗРОБКА ВЕБ-САЙТУ

3.1 Порядок налаштування та встановлення системи та визначення прав доступу до системи

В даному підрозділі розглянемо щодо алгоритмів ще дві схеми – діаграму розгортання та діаграму компонентів.

Діаграма розгортання — діаграма на якій відображаються обчислювальні вузли під час роботи програми, компоненти, та об'єкти, що виконуються на цих вузлах. Компоненти відповідають представленню робочих екземплярів одиниць коду. Компоненти, що не мають представлення під час роботи програми на таких діаграмах не відображаються; натомість, їх можна відобразити на діаграмах компонентів. Діаграма розгортання відображає робочі екземпляри компонентів, а діаграма компонентів, натомість, відображає зв'язки між типами компонентів.

Діаграма розгортання в UML моделює фізичне розгортання артефактів на вузлах.

Вузли представляються, як прямокутні паралелепіпеди з артефактами, розташованими в них, зображеними у вигляді прямокутників. Вузли можуть мати підвузли, які представляються, як вкладені прямокутні паралелепіпеди. Один вузол діаграми розгортання може концептуально представляти безліч фізичних вузлів, таких, як кластер серверів баз даних.

Існує два типи вузлів:

- Вузол пристрою;
- Вузол середовища виконання.

Вузли пристроїв - це фізичні обчислювальні ресурси зі своєю пам'яттю і сервісами для виконання програмного забезпечення, такі як звичайні ПК, мобільні телефони.

Вузол середовища виконання - це програмний обчислювальний ресурс, який працює всередині зовнішнього вузла і який надає собою сервіс, який виконує інші виконувані програмні елементи.

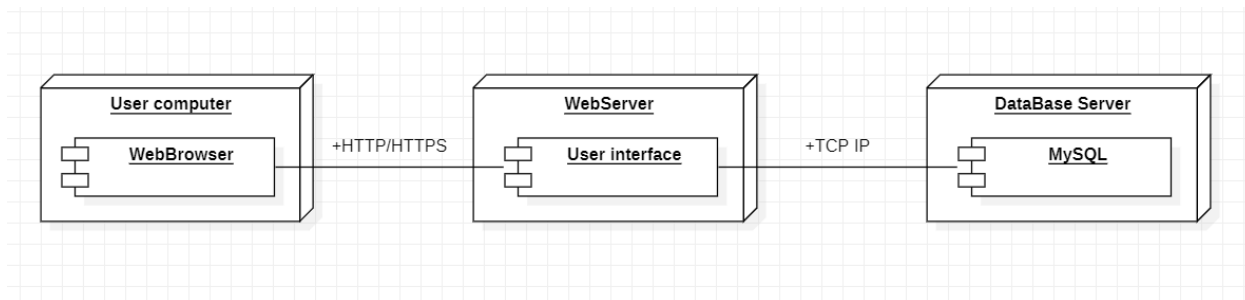


Рисунок 3.1 – Діаграма розгортання

На діаграмі розгортання (рис. 3.1) знаходиться вузол Web Server, саме там знаходяться програмне забезпечення та користувацький інтерфейс системи.

Діаграма компонентів — діаграма, на якій відображаються компоненти, залежності та зв'язки між ними.

Діаграма компонент відображає залежності між компонентами програмного забезпечення, включаючи компоненти вихідних кодів, бінарні компоненти, та компоненти, що можуть виконуватись. Модуль програмного забезпечення може бути представлено як компоненту. Деякі компоненти існують під час компіляції, деякі — під час компонування, а деякі під час роботи програми.

Діаграма компонент відображає лише структурні характеристики, для відображення окремих екземплярів компонент слід використовувати діаграму розгортання.

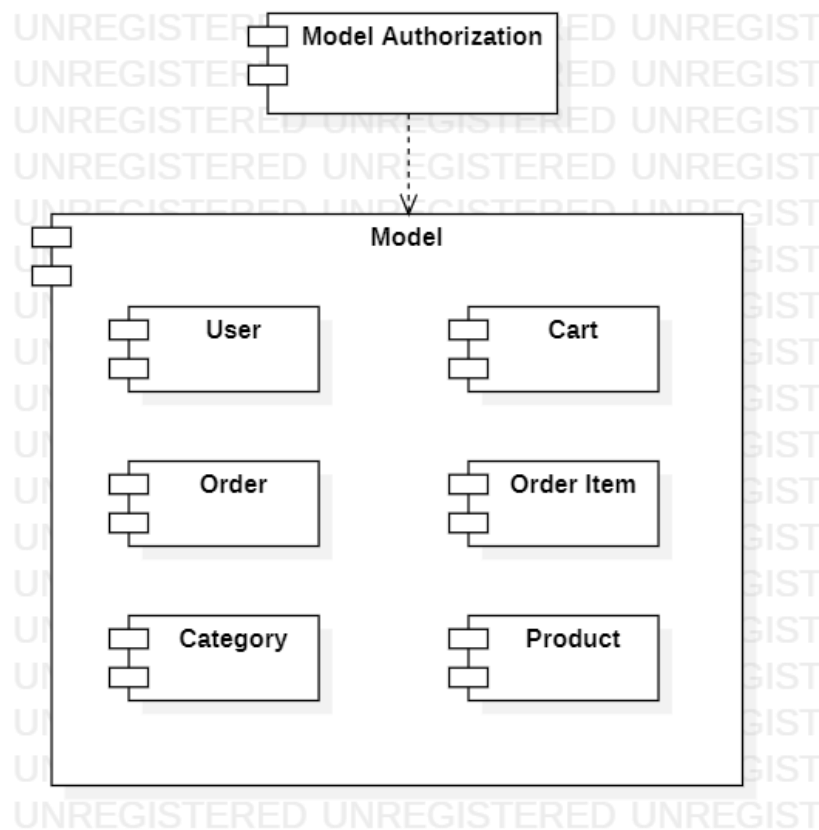


Рисунок 3.2 – Діаграма компонентів

Демонстрація діаграми компонентів програмного комплексу на рис. 3.2.

3.2 Структура інтерфейсу веб-сайту з продажу зоотоварів

В даному підрозділі буде продемонстровано роботу створюваної веб системи.

На початку нас вітає головна сторінка сайту. На ній наявний слайдер та рекомендовані товари внизу. На кожній сторінці є навігаційна панель, яку ми бачимо зверху.

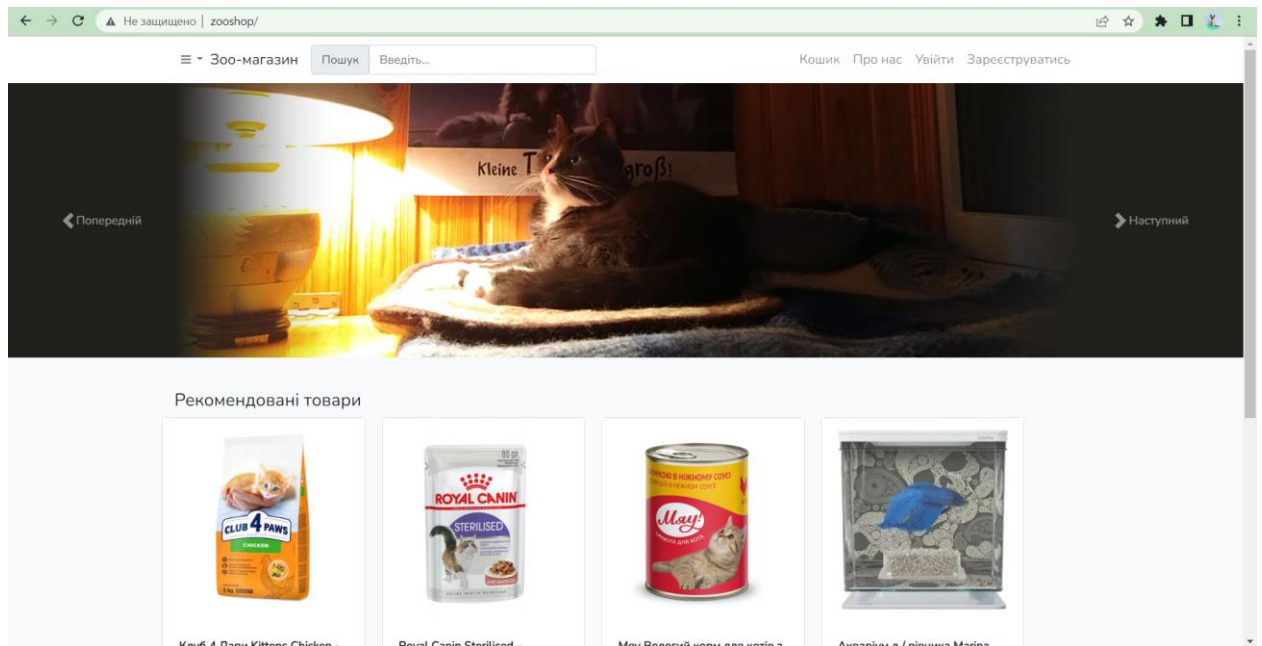


Рисунок 3.3 – Перша сторінка веб додатку

Ми можемо зареєструвати новий аккаунт або увійти в уже існуючий.

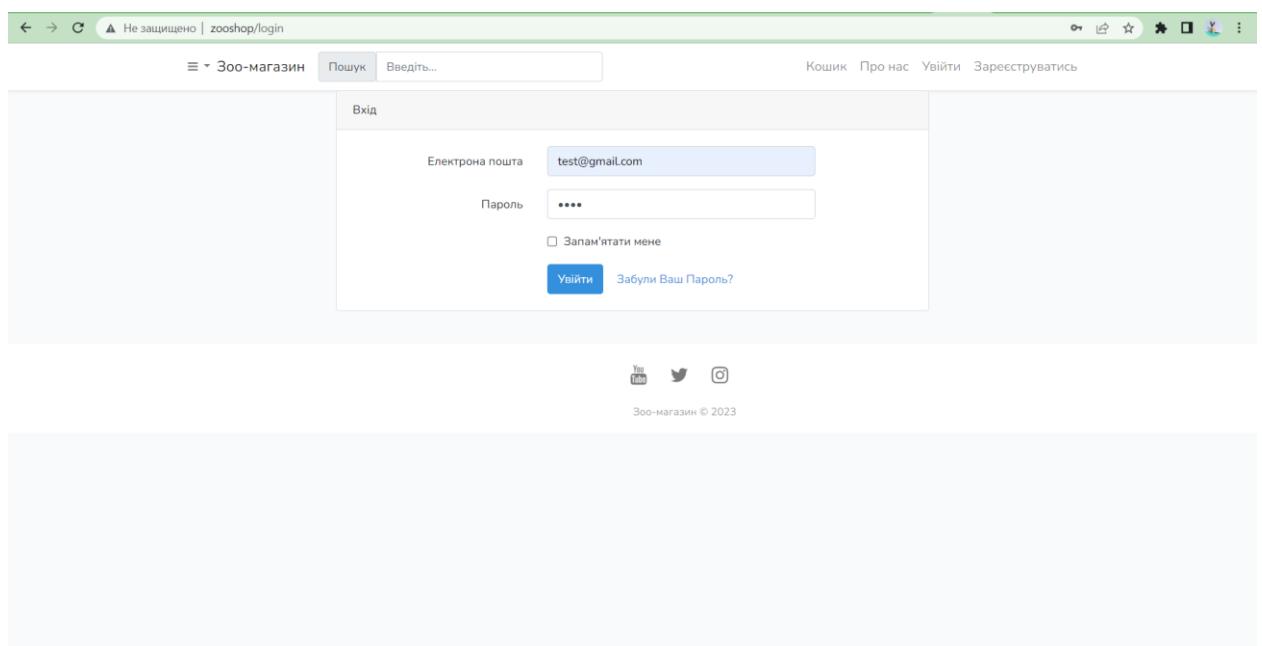


Рисунок 3.4 – Вхід до сайту

При введенні некоректних даних веб сайт видає помилку.

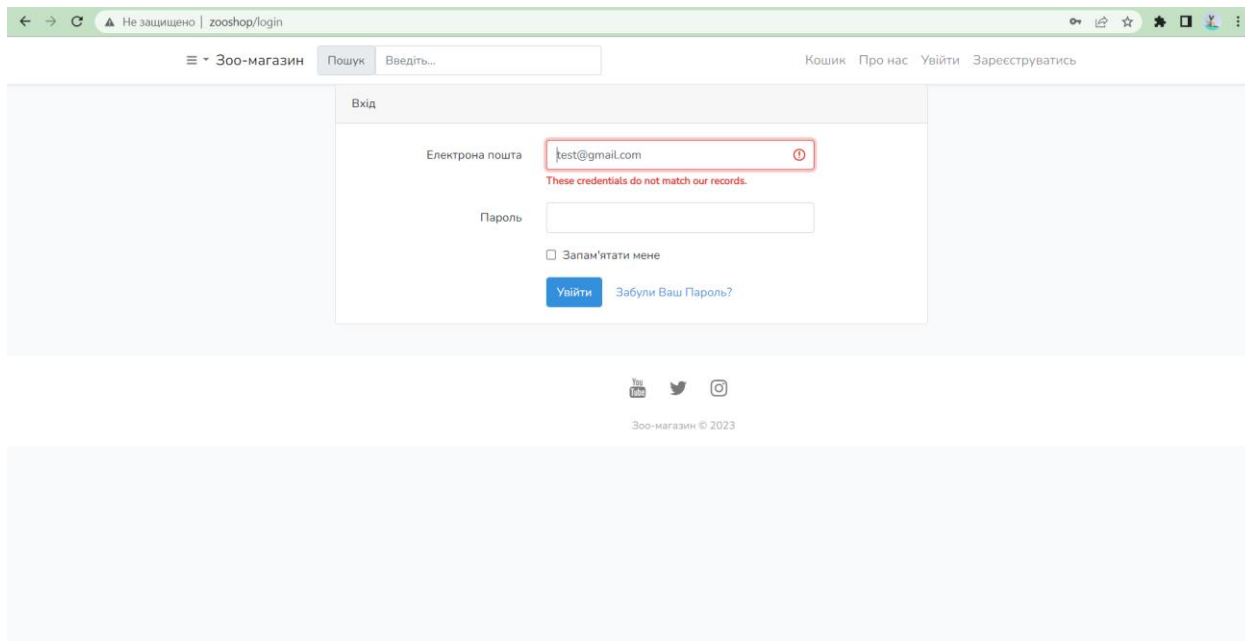


Рисунок 3.5 – Валідація форми

При успішному вході ми будемо бачити своє ім'я в навігаційній панелі, якщо натиснути, відкриється список, який ми побачимо на рис. 3.6.

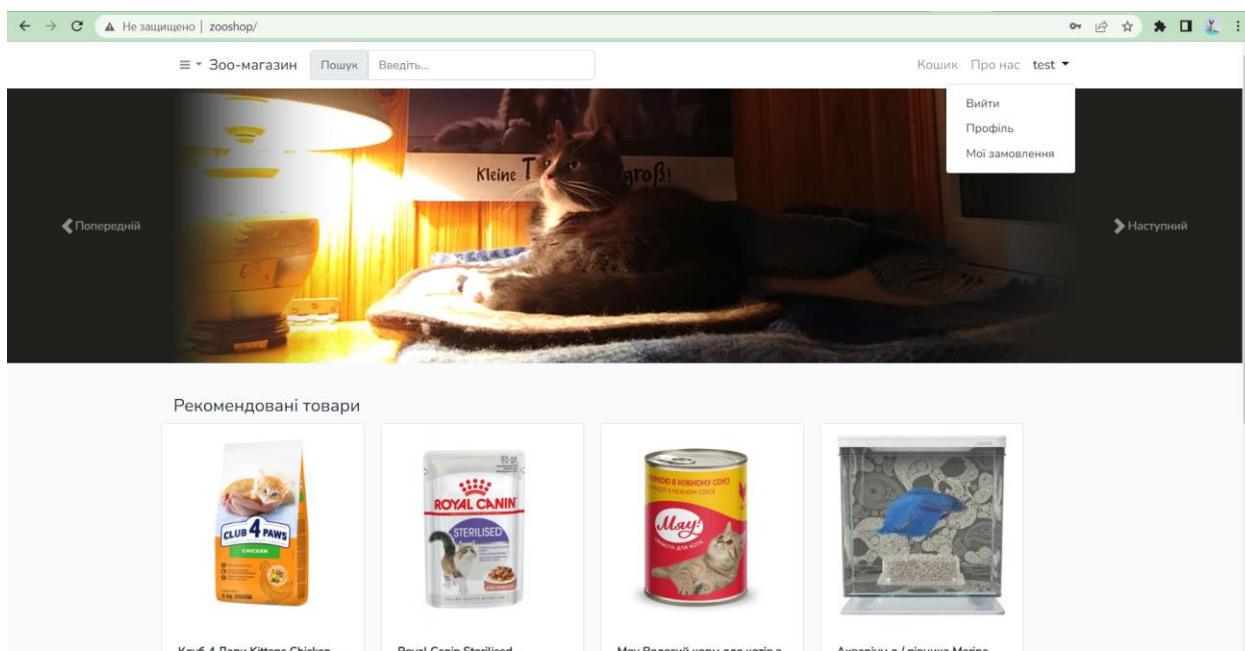


Рисунок 3.6 – Авторизація

Перейдемо до основної частини даної веб орієнтованої системи – товарів. Ми можемо їх переглядати, додавати в кошик та створювати замовлення на основі цього. Щоб переглянути товари, потрібно натиснути на значок списку поряд з назвою. Вибравши категорію, ми побачимо відповідний список товарів.

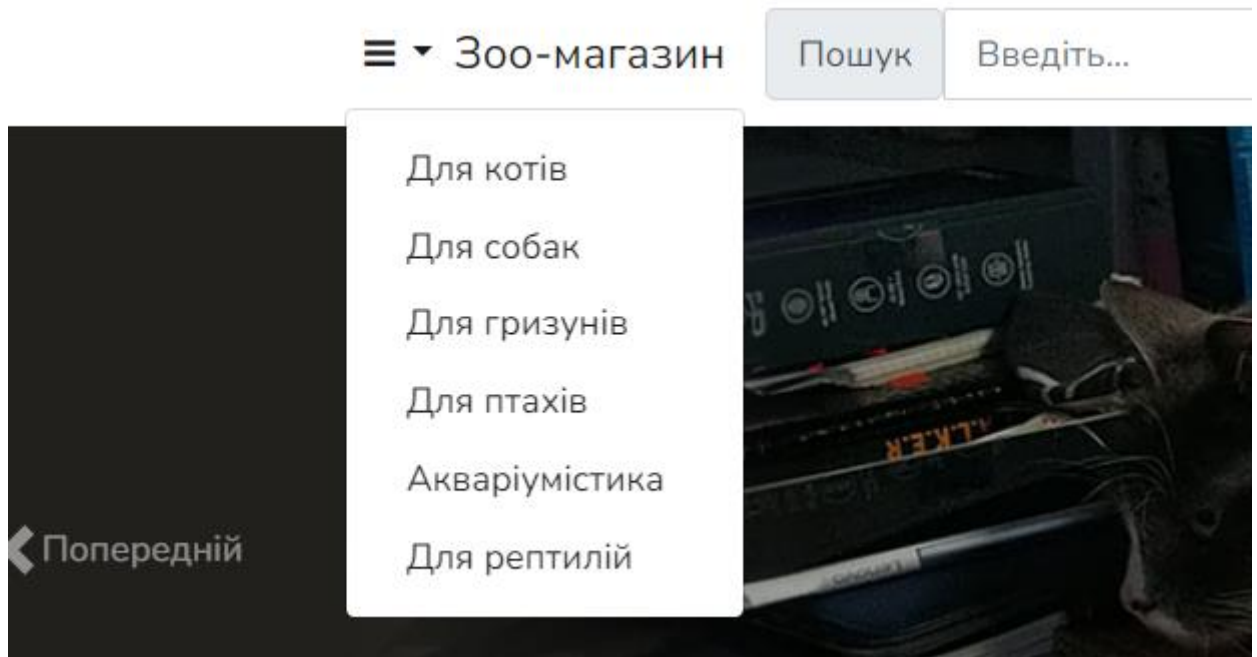


Рисунок 3.7 – Категорії

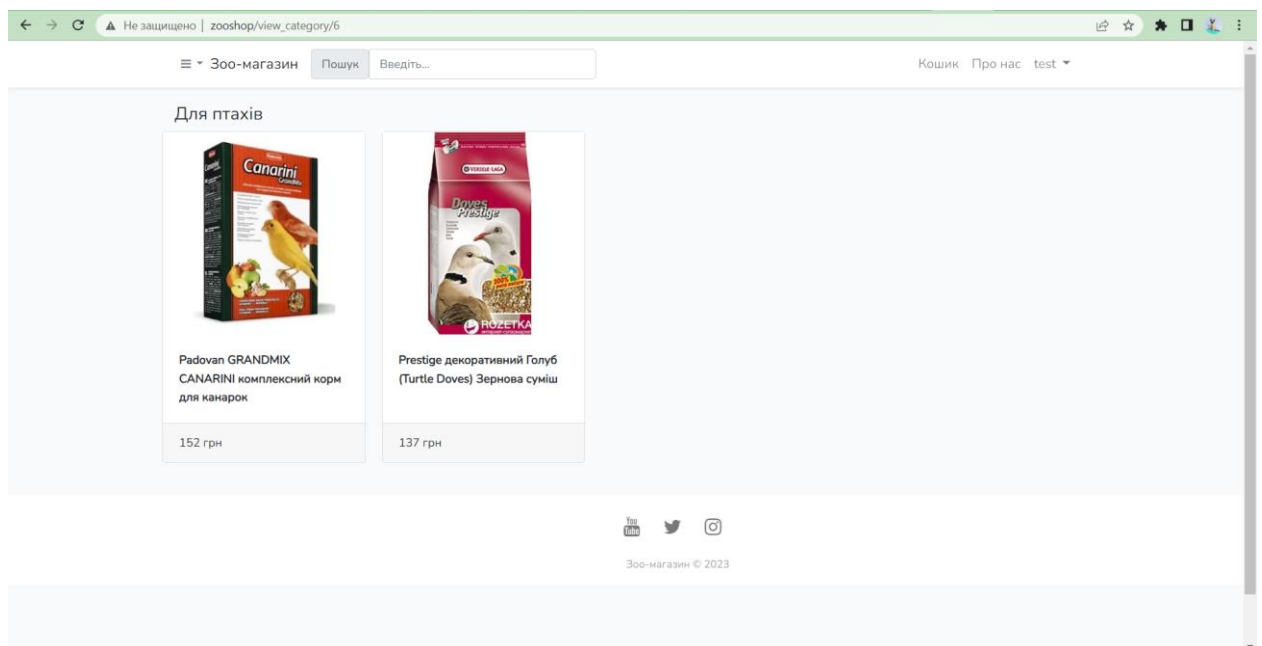


Рисунок 3.8 – Товари за категорією «Для птахів»

При натисканні на назву, ми перейдемо до самої сторінки цього товару. Там можна вибрати кількість продукту та додати в кошик.

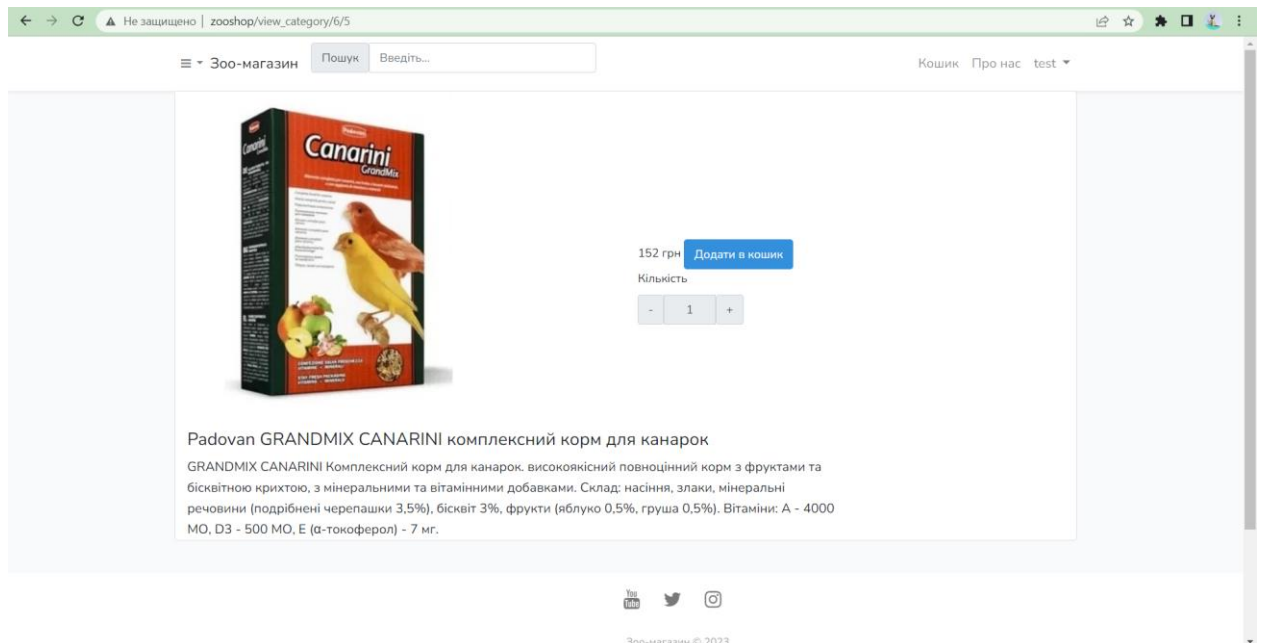


Рисунок 3.9 – Сторінка товару корму для канарок

Якщо спробувати додати товар у кошик не увійшовши в аккаунт, то нічого не відбудеться, так як для збереження даних певний контролер перевіряє авторизацію. А при спробі відкрити кошик нас перекине знову ж таки до форми входу.

Але якщо ми додаємо товар як авторизований користувач, то можемо переглянути товари у кошику. В даному випадку було додано 2 штуки корму для канарок. В кошику ми можемо також побачити повну суму, змінити кількість або видалити товар.

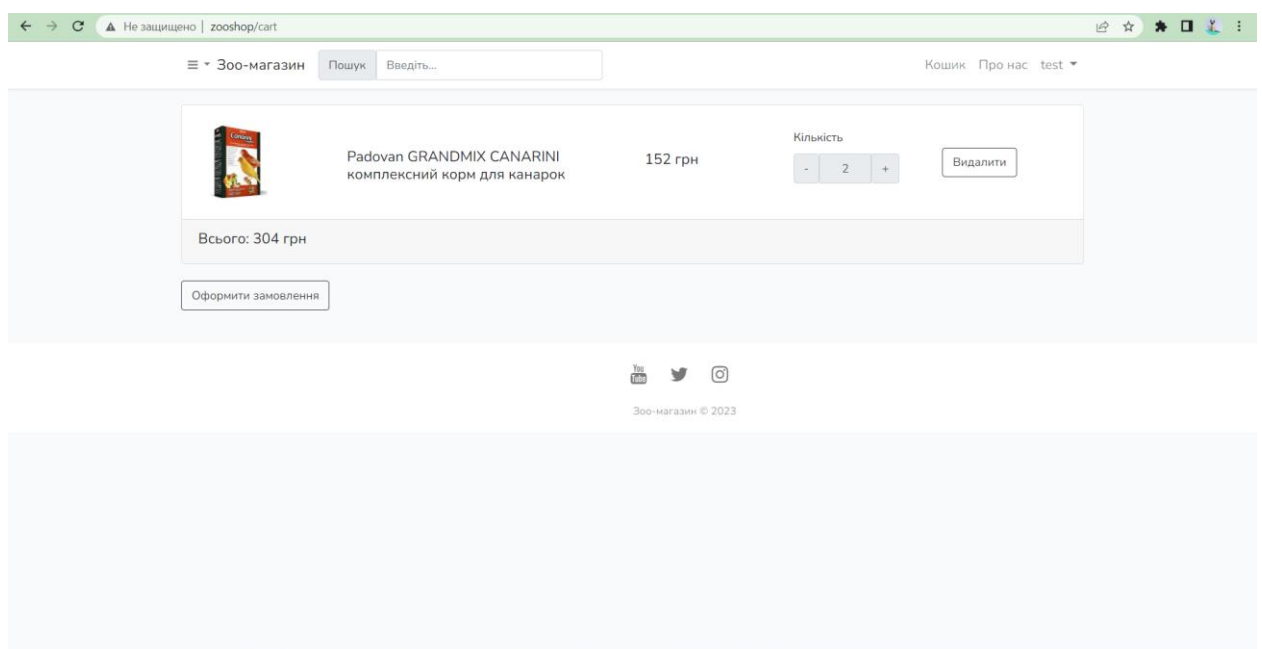


Рисунок 3.10 – Товари у кошику

Якщо більше не хочемо додавати товари, можемо створити замовлення.

При створенні нового замовлення сайт переходить на нову сторінку з формою, яку потрібно заповнити. Збоку знаходиться список товарів. У формі також наявна валідація номеру телефону. Електронна пошта автоматично підставляється з тієї, яку користувач вводить при реєстрації, але її можна змінити (рис. 3.11).

Зоо-магазин

Пошук Введіть...

Кошик Про нас test

Особисті дані

Ім'я

Прізвище

Електронна пошта
test@gmail.com

Номер телефону

Доставка

Місто

Адреса

Коментар (не обов'язково)

Деталі замовлення

Товар	К-ть	Сума
Padovan GRANDMIX CANARINI комплексний корм для канарок	2	304 грн

Всього: 304 грн

Підтвердити замовлення

Рисунок 3.11 – Форма замовлення

При успішному створенні нас перенаправить до головної сторінки. Переглянути замовлення можна натиснувши на своє ім'я в навігаційній панелі та вибравши з випадаючого списку «Мої замовлення».

Зоо-магазин

Пошук Введіть...

Кошик Про нас test

Мої замовлення

Загальна сума	Дата замовлення	
2254.1 грн	2023-06-11 12:46:58	Переглянути
4687.1 грн	2023-06-11 12:51:44	Переглянути
4645.2 грн	2023-06-11 13:03:31	Переглянути
83.8 грн	2023-06-11 13:22:26	Переглянути
41.9 грн	2023-06-12 01:46:59	Переглянути
125.7 грн	2023-06-12 13:52:30	Переглянути
304 грн	2023-06-15 20:06:20	Переглянути

Рисунок 3.12 – Список замовлень

Наше останнє замовлення знаходиться внизу. Натиснувши на кнопку «Переглянути» можна побачити деталі конкретного замовлення.

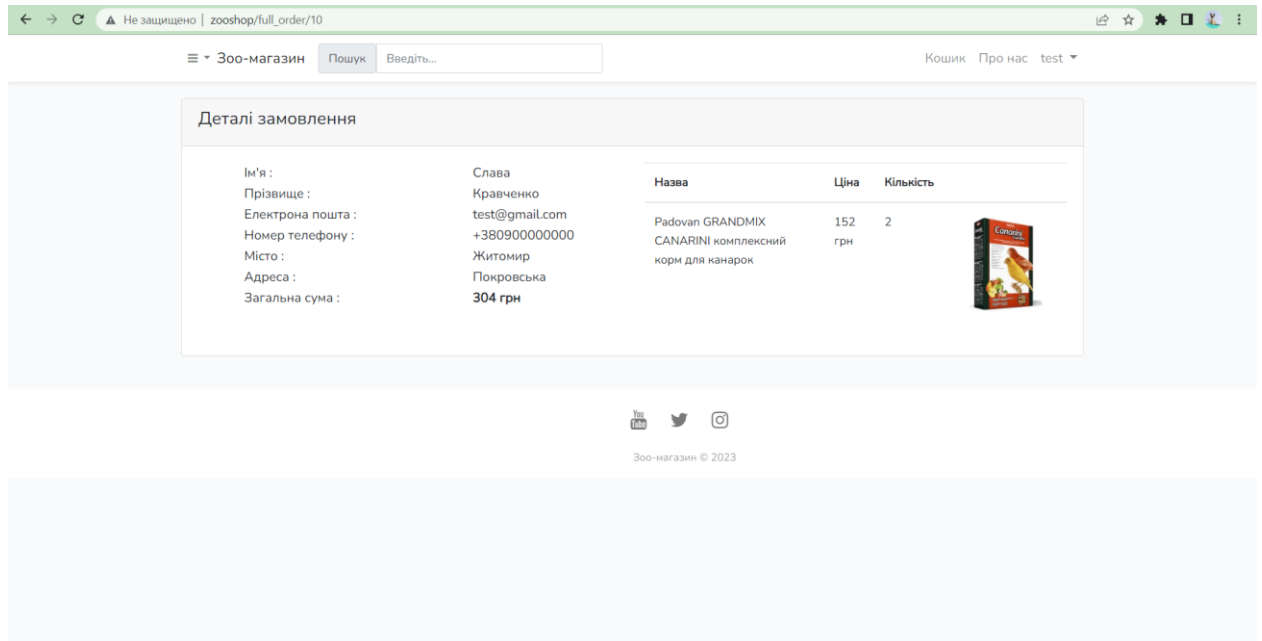


Рисунок 3.13 – Деталі замовлення

Також один із важливих функціоналів сайту – пошук товарів. Пошукове поле розташоване зверху, на навігаційній панелі.

Приклад використання (рис. 3.14 та 3.15).

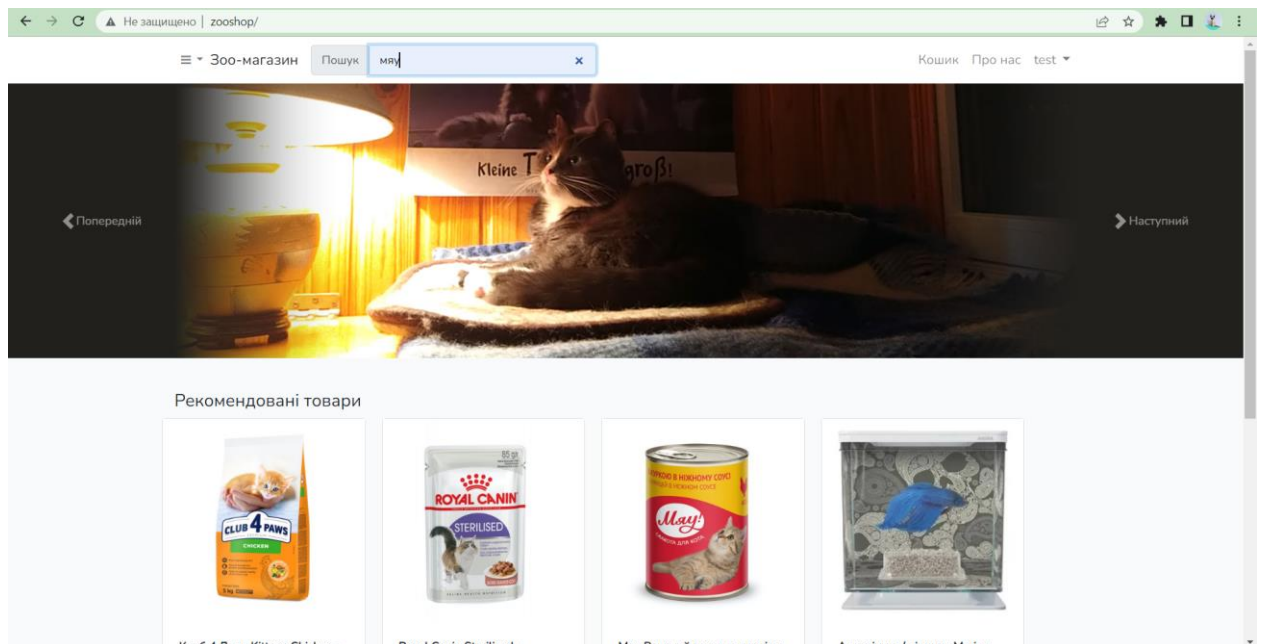


Рисунок 3.14 – Введення назви

При натисканні на ентер сайт перенаправляє до сторінки товару, якщо такий існує. Інакше він повертає назад, звідки ми шукали.

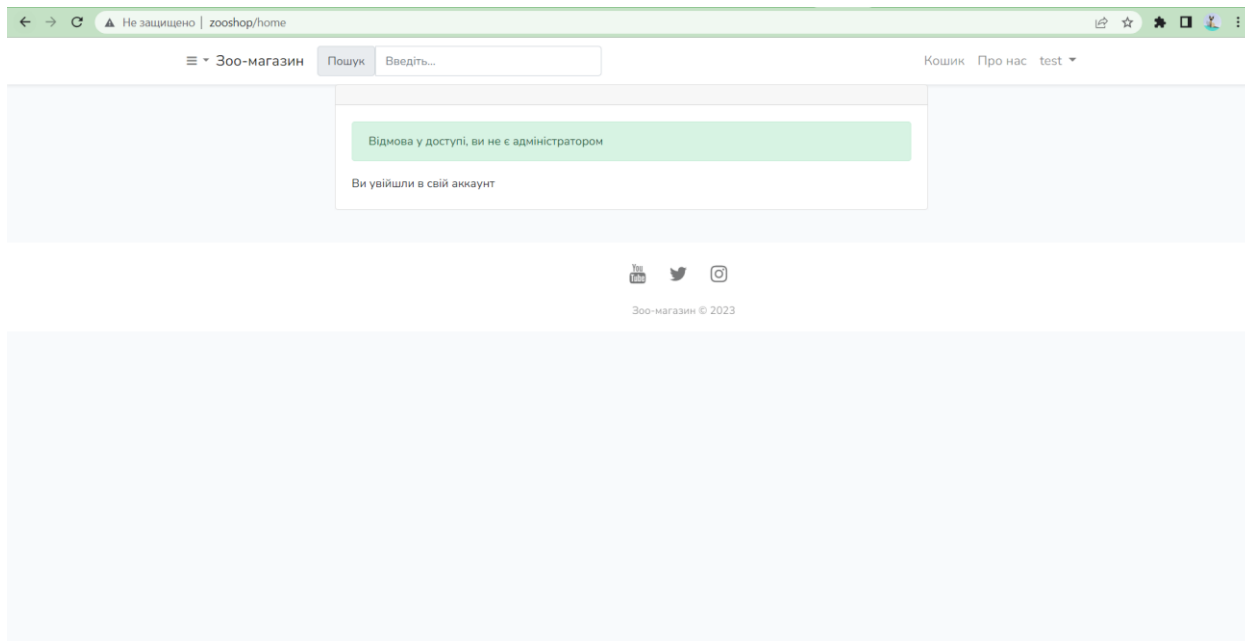


Рисунок 3.17 – Відмова у доступі

В панелі збоку є кнопки-посилання до керування відповідними даними. Розглянемо на прикладі роботи з товарами.

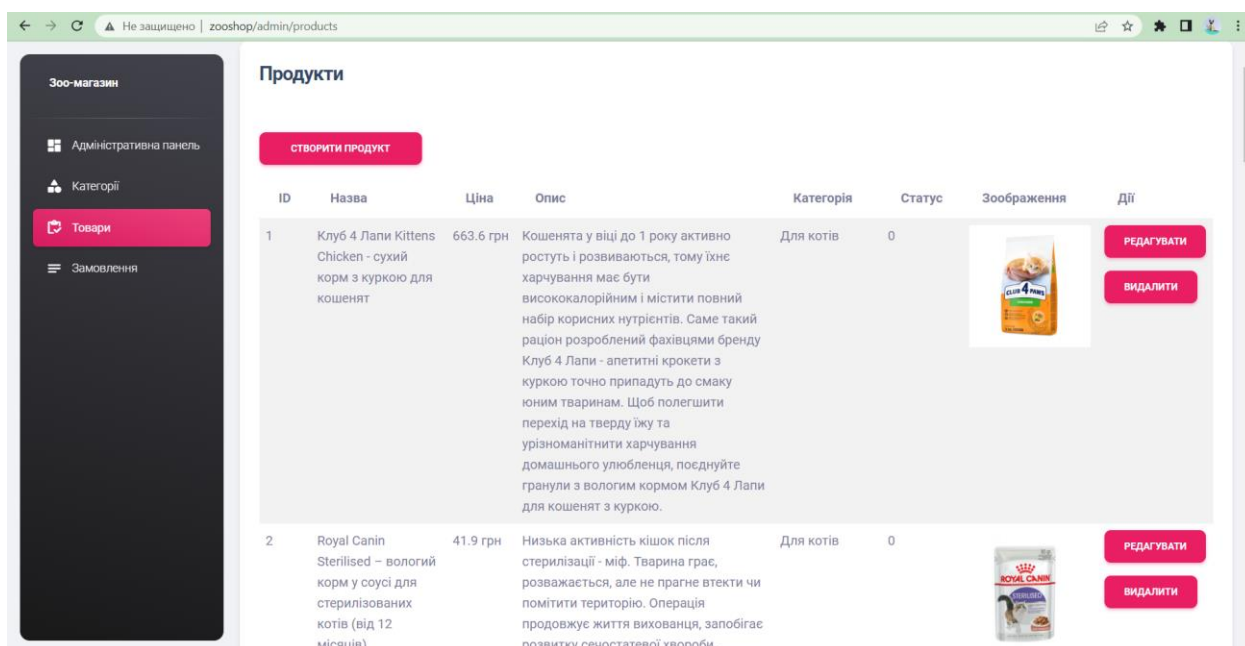


Рисунок 3.18 – Список усіх товарів

Натискаючи на кнопку, перше що ми бачимо це список всіх наявних товарів. Тут адміністратор може додавати, редагувати і видаляти товари.

ЗОО-МАГАЗИН

- Адміністративна панель
- Категорії
- Товари
- Замовлення

Головна / Адміністративна панель
Адміністративна панель

Пошук

Вийти

Додати новий продукт

Назва: Роккі 1 супер-меню корм для щурів, хом'яків і мишей

Ціна (грн): 38

Опис: Вітамінізований корм для дрібних гризунів Роки 1 супер-меню. Ідеально підходить для годування хом'яків та щурів.

Категорія: Для гризунів

Статус (виберіть якщо товар в наявності): ☒

Зображення: Выберите файл4.jpeg

ДОДАТИ

© 2023, made with ❤️ by Creative Tim for a better web.

Рисунок 3.19 – Заповнена форма створення товару

Потрібно правильно заповнити всі данні. Для категорії можна вибрати з випадуючого списку. Статус визначає чи є даний товар зараз у наявності, за замовчуванням при створенні галочка вибрана, тобто товар наявний. Після успішного створення сайт перенаправляє до списку усіх товарів (рис. 3.20).

ЗОО-МАГАЗИН

- Адміністративна панель
- Категорії
- Товари
- Замовлення

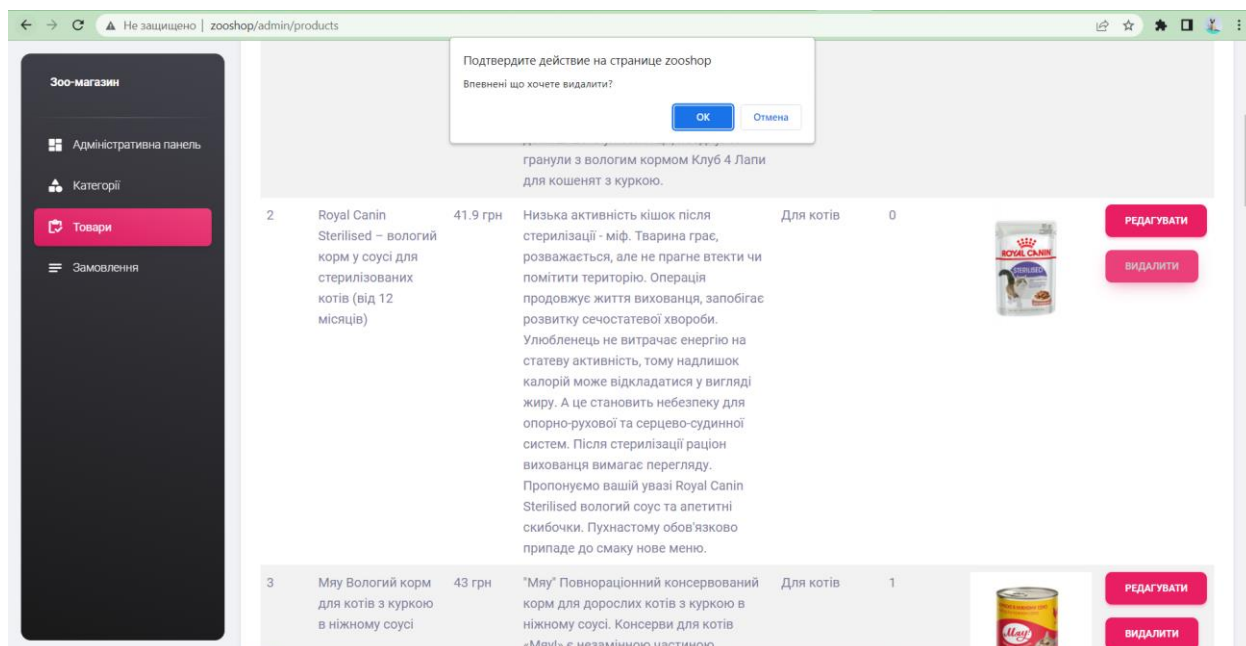
походження, злаки, олії та жири, мінеральні речовини, джерела вуглеводів, риба та рибні субпродукти.

ID	Назва	Ціна	Опис	Категорія	Статус	Дії
9	Лампа інфрачервона для обігріву тварин	185 грн	Високоякісна інфрачервона лампа для обігріву тварин модель E27 (220V, 250W). Використовується для обігріву свинарників, для утримання птиці, для обігріву та поліпшення росту молодяку тварин і птахів (поросят, курчат, каченят, гусенят тощо).	Для рептилій	0	РЕДАГУВАТИ ВИДАЛИТИ
10	Креветки сушені	56 грн	Креветки сушені	Для рептилій	1	РЕДАГУВАТИ ВИДАЛИТИ
11	Роккі 1 супер-меню корм для щурів, хом'яків і мишей	38 грн	Вітамінізований корм для дрібних гризунів Роки 1 супер-меню. Ідеально підходить для годування хом'яків та щурів.	Для гризунів	0	РЕДАГУВАТИ ВИДАЛИТИ

© 2023, made with ❤️ by Creative Tim for a better web.

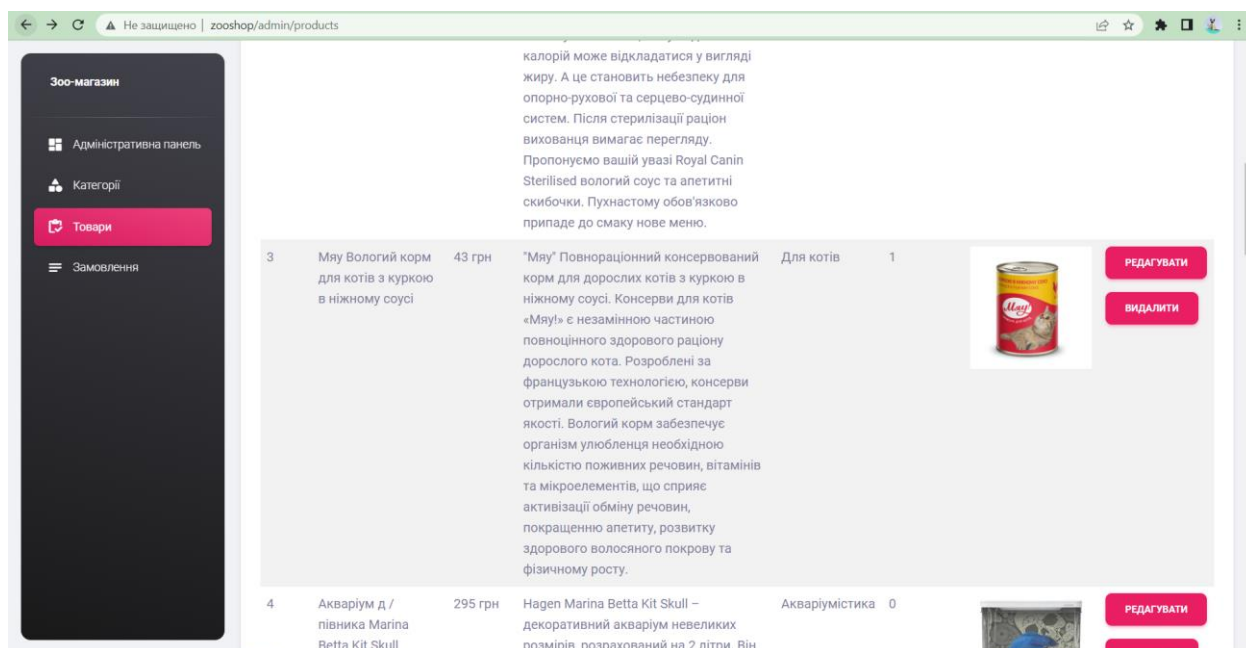
Рисунок 3.20 – Список товарів

При видаленні першим чином з'являється попереджувальне вікно (рис. 3.21).



Форма редагування виглядає дуже схоже на форму створення, але вона перезаписує вже існуючі дані.

Змінімо статус наявності товару нижче.



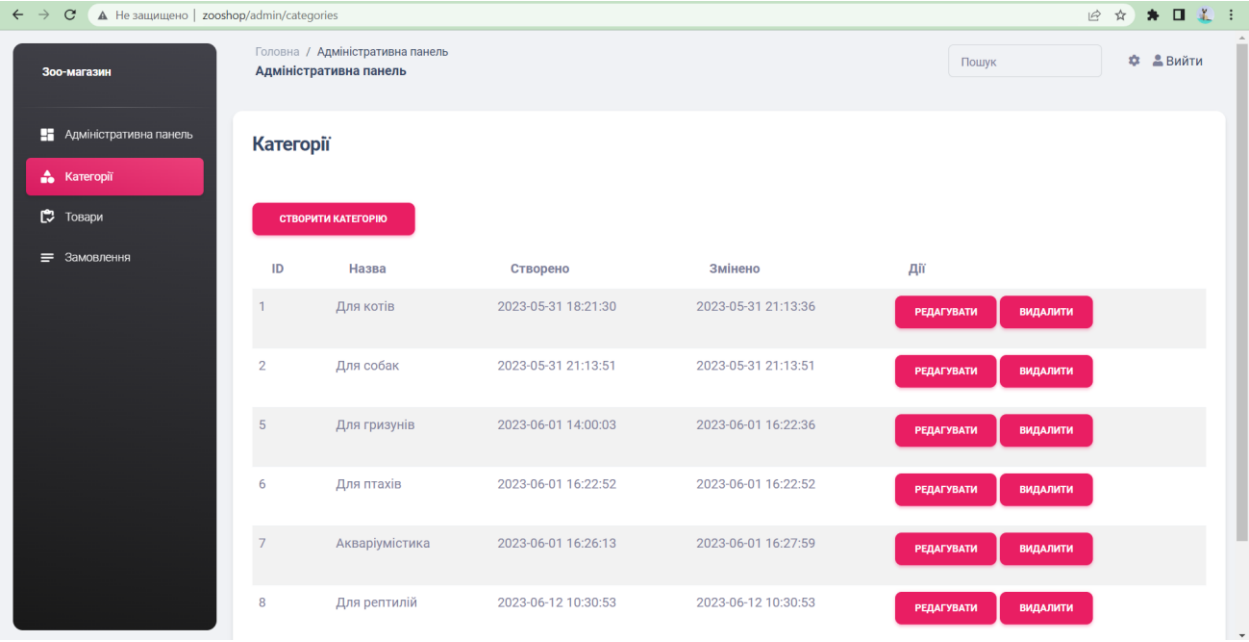


Рисунок 3.25 – Список категорій

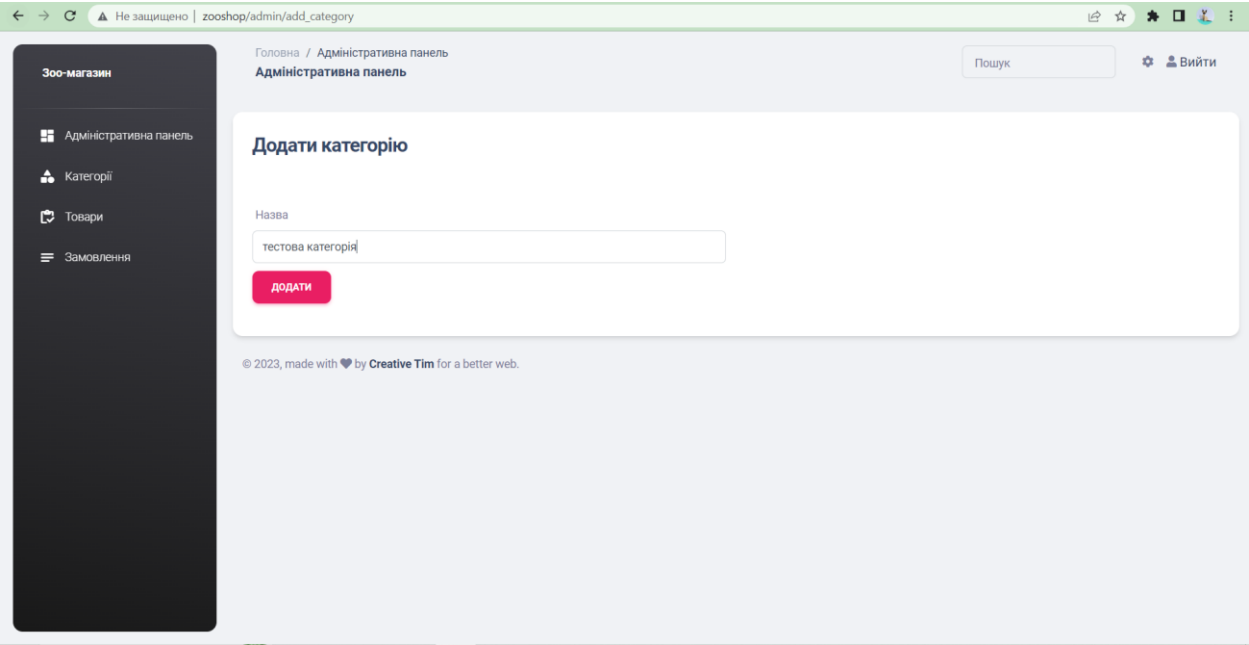


Рисунок 3.26 – Додавання категорії

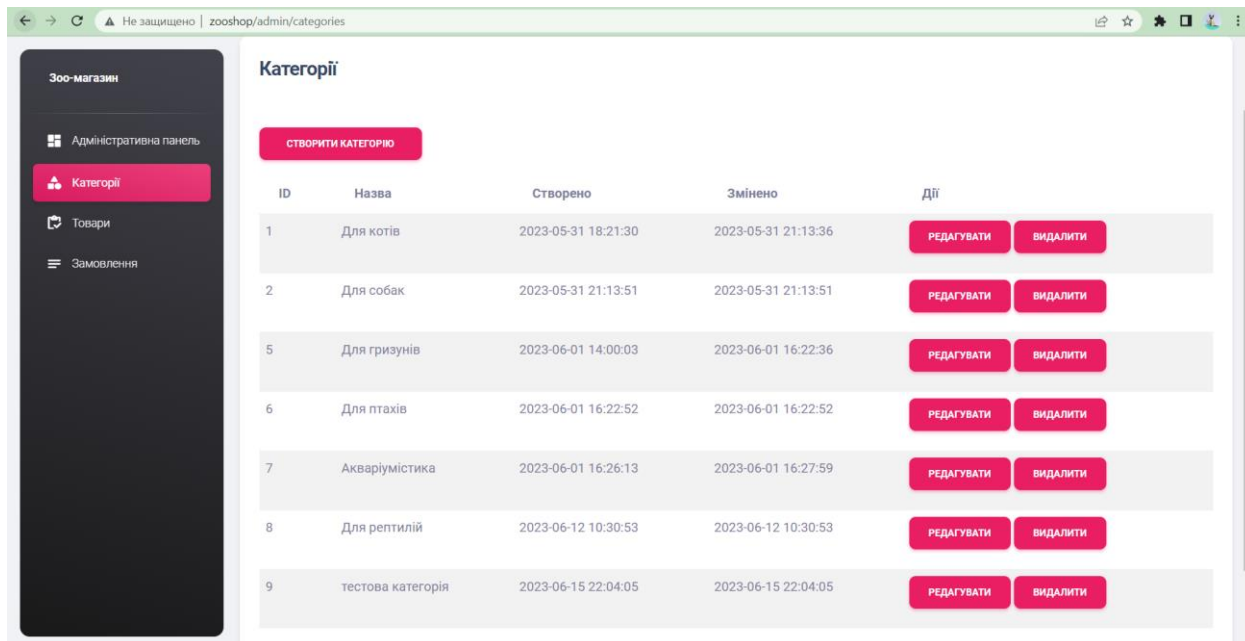


Рисунок 3.27 – Список з новоствореною категорією

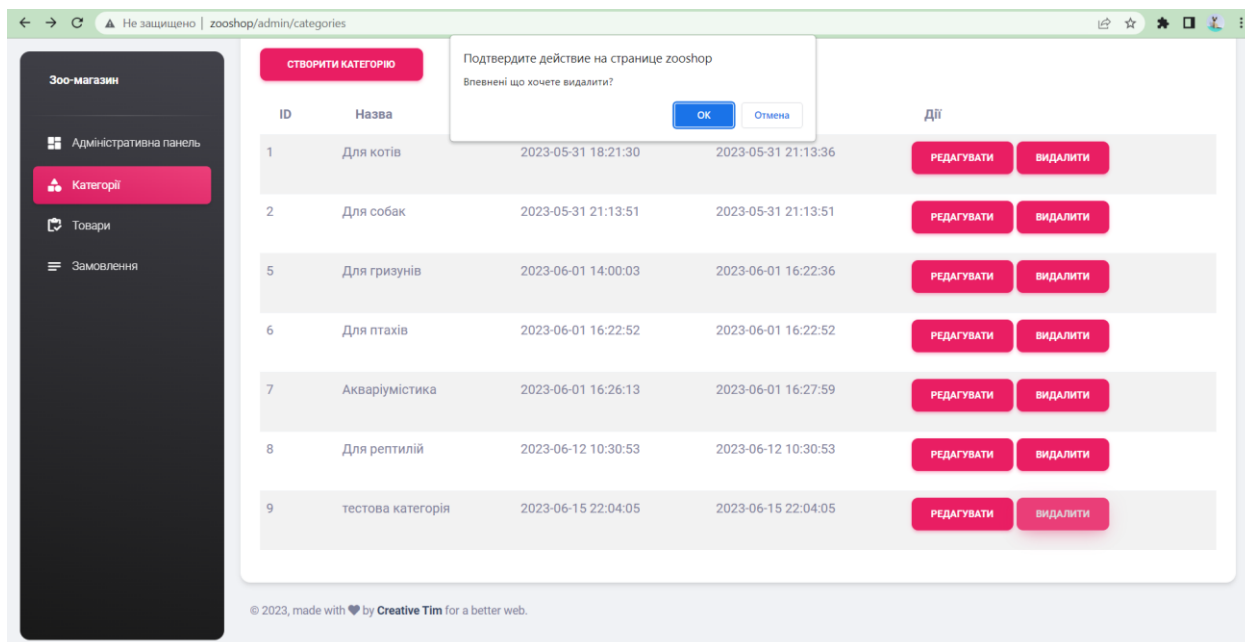


Рисунок 3.28 – Вікно підтвердження видалення

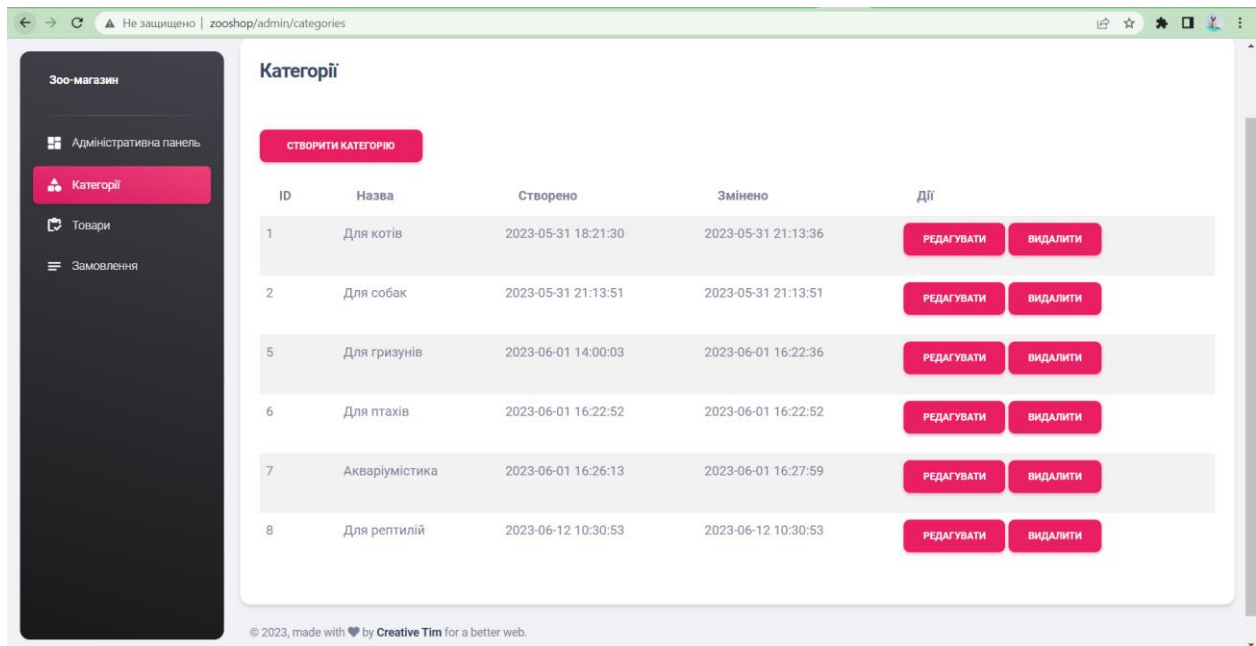


Рисунок 3.29 – Результат видалення

З усього вище описаного можна зробити висновок, що веб орієнтована система справляється з поставленою задачею, вимоги, поставлені у першому розділі, були виконані.

3.3. Тестування роботи програмного продукту

Тестування – це процес перевірки відповідності заявленим вимогам, один з найважливіших процесів у розробці програмного забезпечення. Тому, для перевірки коректності роботи даної веб орієнтованої системи. Спробуємо скористатися основним функціоналом веб сайту, щоб перевірити його працездатність.

1. Вхід в аккаунт

Дії для відтворення: заповнити форму входу даними користувача та натиснути кнопку «Увійти».

Очікувані результати: користувач увійде до аккаунту і буде перенаправлений на домашню сторінку.

Фактичні результати: користувач увійде до аккаунту і буде перенаправлений на домашню сторінку.

2. Додавання товару до кошику

Дії для відтворення: зайти на сторінку будь-якого товару та натиснути «Додати в кошик».

Очікувані результати: товар з'явиться в списку на сторінці «Кошик».

Фактичні результати: товар з'явиться в списку на сторінці «Кошик».

3. Зміна кількості товару у кошику

Дії для відтворення: увійти на сторінку «Кошик» та натиснути на кнопку «+» або «-» біля конкретного товару, змінюючи його кількість.

Очікувані результати: кількість товару автоматично змінюється і залишається в базі.

Фактичні результати: кількість товару автоматично змінюється і залишається в базі.

4. Створення замовлення

Дії для відтворення: авторизованому користувачу зайти на сторінку «Кошик» та вибрати кнопку «Оформити замовлення». На сторінці замовлення ввести коректні дані та натиснути «Підтвердити замовлення».

Очікувані результати: замовлення створенно успішно, дані товари зникнуть з кошику, а номер замовлення з'явиться на сторінці «Мої замовлення», де також можна переглянути його деталі.

Фактичні результати: замовлення створенно успішно, дані товари зникнуть з кошику, а номер замовлення з'явиться на сторінці «Мої замовлення», де також можна переглянути його деталі.

5. Пошук по назві товару

Дії для відтворення: В полі пошуку в панелі навігації зверху ввести назву.

Очікувані результати: веб сайт перенаправить до товару з такою назвою, якщо він існує, інакше перенаправить назад.

Фактичні результати: веб сайт перенаправить до товару з такою назвою, якщо він існує, інакше перенаправить назад.

Висновки до розділу 3

В цьому розділі було продемонстровано порядок налаштування системи та визначення прав доступу до системи, структуру веб сайту та тестування роботи сайту. Було продемонстровано всю роботу сайту, показано обробку різних можливих подій, які могли б викликати помилки або відкрити доступ клієнтам до адміністративних прав. Також були розглянуті схеми розгортання та компонентів.

					ІПЗ.КР.Б – 121-23 – ІЗ	Арк.
						57

ВИСНОВКИ

У даній кваліфікаційній роботі було спроектовано, реалізовано та протестовано веб орієнтовану систему «Веб-сайт з продажу зоотоварів». Протягом виконання даного завдання було продемонстровано роботу з мовою PHP, використання фреймворку Laravel та базою даних MySQL.

Були реалізовані такі функції:

- Збереження даних товарів, категорій, користувачів;
- Перегляд контенту на сторінках веб сайту (товарів, категорій, замовлень);
- Реєстрація та авторизація;
- Основний функціонал – взаємодія користувачів з товарами, тобто додавання їх у кошик, створення замовлення на основі товарів у кошику;
- Адміністративна частина сайту, яка дозволяє керувати даними не тільки напряму в базі даних.

Виходячи з вище перерахованого, даний проект був виконаний відповідно до поставлених вимог та завдання, є функціонуючою робочою системою.

Відповідно було вивчено і засвоєно особливості роботи з фреймворком Laravel, враховано всі можливості даного шаблону, отриманні нові знання у роботі з даною веб орієнтованою системою. Отриманні навички та знання будуть корисні у майбутніх проектах.

					ДУ «Житомирська політехніка».23.121.23.000 – КР	Арк.
						58
Змн.	Арк.	№ докум.	Підпис	Дата		

ЛІТЕРАТУРА

1. Олександр Марголін. UML для бізнес-моделювання: для чого потрібні діаграми процесів [Електронний ресурс] / Олександр Марголін – 2021 – Режим доступу до ресурсу:
<https://evergreens.com.ua/ua/articles/uml-diagrams.html>
2. Можливості діаграми класів [Електронний ресурс] – Режим доступу до ресурсу: <http://www.tsatu.edu.ua/kn/wp-content/uploads/sites/16/laboratorna-robota-12-diahrama-klasiv.pdf>
3. Діаграма послідовності [Електронний ресурс] – Режим доступу до ресурсу: <https://sites.google.com/site/flashukrainian/cxefpagxo/uml/diagrama-poslidovnosti?overridemobile=true>
4. Діаграма компонентів [Електронний ресурс] – Режим доступу до ресурсу: <https://sites.google.com/site/dovidnyk2075/cxefpagxo/uml/diagrama-komponentiv>
5. Діаграма розгортання [Електронний ресурс] – Режим доступу до ресурсу: <https://studfile.net/preview/5010027/page:6/>
6. What is Use Case Diagram? [Електронний ресурс] – Режим доступу до ресурсу: <https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-use-case-diagram/>
7. MVC [Електронний ресурс] – Режим доступу до ресурсу: <https://brander.ua/technologies/mvc>
8. Laravel [Електронний ресурс] – Режим доступу до ресурсу: <https://laravel.com/>
9. Laravel – Artisan [Електронний ресурс] – Режим доступу до ресурсу: <https://www.w3schools.in/laravel/artisan>
10. Laravel Pagination Tutorial [Електронний ресурс] – Режим доступу до ресурсу: <https://laraveltuts.com/laravel-pagination-tutorial/>
11. Artisan Console [Електронний ресурс] – Режим доступу до ресурсу: <https://laravel.com/docs/10.x/artisan>

12. Database: Migrations [Електронний ресурс] – Режим доступу до ресурсу: <https://laravel.com/docs/10.x/migrations>
13. CSRF Protection [Електронний ресурс] – Режим доступу до ресурсу: <https://laravel.com/docs/10.x/csrf#csrf-x-csrf-token>
14. 5.1.15 MySQL Server Time Zone Support [Електронний ресурс] – Режим доступу до ресурсу: <https://dev.mysql.com/doc/refman/8.0/en/time-zone-support.html>
15. Спостерігач [Електронний ресурс] – Режим доступу до ресурсу: <https://refactoring.guru/uk/design-patterns/observer>
16. Autocomplete [Електронний ресурс] – Режим доступу до ресурсу: <https://jqueryui.com/autocomplete/>
17. PHP [Електронний ресурс] – Режим доступу до ресурсу: <http://programming.in.ua/web-design/allphp/30-about-php.html>
18. MySQL [Електронний ресурс] – Режим доступу до ресурсу: <https://brander.ua/technologies/mysql>

ДОДАТКИ

					ДУ «Житомирська політехніка».23.121.23.000 – КР	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		61

Специфікація прецедентів

К1. Перегляд послуг

K1	Клієнт	Перегляд товарів	Клієнт має можливість переглянути усі наявні товари.
----	--------	------------------	--

Основна діюча особа: Клієнт.

Інші учасники прецеденту: відсутні;

Зв'язки з іншими варіантами використання: відсутні;

Короткий опис:

Даний варіант дозволяє клієнту на сторінці програми переглянути на усі товари у вигляді детального списку із наявним прайс листом.

К2. Створення запиту на послугу

K2	Клієнт	Створення замовлення	Клієнт має можливість замовити товар.
----	--------	----------------------	---------------------------------------

Основна діюча особа: Клієнт.

Інші учасники прецеденту: відсутні;

Зв'язки з іншими варіантами використання: відсутні;

Короткий опис:

Даний варіант використання дозволяє клієнту створити замовлення на сайті.

А1. Перегляд надаваних послуг

A1	Адміністратор	Перегляд товарів	Адміністратор має можливість переглянути усі наявні товари.
----	---------------	------------------	---

Основна діюча особа: Адміністратор.

Інші учасники прецеденту: відсутні;

Зв'язки з іншими варіантами використання: відсутні;

Короткий опис:

Даний варіант дозволяє адміністратору переглянути усі наявні поточні товари.

А2. Оформлення та дії над товарами

А2	Адміністратор	Створення та дії над товарами.	Адміністратор має можливість створювати, редагувати та видаляти.
----	---------------	--------------------------------	--

Основна діюча особа: Адміністратор.

Інші учасники прецеденту: відсутні;

Зв'язки з іншими варіантами використання: відсутні;

Короткий опис:

Даний варіант дозволяє адміністратору створювати, редагувати та видаляти товари.

А3. Перегляд бази клієнтів

А3	Адміністратор	Перегляд бази клієнтів	Адміністратор має можливість переглядати бази клієнтів
----	---------------	------------------------	--

Основна діюча особа: Адміністратор.

Інші учасники прецеденту: відсутні;

Зв'язки з іншими варіантами використання: відсутні;

Короткий опис:

Даний варіант використання дозволяє адміністратору переглядати базу даних клієнтів.

А4. Ведення обліку клієнтів

А4	Адміністратор	Введення обліку клієнтів	Адміністратор має можливість ввести облік клієнтів
----	---------------	--------------------------	--

Основна діюча особа: Адміністратор.

Інші учасники прецеденту: відсутні

					ДУ «Житомирська політехніка».23.121.23.000 – КР	Арк.
						63
Змн.	Арк.	№ докум.	Підпис	Дата		

Зв'язки з іншими варіантами використання: відсутні.

Короткий опис:

Даний варіант використання дозволяє адміністратору ввести облік клієнтів, редагувати чи видаляти дані про клієнтів, працювати з базою клієнтів.

А5. Перегляд звітів та аналітики

А5	Адміністратор	Перегляд звітів та аналітики	Адміністратор має змогу переглядати звіти та аналітику інформаційної системи.
----	---------------	------------------------------	---

Основна діюча особа: Адміністратор.

Інші учасники прецеденту: відсутні;

Зв'язки з іншими варіантами використання: відсутні;

Короткий опис:

Даний варіант використання дозволяє адміністратору переглядати звіти та аналітику по інформаційній системі.

Лістинг коду

Admin/ProductController

```

<?php

namespace App\Http\Controllers\Admin;

use App\Category;
use App\Http\Controllers\Controller;
use App\Product;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\File;

class ProductController extends Controller
{
    public function index()
    {
        $product = Product::all();
        return view('admin.product.index', compact('product'));
    }

    public function add()
    {
        $category = Category::all();
        return view('admin.product.add', compact('category'));
    }

    public function create(Request $request)
    {
        $product = new Product();
        if ($request->hasFile('image')) {
            $file = $request->file('image');
            $ext = $file->getClientOriginalExtension();
            $filename = time() . '.' . $ext;
            $file->move('assets/products', $filename);
            $product->image = $filename;
        }
        $product->name = $request->input('name');
        $product->price = $request->input('price');
        $product->description = $request->input('description');
        $product->category_id = $request->input('category_id');
    }
}

```

					ДУ «Житомирська політехніка».23.121.23.000 – КР	Арк.
						65
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        $product->status = $request->input('status') == TRUE ? '0' : '1';
        $product->save();
        return redirect('admin/products');
    }

    public function edit($id)
    {
        $product = Product::find($id);
        $category = Category::all();
        return view('admin.product.edit', compact('product', 'category'));
    }

    public function update(Request $request, $id)
    {
        $product = Product::find($id);
        if ($request->hasFile('image')) {
            $path = 'assets/products/' . $product->image;
            if (File::exists($path)) {
                File::delete($path);
            }
            $file = $request->file('image');
            $ext = $file->getClientOriginalExtension();
            $filename = time() . '.' . $ext;
            $file->move('assets/products', $filename);
            $product->image = $filename;
        }
        $product->name = $request->input('name');
        $product->price = $request->input('price');
        $product->description = $request->input('description');
        $product->category_id = $request->input('category_id');
        $product->status = $request->input('status') == TRUE ? '0' : '1';
        $product->update();
        return redirect('admin/products');
    }

    public function delete($id)
    {
        $product = Product::find($id);
        if ($product->image) {
            $path = 'assets/products/' . $product->image;
            if (File::exists($path)) {

```

```

        File::delete($path);
    }
}
$product->delete();
return redirect('admin/products');
}
}

```

Admin/CategoryController

<?php

```

namespace App\Http\Controllers\Admin;

use App\Category;
use App\Http\Controllers\Controller;
use Illuminate\Http\Request;

class CategoryController extends Controller
{
    public function index()
    {
        $category = Category::all();
        return view('admin.category.index', compact('category'));
    }

    public function add()
    {
        return view('admin.category.add');
    }

    public function create(Request $request)
    {
        $category = new Category();
        $category->name = $request->input('name');
        $category->save();
        return redirect('admin/categories');
    }

    public function edit($id)
    {
        $category = Category::find($id);
    }
}

```

| | | | | | | |
|------|------|----------|--------|------|---|------|
| | | | | | ДУ «Житомирська політехніка».23.121.23.000 – КР | Арк. |
| | | | | | | 67 |
| Змн. | Арк. | № докум. | Підпис | Дата | | |

```

        return view('admin.category.edit', compact('category'));
    }

    public function update(Request $request, $id)
    {
        $category = Category::find($id);
        $category->name = $request->input('name');
        $category->update();
        return redirect('admin/categories');
    }

    public function delete($id)
    {
        $category = Category::find($id);
        $category->delete();
        return redirect('admin/categories');
    }
}

```

FrontendController

```

<?php

namespace App\Http\Controllers;

use App\Category;
use App\Product;
use App\Http\Controllers\Controller;
use Illuminate\Http\Request;

class FrontendController extends Controller
{
    public function index()
    {
        $category = Category::all();
        $product = Product::take(4)->get();
        return view('frontend.index', compact('category', 'product'));
    }

    public function viewcategory($id)
    {
        if (Category::where('id', $id)->exists()) {

```

| | | | | | | |
|------|------|----------|--------|------|---|------|
| | | | | | ДУ «Житомирська політехніка».23.121.23.000 – КР | Арк. |
| | | | | | | 68 |
| Змн. | Арк. | № докум. | Підпис | Дата | | |

```

        $category = Category::find($id);
        $product = Product::where('category_id', $category->id)->get();
        return view('frontend.product.index', compact('product', 'category'));
    } else {
        return redirect('/');
    }
}

public function view()
{
    $product = Product::all();
    return view('frontend.product.index', compact('product'));
}

public function viewproduct($category_id, $prod_id)
{
    if (Category::where('id', $category_id)->exists()) {
        if (Product::where('id', $prod_id)->exists()) {
            $product = Product::find($prod_id);
            return view('frontend.product.view', compact('product'));
        } else {
            return redirect('/');
        }
    } else {
        return redirect('/');
    }
}

public function about()
{
    return view('frontend.about');
}

public function productList()
{
    $product = Product::select("name")->get();
    $data = [];

    foreach ($product as $el)
    {
        $data[] = $el["name"];
    }
}

```

```

    }
    return $data;
}

public function productSearch(Request $request)
{
    $search_prod = $request->product_name;

    if ($search_prod != "")
    {
        $product = Product::where("name", "LIKE", "%$search_prod%")->first();
        if ($product)
        {
            return redirect('view_category/'.$product->category_id.'/'.$product->id);
        }
        else {
            return redirect()->back();
        }
    }
    else {
        return redirect()->back();
    }
}
}

```

CartController

```

<?php

namespace App\Http\Controllers;

use App\Cart;
use App\Product;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Auth;

class CartController extends Controller
{
    public function viewcart()
    {
        $cartitems = Cart::where('user_id', Auth::id())->get();
    }
}

```

| | | | | | | |
|------|------|----------|--------|------|---|------|
| | | | | | ДУ «Житомирська політехніка».23.121.23.000 – КР | Арк. |
| | | | | | | 70 |
| Змн. | Арк. | № докум. | Підпис | Дата | | |

```

        return view('frontend.cart', compact('cartitems'));
    }

    public function addProduct(Request $request)
    {
        $product_id = $request->input('product_id');
        $product_amount = $request->input('product_amount');
        if (Auth::check()) {
            $prod_check = Product::where('id', $product_id)->first();
            if ($prod_check) {
                if (Cart::where('product_id', $product_id)->where('user_id',
Auth::id())->exists()) {
                    return response()->json(['status' => $prod_check->name . " вже
додано в кошик"]);
                } else {
                    $cartItem = new Cart();
                    $cartItem->product_id = $product_id;
                    $cartItem->user_id = Auth::id();
                    $cartItem->product_amount = $product_amount;
                    $cartItem->save();
                    return response()->json(['status' => $prod_check->name . " додано
в кошик"]);
                }
            } else {
                return response()->json(['status' => "Продукту не існує"]);
            }
        } else {
            return response()->json(['status' => "Увійдіть в свій аккаунт"]);
        }
    }

    public function deleteProduct(Request $request)
    {
        $product_amount = $request->input('product_amount');
        if (Auth::check()) {
            $product_id = $request->input('product_id');
            if (Cart::where('product_id', $product_id)->where('user_id', Auth::id())-
>exists()) {
                $cartItem = Cart::where('product_id', $product_id)->where('user_id',
Auth::id())->first();
                $cartItem->delete();
            }
        }
    }

```

| | | | | | | |
|------|------|----------|--------|------|---|------|
| | | | | | ДУ «Житомирська політехніка».23.121.23.000 – КР | Арк. |
| | | | | | | 71 |
| Змн. | Арк. | № докум. | Підпис | Дата | | |

```

        return response()->json(['status' => "Видалено"]);
    } else {
        return response()->json(['status' => "Товару нема"]);
    }
} else {
    return response()->json(['status' => "Увійдіть в свій аккаунт"]);
}
}

public function editProduct(Request $request)
{
    $product_id = $request->input('product_id');
    $product_amount = $request->input('product_amount');

    if (Auth::check()) {
        if (Cart::where('product_id', $product_id)->where('user_id', Auth::id())->exists()) {
            $cart = Cart::where('product_id', $product_id)->where('user_id', Auth::id())->first();
            $cart->product_amount = $product_amount;
            $cart->update();
            return response()->json(['status' => "оновлено"]);
        }
    }
}
}

```

OrderController

```
<?php
```

```
namespace App\Http\Controllers;
```

```
use App\Cart;
```

```
use App\Order;
```

```
use App\OrderItem;
```

```
use App\Product;
```

```
use Illuminate\Http\Request;
```

```
use Illuminate\Support\Facades\Auth;
```

```
class OrderController extends Controller
```

```
{
```

| | | | | | | |
|------|------|----------|--------|------|---|------|
| | | | | | ДУ «Житомирська політехніка».23.121.23.000 – КР | Арк. |
| | | | | | | 72 |
| Змн. | Арк. | № докум. | Підпис | Дата | | |

```

public function index()
{
    $oldcartitems = Cart::where('user_id', Auth::id())->get();
    foreach ($oldcartitems as $item) {
        if (!Product::where('id', $item->product_id)->where('status', '0')->exists()) {
            $removeItem = Cart::where('user_id', Auth::id())->where('product_id', $item->product_id);
            $removeItem->delete();
        }
    }
    $cartitems = Cart::where('user_id', Auth::id())->get();
    return view('frontend.order', compact('cartitems'));
}

public function createorder(Request $request)
{
    $order = new Order();
    $order->user_id = Auth::id();
    $order->firstname = $request->input('firstname');
    $order->lastname = $request->input('lastname');
    $order->email = $request->input('email');
    $order->phone = $request->input('phone');
    $order->city = $request->input('city');
    $order->address = $request->input('address');
    $order->comment = $request->input('comment');
    $order->amount = $request->input('amount');
    $order->save();

    $cartitems = Cart::where('user_id', Auth::id())->get();
    foreach ($cartitems as $item) {
        OrderItem::create([
            'order_id' => $order->id,
            'product_id' => $item->product_id,
            'quantity' => $item->product_amount,
            'cost' => $item->products->price,
        ]);
    }
    $cartitems = Cart::where('user_id', Auth::id())->get();
    $cartitems->each->delete();
}

```

| | | | | | | |
|------|------|----------|--------|------|---|------|
| | | | | | ДУ «Житомирська політехніка».23.121.23.000 – КР | Арк. |
| | | | | | | 73 |
| Змн. | Арк. | № докум. | Підпис | Дата | | |

```

        return redirect('/');
    }

    public function vieworder()
    {
        $orders = Order::where('user_id', Auth::id())->get();
        return view('frontend.orders.view', compact('orders'));
    }

    public function fullorder($id)
    {
        $order = Order::where('id', $id)->where('user_id', Auth::id())->first();
        return view('frontend.orders.orderitems', compact('order'));
    }
}

```

| | | | | | | |
|------|------|----------|--------|------|---|------|
| | | | | | ДУ «Житомирська політехніка».23.121.23.000 – КР | Арк. |
| | | | | | | 74 |
| Змн. | Арк. | № докум. | Підпис | Дата | | |