ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
МІСЬКОГО ГОСПОДАРСТВА ІМЕНІ О. М. БЕКЕТОВА

Пояснювальна записка
до кваліфікаційної роботи бакалавра

на тему:

ПРОЄКТУВАННЯ ВЕБОРІЄНТОВАНОЇ СИСТЕМИ ЕЛЕКТРОННОЇ КОМЕРЦІЇ “ANIMESHKA” ІЗ ЗАСОБАМИ АДАПТИВНОГО ІНТЕРФЕЙСУ ТА АДМІНІСТРУВАННЯ

Виконав: здобувач вищої освіти
групи КН 2022-1
спеціальності
122 «Комп’ютерні науки»

Ілля ДМИТРЕНКО

Керівник:
к.т.н., доц. Марина БУЛАЄНКО

Рецензент:
к.т.н., доц. Микола КАРПЕНКО

м. Харків – 2026 рік

Харківський національний університет міського господарства імені О. М. Бекетова
(повне найменування закладу вищої освіти)
Навчально-науковий Інститут енергетичної, інформаційної та транспортної інфраструктури
Кафедра комп'ютерних наук та інформаційних технологій
Рівень вищої освіти перший (бакалаврський)
Спеціальність 122 «Комп'ютерні науки»

ЗАТВЕРДЖУЮ
Завідувач кафедри КНтаІТ

Марина НОВОЖИЛОВА
«_23_» _червня__2026 року

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Дмитренку Іллі Вадимовичу
(прізвище, ім'я, по батькові)

1. Тема роботи «Проєктування веборієнтованої системи електронної комерції "Animeshka" із засобами адаптивного інтерфейсу та адміністрування»

керівник роботи к.т.н., доц. Булаєнко М.В.
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом закладу вищої освіти від «22» травня 2026 р. № 440-03

2. Термін подання студентом роботи 16.06.2026р.

3. Вихідні дані до роботи: Рекомендації для проєктування веборієнтованої системи електронної комерції "Animeshka" із засобами адаптивного інтерфейсу та адміністрування

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
Дослідження ринку та аналогів сервісів для веб-платформи електронної комерції; створення технічного завдання; вибір інструментів розробки і тестування застосунку.
5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)
Презентація – 19 слайдів

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Розділ I	Марина БУЛАЄНКО	15.05.2026	20.05.2026
Розділ II	Марина БУЛАЄНКО	21.05.2026	25.05.2026
Розділ III	Марина БУЛАЄНКО	26.05.2026	06.06.2026
Розділ IV	Вікторія МАЛИШЕВА	07.06.2026	15.06.2026

7. Дата видачі завдання 15.05.2026 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Вибір теми кваліфікаційної роботи	15.05.2026	Виконано
2	Затвердження тем, наукових керівників, завдань та календарного плану підготовки кваліфікаційної роботи	16.05.2026	Виконано
3	Написання I розділу	20.05.2026	Виконано
4	Написання II розділу	25.05.2026	Виконано
5	Написання III розділу	06.06.2026	Виконано
6	Написання IV розділу	11.06.2026	Виконано
7	Подання кваліфікаційної роботи керівнику	13.06.2026	Виконано
8	Робота по усуненню зауважень керівника, уточнення і доповнення практичного матеріалу, оформлення додатків до кваліфікаційної роботи	16.06.2026	Виконано
9	Подання доопрацьованого варіанту кваліфікаційної роботи керівнику	18.06.2026	Виконано
10	Захист матеріалів кваліфікаційної роботи на засіданні кафедри	21.06.2026	Виконано
11	Офіційний захист матеріалів кваліфікаційної роботи на засіданні екзаменаційної комісії	24.06.2026	Виконано

Студент Ілля ДМИТРЕНКО

Керівник роботи Марина БУЛАЄНКО

АНОТАЦІЯ

Пояснювальна записка до кваліфікаційної роботи бакалавра здобувача вищої освіти групи КН 2022-1 Дмитренка Іллі Вадимовича на тему «Проєктування веборієнтованої системи електронної комерції “Animeshka” із засобами адаптивного інтерфейсу та адміністрування» складається з 4 розділів. Робота містить 65 сторінок тексту, 18 рисунків, 4 таблиці та 25 джерел.

Кваліфікаційна робота бакалавра зосереджена на розробці інтернет-магазину з продажу аніме-фігурок з системою фільтрації та інтуїтивно зрозумілим інтерфейсом. Особливу увагу приділено проєктуванню системи, яка б задовольняла потреби цільової аудиторії, з реалізацією адаптивного дизайну та зручної панелі адміністрування.

У першому розділі роботи проаналізовано предметну область і особливості продажу товарів; проведено огляд наявних аналогів та виявлено їхні недоліки щодо масштабування зображень і фільтрації. На основі аналізу сформовано технічне завдання на розробку вебзастосунку.

У другому розділі спроєктовано інформаційне та математичне забезпечення майбутньої платформи. Побудовано реляційну базу даних та об'єктно-орієнтовану модель; розроблено алгоритми для розрахунку кінцевої вартості товарів з урахуванням знижок, створено механізми управління сесійним кошиком, фільтрації каталогу та безпечного оформлення замовлення з використанням атомарних транзакцій.

У третьому розділі здійснено програмну реалізацію системи мовою Python на базі вебфреймворку Django. Створено клієнтський інтерфейс з підключенням модулів реєстрації, кошика та списку бажань; налаштовано закриту адміністративну панель для менеджерів магазину; розроблено керівництво користувача.

У четвертому розділі розглянуто питання охорони праці під час розробки програмного забезпечення.

Ключові слова: ІНТЕРНЕТ-МАГАЗИН, ЕЛЕКТРОННА КОМЕРЦІЯ, АНІМЕ-ФІГУРКИ, ВЕБФРЕЙМВОРК DJANGO, БАЗА ДАНИХ, АДАПТИВНИЙ ІНТЕРФЕЙС, АДМІНІСТРУВАННЯ.

ABSTRACT

The explanatory note to the bachelor's thesis of the student of group KN 2022-1, Illia Dmytrenko, on the topic "Designing the web-oriented e-commerce system 'Animeshka' with tools for adaptive interface and administration" consists of 4 chapters. The work contains 65 pages of text, 18 figures, 4 tables, and 25 references.

The bachelor's thesis focuses on the development of an online store for selling anime figures with a filtering system and an intuitive interface. Special attention is paid to designing a system that satisfies the needs of the target audience, with the implementation of an adaptive design and a convenient administration panel.

In the first chapter of the work, the subject area and the features of selling goods are analyzed; a review of existing analogues is conducted, and their shortcomings regarding image scaling and filtering are identified. Based on the analysis, a technical specification for the development of the web application is formed.

In the second chapter, the information and mathematical support of the future platform is designed. A relational database and an object-oriented model are built; algorithms for calculating the final cost of goods taking discounts into account are developed; and mechanisms for managing the session cart, catalog filtering, and secure order processing using atomic transactions are created.

In the third chapter, the software implementation of the system is carried out in Python using the Django web framework. A client interface is created with the integration of registration, shopping cart, and wishlist modules; a closed administration panel for store managers is configured; and a user manual is developed.

In the fourth chapter, the issues of occupational health and safety during software development are considered.

Keywords: ONLINE STORE, E-COMMERCE, ANIME FIGURES, DJANGO WEB FRAMEWORK, DATABASE, ADAPTIVE INTERFACE, ADMINISTRATION.

ЗМІСТ

ВСТУП

У сучасних умовах електронна комерція стала одним із найпопулярніших каналів продажу товарів [1]. Особливо це стосується нішевих ринків, зокрема сфери продажу колекційних товарів та аніме-фігурок.

Існуючі рішення, конструктори сайтів, мають обмеження щодо роботи функціоналу, фільтрації, масштабування та дизайну. Тому проєктування власної системи із застосуванням засобів адаптивного інтерфейсу з панелю адміністрування є досить актуальним завданням.

Мета дослідження: спроєктувати систему електронної комерції «Animeshka», яка забезпечить зручний процес купівлі товарів для покупців.

Для досягнення мети було сформульовано такі завдання:

1. Провести аналіз предметної області та визначити функціональні вимоги до розроблюваної системи.
2. Спроєктувати архітектуру бази даних для збереження інформації про товари, користувачів та транзакції замовлень.
3. Розробити математичне та алгоритмічне забезпечення для реалізації пошуку, системи фільтрації та кошика покупок.
4. Створити адаптивний інтерфейс та налаштувати панель адміністрування для управління каталогом товарів та замовлень.

Об’єктом дослідження є процес розробки інтернет магазину для продажу товарів.

Предметом дослідження є методи, алгоритми та засоби побудови інтернет-магазину.

Для досягнення поставленої мети необхідно виконати завдання:

- провести аналіз структури інтернет магазинів;
- розглянути існуючі рішення та їх особливості;
- написати вимоги до розробки;
- реалізувати рішення.

РОЗДІЛ 1 ЗАГАЛЬНІ ПОЛОЖЕННЯ

1.1. Опис предметного середовищ

Інтернет-магазин – це платформа, на якій користувачі можуть переглянути каталог товарів, оформити замовлення, здійснити оплату та замовити доставку. Сучасний інтернет-магазин є веб-застосунком, призначеним для автоматизації взаємодії між продавцем і покупцем.

Інтернет фактично стер географічні кордони для продажів [2], а його доступність та доступність мобільних пристроїв, зумовили щорічне збільшення частки онлайн-покупців у світі.

Загалом, можна зазначити такі переваги інтернет-магазину [3]:

1. Можливість оформлення замовлення в будь-який час, в будь-якому місці. Вам не потрібно витрачати час на пошук спеціалізованого магазину, товар буде доставлено вам додому або у поштомат.
2. Онлайн-шопінг дає змогу порівняти вартість товарів на різних платформах, та заощадити кошти, подивитись детальні характеристики та перед здійсненням замовлення перевірити відгуки щодо товару
3. Інтерактивні інструменти, а саме кошик та список бажань, дозволяють користувачам зберігати обрані позиції та відкладати покупку на певний час.

До основних мінусів можна віднести:

1. Відсутність можливості фізичного контакту з товаром до моменту його отримання.
2. Доставка товару може бути затримана продавцем, компанією доставки або з інших причин.

Особисто на мій погляд, недоліки можна вважати мінімальними в порівнянні з перевагами. Таким чином, розробка власного інтернет-магазину є раціональним рішенням, для ведення бізнесу.

Функціонування інтернет-магазину з продажу аніме-фігурок, має деякі особливості на відміну від класичного інтернет-магазину, що зумовлено властивостями самого товару. Аніме-фігурки різняться за технічними та комерційними характеристиками, які мають бути відображені у базі даних інтернет-магазину.

Основними критеріями можна вважати:

1. Категорії, які визначають тип товарів у каталозі (наприклад, статичні фігурки, рухомі екшн-фігурки, чибі, тощо).
2. Виробник або бренд.
3. Країна походження.
4. Медіафраншиза або Всесвіт, яка визначає першоджерело (аніме, манґа, світ відеоігор), за мотивами якого створено товар.
5. Персонаж.
6. Фізичні та матеріальні характеристики

1.2. Функціональна модель

Функціональна модель інтернет-магазину (рис 1.1) описує, як саме працює система, з яких частин вона складається та як ці частини взаємодіють між собою, яким чином основні процеси магазину, перегляд товарів, оформлення замовлень, керування каталогом та обробка покупок реалізуються у вигляді окремих модулів веб-застосунку [4].

Для зручності роботи інтернет-магазин поділяється на дві основні частини: користувацьку та адміністративну. Користувацька частина призначена для покупців. Адміністративна частина використовується адміністратором магазину.

Рисунок 1.1 – Функціональна модель інтернет магазину

Покупець заходить на сайт і потрапляє на головну сторінку. Він може гортати товари, шукати фігурки через пошуковий рядок, користуватися фільтрами. Натискаючи на карточку з товаром, покупець потрапляє на сторінку товару, де розглядає фото з різних ракурсів, читає опис і перевіряє, чи є вона в наявності.

Покупець може додати товар до кошика й оформити замовлення. Тут він залишає свою контактну адресу, обирає спосіб доставки та оплати. Як тільки замовлення підтверджено, сайт надсилає ці дані в базу даних з замовленнями, а клієнт у своєму кабінеті може бачити історію покупок.

Адміністратор керує всім магазином через адмін-панель. Його робота полягає у спостереженні за актуальністю каталогу, додаванні нових товарів, завантаженні фото, створенні нових категорій та брендів, моніторингу ціни, та знижок. Окремим процесом діяльності адміністратора є обробка замовлень. Отримуючи сповіщення про нову заявку, менеджер перевіряє коректність введених даних покупця, підтверджує факт оплати, організовує фізичну збірку посилки на складі та змінює логістичні статуси в системі, контролює залишки товарів на складі, щоб клієнти не могли випадково замовити те, чого вже немає.

Наш веб-застосунок взаємодіє з базою даних, у якій зберігається інформація про користувачів, товари, категорії, замовлення та платежі. Для

реалізації серверної логіки використовується веб-фреймворк Django [5]. Він забезпечує швидку маршрутизацію запитів, обробку даних та взаємодію з базою даних через вбудовану технологію ORM, яка автоматично трансформує класи моделей Python у таблиці реляційної СУБД.

Користувач взаємодіє із серверною частиною через HTTP-запити [6], а сервер обробляє їх та повертає результати у вигляді динамічних веб-сторінок або даних. На ринку присутня велика кількість готових CMS та конструкторів сайтів, але більшість із них мають обмеження при створенні таких вузькоспеціалізованих інтернет-магазинів. Як було зазначено раніше, специфіка обліку аніме-фігурок вимагає збереження великої кількості унікальних атрибутів, що ускладнює використання стандартних шаблонів. Крім того, такі платформи зазвичай вимагають фінансових витрат на підписку, в поріняні, хостинг власного сайту, коштує набагато дешевше. Створення власного кастомного інтернет-магазину на базі Django хоч і потребує певних затрат, але значно менших, а також забезпечує гнучкість, масштабованість, підтримку та безпечність роботи веб-застосунку.

1.3. Огляд наявних аналогів

Для початку переглянемо інтернет-магазини за схожою тематикою, з українського ринку. Це дозволить виявити переваги та недоліки існуючих магазинів та визначити основні вимоги.

Перший інтернет-магазин – «Pulsar» (рис 1.2).

Верхня частина сайту містить елементи автентифікації користувачів, присутня можливість зміни мовної локалізації, контактні номери телефонів, посилання на інформаційні блоки «Блог», «Гарантії», «Про магазин», а також інтерактивні модулі пошуку, кошика та списку «Бажання».

Головна сторінка має чітку структуру навігації. Ліва частина інтерфейсу відведена під багаторівневий блок фільтрації.

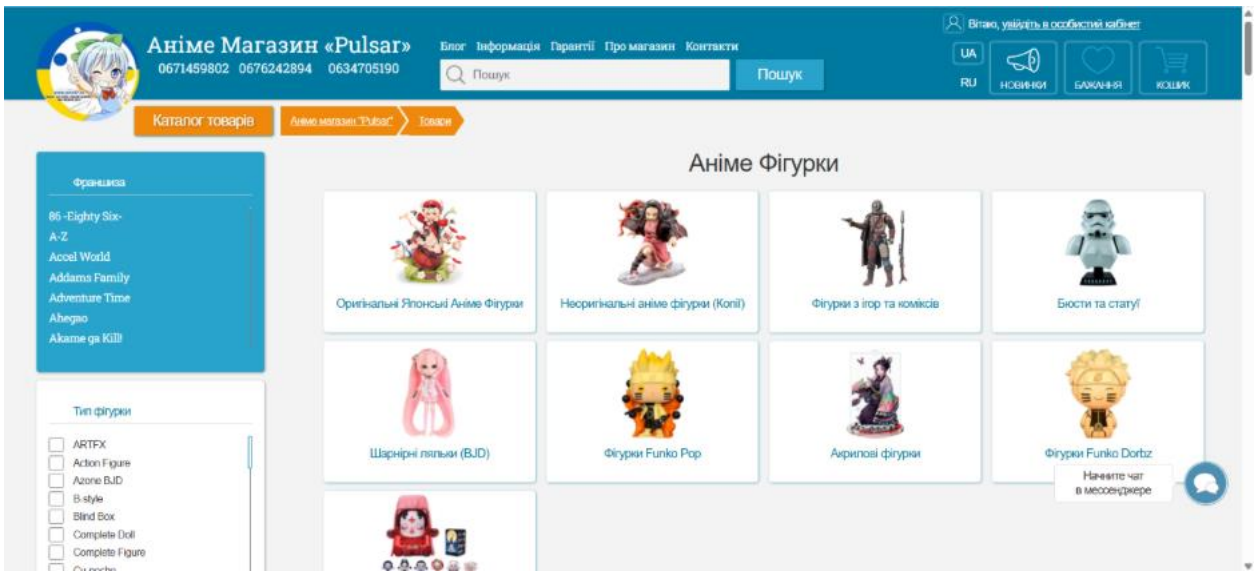

Рисунок 1.2 – Сторінка з аніме фігурками сайту «Pulsar.ua»

Центральна частина сторінки містить рекомендовані категорії фігурок, а нижче карткову структуру категорій каталогу.

Картка товару забезпечує виведення зображень фігурки, характеристик, вартості, статусів наявності та рейтингу.

Нижня частина сайту включає дублюючі посилання на каталоги, правову інформацію, адреси фізичних представництв та віджети соціальних мереж.

Функціональними особливостями можна вважати наявність модулю порівняння характеристик товарів.

Зовнішній вигляд інтерфейсу є застарілим і перевантаженим графічними елементами. Відсутність сучасного мінімалістичного дизайну знижує рівень довіри до магазину.

Значна частина карток товарів містить лише одне зображення фігурки. Для колекційного сегмента, де детальність скульптури є критичною ознакою, цього недостатньо.

Другий інтернет-магазин для аналізу – «Pecorineshop» (рис 1.3)

На рисунку зображено головну сторінку сайту.

Рисунок 1.3 – Головна сторінка сайту – «Pecorineshop.com»

Верхня панель керування спрощена та розвантажена від надмірної текстової інформації, в порівнянні з першим сайтом. Вона містить бургер-меню, рядок інтерактивного пошуку, а також мовний перемикач, що за замовчуванням встановлює українську локалізацію. Права частина навігаційної панелі містить виключно графічні інтуїтивно зрозумілі іконки: порівняння, список бажань, особистий кабінет користувача та кошик.

Центральне місце на головній сторінці займає графічний банер із маркетинговим позиціонуванням «Оригінальні фігурки з Японії». Візуальне оформлення виконано в сучасній колірній гамі з використанням теплих тонів. Нижче головного банера реалізовано плитковий блок динамічного виведення контенту «Останні оновлення», який автоматично підвантажує картки нових надходжень товарів із бази даних.

Як чітко видно на прикладі сторінки картки товару (рис. 1.4), магазин має критичні проблеми з адаптацією.

Головне зображення продукту не масштабується належним чином, через що воно обрізається та займає занадто велику площу екрана, заважаючи сприйняттю інформації.

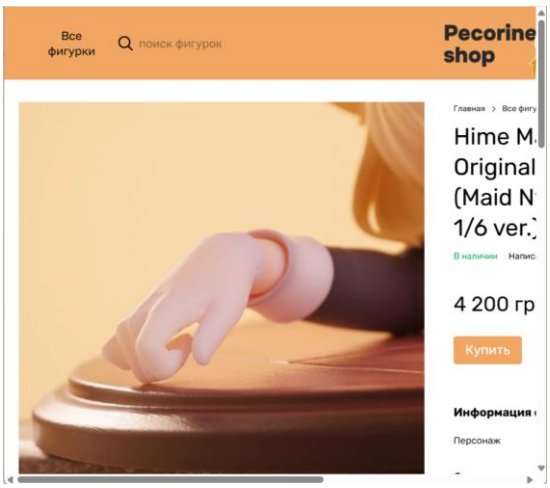

Рисунок 1.4 – Проблеми з масштабуванннавям

Останнім магазином для аналізу візьмемо універсальний маркетплейс, «Rozetka» (рис 1.5), який на відміну від нішевих магазинів, оперує мільйонами товарів у різних категоріях.

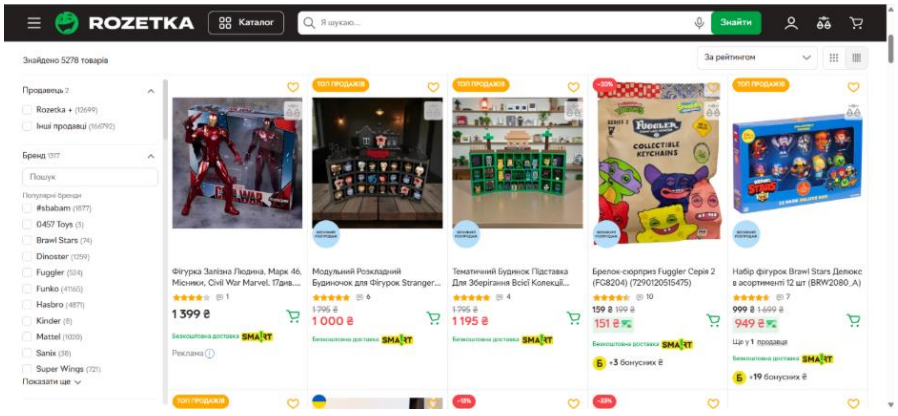

Рисунок 1.5 – Сторінка з категорією фігурок на сайті «Rozetka.ua»

Дизайн професійно спроєктований в сучасній мінімалістичній стилістиці, картки продуктів ідеально структуровані

На жаль система фільтрації маркетплейсу адаптована під загальні категорії мас-маркету і повністю не призначена для колекційних аніме-фігурок. Користувач позбавлений можливості відфільтрувати товари за конкретною медіафраншизою, ім'ям персонажа, масштабом.

Оскільки категорія є загальною, у результаті із класичними фігурками потрапляють сторонні товари, брелки-сюрпризи чи дитячі ігрові набори, що розмиває цільовий асортимент магазину.

1.4. Постановка задачі

Метою роботи є розробка інтернет магазину з продажу аніме-фігурок з гнучкою системою фільтрації та інтуїтивно зрозумілим інтерфейсом.

Сайт повинен мати:

1. Систему пошуку товару
2. Можливість додавання товару у кошик та списку бажань
3. Модуль фільтрації товарів
4. Сучасний мінімалістичний та адаптивний дизайн

Об’єктом дослідження є процес проєктування та програмна реалізація системи електронної комерції.

Предметом дослідження вважаються методи та інструменти розробки веб-застосунку на базі фреймворку Django.

Цілями проєкту можна вважати:

1. Розробку реляційної бази даних
2. Написання серверної частини
3. Верстка клієнтської частини сайту
4. Створення адаптивного дизайну інтерфейсу магазину

Сайт має захищати персональні дані клієнтів, відповідати мінімальним вимогам швидкості завантаження, а також, бути логічно зв’язаним, для інтуїтивної взаємодії між юзером та інтерфейсом.

Для виконання роботи потрібно:

1. Мати навички проведення аналізу предметної області та дослідження аналогів;
2. Знання принципів UI/UX дизайну для створення сучасного та адаптивного інтерфейсу [7];

3. Володіти технологіями фронтенд-розробки: HTML5, CSS3 [8];
4. Знати інструменти бекенд-розробки та баз даних: мову Python, вебфреймворк Django та реляційні СУБД (SQL/Django ORM).

Висновки до розділу

З стрімким зростанням популярності аніме та збільшенням кількості користувачів стає зрозумілим, що існуючі інтернет магазини не відповідають сучасним реаліям. Аналіз аналогів показав, що людям потрібна новий, інтернет-магазин, з сучасним мінімалістичним дизайном, адаптивним інтерфейсом, орієнтованим, як на користувачів ПК так і мобільних пристроїв, інтуїтивним інтерфейсом та фільтрами, орієнтованими на цю нішу товару.

РОЗДІЛ 2 ІНФОРМАЦІЙНЕ ТА МАТЕМАТИЧНЕ ЗАБЕЗПЕЧЕННЯ

2.1. Аналіз предметної області

Наш проєкт системи електронної комерції “Animeshka” повинен мати певну структуру інформації між клієнтською та серверною частинами, для коректної роботи фільтрації, керування кошиком, списком бажань та модулем адміністрування [9].

2.1.1 Вхідні данні

Вхідними даними можна вважати сукупність об’єктів, параметрів, та транзакційні запити які надходять для подальшої обробки сервером та збереження в СУБД.

Класифікуємо інформацію за її джерелами:

1. Дані адміністративного контенту, котрі вводяться адміністратором через адміністративну панель:
    - Параметри товару, назва аніме-фігурки, текстовий опис, ціна, відсоток знижки, актуальні залишки на складі.
    - Нішові класифікаційні ознаки, категорія фігурки, бренд або компанія-виробник, країна походження.
    - Контекстні атрибути, всесвіт, ім'я конкретного персонажа, а також характеристики й матеріал виготовлення.
    - Медіа-контент, набір растрових зображень продукту.
2. Ідентифікаційні та профільні дані, які вводяться користувачем:
    - Дані автентифікації, адреса електронної пошти та пароль, який передається у зашифрованому вигляді.
    - Персональні дані покупця, прізвище, ім'я, контактний номер телефону та адреса доставки.
    - Операційні дані та запити взаємодії.

- Параметри пошуку та фільтрації, текстові рядки пошукових запитів, checkbox для вибору фільтрів.
- Запити керування станом сесії, ідентифікатори товарів при додаванні чи видаленні позицій з кошика або списку бажань.
- Дані оформлення транзакції, обраний клієнтом спосіб доставки, спосіб оплати та підтвердження замовлення.

2.1.2 Вихідні данні

Вихідними даними можна вважати відповіді сервера, у результаті виконання коду

Вихідні дані розподіляються на дві цільові групи:

1. Дані динамічного інтерфейсу клієнта: візуалізований каталог та картки товарів, адаптовані під пристрій користувача; результати вибірки, відфільтровані списки аніме-фігурок, за введеними критеріями користувача; інформація про транзакції, унікальний номер сформованого замовлення, згенерований чек із фінальною вартістю покупки та статус посилки;
2. Керуюча інформація: реєстр замовлень для обробки; структуровані дані про нові заявки покупців, що відображаються в адмін-панелі для перевірки оплати; комплектації посилки на складі та подальшої зміни статусів замовлень;

2.2. Проєктування системи

2.2.1. Проєктування бази даних

Для програмної реалізації системи електронної комерції “Animeshka” обрано реляційну модель бази даних. Взаємодія із СУБД реалізується за допомогою вбудованої технології Django ORM [10], яка автоматично трансформує класи моделей Python у реляційні таблиці.

Проєктування бази даних виконано з урахуванням усіх специфік предметної області.

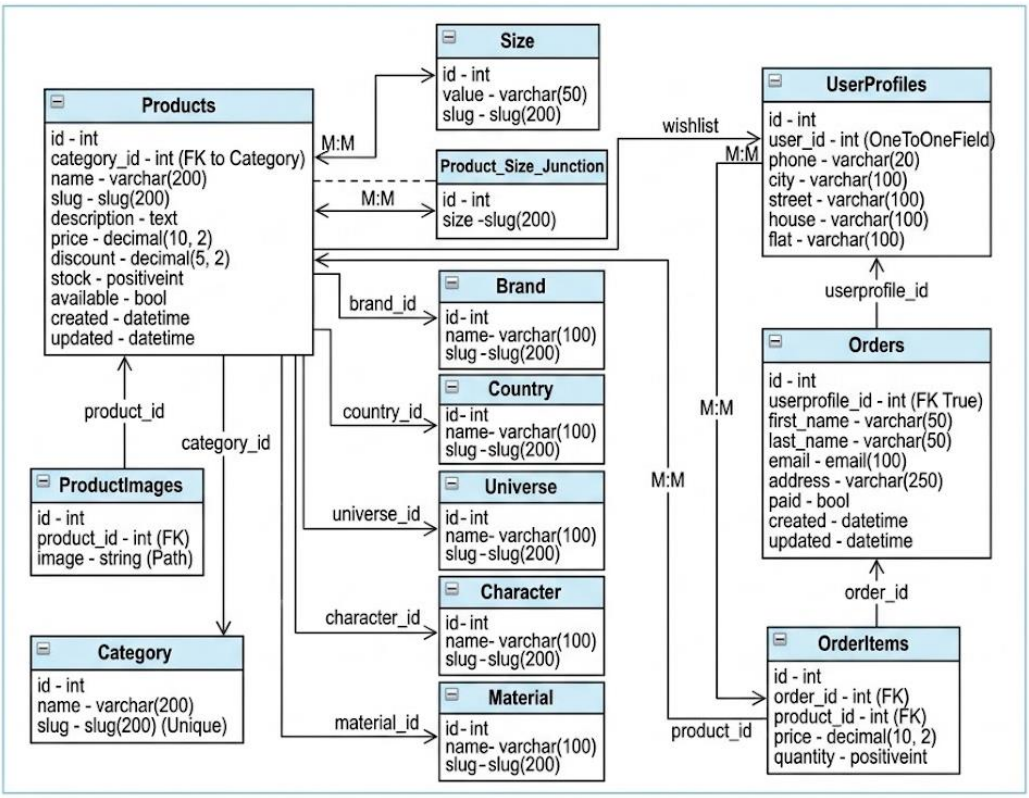

Рисунок 2.1 – Логічна схема бази даних

Всю інфраструктуру збереження даних системи розподілено на чотири логічні блоки (рис 2.1):

1. Довідники нішевих атрибутів, Category, Brand, Country, Universe, Character, Size, Material.
2. Ядро каталогу товарів, Product, ProductImage.
3. Модуль користувачів, UserProfile на базі User.
4. Транзакційний модуль Order, OrderItem.

Сутності та їх атрибути:

Category

- id – унікальний ідентифікатор категорії
- name – назва категорії

- slug – унікальний URL-ідентифікатор

Brand / Country / Universe / Character / Material

- id – унікальний ідентифікатор ознаки
- name – унікальне ім'я або назва ознаки
- slug – унікальний URL-ідентифікатор для фільтрації товарів на сайті

Size

- id – унікальний ідентифікатор розміру
- value – значення масштабу фігурки
- slug – унікальний URL-ідентифікатор для фільтрації

Product

- id – унікальний ідентифікатор товару
- category – зв'язок з категорією
- name – найменування товару
- slug – URL-слаг
- description – текстовий опис фігурки
- price – базова ціна товару
- discount – знижка на товар у відсотках
- stock – кількість одиниць товару на складі
- available – прапорець (так/ні) відображення товару на сайті
- created – дата та час створення картки товару
- updated – дата та час останнього оновлення даних про товар
- brand – зв'язок з брендом
- country – зв'язок з країною
- universe – зв'язок із всесвітами аніме
- character – зв'язок з персонажами
- size – зв'язок з розмірною сіткою
- material – зв'язок з матеріалами

ProductImage

- id – унікальний ідентифікатор зображення
- product – зв'язок з товаром
- image – шлях до файлу зображення на сервісі

UserProfile

- id – унікальний ідентифікатор профілю
- user – строгий зв'язок з системним користувачем Django
- phone – контактний телефон покупця
- city – місто доставки
- street – вулиця доставки
- house – номер будинку
- flat – номер квартири
- wishlist – список бажаних товарів

Order

- id – унікальний ідентифікатор замовлення
- user – зв'язок з покупцем
- first_name – ім'я отримувача посилки
- last_name – прізвище отримувача посилки
- email – електронна пошта для сповіщень про замовлення
- address – повна логістична адреса доставки
- paid – статус оплати
- created – дата та час створення замовлення
- updated – дата та час зміни статусу замовлення

OrderItem

- id – унікальний ідентифікатор позиції
- order – зв'язок з документом замовлення
- product – зв'язок з купленим товаром
- price – фіксація ціни на момент безпосередньої покупки
- quantity – кількість куплених одиниць фігурок в цій позиції

На цьому побудову бази даних можна вважати завершеною.

Побудова об’єктно орієнтовної моделі

В основі реалізації нашого інтернет магазину є проектування Model-View-Template [11] (рис 2.2), який використовується у Django. ORM дозволяє описати структуру бази даних у вигляді класів, де кожен клас – таблиця в базі даних, а атрибути класу – поля.

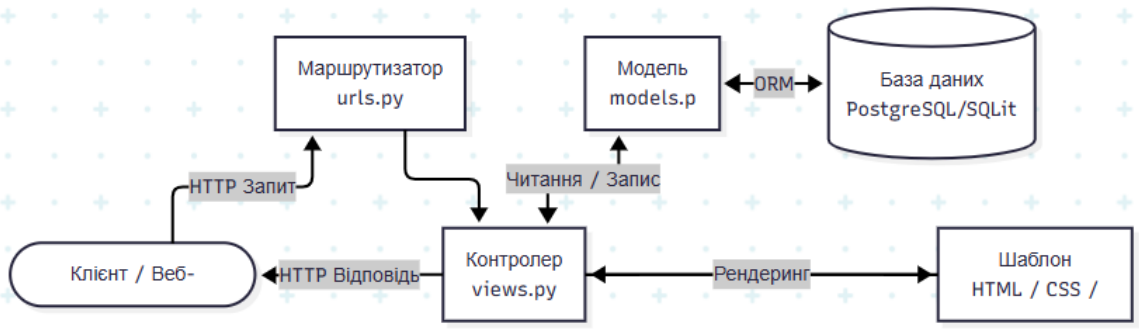

Рисунок 2.2 – Архітектура взаємодії компонентів системи

При проєктуванні БД було розробленно три основні сутності:

1. Каталог товарів та система класифікації.
2. Управління користувачами та їх персональними даними.
3. Система оформлення та обробки замовлень.

Основна сутність – Product, який містить всю інформацію про товар.

Для реалізації системи фільтрації в інтернет магазині побудовано систему зв'язків. Зв'язок "один-до-багатьох" реалізовано за допомогою зовнішніх ключів для сутностей: Category, Brand.

Зв'язки типу "багато-до-багатьох" – один товар може належати до кількох всесвітів, містити кількох персонажів, випускатися в різних розмірах та виготовлятися з комбінації матеріалів. Для реалізації цієї логіки створено класи: Universe, Character, Size, та Material.

Також в класі Product було реалізовано метод get_absolute_url для генерації посилань на сторінку товару, а також властивість sell_price, яка

розраховує кінцевої вартості товару з урахуванням знижки. А для зберігання зображень товару спроектовано ProductImage, що дозволяє завантажувати необмежену кількість зображень для кожної позиції.

Важливим аспектом проектування інтернет-магазину є гнучка та безпечна системи автентифікації, авторизації та управління персональними даними покупців. Для її реалізації була використана інтеграція вбудованої підсистеми користувачів django.contrib.auth та спеціалізована модель розширення профілю UserProfile.

Стандартна модель User у Django забезпечує базовий функціонал, такий як збереження хешованих паролів, верифікація електронної пошти та розмежування прав доступу. А модель UserProfile розширює цей функціонал, додаючи атрибути, такі як, збереження адреси користувача та номер телефону, а головне – можливість реалізації списку бажань. Зв'язок між моделлю та профілем – "один-до-одного". Це гарантує, унікальність кожному обліковому запису.

Для підвищення автономності та стабільності системи, процес створення профілю було автоматизовано за допомогою декоратора @receiver та сигналу post_save.

Їх робота полягає в наступному: у момент створення нового профілю, сигнал post_save автоматично ініціює виконання методу create_user_profile, який створює пустий об'єкт UserProfile, зв'язаний із цим користувачем. Метод save_user_profile забезпечує синхронне оновлення та збереження даних профілю.

Для забезпечення повноцінного функціонування інтернет-магазину побудована структура бази даних взаємодіє з контролерами Views, формуючи складні вибірки QuerySets за допомогою механізмів Django ORM.

Важливим аспектом магазину є реалізація механізму пошуку та фільтрації у каталозі товарів. Для забезпечення пошуку за назвою та описом товару використовуються Q-об'єкти. Які дозволяють генерувати SQL-запити.

Фільтрація товарів за категоріями, брендами, матеріалами та іншими характеристиками написана через додавання умов до базового запиту на основі GET-параметрів. Оскільки застосовуються зв'язки типу "багато-до-багатьох", ORM може повертати дубльовані записи. Тому на етапі формування вибірки застосовується метод distinct(), який гарантує унікальність кожного товару.

На відміну від замовлень, кошик покупця реалізовано на базі механізму сесій. Він дозволяє зберігати дані анонімних користувачів у оперативній пам'яті або сховищах, наприклад, Redis. При цьому кошик зберігає лише ідентифікатори товарів, а повна інформація підтягується з моделі Product в момент відображення сторінки.

Управління списком бажань інтегровано за допомогою асинхронних запитів. Контролер toggle_wishlist дозволяє додавати або видаляти зв'язок між UserProfile та об'єктом Product без перезавантаження сторінки.

Для передачі даних про кількість товарів у списку бажань, списків категорій та брендів у всі HTML сторінки, створено контекстні процесори. Вони виконують запити до ORM і роблять ці дані доступними на будь-якій сторінці сайту.

Критичним етапом є процес оформлення замовлення, оскільки він зачіпає одразу кілька таблиць бази даних. У контролері order_create цю проблему вирішено за допомогою контекстного менеджера транзакцій with transaction.atomic() [12].

Цей механізм гарантує дотримання принципів Атомарність, Узгодженість, Ізольованість, Довговічність. Процес збереження замовлення order.save()), створення позицій замовлення OrderItem.objects.create(...) та списання товарів із залишків на складі product.stock-= item['quantity'] об'єднані в єдину транзакцію. Якщо на одному з цих етапів виникне помилка, всі попередні зміни будуть скасовані. Це унеможливлює ситуацію, коли у клієнта списуються кошти, але товар не резервується, або навпаки.

2.3. Математичне та алгоритмічне забезпечення

Алгоритмічне забезпечення інтернет-магазину описує послідовність логічних кроків та операцій, які виконує система. Архітектура побудована таким чином, щоб мінімізувати кількість звернень до бази даних та забезпечити високу швидкість обробки запитів [13]. Основними алгоритмами можна вважати процеси управління кошиком, фільтрації каталогу та оформлення замовлення.

Оскільки кошик реалізовано на базі механізму сесій, він працює як з авторизованими, так і з анонімними користувачами. Алгоритм роботи кошика складається з наступних етапів:

1. Ініціалізація. При кожному HTTP-запиті система перевіряє наявність ключа CART_SESSION_ID у користувача. Якщо кошик існує, він завантажується в оперативну пам'ять як словник; якщо ні – створюється новий порожній об'єкт.

2. Додавання та оновлення товару. При виклику методу add() алгоритм виконує конвертацію ідентифікатора товару в рядковий тип для коректного збереження у JSON-форматі.

3. Валідація залишків. Перед збереженням нової кількості товару алгоритм розраховує final_quantity. Далі відбувається перевірка: чи не перевищує бажана кількість товару залишок на складі. Якщо перевищує, система автоматично коригує кількість до максимально доступної.

4. Підрахунок та агрегація. При ітерації по кошику, алгоритм збирає всі ідентифікатори товарів, робить один оптимізований запит до БД та обчислює загальну вартість кожної позиції та всього кошика (get_total_price).

Генерація каталогу товарів побудована за наступними кроками:

1. Формування базової вибірки. Отримання всіх товарів із прапорцем available=True.

2. Повнотекстовий пошук. Якщо у запиті присутній параметр q, алгоритм застосовує регулярні вирази для пошуку збігів у назві або описі товару.

3. Категоризація. Якщо запит містить ідентифікатор категорії, вибірка фільтрується за відповідним ключем.

4. Мультифільтрація. Алгоритм зчитує GET-параметри та для кожного активного параметра застосовується оператор SQL IN (filter(field__slug__in=slugs)).

5. Нормалізація результату. Оскільки застосовуються фільтри за полями зв'язку "багато-до-багатьох", створюємо метод distinct(), який видаляє можливі дублікати товарів.

Алгоритм оформлення замовлення, (рис 2.3):

1. Валідація вхідних даних. Система перевіряє запит за допомогою форми OrderCreateForm.
2. Відкриття транзакції. Ініціюється блок with transaction.atomic().
3. Створення базового замовлення. Алгоритм створює об'єкт Order. Якщо покупець авторизований, замовлення автоматично прив'язується до його профілю.
4. Кошик та зміна складу. Для кожного елемента з кошика:
    a. Створюється запис OrderItem, що фіксує ідентифікатор товару, його ціну та кількість.
    b. Від поточного залишку товару на складі віднімається замовлена кількість. Оновлений товар зберігається в БД.
5. Закриття транзакції та очищення. Якщо всі ітерації пройшли успішно, зміни комітяться в БД. Сесійний кошик повністю очищується.
6. Завершення. Користувач перенаправляється на сторінку успішного оформлення з відображенням номера замовлення.

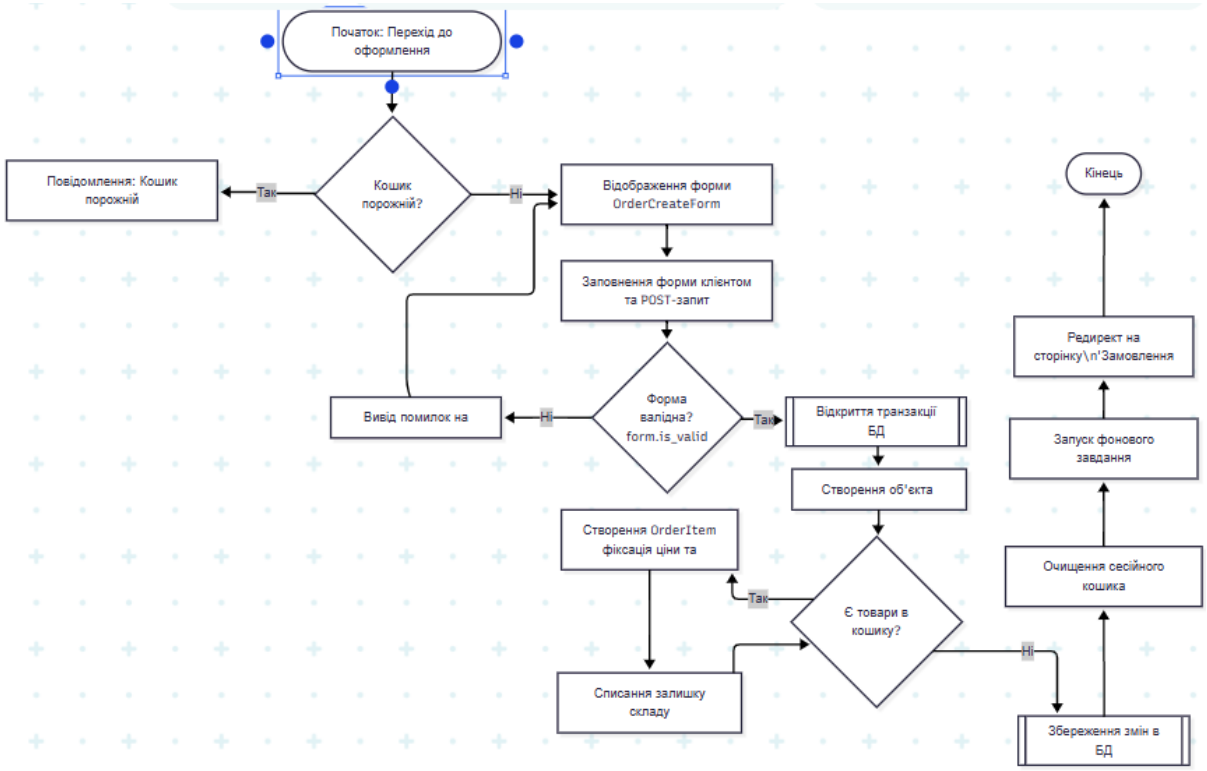

Рисунок 2.3 – Блок-схема алгоритму оформлення замовлення

Математичне забезпечення системи можна розділити на три ключові блоки.

Базовим математичним розрахунком у системі є визначення актуальної ціни продажу товару. Оскільки в моделі даних Product зберігається базова ціна та відсоток знижки, кінцева ціна обчислюється за математичною формулою розрахунку знижки:

$$sellprice = \left[price \cdot \left(1 - \frac{discount}{100}\right)\right] \qquad (2.1)$$

де:

sell_price – кінцева вартість товару, що відображається клієнту;

self.price – базова ціна товару, задана адміністратором;

self.discount – розмір встановленої знижки.

Реалізація на Python:

```
@property
def sell_price(self):
    if self.discount > 0:
        return int(self.price * (1- self.discount / 100))
    return int(self.price)
```

Для уникнення проблем із похибками обчислення чисел з плаваючою комою результати розрахунків приводяться до цілочисельного типу.

Кошик користувача та оформлене замовлення – множиною товарів. Для підрахунку загальної вартості замовлення використовується формула суми цін:

$$Total\ \mathrm{Pr}\,ice = \sum_{i=1}^{n}(price \cdot quantity) \qquad (2.2)$$

де:

get_total_price – загальна сума до сплати за замовлення;

item['quantity'] – загальна кількість позицій;

item['price'] – актуальна ціна продажу товару;

Sum() – кількість одиниць і-го товару.

Реалізація на Python:

```
def get_total_price(self):
    return sum(Decimal(item['price']) * item['quantity'] for item in
               self.cart.values())
```

Для алгоритму фільтрації каталогу:

```
products = products.filter(category=category)
products = products.filter(brand__slug__in=slugs)
products = products.distinct()
```

де:

filter(...) – команда, яка покроково відсіює товари. З кожним новим застосованим фільтром список зменшується.

category та brand__slug__in – конкретні параметри, за якими користувач шукає потрібну річ;

distinct() – команда, яка очищає фінальний список від дублікатів.

Для додавання нових товарів, обробки замовлень менеджерами, управління користувачами реалізовано підсистему управління контентом на базі модуля django.contrib.admin.

Для адаптації стандартної панелі під вимоги магазину, алгоритми відображення були кастомізовані за допомогою класів спадкоємців ModelAdmin:

У класах адміністрування створенно змінні, які керують алгоритмами формування SQL-запитів:

list_display – визначає структуру колонок у загальній таблиці записів.

list_filter – генерує бічну панель фільтрації, динамічно створюючи SQL-запити з оператором WHERE.

search_fields – підключає алгоритм повнотекстового пошуку за обраними текстовими полями бази даних.

prepopulated_fields – генерує URL-ідентифікатор.

Доступ до адміністративної панелі закритий. При спробі переходу на маршрут /admin/, система перевіряє права доступу користувача.

Висновоки до розділу

На основі аналізу предметної області та побуві матетатичного та алгоритмічного забезпечення було:

1. Розроблено структуру бази даних, систему профілів користувачів та систему транзакцій.

2. Описано формули розрахунку ціноутворення. Побудовано алгоритм пошуку та вибірки.
3. Спроєктовано кошик, процес створення замовлення та списання товарів зі складу.
4. Налаштовано адміністративної панель для менеджерів магазину.
5. Створено інтерфейс користувача із застосуванням контекстних процесорів.
6. Впроваджено систему безпеки, що включає захист від CSRF, XSS, SQL-ін'єкцій та хешування паролів [14].

РОЗДІЛ 3 ПРОГРАМНЕ ТА ТЕХНІЧНЕ ЗАБЕЗПЕЧЕННЯ

3.1. Засоби розробки

Для програмної реалізації інтернет-магазину Animeshka я обрав вебтехнології, які забезпечують високу продуктивність, масштабованість та безпеку системи.

В якості мови програмування серверної частини було обрано Python, високорівневу мову з динамічною типізацією, яка має власну бібліотеку для веброзробки [15]. Головним інструментом програмування бекенду є Django-вебфреймворк, що базується на архітектурному патерні MVT. Використання Django дозволило значно пришвидшити процес розробки завдяки вбудованій системі ORM, готовій панелі адміністрування та надійним механізмам захисту інформації.

Для побудови користувацького інтерфейсу застосовано систему шаблонізації Django HTML/CSS [16].

Для зберігання об'єктно-орієнтованої моделі використовується реляційна база даних. Під час процесу розробки та тестування застосовувалась СУБД SQLite [17], яка є стандартним рішенням у Django.

Написання програмного коду здійснювались за допомогою сучасного середовища розробки Visual Studio Code від Microsoft. Цей редактор забезпечує синтаксичний аналіз для Python, має багатий вибір розширень для веброзробки. Для резервного копіювання проєкту було використано розподілену систему контролю версій Git.

3.2. Вимоги до технічного та програмного забезпечення

Для забезпечення безпечної та безперебійної роботи розробленого інтернет-магазину, апаратне та програмне забезпечення, повинно відповідати відповідним вимогам:

1. Вимоги до серверної частини

Апаратне забезпечення:

- Багатоядерний процесор архітектури x86_64 або ARM.
- Оперативна пам'ять – мінімум 2 ГБ. Рекомендований обсяг – 4 ГБ для забезпечення швидкої роботи СУБД.
- Накопичувач – щонайменше 20 ГБ вільного місця. Рекомендується використання SSD накопичувача для забезпечення високої швидкості читання/запису.

Програмне забезпечення:

- Операційна система GNU/Linux або Windows.
- Інтерпретатор Python версії 3.11, а також віртуальне середовище з усіма залежностями з файлу requirements.txt.
- Реляційна СУБД PostgreSQL.

2. Вимоги до клієнтської частини

Апаратне забезпечення:

- Персональний комп'ютер, ноутбук, планшет або смартфон.
- Підключення до мережі Інтернет зі швидкістю від 1 Мбіт/с.

Програмне забезпечення:

- Сучасна операційна система.
- Браузер із підтримкою стандартів HTML5, CSS3 та JavaScript.
- Додаткові вимоги: у налаштуваннях браузера клієнта обов'язково має бути ввімкнена підтримка виконання JavaScript .

3.3. Опис програмної реалізації

Фреймворк Django реалізується через поділ загальної системи на ізольовані компоненти – додатки. Кожен додаток містить власні моделі даних, контролери, форми та шаблони. Такий підхід значно полегшує подальше масштабування коду.

Структура програмного забезпечення складається з головного ядра проєкту та чотирьох основних функціональних додатків (рис 3.1):

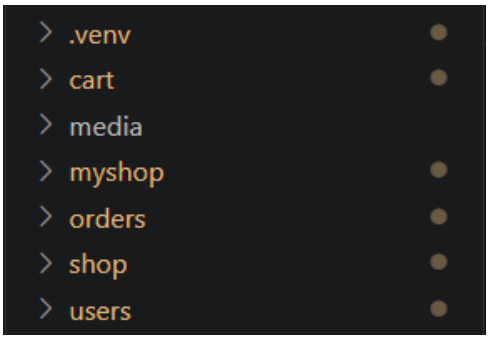

Рисунок 3.1 – Структура директорій

1. Додаток shop – є центральним модулем системи. Він містить моделі товарів, категорій, тощо. Модуль відповідає за виведення вітрини магазину, пошук та фільтрацію каталогу.

2. Додаток cart – відповідає за тимчасове зберігання обраних товарів у сесії користувача. Цей модуль не створює нових таблиць у базі даних, а зберігає словники в оперативній пам'яті сервера, реалізуючи логіку додавання, видалення та підрахунку проміжної вартості товарів.

3. Додаток orders – модуль , який вступає в роботу на етапі оформлення покупки. Він переносить дані з тимчасового кошика до бази даних, формуючи об'єкти Order та OrderItem, а також керує списанням залишків зі складу.

4. Додаток users – відповідає за систему авторизації, управління персональними даними клієнтів, збереження контактної інформації та адрес доставки, а також реалізує код списку бажань.

Механізм маршрутизації відповідає за обробку вхідних HTTP-запитів від клієнтського браузера та їх перенаправлення до відповідних контролерів Views. У проєкті застосовано багаторівневу маршрутизацію.

Головний файл urls.py, розташований у центральній директорії проєкту, не обробляє запити самостійно, а перенапрявляє їх внутрішнім маршрутизаторам за допомогою функції include().

Код головного файлу маршрутизації наведено нижче:

```
from django.contrib import admin
from django.urls import path, include
from django.conf import settings
from django.conf.urls.static import static

urlpatterns = [
    path('admin/', admin.site.urls),
    path('accounts/', include('allauth.urls')),
    path('users/', include('users.urls', namespace='users')),
    path('cart/', include('cart.urls', namespace='cart')),
    path('orders/', include('orders.urls', namespace='orders')),
    path('', include('shop.urls', namespace='shop')),
]
if settings.DEBUG:
    urlpatterns += static(
        settings.MEDIA_URL,
        document_root=settings.MEDIA_ROOT
    )
```

Як видно з наведеного програмного коду, система розроділяє зони відповідальності. Запити, що починаються з префікса /cart/, направляються до модуля кошика, а префікс /orders/ – підсистеми оформлення замовлень. Порожній шлях ('') призначений дозволяє відкривати каталог товарів одразу на головній сторінці сайту.

Додатково реалізовано умовне підключення функції static(). Цей алгоритм активується виключно в режимі розробки для забезпечення коректної віддачі браузеру медіафайлів, завантажених користувачами або адміністраторами.

Модуль каталогу товарів відповідає за формування вітрини, надання користувачеві інструментів пошуку продукції та ознайомлення з характеристиками кожної товару. Програмна реалізація цього модуля

знаходиться у файлі views.py і складається з двох основних функцій: product_list та product_detail.

Для відображення асортименту товарів розроблено контролер product_list. Його головне завдання – сформувати вибірку товару з бази даних.

Код вибірки та пошуку наведено нижче:

```
def product_list(request, category_slug=None):
    category = None
    products = Product.objects.filter(available=True)
    search_query = request.GET.get('q', '').strip()
    if search_query:
        products = products.filter(
            Q(name__iregex=search_query) |
            Q(description__iregex=search_query)
        )


if category_slug:
        category = get_object_or_404(Category, slug=category_slug)
        products = products.filter(category=category)
    filter_params = {
        'brand': request.GET.get('brand'),
        'country': request.GET.get('country'),
        'universe': request.GET.get('universe'),
        'character': request.GET.get('character'),
        'size': request.GET.get('size'),
        'material': request.GET.get('material'),
    }
```

З коду видно, алгоритм розпочинає роботу з відсіювання товарів, які наразі недоступні для продажу . Далі контролер аналізує словник request.GET на наявність пошукового запиту q. Використання об'єктів Q дозволяє виконувати пошук одночасно за назвою та описом товару за допомогою регулярних виразів , ігноруючи регістр введених символів.

Після застосування всіх необхідних фільтрів контролер формує словник, до якого передаються очищені від дублікатів товари, списки категорій, брендів та матеріалів для відображення у бічній панелі (рис 3.2).

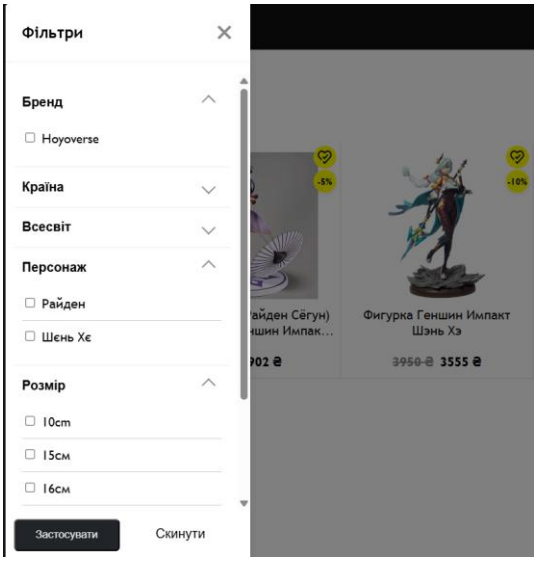

Рисунок 3.2 – Графічний інтерфейс каталогу товарів із системою фільтрації

Для ознайомлення покупця з повною інформацією про конкретну позицію реалізовано контролер product_detail (рис 3.3), який генерує сторінку з інформацією товару, зображенням та можливістю додавання у кошик. Ця функція обробляє запити, що містять унікальний ідентифікатор та посилання товару.

```
def product_detail(request, id, slug):
    product = get_object_or_404(Product,
                                id=id,
                                slug=slug,
                                available=True)
    cart = Cart(request)
    in_cart = str(product.id) in cart.cart
    cart_product_form = CartAddProductForm(product=product)
    return render(request,
                  'shop/product/detail.html',
                  {'product': product,
                   'cart_product_form': cart_product_form,
                   'in_cart': in_cart,
                   })
```

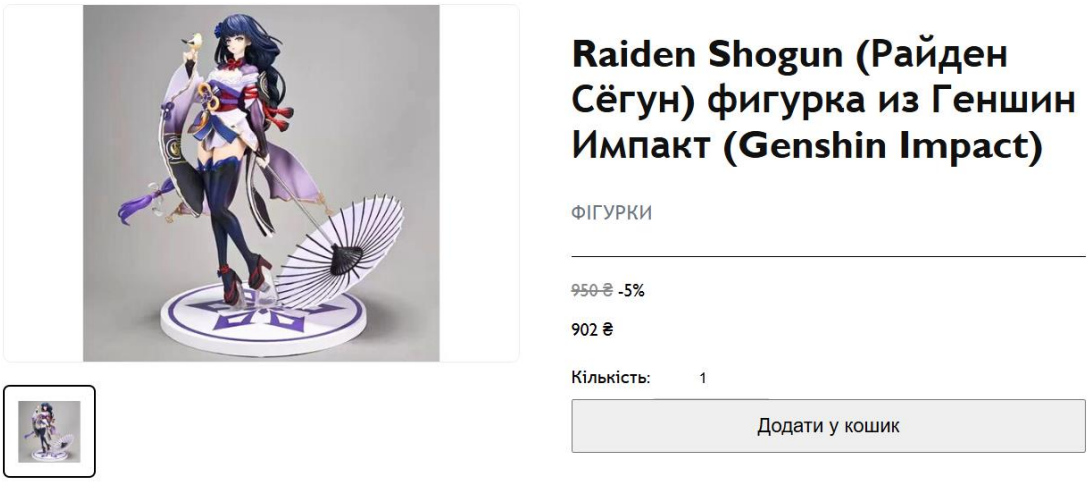

Рисунок 3.3 – Сторінка опису товару з формою додавання до кошика

Критично важливою особливістю цього контролера є використання оптимізованої функції get_object_or_404. Якщо зловмисник або пошуковий бот спробує звернутися до товару, якого не існує в базі даних, або який був знятий з продажу, система автоматично перерве виконання коду і поверне стандартну HTTP-помилку 404 . Це захищає додаток від внутрішніх збоїв сервера.

Крім того контролер створює екземпляр класу кошика і перевіряє, чи не був цей товар доданий туди раніше. Далі ініціалізується форма CartAddProductForm, у яку динамічно передається поточний об'єкт товару. Це необхідно для того, щоб обмежити максимальну кількість товару у списку відповідно до його залишку на складі.

Логіка додавання, оновлення та видалення товарів з кошика реалізована у файлі cart/views.py. Оскільки кошик змінює стан сесії користувача, доступ до контролера зміни кількості товару обмежений методами HTTP-запитів.

Для забезпечення безпеки використовується декоратор @require_POST, який блокує будь-які спроби звернутися до функції через звичайний GET-запит. Для перегляду вмісту кошика (рис 3.4) використовується контролер cart_detail.

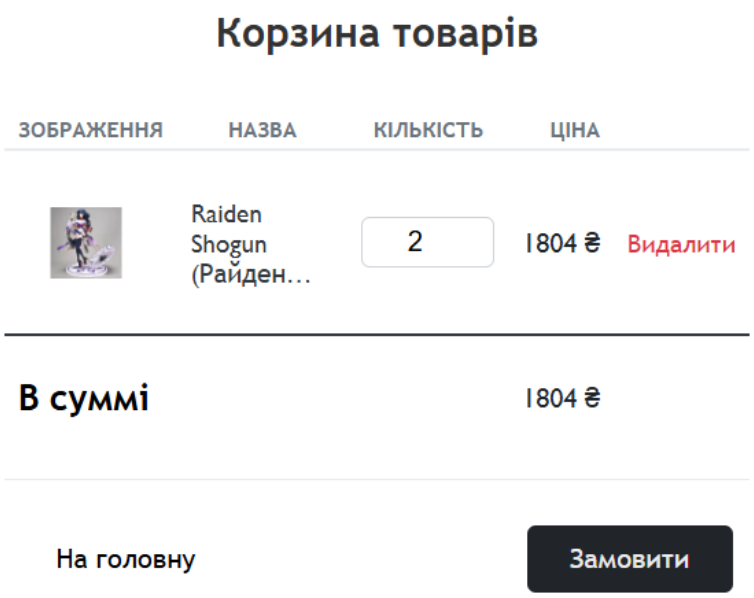

Рисунок 3.4 – Графічний інтерфейс кошика покупок із відображенням доданих товарів

Процес перенесення даних із сесійного кошика до бази даних реалізується контролером order_create у додатку orders.

```
def order_create(request):
    cart = Cart(request)
    if request.method == 'POST':
        form = OrderCreateForm(request.POST)
        if form.is_valid():
            with transaction.atomic():
                order = form.save(commit=False)
                if request.user.is_authenticated:
                    order.user = request.user
                order.save()
                for item in cart:
                    OrderItem.objects.create(order=order,
                                             product=item['product'],
                                             price=item['price'],
                                             quantity=item['quantity'])
                    product = item['product']
                    product.stock -= item['quantity']
                    product.save()
```

```
            cart.clear()
            order_created_task.delay(order.id)
            return redirect(reverse('orders:order_created', args=[order.id]))
    else:
        form = OrderCreateForm()
    return render(request, 'orders/create.html',
                  {'cart': cart, 'form': form})

def order_created(request, order_id):
    order = get_object_or_404(Order, id=order_id)
    return render(request, 'orders/created.html', {'order': order})
```

Блок коду with transaction.atomic(): гарантує, що збереження об'єкта, створення кожної окремої позиції та математичне зменшення залишків товарів на складі будуть виконані як єдина операція. У разі виникнення будь-якої системної помилки на одному з етапів, усі попередні зміни будуть автоматично скасовані, що захищає магазин від розсинхронізації даних.

Після успішної транзакції, система викликає метод cart.clear() для видалення збережених даних із сесії браузера. Сервер повертає користувачеві HTTP-перенаправлення на сторінку подяки за покупку (рис 3.5).

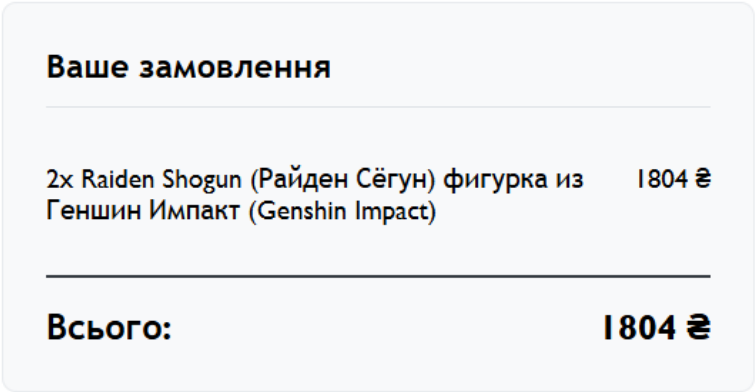

Рисунок 3.5 – Сторінка завершення покупки

Середовище користувача дозволяє покупцям керувати своїми персональними даними (рис 3.6), переглядати історію замовлень та формувати власний список бажань. Оскільки архітектура розділена на базову інформацію для авторизації та специфічних даних клієнта магазину, алгоритм редагування

профілю вимагає синхронної роботи з двома окремими сутностями. Його логіку реалізовано у контролері profile_edit.

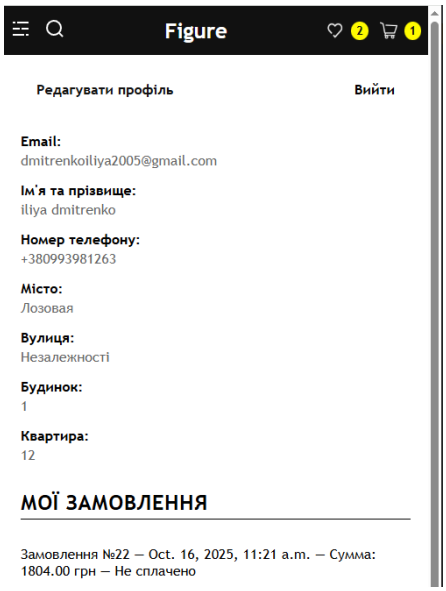

Рисунок 3.6 – Інтерфейс особистого кабінету користувача

Для захисту персональних даних застосовується декоратор @login_required, який гарантує, що доступ до цього контролера мають виключно авторизовані користувачі. Алгоритм ініціалізує одразу дві форми UserUpdateForm та UserProfileUpdateForm, передаючи в них поточні екземпляри об'єктів. Це дозволяє заповнити поля HTML-форми вже існуючими даними клієнта. Після успішної валідації обох форм система виконує збереження змін у базу даних.

Для підвищення зручності користування каталогом , процес додавання товарів до списку бажань (рис 3.7) реалізовано без перезавантаження вебсторінки за допомогою технології AJAX з додаванням коду на мові JavaScript [18]. За серверну обробку таких запитів відповідає контролер toggle_wishlist.

```
@login_required
@require_POST
def toggle_wishlist(request, product_id):
```

```
product = get_object_or_404(Product, id=product_id)
profile = request.user.profile
if product in profile.wishlist.all():
    profile.wishlist.remove(product)
    action = 'removed'
else:
    profile.wishlist.add(product)
    action = 'added'
return JsonResponse({
    'status': 'ok',
    'action': action,
    'wishlist_count': profile.wishlist.count()
})
```

Алгоритм використовує метод перемикача. Після ідентифікації товару за його id система перевіряє, чи присутній цей об'єкт у зв'язках ManyToMany поточного профілю користувача. Якщо товар вже є у списку – він видаляється, якщо ні – додається.

Замість класичного рендерингу HTML-шаблону, цей контролер повертає структуровану відповідь у форматі JsonResponse. Словник містить інформацію про виконану дію та актуальну кількість товарів у списку бажань. JavaScript перехоплює цю відповідь та миттєво оновлює іконку обраного товару та загальний лічильник в навігаційному меню магазину.

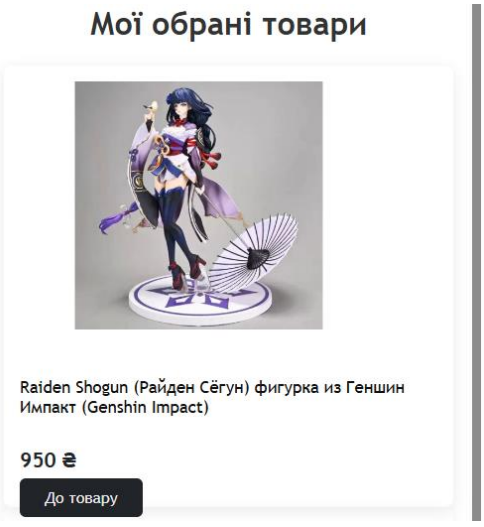

Рисунок 3.7 – Сторінка списку бажань користувача із збереженими товарами

Для забезпечення управління життєвим циклом магазину у системі реалізовано підсистему управління контентом. Її побудовано на базі вбудованого модуля django.contrib.admin, який динамічно генерує графічний інтерфейс для виконання операцій Create, Read, Update, Delete (рис 3.8).

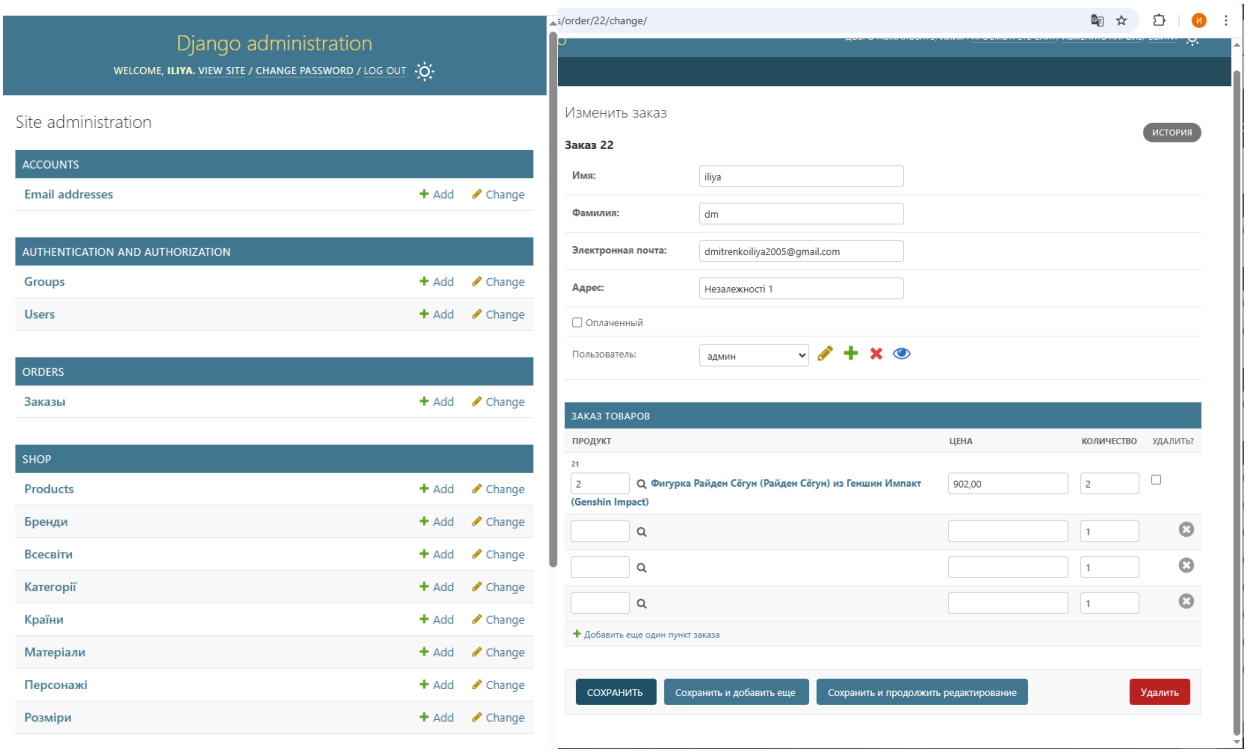

Рисунок 3.8 – Інтерфейс адміністративної панелі

3.4. Керівництво користувача

Оскільки розроблений магазин є вебдодатком, користувачеві не потрібно встановлювати програмне забезпечення, достатньо мати сучасний веббраузер та підключення до мережі Інтернет.

Клієнтська частина інтернет-магазину спроєктована так, щоб процес пошуку та покупки товарів був інтуїтивно зрозумілим.

При переході на головну сторінку вебдодатка користувач потрапляє до загального каталогу товарів.

- У верхній частині екрана розташоване головне навігаційне меню, яке містить рядок глобального пошуку, іконки доступу, списку бажань та кошика та бургер-меню .
- Для пошуку товару користувач може ввести ключове слово у рядок пошуку та натиснути клавішу Enter або іконку лупи (рис 3.9).
- Для звуження результатів пошуку на сторінці каталогу створено панель фільтрації. Користувач може обрати категорію товарів, відфільтрувати їх за характеристиками.

Рисунок 3.9 – Головна сторінка каталогу та використання рядка пошуку

При натисканні на картку товару в каталозі, користувач переходить на сторінку детального опису, де відображається зображення , актуальна ціна, та текстовий опис (рис 3.3).

Якщо користувач авторизований у системі, він може додати товар до списку бажань, натиснувши на іконку у формі серця. Іконка змінить свій стан, а лічильник у верхньому меню збільшиться. При спробі натиснути на кнопку неавторизованим користувачем, система запропонує йому авторизуватися.

Для здійснення покупки користувач повинен виконати наступні кроки:

1. Обрати товар і натиснути кнопку, додати в кошик.

2. Після цього можна продовжити покупки або перейти до кошика, натиснувши на відповідну іконку у верхньому правому куті екрана.
3. На сторінці кошика маємо змогу перевірити обрані товари, змінити їх кількість або зайві позиції.
4. Натиснути кнопку оформити замовлення.
5. На сторінці оформлення заповнюємо форму з даними: ім'я, прізвище, електронна пошта, номер телефону та точна адреса доставки.
6. Підтвердити замовлення, система обробить запит і підвантажить сторінку успішного оформлення, де буде вказано унікальний номер замовлення.

Для отримання доступу до розширеного функціонала користувач повинен зареєструватися.

1. Після авторизації, у розділі профілю, користувач може змінити свою контактну інформацію, яка буде використовуватися для наступних замовлень.
2. У розділі історія замовлень відображається таблиця з усіма попередніми покупками, їх статусами та загальною сумою.
3. У розділі список бажань знаходиться перелік збережених товарів, які користувач планує придбати в майбутньому.

Для управління контентом та обробки замовлень передбачена закрита адміністративна панель. Доступ до неї мають лише користувачі з відповідними правами.

1. Для входу в панель керування необхідно додати /admin/ до адреси сайту.
2. На екрані з'явиться форма авторизації, де необхідно ввести логін та пароль адміністратора.
3. Після входу відкривається інформаційна панель, де всі об'єкти згруповані за додатками (Каталог, Замовлення, Користувачі) (рис 3.10).

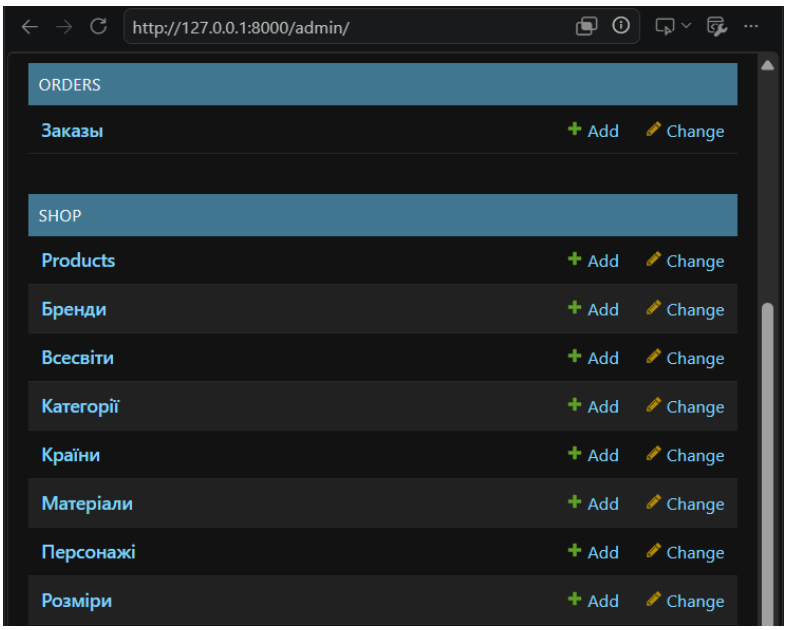

Рисунок 3.10 – Головний екран панелі адміністратора

Для управління каталогом товарів менеджер використовує розділ Shop панелі керування. Створення нової позиції передбачає заповнення базових характеристик у відповідній формі: назви, категорії, ціни з можливою знижкою, кількості на складі та завантаження зображення. Редагування існуючих позицій здійснюється через загальний список товарів із використанням інструментів пошуку та фільтрації.

Обробка покупок здійснюється у розділі orders, куди надходять усі нові заявки. Відкривши замовлення, менеджер отримує доступ до інформації: контактних даних, адреси доставки та переліку придбаних товарів.

Додатково передбачено розділ Users. Адміністратор має змогу переглядати контактні дані користувачів, відновлювати їх паролі за запитом та призначати розширені права доступу для нових співробітників.

Висновки до розділу

Було здійснено практичну реалізацію інтернет-магазину та розроблено документацію для користувачів. За результатами виконання можна зробити наступні висновки:

Для побудови серверної частини використано мову програмування Python та фреймворк Django, що дозволило пришвидшити процес створення бекенду [19].

1. Визначено мінімальні вимоги до апаратного і програмного забезпечення серверного середовища та клієнтських пристроїв.
2. Наведено лістинг програмного коду та описано логіку роботи ключових структур: генерацію вітрини товарів, роботу із кошиком, алгоритм збереження замовлень, профілю користувача та списку бажань.
3. Продемонстровано реалізацію адміністративної панелі.
4. Складено інструкцію експлуатації вебдодатка.

Таким чином, розроблений інтернет-магазин, має зручний та інтуїтивно зрозумілий інтерфейс, реалізує ключові механіки електронної комерції.

РОЗДІЛ 4 ОХОРОНА ПРАЦІ

4.1. Організаційно-правові основи забезпечення безпеки праці

Охорона праці відіграє ключову роль у процесі створення безпечного, здорового та комфортного робочого середовища. Її призначення полягає не лише у запобіганні нещасним випадкам чи професійним захворюванням, але й у збереженні високої працездатності фахівців під час виконання їхніх обов'язків [20]. В умовах використання комп'ютерної техніки, під час проєктування та розробки програмного забезпечення, коли фахівець проводить тривалий час за екраном монітора, правильного освітлення та дотримання оптимальних режимів праці й відпочинку стають критично важливими. Забезпечення належних умов праці безпосередньо впливає на рівень стомлюваності, концентрацію уваги та, як наслідок, на якість та ефективність інтелектуальної діяльності розробника.

Законодавча база України у сфері охорони праці гарантує захист прав працівників. ЇЇ фундаментом є Конституція України, яка на найвищому рівні закріплює право кожного громадянина на безпечні та здорові умови праці, а також на належні санітарно-побутові умови [21].

Основним спеціалізованим документом є Закон України «Про охорону праці». Він роз’яснює механізми реалізації прав і визначає принципи державної політики в цій сфері. Закон розподіляє обов'язки: роботодавець зобов'язаний за власні кошти створити безпечні умови праці на кожному робочому місці, а працівник, зі свого боку, повинен дбати про особисту безпеку, знати та виконувати вимоги інструкцій з охорони праці [22].

Для робочих місць, обладнаних персональними комп'ютерами, також діють специфічні санітарні норми та державні стандарти, які жорстко регламентують параметри мікроклімату, рівень шуму, освітленість та допустимі рівні електромагнітного випромінювання у приміщеннях.

4.2. Характеристика об’єкта та виявлення потенційних небезпек

Об'єктом нашого дослідження є робоче місце розробника програмного забезпечення, яке організоване в домашніх умовах. Робоче середовище - кімната, що має як природне освітлення через вікно, так і систему штучного освітлення. Мікроклімат приміщення підтримується за допомогою побутової системи опалення та регулярного провітрювання.

Робоче місце обладнане стандартним комп'ютерним столом та ергономічним кріслом із можливістю регулювання висоти сидіння і кута нахилу спинки. Основним обладнанням є персональний комп'ютер. На столі також розміщені периферійні пристрої: клавіатура, миша, акустична система.

Специфіка діяльності передбачає тривале перебування у статичній сидячій позі та безперервну фіксацію погляду на екран монітора. Це зумовлює високу концентрацію уваги та нервове напруження.

На основі аналізу умов праці та робочого середовища можна виявити низку небезпек, які будуть супроводжувати процес розробки. Їх можна класифікувати на фізичні, психофізіологічні та небезпеки загального характеру [23].

Таблиця 2.1 – Перелік потенційних небезпек та шкідливих виробничих факторів на робочому місці

Група небезпек	**Назва фактора**	**Джерело небезпеки**	**Можливі наслідки**
Фізичні	Підвищене значення напруги в електричному колі	Комп'ютер, блок живлення, розетки, подовжувач	Ураження електричним струмом
Фізичні	Нераціональне або недостатнє освітлення	Відблиски на моніторі, неправильне розташування світильників	Зниження гостроти зору, швидка втомлюваність очей

Психофізіологічні	Перенапруження зорового аналізатора	Екран монітора під час тривалого читання коду	«Синдром сухого ока», спазм акомодації, головний біль
Психофізіологічні	Статичне фізичне перевантаження	Тривале перебування у незручній позі за столом	Остеохондроз, порушення кровообігу, болі у спині та шиї
Психофізіологічні	Розумове та нервово-емоційне перенапруження	Складні логічні задачі, дедлайни, пошук помилок у коді	Стрес, хронічна втома, порушення сну
Загальні	Загроза виникнення пожежі	Коротке замикання електропроводки або перегрів техніки	Опіки, отруєння продуктами горіння, матеріальні збитки
Загальні	Військова загроза	Ракетні удари, атаки БПЛА у період воєнного стану	Пряма загроза життю та здоров'ю, травмування

4.3. Дослідження ризику реалізації небезпек та розробка заходів безпеки

Оцінка ризику – це систематичний і структурований процес ідентифікації, аналізу та оцінювання ймовірності настання небезпечних подій, а також тяжкості їхніх наслідків для здоров'я людини. Основна мета цієї процедури полягає у визначенні того, чи є поточний рівень ризику прийнятним, та у формуванні бази для прийняття рішень щодо управління безпекою [24].

До ключових задач оцінки ризику належать:

1. систематизація виявлених небезпек на робочому місці;
2. визначення сценаріїв, за яких ці небезпеки можуть призвести до шкоди;
3. пріоритезація ризиків за ступенем їхньої критичності;
4. обґрунтування необхідних профілактичних або захисних заходів.

Особливістю процесу є те, що ризик не завжди можна повністю усунути, тому головним завданням стає його зниження до прийнятного рівня.

Для оцінки ризиків виявлених небезпек під час роботи за комп'ютером використаємо матричний метод, де рівень ризику визначається за двома критеріями: ймовірність виникнення та тяжкість наслідків .

Рівень ризику = Ймовірність × Тяжкість.

Оцінка ризику для обраних небезпек:

Таблиця 4.2 – Матриця оцінки ризику перенапруження зору

Перенапруження зору				
Визначення категорії серйозності небезпеки		Визначення рівня ймовірності небезпеки		Індекс ризику небезпеки
Вид, категорія	Опис	Вид, рівень	Опис	
III – гранична	Зниження гостроти зору, «синдром сухого ока», спазм акомодації	A – часта	Відбувається постійно через специфіку професійної діяльності	3A – неприпустимий (потребує обов'язкових заходів)

Таблиця 4.3 – Матриця оцінки ризику ураження електричним струмом

Ураження електричним струмом				
Визначення категорії серйозності небезпеки		Визначення рівня ймовірності небезпеки		Індекс ризику небезпеки
Вид, категорія	Опис	Вид, рівень	Опис	
I – катастрофічна	Смертельний наслідок або тяжкі опіки, ураження внутрішніх органів та зупинка серця	D – віддалена	Настання малоймовірне за умови використання справної техніки та непошкодженої ізоляції	1D – небажаний (допустимий за умови постійного контролю)

Таблиця 4.4 – Матриця оцінки ризику виникнення пожежі

Виникнення пожежі				
Визначення категорії серйозності небезпеки		Визначення рівня ймовірності небезпеки		Індекс ризику небезпеки
Вид, категорія	Опис	Вид, рівень	Опис	

I – катастрофічна	Пряма загроза життю та здоров'ю, отруєння чадним газом, значна втрата майна	D – віддалена	Може статися вкрай рідко у звичайних умовах експлуатації	1D – небажаний (допустимий за умови постійного контролю)

За результатами проведеної оцінки, найвищий рівень ризику становить перенапруження зору, що вимагає проведення заходів для зниження рівня небезпеки. Небезпеки ураження електричним струмом та виникнення пожежі мають індекс 1D, тобто є допустимими за умови постійного контролю та дотримання базових правил експлуатації техніки.

Для мінімізації виявлених ризиків та створення безпечних умов праці на робочому місці розроблено комплекс профілактичних та захисних заходів.

Для запобігання зниженню гостроти зору та розвитку синдрому сухого ока необхідно дотримуватися режиму праці та відпочинку, роблячи регламентовані перерви по 10-15 хвилин після кожної години роботи за монітором. Під час перерв рекомендується виконувати спеціальну гімнастику. Організаційні заходи обов'язково підкріплюються технічними засобами: використанням монітора з матрицею без мерехтіння, активацією фільтра синього світла та налаштуванням оптимальної яскравості і контрастності екрана [25].

Для мінімізації ризику ураження електричним струмом необхідно час від часу проводити візуальний огляд цілісності кабелів, штепсельних вилок та розеток. Обов'язково знеструмлювати обладнання під час грози або після завершення робочого дня, а також не допускати розміщення ємностей з рідиною поблизу комп'ютерної техніки. З технічного боку безпека забезпечується наявністю захисного заземлення в розетках, використанням якісних мережевих фільтрів із вбудованими запобіжниками та встановленням пристрою захисного відключення в електрощитку приміщення.

Профілактика пожежної небезпеки полягає у забезпеченні вільної циркуляції повітря навколо комп'ютерної техніки - вентиляційні отвори категорично заборонено перекривати, очищення обладнання від пилу для

запобігання перегріву внутрішніх компонентів, а також недопущення перевантаження електричних розеток через підключення великої кількості потужних приладів до одного подовжувача. До ключових засобів захисту належать справні автоматичні вимикачі струму в електромережі та наявність у приміщенні первинних засобів пожежогасіння.

Висновки до розділу

У розділі було розглянуто основи забезпечення безпеки праці та умови роботи розробника програмного забезпечення на домашньому робочому місці. Було визначено, що створення належних умов праці є критично важливим фактором не лише для збереження здоров'я, зниження втомлюваності та забезпечення високої ефективності інтелектуальної діяльності.

Було складено перелік потенційних небезпек, які супроводжують процес розробки. Проведено оцінку ризику для імовірних загроз. Результати аналізу показали, що найвищий рівень ризику має психофізіологічний фактор – перенапруження зорового аналізатора, що зумовлено специфікою постійної роботи за монітором. Інші фізичні та загальні небезпеки, такі як ураження електричним струмом та можливість виникнення пожежі, мають небажаний, але допустимий рівень ризику за умови постійного контролю.

На основі ризиків було розроблено комплекс заходів. Зокрема, для мінімізації впливу на зір рекомендовано дотримання режиму праці та відпочинку та правильне налаштування монітора. Для запобігання фізичним небезпекам дотримуватись інструкції з електро та пожежної безпеки, своєчасний огляд обладнання, використання надійних мережевих фільтрів із заземленням та забезпечення вільної вентиляції техніки.

Дотримання запропонованих рекомендацій дозволить знизити вплив факторів до прийнятного рівня.

ЗАГАЛЬНІ ВИСНОВКИ

У результаті виконання роботи було успішно спроєктовано систему електронної комерції Animeshka. Головною метою проєкту було створення сучасного інтернет-магазину з продажу аніме-фігурок, який задовольняє потреби цільової аудиторії.

Під час розробки було пройдено всі етапи створення програмного забезпечення:

На початковому етапі було проведено аналіз предметної області та існуючих аналогів, таких як Pulsar, Pecorineshop, Rozetka. Дослідження виявило, що наявні платформи мають застарілий інтерфейс, проблеми з масштабуванням або не адаптовані під специфіку колекційних товарів.

Розроблено реляційну базу даних, , побудовано об'єктно-орієнтовану модель на базі архітектури MVT. Також створено математичні моделі та алгоритми для обчислення ціноутворення з урахуванням знижок, реалізації систем фільтрації та механізму оформлення замовлень із використанням атомарних операцій.

Програмну частину виконано за допомогою мови Python та вебфреймворку Django. Впроваджено сесійний кошик, який працює для неавторизованих користувачів, та систему списку бажань, що функціонує асинхронно без перезавантаження вебсторінки. Налаштовано розширену адміністративну панель, яка дозволяє менеджерам керувати каталогом і обробляти нові замовлення. Увесь вихідний код проєкту розміщено у відкритому репозиторії на GitHub за посиланням.

Було проаналізовано домашнє робоче місце розробника програмного забезпечення. Виявлено, що найбільш критичним ризиком є перенапруження зору. Для усунення ризиків було розроблено комплекс заходів.

Загалом, розроблений інтернет-магазин Animeshka є масштабованим та безпечним вебзастосунком. Він повністю відповідає вимогам технічного завдання.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Краус К. М., Краус Н. М., Манжура О. В. Електронна комерція та Інтернет-торгівля : навч.-метод. посіб. Київ : Аграр Медіа Груп, 2021. С. 454-456

2. Рожко В. І. Рекомендації щодо удосконалення маркетингової діяльності інтернет-магазину. Актуальні проблеми соціально-економічного розвитку в контексті євроінтеграції України *:* колективна монографія. Львів ; Торунь : Ліга-Прес, 2023. С. 23–42. DOI: 10.36059/978-966-397-323-4-23

3. olga_cpa. Переваги інтернет-магазинів у порівнянні з традиційними магазинами. Блог Cpashka. 2024. URL: https://cpashka.biz/blog/perevahy-internet-mahazyniv-u-porivnianni-z-tradytsijnymy-mahazynamy/ (дата звернення: 14.06.2026).

4. Булаєнко М.В. Математичні моделі оцінки ризиків у кібербезпеці в умовах цифрової трансформації / XXXVIII Міжнародна науково-практична конференція молодих вчених і студентів «Трансформація економічних систем та інститутів у нових геостратечічних реаліях» 14-15 квітня 2025 р. 2025 р. [Електронне видання]. У 2-х томах. Том 2. Дніпро: Університет імені Альфреда Нобеля, 2025. 67 с. https://duan.edu.ua/wp-content/uploads/2025/04/materialy-khxkhviii-mizhnarod.-nauk.-prakt.-konf._tom_2_2025.pdf

5. Django Software Foundation. Django documentation. Django. 2026. URL: https://docs.djangoproject.com/ (дата звернення: 14.06.2026).

6. Kurose J. F., Ross K. W. Computer Networking: A Top-Down Approach. 2021. С. 94.

7. Булаєнко М.В., Ликова В.І. Аналіз та покращення ux/ui, як складової якості програмного забезпечення. / Матеріали II (VIII) Міжнародної Інтернет-конференції здобувачів вищої освіти і молодих учених / Запоріжжя. 2-4 квітня. 2025 р. С. 192-193. https://knameedu.sharepoint.com/:b:/s/msteams_73a185/Ea_XlAC8icJGtpiD1WgQJ8IBEC7yXUKL9YkBB7J2Qao2CA?e=Pp54lz

8. HTML: HyperText Markup Language. MDN Web Docs. 2025. URL: https://developer.mozilla.org/en-US/docs/Web/HTML (дата звернення: 14.06.2026).

9. Булаєнко М. В., Назаренко Д.А. Реінжиніринг бізнес-процесів управління продажами для підприємства малого бізнесу / Матеріали VI Міжнародної науково-практичної конференції здобувачів вищої освіти і молодих учених «Перспективи розвитку територій: теорія і практика» 16-17 листопада 2022 р. https://science.kname.edu.ua/images/dok/konferentsii/2022/Tezy_2022/2022_molodi_vcheni.pdf

10. Database transactions. Django Project Documentation. URL: https://docs.djangoproject.com/en/6.0/topics/db/transactions/ (дата звернення: 14.06.2026).

11. Django Project MVT Structure. GeeksforGeeks. 2026. URL: https://www.geeksforgeeks.org/python/django-project-mvt-structure/ (дата звернення: 14.06.2026).

12. Xiang ZHU. Python thread safe operations. 2024. URL: https://copdips.com/2024/10/python-thread-safe-operations.html (дата звернення: 14.06.2026).

13. Булаєнко М.В., Підлужний П.А. Інтеграція малих мовних моделей у бізнес-процеси: архітектурні підходи та забезпечення структурної цілісності в системах реального часу // Комунальне господарство міст, Сер.: Інформаційні технології та інженерія. - 2026. - Вип. 1(196). - С. 34-41. DOI: https://doi.org/10.33042/3083-6727-2026-1-196-34-41

14. ДОЛІНОВСЬКИЙ Р. М. Методи захисту веб-застосунку від XSS і CSRF вразливостей, 2023 С. 29-30

15. Булаєнко М. В., Пісарєв Д.С. Моделювання та розробка інтерактивної гри // Комунальне господарство міст, Сер.: Технічні науки та архітектура. - 2021. - Вип. 6(166). - С. 15-19. DOI 10.33042/2522-1809-2021-6-166-15-19

16. CSS: Cascading Style Sheets. MDN Web Docs. URL: https://developer.mozilla.org/en-US/docs/Web/CSS (дата звернення: 14.06.2026).

17. SQLite Documentation. SQLite. URL: https://sqlite.org/docs.html (дата звернення:14.06.2026).

18. JavaScript. MDN Web Docs. URL: https://developer.mozilla.org/en-US/docs/Web/JavaScript (дата звернення: 14.06.2026).\

19. Official Python 3 Documentation.URL: https://docs.python.org/3/ (дата звернення: 14.06.2026).

20. Про охорону праці : Закон України від 14.10.1992 № 2694-XII. База даних «Законодавство України» / Верховна Рада України. URL: https://zakon.rada.gov.ua/laws/show/2694-12 (дата звернення: 14.06.2026)

21. Конституція України : Закон України від 28.06.1996 № 254к/96-ВР. База даних «Законодавство України» / Верховна Рада України. URL: https://zakon.rada.gov.ua/laws/show/254%D0%BA/96-%D0%B2%D1%80 (дата звернення: 14.06.2026).

22. Про затвердження Вимог щодо безпеки та захисту здоров'я працівників під час роботи з екранними пристроями : Наказ Міністерства соціальної політики України від 14.02.2018 № 207. База даних «Законодавство України» / Верховна Рада України. URL: https://zakon.rada.gov.ua/laws/show/z0508-18#Text (дата звернення: 14.06.2026).

23. Системи управління охороною здоров'я та безпекою праці. Вимоги та настанови щодо застосування : ДСТУ ISO 45001:2019. – Київ : ДП «УкрНДНЦ» – 9с.

24. ДСТУ EN 31010:2013. Керування ризиком. Методи загального оцінювання ризику (EN 31010:2010, IDT). Київ : Мінекономрозвитку України. 2с.

25. Охорона праці в галузі інформаційних технологій : навч. посіб. / О. Г. Левченко, О. М. Полукаров та ін. Київ : КПІ ім. Ігоря Сікорського, 2021 С. 158-159

Додаток А

Фрагмент коду головної сторінки

```
<div class="top-container">
<button id="open-filter-btn" class="filter-trigger-button">
   <span>Фільтри</span>
</button>
      <div id="main" class="product-list">
      <h1>{% if category %}{{ category.name }}{% else %}Товари{% endif %}</h1>
      {% for product in products %}
         <div class="item">
            <a href="{{ product.get_absolute_url }}">
               <img src="{% if product.images.all %}{{ product.images.first.image.url }}{% else
%}{% static 'img/no_img.png' %}{% endif %}"
               alt="{{ product.name }}"
               class="product-image">
               </a>
            <div class="name-info">
               <a href="{{ product.get_absolute_url }}">{{ product.name }}</a><br>
            </div>
            {% if user.is_authenticated %}
               <form action="{% url 'shop:toggle_wishlist' product.id %}" method="POST"
class="wishlist-form">
                  {% csrf_token %}
                  <button type="submit"
                        class="wishlist-toggle-btn {% if product in user.profile.wishlist.all
%}active{% endif %}"
                        data-liked-src="{% static 'img/liked_black.png' %}"
                        data-unliked-src="{% static 'img/like_black.png' %}">

                     {% if product in user.profile.wishlist.all %}
                        <img src="{% static 'img/liked_black.png' %}" alt="В избранном">
                     {% else %}
                        <img src="{% static 'img/like_black.png' %}" alt="Добавить в
избранное">
                     {% endif %}
```

```
                </button>
            </form>
          {% else %}
          <div class="wish-login">
            <a href="{% url 'account_login' %}?next={{ request.path }}">
              <img src="{% static 'img/like_black.png' %}" alt="Войдите, чтобы добавить в
избранное">
            </a>
          </div>
          {% endif %}
          {% if product.discount > 0 %}
            <div class="discount-info">
              -{{ product.discount }}%
            </div>
            <div class="price">
              <span class="old-price">{{ product.price }} ₴</span>
              <span class="new-price">{{ product.sell_price }} ₴</span>
          {% else %}
            <span class="price">{{ product.price }} ₴</span>
          {% endif %}
            </div>
        </div>
      {% endfor %}
  </div>
{% endblock %}
```

Фрагмент коду моделі Product

```
class Product(models.Model):
  category = models.ForeignKey(Category, related_name='products',
on_delete=models.CASCADE)
  name = models.CharField(max_length=200, db_index=True)
  slug = models.SlugField(max_length=200, db_index=True)
  description = models.TextField(blank=True)
  price = models.DecimalField(max_digits=10, decimal_places=0)
  discount = models.DecimalField(default=0, max_digits=2, decimal_places=0)
```

```
    stock = models.PositiveIntegerField()
    available = models.BooleanField(default=True)
    created = models.DateTimeField(auto_now_add=True)
    updated = models.DateTimeField(auto_now=True)
    brand = models.ForeignKey(Brand, on_delete=models.SET_NULL, null=True, blank=True)
    country = models.ForeignKey(Country, on_delete=models.SET_NULL, null=True,
blank=True)
    universe = models.ManyToManyField(Universe, blank=True)
    character = models.ManyToManyField(Character, blank=True)
    size = models.ManyToManyField(Size, blank=True)
    material = models.ManyToManyField(Material, blank=True)

    class Meta:
        ordering = ('name',)
        indexes = [
            models.Index(fields=['id', 'slug']),]
        verbose_name = 'product'
        verbose_name_plural = 'products'

    def __str__(self):
        return self.name

    def get_absolute_url(self):
        return reverse('shop:product_detail',
                       args=[self.id, self.slug])

    @property
    def sell_price(self):
        if self.discount > 0:
            return int(self.price * (1 - self.discount / 100))
        return int(self.price)


class ProductImage(models.Model):
```

```
    product = models.ForeignKey(Product, related_name='images',
on_delete=models.CASCADE)
    image = models.ImageField(upload_to='products/%Y/%m/%d')

    def __str__(self):
        return f"Image for {self.product.name}"
```