

**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**  
**НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ОДЕСЬКА ПОЛІТЕХНІКА»**

Навчально-науковий інститут комп'ютерних систем

Кафедра інформаційних систем

Олександр ЦВЕТКОВ

(група AI-214)

**КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА**

**Розробка вебресурсу з підтримки продажу гаджетів та аксесуарів**

*Спеціальність:*

122 Комп'ютерні науки

*Освітньо-професійна програма:*

Комп'ютерні науки

*Керівник:*

Оксана БАБІЛУНГА, к.т.н., доцент

## АНОТАЦІЯ

Цветков О. А. Розробка вебресурсу з підтримки продажу гаджетів та аксесуарів : кваліфікаційна робота бакалавра за спеціальністю «122 Комп'ютерні науки» / Олександр Андрійович Цветков ; керівник Оксана Юріївна Бабілунга. – Одеса : Нац. ун-т “Одес. політехніка”, 2025. – 84 с.

Кваліфікаційна робота містить основну текстову частину обсягом 57 сторінок, список використаних джерел з 21 найменувань на 2 сторінках та додатки на 20 сторінках.

Дана кваліфікаційна робота присвячена розробці вебресурсу для підтримки продажу гаджетів та аксесуарів. Розроблений вебресурс реалізований як вебсайт із клієнтською та серверною частинами, що взаємодіють із базою даних PostgreSQL. У якості серверного фреймворку було обрано Django, що забезпечує швидку розробку, безпеку та масштабованість. Клієнтська частина реалізована за допомогою Alpine.js для динамічної взаємодії на сторінках, а також Bootstrap – для забезпечення адаптивного дизайну.

Розроблений вебсайт включає базову функціональність: реєстрацію та автентифікацію користувачів, перегляд каталогу товарів, пошук і фільтрацію продукції, управління кошиком, створення замовлення з вибором способу доставки.

У процесі досягнення поставленої мети була створена проєктна документація з використанням UML-діаграм, а також підготовлено інструкцію користувача для взаємодії з вебресурсом.

*Ключові слова:* інтернет-магазин, вебресурс, замовлення гаджетів, аксесуари, вебресурс для продажу, Django, Alpine.js, PostgreSQL, Bootstrap.

## ABSTRACT

Tsvetkov A. A. Development of a web resource to support the sale of gadgets and accessories : bachelor's qualification work in the specialty «122 Computer sciences» / Alexandr Andreevich Tsvetkov ; supervisor Oksana Yuryivna Babilunha. – Odesa : Odesa Polytech. Nat. Univ., 2025. – 84 p.

The qualification work contains the main text part of 57 pages, a list of references of 21 titles on 2 pages and appendices on 20 pages.

This qualification work is devoted to the development of a web resource to support the sale of gadgets and accessories. The developed web resource is implemented as a website with client and server parts that interact with the PostgreSQL database. Django was chosen as the server framework, which provides fast development, security, and scalability. The client side is implemented using Alpine.js for dynamic interaction on the pages, as well as Bootstrap to provide responsive design.

The developed website includes basic functionality: user registration and authentication, product catalog browsing, product search and filtering, shopping cart management, order creation with a choice of delivery method.

In the process of achieving this goal, project documentation was created using UML diagrams, and a user manual was prepared for interacting with the web resource.

*Keywords:* online store, website, ordering gadgets, accessories, website for sale, Django, Alpine.js, PostgreSQL, Bootstrap.

## ЗМІСТ

Вступ.....	6
1 Огляд сайтів-аналогів та технологій їх розробки .....	7
1.1 Використання інформаційних технологій в сфері продажу гаджетів і аксесуарів.....	7
1.2 Огляд сайтів-аналогів з продажу гаджетів і аксесуарів .....	8
1.3 Порівняльний аналіз сайтів з продажу гаджетів і аксесуарів .....	14
1.4 Вибір стеку технологій для розробки вебресурсу .....	15
Висновки до розділу 1 .....	24
2 Проєктування вебресурсу з підтримки продажу гаджетів та аксесуарів .....	25
2.1 Мета та основні функції вебресурсу .....	25
2.2 Потоки даних вебресурсу для продажу гаджетів та аксесуарів .....	25
2.3 Проєктування діаграми прецедентів онлайнсервісу .....	29
2.4 Формування функціональних вимог до вебресурсу .....	31
2.5 Нефункціональні вимоги до вебресурсу магазину гаджетів і аксесуарів ....	33
2.6 Ідентифікація архетипу та проєктування інтерфейсу вебресурсу.....	35
2.7 Проєктування логічного моделі вебресурсу .....	38
2.8 Проєктування діаграми шарів вебресурсу .....	40
2.9 Проєктування розгортання вебресурсу .....	41
2.10 Моделювання динамічної поведінки вебресурсу .....	41
2.11 Проєктування моделі даних вебресурсу .....	44
2.12 Уявлення інтерфейсів вебресурсу .....	45
2.13 Забезпечення безпеки вебресурсу .....	46
2.14 Перелік технологій вебресурсу .....	47
Висновки до розділу 2.....	49
3 Програмна реалізація вебресурсу з з підтримки продажу гаджетів і аксесуарів.....	50
3.1 Структура проєкту вебресурсу .....	50
3.2 Уявлення про структуру класів вебресурсу.....	52

3.3 Використання сервісу GitHub для організації управління програмним кодом системи .....	54
3.4 Підрахунок метрик вихідного коду вебресурсу .....	54
3.5 Список якості реалізації вебресурсу .....	55
3.6 Функціональне тестування вебресурсу .....	56
Висновки до розділу 3.....	60
Загальні висновки.....	61
Список використаних джерел.....	63
Додаток А Макети сторінок вебресурсу.....	65
Додаток Б Інструкція користувача .....	67
Додаток В Вихідний код вебресурсу .....	76

## ВСТУП

**Актуальність теми роботи.** У цифрову епоху продаж у інтернеті стала ключовим елементом бізнесу. Онлайн-платформи та мобільні пристрої змінили спосіб купівлі товарів. Тепер споживачі мають доступ до асортименту цілодобово, не виходячи з дому. Особливо динамічно розвивається ринок гаджетів і аксесуарів, адже попит на них постійний, завдяки технологічним новинкам.

Попри зручність онлайн-продажів, підприємці стикаються з проблемами: конкуренція, недовіра клієнтів, висока комісія маркетплейсів. Наприклад, щоденна втрата при продажах на 1000 грн із 10% комісії становить 100 грн, а за рік – 36 000 грн. Це суттєві суми, які доцільніше інвестувати у власний сайт або бренд.

Створення інтернет-магазину – розумна альтернатива: витрати (оренда хостингу, домен, підтримка) складають мінімально 12300 грн на рік. Власний ресурс дозволяє будувати бренд, контролювати взаємодію з клієнтами та уникати залежності від сторонніх платформ, пристосовуючись під них, що забезпечує стабільність і конкурентну перевагу.

**Мета і задачі роботи.** Метою даної роботи є розробка вебресурсу з підтримки продажу гаджетів і аксесуарів.

Для досягнення поставленої мети у кваліфікаційній роботі необхідно вирішити такі задачі:

- провести огляд існуючих вебресурсів з продажу гаджетів і аксесуарів;
- здійснити проектування власного вебресурсу з продажу гаджетів;
- провести програмну реалізацію спроектованого вебресурсу;
- провести тестування розробленого програмного забезпечення.

**Об'єкт роботи.** Процес інформаційної підтримки продажу.

**Предмет роботи.** Алгоритми, засоби та технології створення вебресурсу для продажу гаджетів і аксесуарів.

Практична цінність даної розробки полягає у створенні ефективної альтернативи торгівлі на маркетплейсах, де продавці змушені щодня сплачувати високі комісії, що суттєво зменшують прибуток.

# 1 ОГЛЯД САЙТІВ-АНАЛОГІВ ТА ТЕХНОЛОГІЙ ЇХ РОЗРОБКИ

## 1.1 Використання інформаційних технологій в сфері продажу гаджетів і аксесуарів

Використання інформаційних технологій у сфері продажу гаджетів і аксесуарів стало важливим фактором для розвитку бізнесу та підвищення зручності для клієнтів. Онлайн-платформи дозволяють компаніям продавати техніку та аксесуари цілодобово, охоплюючи значно ширшу аудиторію, ніж у випадку з фізичними магазинами. За допомогою сучасних вебсайтів користувачі можуть легко переглядати каталог товарів, читати характеристики, дивитися фото чи відеоогляди, ознайомлюватися з відгуками інших покупців і швидко оформлювати замовлення.

Автоматизовані системи управління продажами допомагають спростити облік товарів, контроль наявності на складі, обробку замовлень та формування фінансової звітності. Також часто використовуються CRM-системи, які зберігають інформацію про клієнтів і дозволяють персоналізувати взаємодію з ними – наприклад, надсилати індивідуальні пропозиції або повідомлення про новинки.

Ще один важливий напрямок – це онлайн-маркетинг. Завдяки SEO, рекламі в соціальних мережах та контекстній рекламі компанії можуть ефективно привертати нових покупців і підтримувати зв'язок із постійними клієнтами. Інформаційні технології також допомагають відслідковувати ефективність рекламних кампаній і вчасно вносити корективи.

Не менш важливою є організація доставки та логістики. Багато онлайн-магазинів впроваджують системи для автоматичного розрахунку строків доставки, відстеження відправлень та інформування клієнта про статус замовлення. Це підвищує довіру до сервісу та формує позитивне враження від покупки.

У підсумку, ІТ-рішення дають змогу не лише покращити обслуговування клієнтів, а й підвищити загальну ефективність роботи компанії, зробити процес продажу швидким, зручним і сучасним.

## 1.2 Огляд сайтів-аналогів з продажу гаджетів і аксесуарів

Перш ніж вибрати технології для створення вебресурсу, орієнтованого на продажі гаджетів і аксесуарів, і приступити до проєктування додатку, важливо визначити ключові потреби клієнтів таких платформ. Варто також проаналізувати вже існуючі сайти, що працюють у цій сфері. Додатково огляд проводиться для виявлення недоліків і незручностей, з якими стикаються користувачі при купівлі гаджетів і аксесуарів. Це допоможе уникнути подібних проблем при створенні власної системи.

В результаті аналізу українських сайтів було відібрано три платформи, які будуть використовуватися як аналоги. Серед них:

- «World of Gargets» [1];
- «KiwiStore» [2];
- «GadgetShop» [3].

Тепер потрібно уважно розглянути, як працюють ці сайти: як відбувається покупка, що в них зручного, а що – ні, і яке загальне враження залишає користування ними.

### 1.2.1 Інтернет-магазин «World of Gadgets».

Інтернет-магазин «World of Gadgets» – одна з українських онлайн-платформ, що спеціалізується на продажу гаджетів і аксесуарів (рис. 1.1) [1]. Головна сторінка сайту має сучасний дизайн, що одразу створює позитивне враження у відвідувачів.

Структура вебсайту продумана та забезпечує зручну навігацію по асортименту товарів. У верхній частині сторінки розташоване головне меню з розділами, такими як «Гаджети», «Аксесуари», «Чохли», «Гаджети для дітей», «Аудіо», «Захист екрана» та «Аксесуари для авто». Кожен із цих розділів містить підкатегорії, що полегшує пошук потрібного продукту.

Структура вебсайту продумана та забезпечує зручну навігацію по асортименту товарів. У верхній частині сторінки розташоване головне меню з розділами, такими як «Гаджети», «Аксесуари», «Чохли», «Гаджети для дітей»,



«Аудіо», «Захист екрана» та «Аксесуари для авто». Кожен із цих розділів містить підкатегорії, що полегшує пошук потрібного продукту.

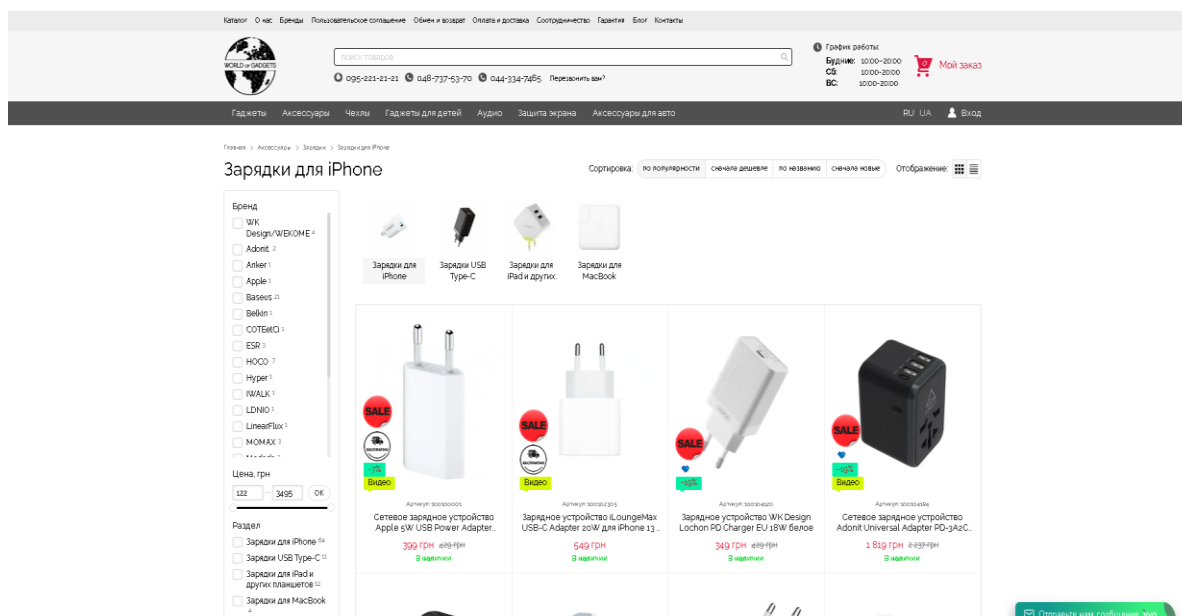


Рисунок 1.1 – Знімок екрану головної сторінки інтернет-магазину «World of Gadgets» (джерело: [1])

Структура вебсайту продумана та забезпечує зручну навігацію по асортименту товарів. У верхній частині сторінки розташоване головне меню з розділами, такими як «Гаджети», «Аксесуари», «Чохли», «Гаджети для дітей», «Аудіо», «Захист екрана» та «Аксесуари для авто». Кожен із цих розділів містить підкатегорії, що полегшує пошук потрібного продукту.

Однією з ключових особливостей «World of Gadgets» є великий асортимент товарів, що включає розумні лампи, камери, метеостанції, дверні замки, розетки, термостати та колонки. Крім того, представлені гаджети для здоров'я, геймінгу, розумні годинники, фітнес-браслети та багато іншого. Це дозволяє клієнтам знайти всі необхідні пристрої в одному місці.

Кожен продукт на сайті супроводжується детальним описом, що включає інформацію про виробника, технічні характеристики та рекомендації щодо використання. До товарів прикріплені якісні фотографії, а в деяких випадках і відеоогляди, що допомагає покупцям краще уявити обраний продукт.

На сайті також є блог, де публікуються статті про різні види гаджетів, поради щодо їх використання та огляди новинок. Це створює додаткову цінність для відвідувачів і сприяє формуванню довіри до магазину.

«World of Gadgets» підтримує кілька способів оплати, включаючи оплату банківською картою та готівкою при отриманні товару. Доставка здійснюється по всій території України за допомогою кур'єрських служб, що забезпечує зручність для клієнтів.

Для зручності покупців на сайті реалізована система відгуків і рейтингів товарів, що допомагає новим користувачам приймати обґрунтовані рішення під час вибору продукції. Команда підтримки готова допомогти у вирішенні будь-яких питань, забезпечуючи високий рівень обслуговування клієнтів.

Загалом, інтернет-магазин «World of Gadgets» пропонує користувачам зручний інтерфейс, широкий асортимент товарів і якісне обслуговування, що робить процес купівлі гаджетів і аксесуарів приємним та ефективним.

### **1.2.2 Інтернет-магазин «KIWI Store».**

Інтернет-магазин «KIWI Store» – це українська онлайн-платформа, що спеціалізується на продажу аксесуарів для пристроїв Apple та інших гаджетів. Головна сторінка сайту привертає увагу яскравими банерами та сучасним дизайном, створюючи позитивне враження у відвідувачів (рис. 1.2) [2].

Структура сайту продумана та забезпечує зручну навігацію по асортименту товарів. У верхній частині сторінки розташоване головне меню з розділами, такими як «Чохли», «Скло», «Аксесуари», «Гаджети», «Аудіо», «Для Фото і Відео» та «Аксесуари для Ноутбуків». Кожен із цих розділів містить підкатегорії, що полегшує пошук потрібного продукту.

Однією з ключових особливостей «KIWI Store» є великий асортимент товарів, що включає чохли для iPhone, iPad та Apple Watch, захисне скло, бездротові зарядні пристрої, ремінці для Apple Watch, кабелі, павербанки, дитячі смарт-годинники, екшн-камери та аксесуари до них, портативні колонки, навушники, мікрофони, штативи та багато іншого. Це дозволяє клієнтам знайти всі необхідні пристрої та аксесуари в одному місці.

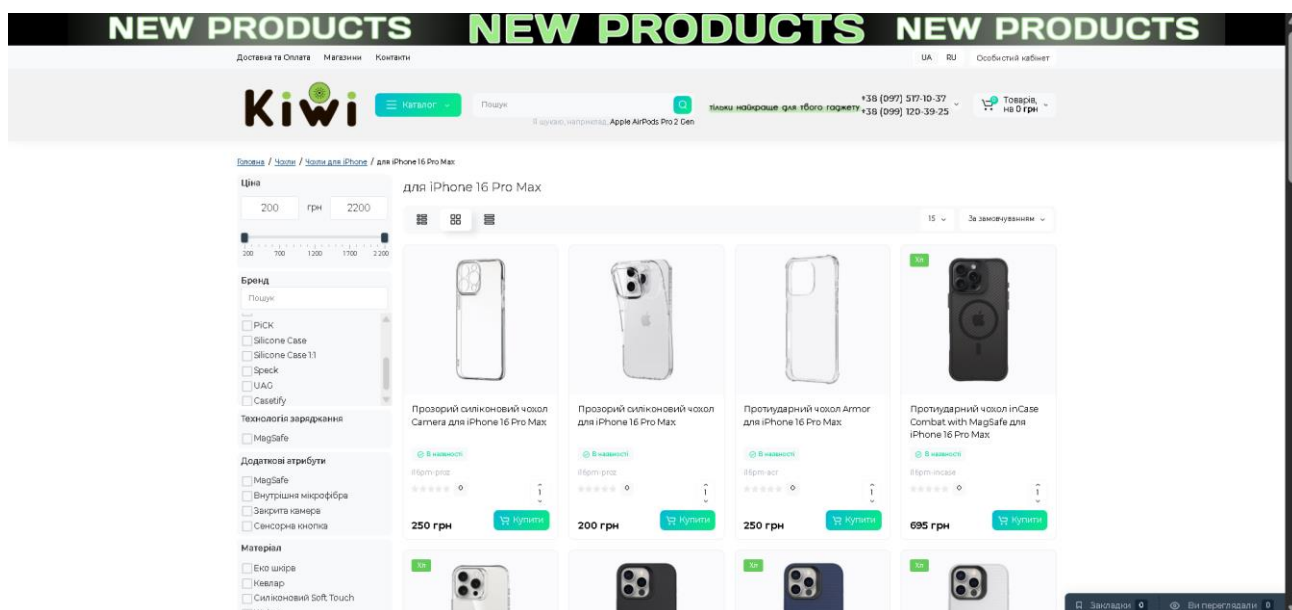


Рисунок 1.2 – Знімок екрану головної сторінки інтернет-магазину  
«KIWI Store» (джерело: [2])

Однією з ключових особливостей «KIWI Store» є великий асортимент товарів, що включає чохла для iPhone, iPad та Apple Watch, захисне скло, бездротові зарядні пристрої, ремінці для Apple Watch, кабелі, павербанки, дитячі смарт-годинники, екшн-камери та аксесуари до них, портативні колонки, навушники, мікрофони, штативи та багато іншого. Це дозволяє клієнтам знайти всі необхідні пристрої та аксесуари в одному місці.

Кожен продукт на сайті супроводжується детальним описом, що включає інформацію про виробника, технічні характеристики та рекомендації щодо використання. До товарів прикріплені якісні фотографії, що допомагає покупцям краще уявити обраний продукт.

«KIWI Store» підтримує кілька способів оплати, включаючи оплату банківською картою та готівкою при отриманні товару. Доставка здійснюється по всій території України за допомогою кур'єрських служб, що забезпечує зручність для клієнтів.

Для зручності покупців на сайті реалізована система відгуків і рейтингів товарів, що допомагає новим користувачам приймати обґрунтовані рішення під час

вибору продукції. Команда підтримки готова допомогти у вирішенні будь-яких питань, забезпечуючи високий рівень обслуговування клієнтів.

Загалом, інтернет-магазин «KIWI Store» пропонує користувачам зручний інтерфейс, широкий асортимент товарів і якісне обслуговування, що робить процес купівлі аксесуарів і гаджетів приємним та ефективним.

### 1.2.3 Інтернет-магазин «Gadget-Shop».

«Gadget-Shop» – це інтернет-магазин, що спеціалізується на продажу сучасних гаджетів та аксесуарів в Україні. Сайт має стильний, сучасний дизайн, який поєднує яскраві банери та спеціальні пропозиції, що одразу привертають увагу відвідувачів і мотивують до покупки (рис. 1.3) [3].

Архітектура сайту продумана таким чином, щоб забезпечити зручну навігацію навіть для тих, хто не має великого досвіду в онлайн-шопінгу. У верхній частині сторінки розташоване головне меню, яке включає категорії: «Аксесуари для гаджетів», «Товари для дому», «Дитячі товари», «Автотовари», «Спорт і захоплення», «Краса, здоров'я, догляд», «Гаджети для бізнесу», «Фото-відео-аудіо», «Гаджети для тварин» і «Інструменти й обладнання». Кожна категорія містить підкатегорії, що дозволяє легко знайти потрібний товар.

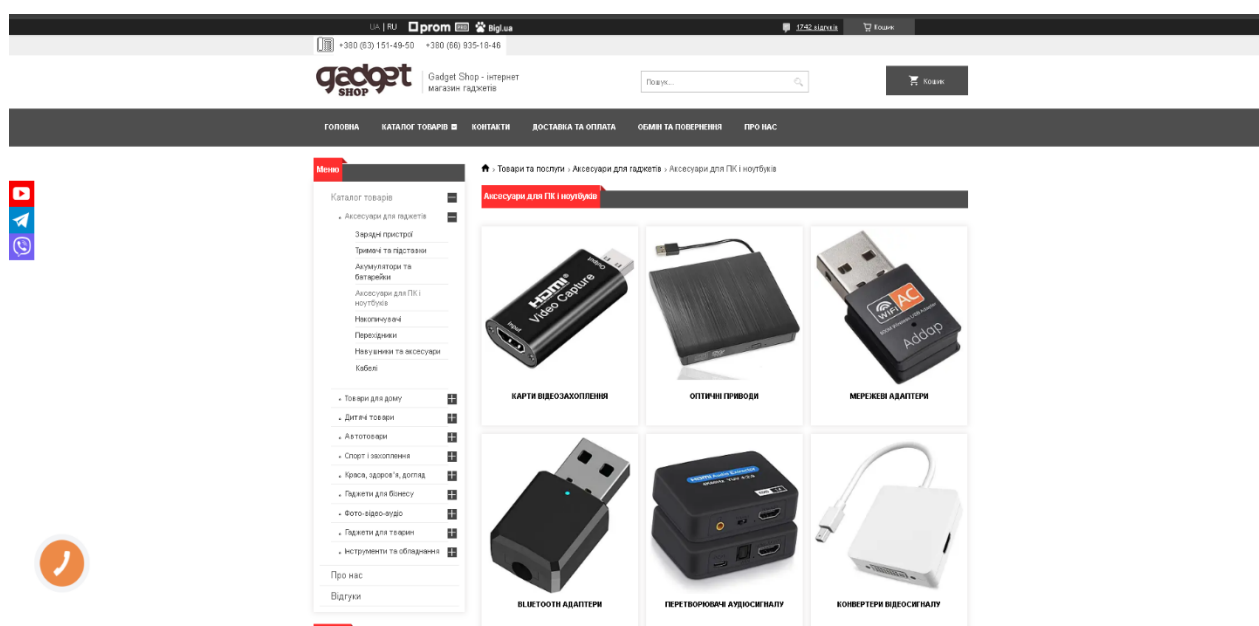


Рисунок 1.3 – Знімок екрану головної сторінки інтернет-магазину «Gadget-Shop» (джерело: [3])

Функціонал сайту включає в себе розширену пошукову систему і зручні фільтри, які допомагають відвідувачам швидко сортувати товари за ціною, популярністю, новизною та іншими характеристиками.

Товарні сторінки містять детальні описи продукції: від технічних характеристик до рекомендацій щодо використання. До кожного товару додаються якісні зображення, що дозволяють роздивитися продукт з різних ракурсів. У деяких випадках додаються відеоогляди, що допомагають покупцям краще зрозуміти функціонал гаджета перед покупкою.

Система відгуків дозволяє клієнтам залишати свої коментарі про продукцію, допомагаючи майбутнім покупцям робити більш зважений вибір.

Блог – це ще один важливий елемент сайту. Тут публікуються огляди новинок техніки, поради з використання гаджетів, порівняння пристроїв і інші корисні матеріали. Блог не лише надає корисну інформацію, а й сприяє додатковому залученню відвідувачів на сайт.

Оплата і доставка представлені кількома зручними варіантами. Покупці можуть розрахуватися банківською карткою або готівкою при отриманні. Доставка здійснюється по всій Україні за допомогою провідних кур'єрських служб. Товари ретельно упаковуються, щоб уникнути пошкоджень під час транспортування.

Клієнтський сервіс є однією з сильних сторін Gadget-Shop. Служба підтримки готова допомогти клієнтам у вирішенні будь-яких питань, забезпечуючи швидкий і професійний зворотний зв'язок через онлайн-чат, телефон або електронну пошту.

Для постійних клієнтів діє система знижок і бонусів, що стимулює повторні покупки та формує лояльність до магазину.

Загалом, «Gadget-Shop» – це сучасний, зручний і функціональний інтернет-магазин, який пропонує широкий вибір гаджетів і аксесуарів, поєднуючи якісний сервіс, зручний інтерфейс і приємний шопінг.

### 1.3 Порівняльний аналіз сайтів з продажу гаджетів та аксесуарів

За результатами аналізу трьох інтернет-магазинів – «World of Gadgets», «KIWI Store» та «Gadget-Shop» – була створена порівняльна таблиця їхнього функціоналу (табл. 1.1).

Таблиця 1.1 – Результат порівняння сайтів з продажу гаджетів та аксесуарів

Функціонал	World of Gadgets	KIWI Store	Gadget-Shop	Вебресурс, який розробляється
Кошик покупок із функцією збереження	Ні	Так	Так	Так
Самовивіз з локального магазину	Так	Так	Ні	Так
Система рейтингів та відгуків	Так	Так	Так	Так
Адаптивний дизайн для мобільних пристроїв	Ні	Так	Ні	Так
Можливість додати товари в кошик, якщо не авторизований	Ні	Так	Так	Так
Наявність навігації для фото у товарах	Так	Ні	Так	Так
Онлайн-консультації, гаряча лінія	Ні	Так	Так	Так

В результаті огляду сайтів-аналогів, їхніх переваг та недоліків, а також аналізу ключових елементів, були визначені основні вимоги до створення власного вебресурсу з продажу гаджетів та аксесуарів:

1) простий та зрозумілий дизайн – сайт має бути привабливим, мати чітку структуру та зручну навігацію, щоб навіть нові користувачі могли легко знайти потрібний товар;

2) адаптивність – сайт повинен коректно працювати на різних пристроях: смартфонах, планшетах, комп'ютерах;

3) каталог продукції – важливо створити структурований каталог із детальними описами, якісними зображеннями та технічними характеристиками;

4) функція пошуку та фільтрації – необхідно інтегрувати зручну систему пошуку та фільтрів (за ціною, популярністю, категорією, брендом тощо);

5) система замовлення – процес оформлення замовлення повинен бути інтуїтивним та підтримувати різні види оплат (після оплата при отриманні замовлення на пошті, готівка або картою при отриманні самовивозом);

6) система відгуків та рейтингів – можливість залишати відгуки підвищує довіру до магазину та допомагає новим покупцям робити зважений вибір. Контактна інформація та підтримка – наявність онлайн-чату, телефону, електронної пошти та швидкий зворотний зв'язок підвищує рівень обслуговування;

7) онлайн-консультації – можливість отримати швидку відповідь від фахівця підвищує якість обслуговування;

8) мобільний додаток – додаток для смартфонів дасть клієнтам ще більше зручності у виборі та покупці товарів.

Дотримання цих вимог дозволить створити зручний, сучасний та ефективний інтернет-магазин, який буде конкурентоспроможним на ринку гаджетів та аксесуарів.

#### **1.4 Вибір стеку технологій для розробки вебресурсу**

Для розробки вебресурсу буде використана класична архітектура: клієнтський додаток у вигляді вебсайту, серверний додаток та систему управління базою даних (СУБД).

Для вибору технологічного стеку потрібно обрати наступні технології:

- а) мова програмування для розробки клієнтського додатку;
- б) технології для розробки клієнтського додатку;
- в) мова програмування для розробки серверного додатку;
- г) технології для розробки серверного додатку;
- д) видір архітектурного шаблону
- д) СУБД для збереження даних.

### 1.4.1 Вибір CSS-фреймворку для стилізації клієнтського додатку.

Для реалізації клієнтської частини вебресурсу особливу увагу слід приділити вибору CSS-фреймворку, оскільки саме від нього значною мірою залежить візуальна привабливість, зручність використання інтерфейсу та швидкість розробки. Сучасні CSS-фреймворки надають широкі можливості для створення адаптивного та функціонального дизайну, а також значно полегшують підтримку коду в майбутньому. Серед найбільш відомих варто виділити три рішення: Bootstrap, Tailwind CSS та Bulma, кожне з яких має свої переваги та недоліки.

Bootstrap є одним із найдавніших і водночас найпопулярніших CSS-фреймворків. Він пропонує велику бібліотеку готових інтерфейсних компонентів, а також підтримку JavaScript-плагінів, що дозволяють реалізовувати динамічну поведінку елементів. Однією з головних переваг Bootstrap є вбудована адаптивність – фреймворк автоматично підлаштовує інтерфейс під розміри екрана. Крім того, він має широке ком'юніті, гарну документацію та підтримку різними плагінами і шаблонами, що значно пришвидшує розробку [4].

Tailwind CSS, на відміну від Bootstrap, дотримується утилітарного підходу. Він не нав'язує жодного готового дизайну, а замість цього пропонує велику кількість CSS-класів, які відповідають за окремі стилі (відступи, кольори, шрифти тощо). Це дозволяє створювати абсолютно унікальні інтерфейси, не обмежуючись заготовленими шаблонами. Tailwind вимагає трохи більше уваги до структури HTML-коду, але водночас відкриває широкі можливості для кастомізації [5].

Bulma – менш поширений, проте цікавий фреймворк, що базується на Flexbox-моделі. Його відрізняє простий синтаксис, зрозуміле компонування та легкість у засвоєнні. Bulma ідеально підходить для проєктів середньої складності, де потрібен швидкий старт без надлишкової складності [6].

З огляду на цілі проєкту – створення вебсайту для підтримки продажу гаджетів та аксесуарів – було прийнято рішення використовувати Bootstrap. Це обумовлено кількома факторами: по-перше, його широке розповсюдження забезпечує кращу підтримку спільнотою та документацією, що знижує поріг входу для нових розробників, які можуть долучитися до підтримки сайту в майбутньому.



По-друге, велика кількість готових компонентів дозволяє скоротити час на розробку базового функціоналу. І нарешті, перевірена адаптивна сітка Bootstrap робить інтерфейс зручним для користувачів на будь-яких пристроях – від смартфонів до десктопів.

#### **1.4.2 Вибір клієнтського фреймворку.**

У сучасній веброзробці фронтенд відіграє ключову роль у формуванні зручного, швидкого та інтерактивного інтерфейсу. Для цього застосовуються JavaScript-фреймворки, які дозволяють реалізовувати динамічну поведінку без постійного перезавантаження сторінки. Найбільш вживаними рішеннями є Vue.js, Angular та Alpine.js – кожен із них має свої переваги та найкраще підходить для певних типів проєктів.

Angular – потужний, модульний фреймворк від Google. Він надає повноцінний набір інструментів для побудови великих корпоративних застосунків: двостороннє зв'язування, вбудований роутер, DI, системи валідації, підтримка TypeScript. Angular чудово підходить для великих команд, проєктів зі складною логікою та суворими вимогами до структури коду. Проте його громіздкість, висока крива навчання та значна кількість шаблонного коду можуть ускладнити використання в невеликих або швидких проєктах [7].

Vue.js – прогресивний фреймворк, що надає баланс між простотою та потужністю. Його легкий синтаксис, реактивна система даних і можливість поступового впровадження роблять його ідеальним для середніх проєктів. Vue має гарну документацію, активну спільноту та багатий набір офіційних інструментів, що полегшує масштабування. Проте навіть Vue потребує певної структури, налаштування середовища та використання збирача, що трохи підвищує поріг входу [8].

Alpine.js – мікрофреймворк, натхненний філософією Vue, але реалізований у максимально простий і легкий спосіб. Його головна ідея – забезпечити базову реактивність і управління поведінкою інтерфейсу без складної інфраструктури. Alpine не потребує збірки, дозволяє писати логіку прямо в HTML, важить менше 10 Кбайт і блискавично працює. Це ідеальний інструмент для швидкого створення

інтерактивних елементів, особливо в поєднанні з серверною генерацією сторінок [9].

Таблиця 1.2 – Результат порівняння клієнтських фреймворків Java Script

Критерій	Angular	Vue.js	Alpine.js
Складність навчання	Висока	Середня	Низька
Розмір фреймворку	Великий	Середній	Дуже малий
Швидкість старту проекту	Повільна	Помірна	Швидка
Потужність і масштабованість	Висока	Висока	Середня
Простота інтеграції з Django	Помірна	Висока	Дуже висока
Потреба в збірці (webpack тощо)	Так	Так	Ні
Ідеальне застосування	Корпоративні SPA	Середні/масштабовані SPA	Легкі, швидкі інтерфейси

З огляду на характер даного проекту – розробка вебресурсу для підтримки продажу гаджетів та аксесуарів – найважливішими критеріями стали швидкість розробки, простота підтримки та можливість легкого розгортання. Саме за цими параметрами Alpine.js виявився найоптимальнішим вибором.

Таким чином, хоча Angular перемагає у сфері складних бізнес-застосунків, а Vue.js є прекрасним вибором для більшості проектів середньої складності, у випадку цього проекту найбільш вигідним за всіма ключовими параметрами став саме Alpine.js – легкий, ефективний і практичний інструмент для сучасного вебу.

#### **1.4.3 Вибір серверного фреймворку.**

При розробці серверної частини вебресурсу важливо обрати такий фреймворк, який відповідатиме вимогам проекту з точки зору функціональності, безпеки, швидкості розробки та подальшого супроводу. Ринок сучасної

веброзробки пропонує багато рішень, серед яких особливо виділяються Django, Express.js та Spring Boot – кожен із них має свої сильні сторони та специфіку застосування.

Django – це високорівневий фреймворк для мови Python, який побудований за принципом «все в комплекті» (batteries included). Він пропонує цілу екосистему вбудованих інструментів: ORM (об'єктно-реляційне відображення), систему маршрутизації, механізми автентифікації, панель адміністратора, засоби захисту від поширених вебзагроз (XSS, CSRF, SQL-ін'єкцій) та багато іншого. Django активно використовується для створення як невеликих сайтів, так і масштабних вебдодатків, завдяки чому має широку спільноту та стабільну документацію. Його структуру можна назвати «досить суворою», що, з одного боку, обмежує надмірну гнучкість, а з іншого – спрощує підтримку і масштабування проєктів [10].

Express.js – популярний фреймворк для побудови серверів на платформі Node.js. Він дотримується мінімалістичного підходу: базова установка містить лише найнеобхідніше, а розширення можливостей відбувається через підключення зовнішніх модулів. Завдяки цьому Express дуже гнучкий і дозволяє тонко налаштовувати архітектуру додатка. Фреймворк ідеально підходить для створення REST API, мікросервісів і легких серверних застосунків. Однак, у порівнянні з Django, він не має такої кількості вбудованих функцій, тому розробник повинен самостійно обирати і налаштовувати додаткові інструменти, включно з системою безпеки та ORM [11].

Spring Boot – це фреймворк для Java, орієнтований на розробку корпоративних додатків. Він автоматизує більшість процесів конфігурації та дозволяє швидко створювати масштабовані і стабільні серверні системи. Завдяки потужному набору бібліотек і підтримці великої кількості інтеграцій, Spring Boot часто застосовується в банківських, фінансових та e-commerce системах, де важлива надійність, продуктивність і безпека. Водночас, поріг входу в Spring Boot вищий, ніж у Django чи Express, що може ускладнити його використання у відносно простих проєктах [12].

Після аналізу переваг і недоліків кожного з фреймворків, для реалізації серверної частини проєкту було обрано Django. Це рішення обумовлене кількома чинниками:

- 1) Django дозволяє швидко розгорнути повноцінний вебдодаток із мінімальними зусиллями, завдяки великій кількості готових рішень;
- 2) високий рівень безпеки за замовчуванням особливо важливий у контексті роботи з персональними даними користувачів та онлайн-замовленнями;
- 3) використання Python спрощує інтеграцію з іншими інструментами для аналітики, обробки даних чи автоматизації;
- 4) наявність адміністративної панелі значно полегшує управління вмістом сайту без потреби створювати її вручну.

Таким чином, Django став найбільш оптимальним вибором для проєкту, орієнтованого на онлайн-продаж гаджетів та аксесуарів, де важливі швидкість розробки, зручність підтримки і надійність.

#### **1.4.4 Вибір СУБД для реалізації вебресурсу.**

У розробці вебресурсів, особливо тих, що мають справу з великим обсягом даних, критично важливо обрати надійну систему управління базами даних. Вона повинна не лише ефективно зберігати й обробляти інформацію, але й забезпечувати стабільну роботу при зростанні навантаження, підтримувати транзакції, гнучке масштабування та інтеграцію з обраним серверним фреймворком.

Серед найпопулярніших реляційних СУБД можна виділити MySQL, PostgreSQL та SQLite – кожна з них має свої переваги і рекомендовані сценарії використання.

MySQL – це широко розповсюджена система з відкритим кодом, яка історично зарекомендувала себе як надійна основа для багатьох вебплатформ. Вона забезпечує високу швидкість виконання запитів, підтримку транзакцій, індексування та реплікацію даних. MySQL часто використовується в поєднанні з PHP або CMS (наприклад, WordPress), а також підтримується багатьма хостингами «з коробки». Проте в деяких випадках їй бракує гнучкості при роботі зі складними типами даних і розширеною логікою запитів [13].

PostgreSQL – це потужна об'єктно-реляційна СУБД, яка відзначається широкими функціональними можливостями. Вона підтримує складні SQL-запити, транзакції з багаторівневою ізоляцією, розширені типи даних, тригери, збережені процедури та інші інструменти для реалізації бізнес-логіки на рівні бази. PostgreSQL має високу надійність, розширювану архітектуру, активну спільноту і добре інтегрується з фреймворком Django, де вона часто використовується за замовчуванням [14].

SQLite – це вбудована в Django легка база даних, яка не потребує окремого сервера для роботи. Її зручно використовувати у невеликих додатках, мобільних застосунках або на етапі прототипування. Проте через обмеженість у функціоналі вона не підходить для масштабованих вебпроектів з великою кількістю одночасних запитів і складною структурою даних [15].

З огляду на вимоги до надійності, масштабованості та тісної інтеграції з Django, у рамках даного проекту було обрано PostgreSQL як основну СУБД. Вона забезпечує стабільну роботу при високому навантаженні, гнучкі можливості для аналітики та безпечне зберігання даних. Крім того, її активна підтримка в Django дозволяє з мінімальними зусиллями реалізувати повноцінний бекенд.

#### **1.4.5 Вибір системи контролю версій для розробки вебресурсу.**

У процесі розробки сучасних вебдодатків надзвичайно важливою є наявність ефективної системи контролю версій (СКВ), яка забезпечує збереження історії змін у коді, можливість паралельної роботи кількох розробників, контроль якості та підтримку безперервної інтеграції. Вибір правильної СКВ відіграє ключову роль у забезпеченні стабільності, продуктивності команди та керованості всього проекту. Найбільш популярні сучасні системи базуються на Git – розподіленій системі контролю версій, що дозволяє розробникам локально зберігати історію змін, а пізніше синхронізувати її з віддаленим сервером. Git підтримується багатьма платформами, серед яких варто розглянути Azure DevOps, GitHub, GitLab. Оцінімо кожен з них у контексті потреб проекту.

Azure DevOps – рішення від Microsoft, що об'єднує інструменти для управління життєвим циклом програмного забезпечення: контроль версій,

планування задач, автоматичне тестування, CI/CD, аналітику та реліз-менеджмент. Платформа добре інтегрується з іншими сервісами Microsoft (наприклад, Azure, Visual Studio), що робить її вигідним вибором для великих компаній, орієнтованих на екосистему Microsoft. Для невеликих команд чи проєктів вона може бути надмірно складною у налаштуванні [16].

GitHub – це провідна хмарна платформа для хостингу Git-репозиторіїв, яка широко використовується як в індивідуальній, так і в командній розробці. Однією з головних переваг GitHub є інтуїтивно зрозумілий вебінтерфейс, а також багатий набір інструментів: система pull-запитів, оглядів коду, автоматичних перевірок (linting, тестування), інтеграція з CI/CD-сервісами (наприклад, GitHub Actions), візуалізація змін, підтримка Wiki та Issue Tracking. Величезна спільнота та велика кількість публічних репозиторіїв роблять GitHub зручним середовищем для відкритої розробки, а також для пошуку прикладів та рішень [17].

GitLab – потужна DevOps-платформа, яка, крім стандартного функціоналу Git, пропонує повністю інтегроване середовище для безперервної інтеграції та доставки (CI/CD), управління проєктами, безпеки, моніторингу та управління релізами. Однією з ключових переваг GitLab є можливість самостійного розгортання інстансу на власному сервері – це важливо для організацій із підвищеними вимогами до конфіденційності або обмеженим доступом до зовнішніх сервісів. GitLab також підтримує імпорт із GitHub, що полегшує міграцію [18].

GitHub забезпечує усі необхідні можливості для ефективної командної розробки, стабільного управління версіями коду та інтеграції з іншими інструментами, що і визначило його як оптимальний вибір для даного вебпроєкту.

#### **1.4.6 Використання патернів проєктування у розробці вебресурсу.**

На етапі проєктування архітектури вебресурсу важливо обрати відповідні патерни проєктування, які забезпечать зрозумілу, гнучку та підтримувану структуру додатку. Застосування перевірених архітектурних підходів дозволяє зменшити зв'язність коду, полегшити його тестування та розширення, а також підвищити надійність системи. У межах розробки вебресурсу для продажу гаджетів

та аксесуарів, де ми плануємо використовувати Django як серверний фреймворк, доцільно розглянути найбільш популярні патерни: Model-View-Template (MVT), Decorator та Observer.

Model-View-Template (MVT) – це архітектурний патерн, який лежить в основі фреймворку Django і забезпечує чітке розділення відповідальностей між різними компонентами системи. Модель (Model) відповідає за зберігання та обробку даних, представлення (View) – за логіку обробки запитів, а шаблон (Template) – за виведення інформації користувачу. Такий підхід дозволяє створювати масштабовані та підтримувані системи, де логіка бізнес-процесів, взаємодії з базою даних та відображення інтерфейсу не перетинаються між собою [19].

Decorator – це структурний патерн, що дозволяє розширювати функціональність окремих елементів системи без зміни їхнього початкового коду. У Django цей підхід реалізується за допомогою функціональних декораторів, які накладаються на представлення (views). Наприклад, такі декоратори, як `@login_required` або `@permission_required`, забезпечують перевірку прав доступу до певних ресурсів. Такий механізм дозволяє гнучко управляти авторизацією та розширювати функціональність без дублювання коду, що особливо актуально для адміністративної частини вебресурсу [20].

Observer – це поведінковий патерн, який реалізується у Django через механізм сигналів (signals). Він дозволяє реалізувати реакцію на події в системі, такі як збереження або видалення об'єктів моделі, без необхідності вносити зміни у логіку представлень чи моделей. Наприклад, сигнал `post_save` може бути використаний для автоматичної відправки повідомлення користувачу після створення замовлення або для оновлення пов'язаних записів. Такий підхід дозволяє зменшити зв'язність між компонентами і підвищити гнучкість архітектури [21].

Таким чином, використання патерну Model-View-Template дозволяє побудувати архітектурно правильну систему з чітким розподілом обов'язків, високим рівнем гнучкості та підтримуваності. Ця інтеграція в рамках обраного технологічного стека на основі Django забезпечить стабільність, логічну структуру та можливість подальшого розширення вебресурсу.

## Висновки до розділу 1

У першому розділі даної кваліфікаційної роботи було здійснено аналіз сучасного стану ринку онлайн-продажів гаджетів та аксесуарів, досліджено особливості вже існуючих вебресурсів цієї тематики та сформовано перелік ключових функціональних і нефункціональних вимог до майбутньої системи.

З метою кращого розуміння потреб користувачів було проведено порівняльний огляд кількох інтернет-магазинів, що дозволило виявити як ефективні рішення в побудові інтерфейсу та структури сайту, так і типові недоліки, яких варто уникнути під час розробки власного вебресурсу. На основі цього було сформульовано перелік вимог до проєкту, зокрема: адаптивність інтерфейсу, підтримка системи рейтингу, гнучка система фільтрації, тощо.

В результаті технічного обґрунтування було обрано стек технологій, що найкраще відповідає потребам проєкту. Серверну частину буде реалізовано з використанням фреймворку Django, який забезпечує високий рівень безпеки та наявність вбудованих інструментів для управління контентом і користувачами.

Для клієнтської частини обрано Alpine.js, що дозволяє створити легкий, інтерактивний інтерфейс без перевантаження логіки.

У якості CSS-фреймворку використовується Bootstrap, що забезпечує адаптивність, зручність компонування елементів та швидке стилізування сторінок.

База даних реалізована за допомогою PostgreSQL, що дозволяє працювати зі складними запитами та гарантує надійність зберігання даних.

Для контролю версій використано платформу GitHub, яка сприяє структурованій командній розробці та збереженню історії змін.

Також для організації вихідного коду була обрана система контролю версій GitHub, у якості основних патернів проєктування для розробки вебресурсу було обрано патерн Model-View-Template (MVT), який є стандартом у Django.



## **2 ПРОЄКТУВАННЯ ВЕБРЕСУРСУ З ПІДТРИМКИ ПРОДАЖУ ГАДЖЕТІВ ТА АКСЕСУАРІВ**

### **2.1 Мета та основні функції вебресурсу**

Метою створення даного вебресурсу є розробка зручної, функціональної та безпечної онлайн-платформи, що дозволяє реалізовувати процес продажу гаджетів та аксесуарів через інтернет. Вебресурс має забезпечити комфортну взаємодію між кінцевим користувачем, який шукає і замовляє продукцію, та адміністративною частиною системи, яка керує контентом, обробляє замовлення та супроводжує продаж.

Серед основних функцій вебресурсу варто виділити:

- 1) формування структурованого каталогу товарів з можливістю фільтрації та пошуку;
- 2) підтримку реєстрації, авторизації та персонального кабінету користувача;
- 3) реалізацію кошика з можливістю додавання, видалення товарів і оформлення замовлення;
- 4) обробку замовлень з урахуванням способу доставки та адреси доставки;
- 5) інтеграцію з логістичними службами для списку актуальних відділень і замовленням користувачем товару у зручне для себе відділення;
- 6) можливість залишати відгуки та оцінки до товарів;
- 7) управління товарними позиціями, описами та медіаконтентом з боку адміністратора;
- 8) реалізацію базових механізмів захисту персональних даних та обробки платіжної інформації.

### **2.2 Потоки даних вебресурсу для продажу гаджетів та аксесуарів**

Потоки даних представляють собою цілеспрямований рух інформації між різними частинами системи, що забезпечує її повноцінне функціонування. Вони

охоплюють як зовнішню взаємодію між користувачем і вебресурсом, так і внутрішню передачу даних між логічними модулями системи, зокрема між клієнтською частиною, сервером і базою даних. Правильна організація інформаційних потоків є ключовою умовою стабільної роботи ресурсу, зниження затримок та забезпечення коректної обробки запитів.

Діаграма потоків даних вебресурсу для продажу гаджетів та аксесуарів наведена на рис. 2.1.

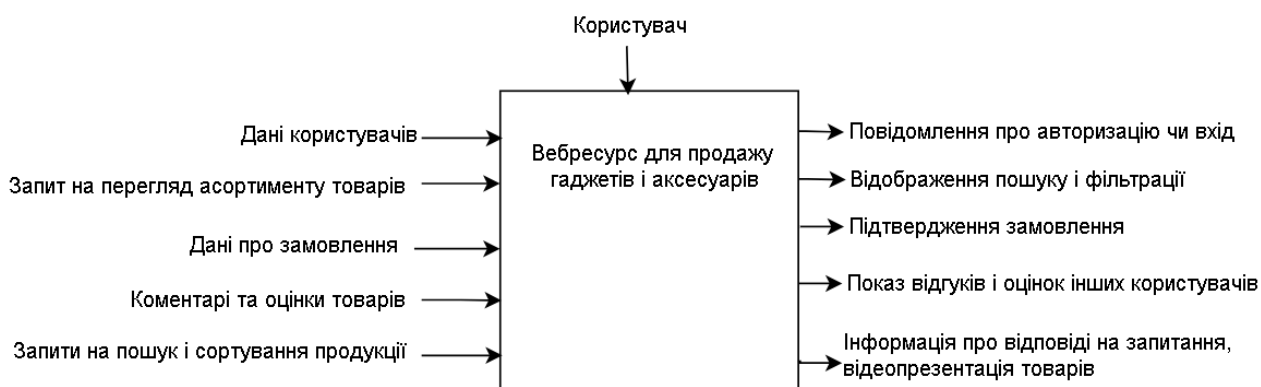


Рисунок 2.1 – Діаграма інформаційних потоків вебресурсу

У процесі аналізу функціональної структури майбутнього вебдодатку було ідентифіковано основні вхідні інформаційні потоки, які формуються на основі дій користувачів:

1) дані реєстрації та авторизації користувачів – на етапі створення облікового запису або входу в систему користувач заповнює форму із зазначенням особистих даних (електронна адреса, пароль). Ця інформація передається на сервер, де здійснюється перевірка її коректності, у випадку успішної обробки – зберігається у базі даних з дотриманням вимог безпеки;

2) запити на перегляд каталогу товарів – при зверненні до розділу з асортиментом система ініціює звернення до бази даних для отримання актуального переліку доступних товарів. Дані проходять фільтрацію та форматування для подальшого відображення у клієнтській частині з урахуванням адаптивного дизайну;

3) інформація про замовлення – під час оформлення покупки формується комплексний запит, що містить відомості про обрані товари, контактну інформацію, адресу та спосіб доставки. Отримані дані обробляються сервером, зберігаються у відповідних таблицях бази даних та надалі використовуються для генерації звітів, оновлення залишків і відстеження замовлень;

4) відгуки та рейтинги – після здійснення покупки користувач має змогу залишити текстовий коментар та оцінку товару. Ці дані надсилаються на сервер, проходять базову перевірку (на наявність заборонених символів або спаму), після чого зберігаються в базі даних і відображаються в інтерфейсі користувачів;

5) запити на пошук і сортування товарів – під час використання пошукового рядка.

Таким чином, всі інформаційні потоки в системі тісно взаємопов'язані і забезпечують злагоджену роботу вебресурсу. Їх правильна побудова є основою для реалізації логіки вебресурсу для підтримки продажу гаджетів і аксесуарів, адаптації інтерфейсу під користувача та забезпечення безперервного обміну даними між клієнтом і сервером.

Основні вихідні потоки системи формуються як відповідь на дії користувача та результат обробки запитів сервером. Вони охоплюють усі етапи взаємодії з вебресурсом – від підтвердження авторизації до завершення покупки та перегляду додаткового інформаційного контенту. Правильна організація вихідних потоків є критично важливою для забезпечення зручного користувацького досвіду, оперативного інформування та цілісного функціонування платформи.

Повідомлення про успішну реєстрацію або вхід – після обробки даних, введених користувачем під час реєстрації або авторизації, система повертає відповідь про успішне створення акаунту або підтвердження входу. Це може бути як текстове повідомлення, так і автоматичне перенаправлення до персоналізованого кабінету користувача.

Відображення результатів пошуку та фільтрації – у відповідь на пошуковий запит або зміну параметрів фільтрації (наприклад, за ціною, рейтингом чи категорією) сервер надсилає відсортований список товарів, який відповідає

зазначеним критеріям. Система також може відображати повідомлення у випадку відсутності результатів за заданими параметрами.

Інформація про вміст кошика – після додавання товарів до кошика користувач має змогу переглянути поточний його вміст. У відповідь на запит сервер генерує перелік обраних товарів із зазначенням кількості, вартості, загальної суми та можливості змінити вміст (видалення, зміна кількості або повернення до перегляду). Вміст кошика зберігається в сесії або прив'язується до облікового запису для забезпечення безперервності при зміні сторінок або повторному вході.

Підтвердження замовлення – після завершення оформлення покупки система формує відповідь, яка включає деталі замовлення: найменування товарів, підсумкову суму, спосіб оплати, адресу доставки та унікальний номер замовлення. Ці дані також можуть дублюватися на електронну пошту користувача або зберігатися в особистому кабінеті.

Показ відгуків та оцінок інших користувачів – під час перегляду сторінки конкретного товару система завантажує та відображає коментарі, залишені іншими покупцями, а також середній рейтинг на основі усіх оцінок. Це дозволяє новим користувачам формувати обґрунтовану думку про товар перед прийняттям рішення про покупку.

Інформація про контактні дані, відповіді на часті запитання та демонстраційні матеріали – у відповідь на звернення до відповідних розділів сайту сервер повертає контент, який містить координати служби підтримки, відповіді на поширені питання, відеоогляди, інструкції з використання продукції та інші допоміжні матеріали, що сприяють кращому ознайомленню з функціоналом вебресурсу.

Такий підхід до формування вихідних потоків даних гарантує цілісність користувацького досвіду, швидкий зворотний зв'язок на дії користувача та ефективне відображення динамічного контенту. У сукупності це сприяє підвищенню задоволеності користувачів, зростанню конверсій та загальній стабільності роботи системи.

## 2.3 Проєктування діаграми прецедентів онлайн-сервісу

На етапі проєктування інформаційної системи одним із важливих завдань є візуалізація функціональних можливостей платформи та взаємодії між користувачами і системою. Для цього застосовується діаграма варіантів використання (Use Case Diagram), що належить до нотації UML і дозволяє чітко визначити ролі (акторів) та функціональні сценарії їхньої взаємодії з вебресурсом. Такий підхід дозволяє систематизувати вимоги до платформи, виявити основні потоки дій та оптимізувати подальший процес реалізації.

У межах даної системи було виокремлено ключових акторів, які відіграють різні ролі у процесі функціонування вебплатформи:

а) гість – користувач, який відвідує вебресурс без авторизації. Гість має обмежений набір функцій: перегляд каталогу товарів, пошук продукції за заданими параметрами, читання описів та відгуків, і навіть додавання товарів до кошику. При цьому функціональність оформлення замовлення, доступ до особистого кабінету та залишення коментарів для нього недоступні;

б) користувач – зареєстрований відвідувач сайту, який має доступ до повного функціоналу. Він може додавати товари до кошика, оформлювати замовлення з вибором способу оплати та доставки, залишати відгуки якщо купував товар, ставити оцінки і змінювати особисті дані в акаунті. Це основна роль у системі, яка визначає ключові потоки взаємодії з платформою;

в) адміністратор – користувач із розширеними правами доступу, відповідальний за управління контентом ресурсу. До його обов'язків належить додавання нових товарів, редагування описів і цін, модерація відгуків, створення акцій а також контроль за виконанням замовлень. Адміністратор також може створювати або блокувати облікові записи співробітників;

г) API Нової Пошти – це зовнішній програмний інтерфейс, що використовується для автоматизованої взаємодії вебресурсу з логістичною системою перевізника. Основним призначенням цього API є забезпечення можливості інтегрованого доступу до функціоналу доставки: отримання списку

відділень, створення накладних, розрахунків вартості доставки, відстеження відправлень тощо.

Діаграма варіантів використання для вебресурсу з продажу гаджетів та аксесуарів представлена на рис. 2.2.

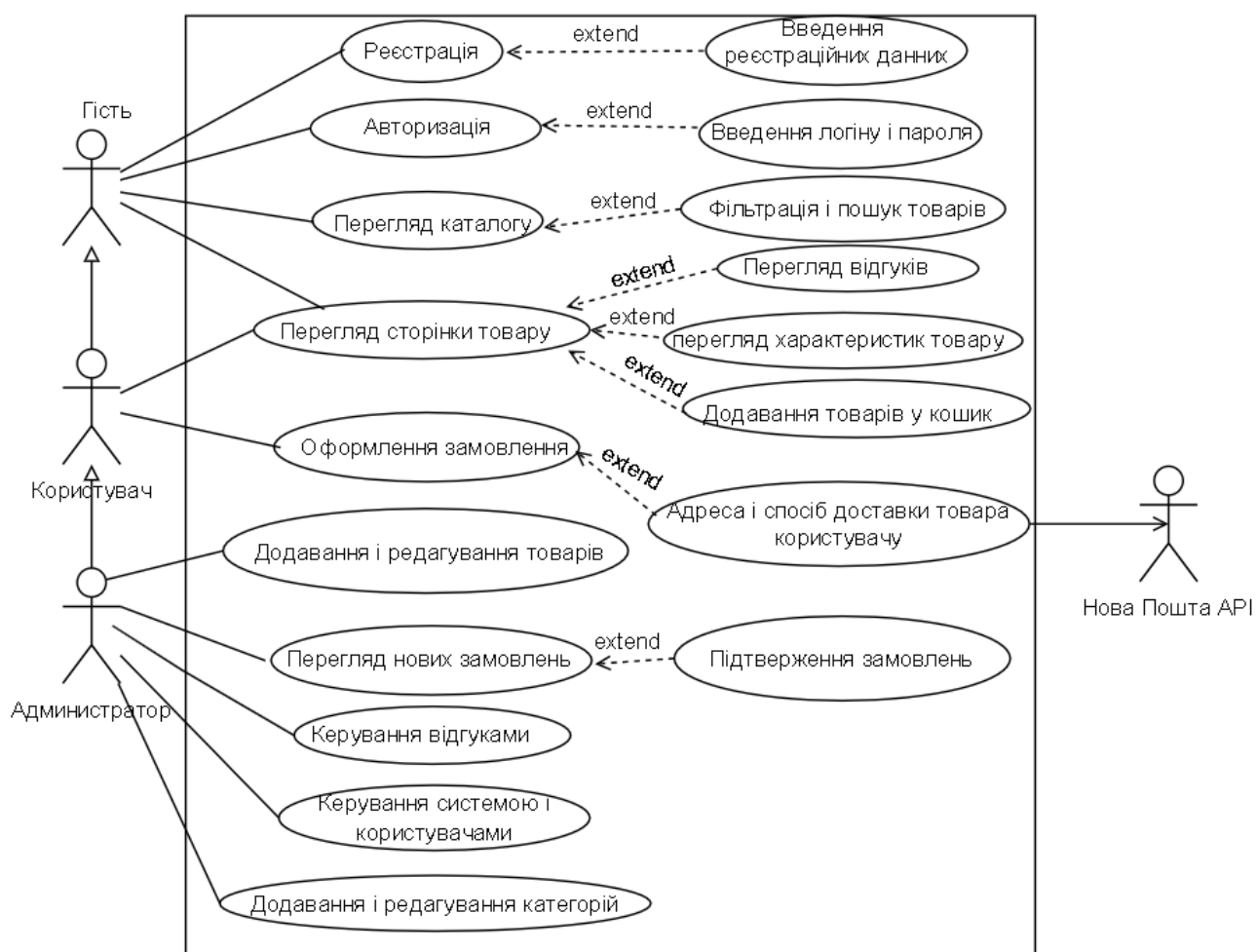


Рисунок 2.2 – Діаграма варіантів використання вебресурсу

Загалом, правильне визначення акторів та їхніх варіантів використання є критичним для побудови стійкої та логічно обґрунтованої архітектури системи. Це дозволяє уникнути плутанини під час розробки функціоналу, спрощує подальше тестування та гарантує, що всі ключові сценарії взаємодії з вебресурсом будуть належно реалізовані.

## 2.4 Формування функціональних вимог до вебресурсу

Після побудови UML-діаграми варіантів використання наступним кроком є деталізація сценаріїв взаємодії користувачів із функціоналом платформи. Це дозволяє більш точно окреслити технічні вимоги до системи та сформулювати основу для реалізації логіки вебресурсу. Нижче подано опис основних функціональних вимог (англ. Functional Requirement, FR), які визначають ключові сценарії роботи користувачів із платформою.

FR1. Автентифікація. Цей сценарій описує процес входу зареєстрованого користувача до системи.

FR1.1. Користувач відкриває сторінку авторизації.

FR1.2. Вводить електронну пошту та пароль.

FR1.3. Натискає на кнопку входу.

FR1.4. Система перевіряє достовірність вказаних даних у базі.

FR1.5. У разі успіху – здійснюється вхід до акаунту, у разі помилки – виводиться відповідне повідомлення.

FR2. Реєстрація. Сценарій охоплює процедуру створення нового облікового запису.

FR2.1. Користувач переходить на сторінку реєстрації.

FR2.2. Заповнює форму із зазначенням персональних даних (email, пароль, підтвердження паролю).

FR2.3. Погоджується з умовами користування сервісом.

FR2.4. Натискає кнопку реєстрації.

FR2.5. Система перевіряє валідність даних та створює нового користувача в базі.

FR2.6. Користувач отримує повідомлення про успішну реєстрацію та доступ до авторизації.

FR3. Перегляд каталогу. Сценарій відображає базову взаємодію з каталогом товарів.

FR3.1. Користувач переходить до розділу каталогу.

FR3.2. Система виводить перелік товарів із назвою і цінами.

FR3.3. За необхідності користувач застосовує фільтри (категорія, діапазон цін).

FR3.4. Користувач має змогу перейти до сторінки конкретного товару.

FR4. Детальний перегляд товару. Описує процес ознайомлення з конкретною позицією в каталозі.

FR4.1. Користувач обирає товар зі списку.

FR4.2. Система відображає сторінку з повною інформацією (опис, технічні характеристики, фото, ціна, наявність на складі).

FR4.3. Користувач має можливість додати товар до кошика.

FR5. Додавання товару до кошика. Реалізує початковий етап процесу покупки. FR5.1. Користувач перебуває на сторінці товару.

FR5.2. Натискає кнопку додавання до кошика.

FR5.3. Система фіксує обраний товар у поточному кошику та повідомляє про успішне додавання.

FR6. Перегляд кошика. Цей сценарій описує дії користувача з поточним вмістом кошика.

FR6.1. Користувач відкриває кошик через відповідну іконку або кнопку.

FR6.2. Система виводить список обраних товарів із зазначенням кількості та підсумкової вартості.

FR6.3. Користувач може редагувати кількість товарів або видалити окремі позиції.

FR6.4. У разі готовності – користувач переходить до оформлення замовлення.

FR7. Оформлення замовлення. Визначає повний цикл створення замовлення.

FR7.1. Користувач натискає кнопку оформлення з вікна кошика.

FR7.2. Заповнює форму з контактними та адресними даними.

FR7.3. Обирає спосіб доставки з доступних варіантів.

FR7.4. Перевіряє зведену інформацію про замовлення.

FR7.5. Підтверджує оформлення замовлення.



FR7.6. Система обробляє замовлення, створює запис у базі даних і повертає підтвердження.

FR8. Написання відгуку та перегляд відгуків про товар. Цей сценарій описує процес взаємодії користувача з модулем відгуків.

FR8.1. Користувач відкриває сторінку обраного товару.

FR8.2. Активує вкладку або секцію, що містить користувацькі відгуки.

FR8.3. Система відображає вже наявні відгуки з коментарями та рейтингами.

FR8.4. Користувач, за наявності авторизації і якщо він купував цей товар, може додати власний відгук, вказавши текстове повідомлення та оцінку у вигляді кількох зірок або шкали.

FR8.5. Система здійснює перевірку введеної інформації, зберігає її у базі даних і додає новий відгук до загального списку.

FR9. Редагування особистої інформації Сценарій охоплює зміну персональних даних у профілі користувача.

FR9.1. Користувач проходить авторизацію в особистому акаунті.

FR9.2. Переходить до розділу профілю, доступного через основне меню або іконку акаунту.

FR9.3. Вносить зміни у необхідні поля – ім'я, прізвище, адресу доставки, номер телефону або інші атрибути.

FR9.4. Активує кнопку збереження змін.

FR9.5. Система оновлює дані в базі користувачів та підтверджує успішне збереження.

## **2.5 Нефункціональні вимоги до вебресурсу магазину гаджетів та аксесуарів**

Нефункціональні вимоги (англ. Non-Functional Requirement, NFR) визначають загальні характеристики системи, що не стосуються безпосередньо її функціональності, але мають критичне значення для користувацького досвіду,

технічної стійкості та безпеки. Вони охоплюють продуктивність, доступність, сумісність, захист даних та можливість масштабування.

### **2.5.1 Продуктивність і швидкодія.**

NFR1. Вебплатформа повинна завантажуватись у браузері користувача не довше ніж за 5 с, за умов середньої швидкості з'єднання.

NFR2. Середній час відповіді сервера на будь-який запит не має перевищувати 500 мс у стандартних умовах.

NFR3. Загальний обсяг сторінки при завантаженні (включаючи зображення, шрифти, CSS і JavaScript) не повинен перевищувати 3 Мбайти, щоб уникнути перевантаження клієнтського інтерфейсу.

### **2.5.2 Доступність та надійність.**

NFR4. Система повинна бути доступною щонайменше 90% часу протягом календарного місяця, навіть з урахуванням обмежень, пов'язаних із хостингом.

NFR5. База даних повинна архівуватися не рідше одного разу на добу для запобігання втраті даних.

NFR6. У разі збоїв чи помилок платформа має бути відновлена у робочий стан не пізніше ніж за 1 годину (наприклад, через резервне встановлення).

### **2.5.3 Сумісність і адаптивність.**

NFR7. Сайт повинен стабільно функціонувати у сучасних браузерах, таких як Google Chrome, Mozilla Firefox та Opera.

NFR8. Інтерфейс має коректно масштабуватися під мобільні пристрої з мінімальною шириною екрана від 360 пікселів, що відповідає розмірам більшості смартфонів.

### **2.5.4 Безпека інформації.**

NFR9. Усі передані користувачем дані повинні захищатися протоколом HTTPS; рекомендується використання безкоштовного SSL-сертифікату, наприклад, Let's Encrypt.

NFR10. Паролі користувачів мають зберігатися у захищеному вигляді з використанням щонайменше алгоритмів хешування MD5 або SHA-256, з можливістю подальшого вдосконалення безпеки.

NFR11. Для запобігання автоматизованому доступу слід реалізувати базовий захист, зокрема CAPTCHA на формі входу або реєстрації.

### 2.5.5 Масштабованість і підтримка.

NFR12. Адміністративна частина сайту повинна містити зручний інтерфейс для додавання нових товарів, не вимагаючи безпосереднього редагування коду.

NFR13 Структура вебресурсу має дозволяти розширення категорій товарів і додавання нових функцій без необхідності повної реконструкції архітектури.

## 2.6 Ідентифікація архетипу та проєктування інтерфейсу вебресурсу

Вебресурс відноситься до архетипу Web Application (WA) – класичний вебдодаток, що надає інтерфейс через HTML, CSS та JavaScript.

Окрім клієнтської частини, вебресурс складається з серверної частини на базі фреймворку Django та СУБД PostgreSQL для збереження даних.

Ця структура забезпечує масштабованість, швидкодію та зручність розробки і підтримки вебресурсу.

Проєктування системи вікон вебресурсу з продажу гаджетів і аксесуарів відбувається за допомогою діаграми станів, яка наведена на рис. 2.3.

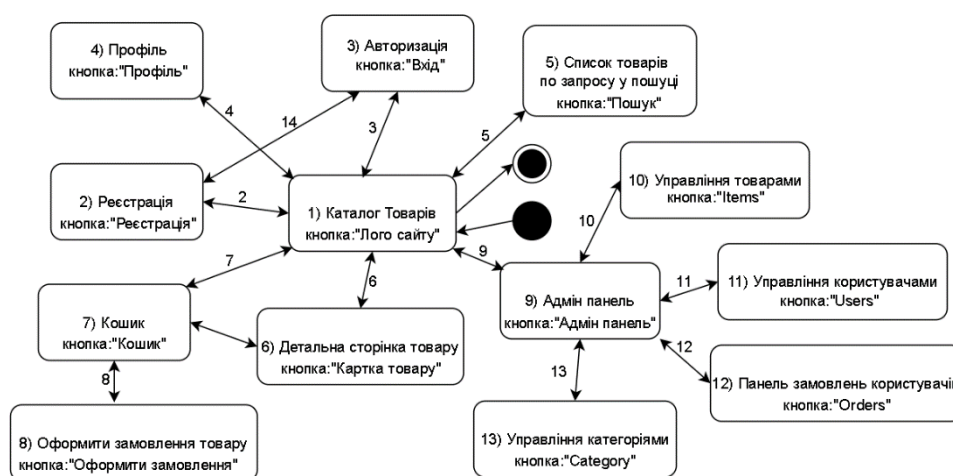


Рисунок 2.3 – Діаграма станів вебресурсу

Після визначення системи екранів вебресурсу з продажу гаджетів і аксесуарів необхідно розглянути макети основних вікон вебресурсу та надати їх стислий опис.

На рис. 2.4 зображений макет головного екрану каталога вебресурсу з підтримки продажу гаджетів і аксесуарів. В верхній частині вікна розташований Логотип сайту.

Далі нижче є вікно з каталогом товарів і фільтром де присутні різні типи товарів. Поруч з Каталогом є блок з Пошуком товарів. Це дуже облегшує пошук потрібних товарів користувачу по назві. Поруч з пошуком є блок з Профілем користувача, де знаходиться інформація про користувача. Поруч з Профілем є посилання на список обраних товарів (Favorites).

Далі на екрані міститься сітка з карточками товарів. Кожна картка містить фотографію товару, різні позначки (наприклад, якщо товар продається зі знижкою або цей товар є новинкою). Далі йде опис товару, його рейтинг та вартість.

Для того, щоб перейти до вікна товару, користувачу потрібно натиснути на карту товару.

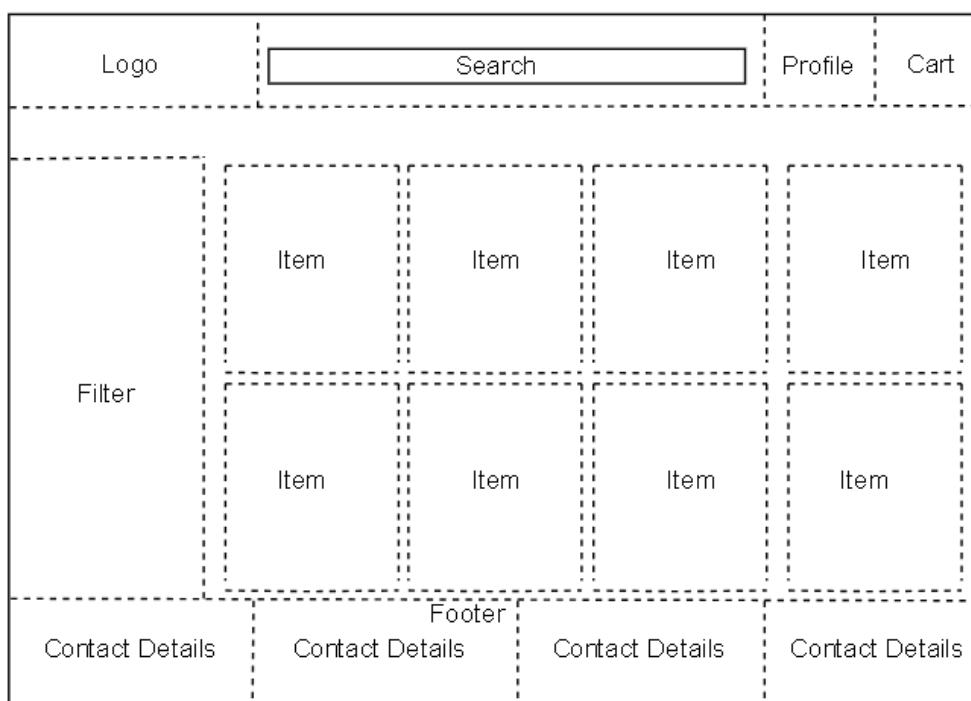


Рисунок 2.4 – Макет екрану головного вікна вебресурсу

В нижній частині вікна знаходиться футер, в якому містяться різні контактні адреси та посилання на соціальні мережі фірми.

На рис. 2.5 зображений макет екрану детального перегляду товару.

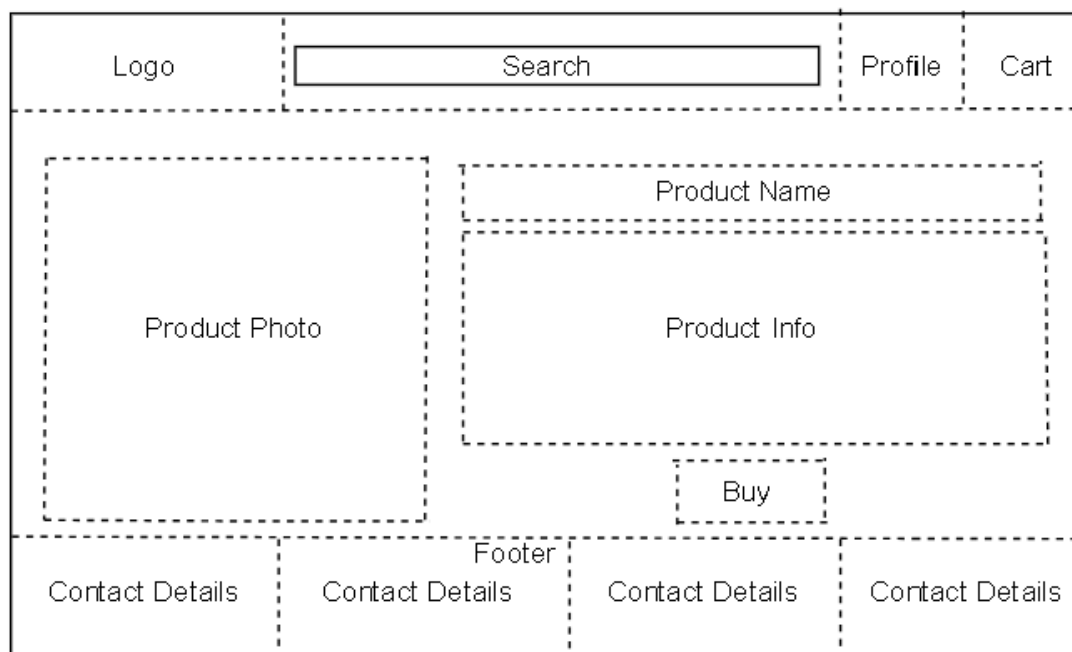


Рисунок 2.5 – Макет екрану перегляду товарів

Верхня частина екрану залишається незмінною. Логотип, Контактні данні, Випадаюча панель з каталогом товарів, Пошук, Профіль користувача і Обрані товари все залишається.

Нижче у вікні додаю блоки з детальною інформацією про товар, детальні характеристики товару і відгуки інших користувачів.

Нижче є блок з фотографіями товару. Поруч з фото є блок з іменем товару, а нижче знаходиться блок з стислою інформацією про товар.

Нижче, під цим блоком знаходиться дві кнопки з додаванням до списку Сподобавшихся товарів і кнопка для додання товару до покупки. Користувач може перейти до його оплати за допомогою платіжного шлюзу.

На рис. 2.6 зображений макет екрану замовлення товарів вебресурсу. В цьому вікні користувач може сформувати замовлення та перейти до його оплати за допомогою платіжного шлюзу.

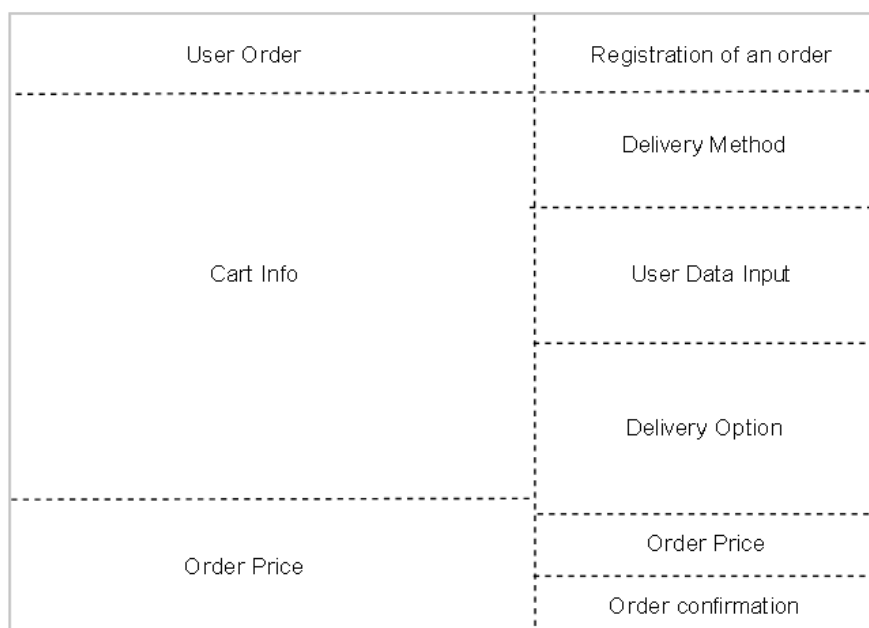


Рисунок 2.6 – Макет екрану замовлення товарів вебресурсу

Верхня частина екрану залишається незмінною, як і на попередніх екранах. В правій частині екрану міститься блок з контактними даними, з права логотип.

В середині знаходиться блоки з введенням контактих даних користувача (ім'я, прізвище, номер телефону тощо). Далі йдуть параметрів доставки та оплати. Користувачу треба вказати спосіб доставки товару, місто, і номер відділення пошти.

В правій частині екрану міститься інформація про товарні позиції і загальну вартість замовлення. Для підтвердження замовлення користувачу треба натиснути на кнопку підтвердження замовлення. Інші макети екранів наведені в Додатку А даної роботи.

## 2.7 Проєктування логічної моделі вебресурсу

Формування логічного уявлення вебресурсу, орієнтованого на продаж гаджетів та аксесуарів, є важливою складовою процесу проєктування, яка закладає основу для побудови цілісної архітектури системи. Побудова логічної моделі дозволяє окреслити основні структурні елементи платформи, визначити зв'язки між ними та описати механізми їхньої взаємодії. Це не лише сприяє кращому

розумінню загальної організації вебресурсу, а й допомагає забезпечити узгодженість між різними компонентами на етапі реалізації.

Логічне моделювання дозволяє абстрагуватись від фізичної реалізації компонентів і зосередитися на структурі системи, її функціональних складових та принципах взаємодії між ними. Таке уявлення дає змогу ефективно виявити потенційні вузькі місця у проєкті ще до початку реалізації, а також полегшує подальше документування та підтримку проєкту.

У процесі побудови логічної моделі були визначені ключові підсистеми: модуль управління користувачами, каталог товарів, система замовлень, інтерфейс для адміністрування контенту, механізм обробки відгуків, платіжна інтеграція та модуль аналітики. Зв'язки між ними побудовані на основі взаємозалежностей, що виникають у процесі обробки даних і реалізації бізнес-логіки.

Візуалізація логічної структури системи наведена на рис. 2.7. Вона відображає ключові логічні сутності, їхню ієрархію, а також способи взаємодії між окремими частинами вебресурсу. Такий підхід значно спрощує етапи подальшої розробки, модульного тестування та впровадження нових функцій.

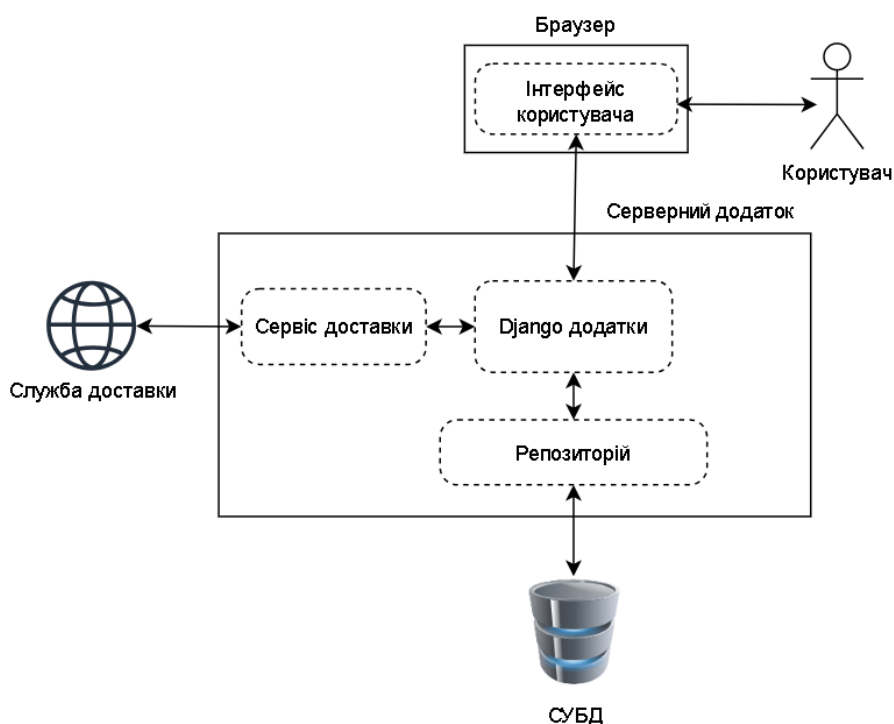


Рисунок 2.7 – Діаграма логічного уявлення вебресурсу

## 2.8 Проектування діаграми шарів вебресурсу

Для ефективної розробки вебресурсу з продажу гаджетів та аксесуарів використовується діаграма шарів, що демонструє архітектурну структуру системи та взаємозв'язки між її компонентами. Вона спрощує розробку, тестування й обслуговування.

У рамках Django застосовується архітектурний шаблон MVT (Model–View–Template), де: Model відповідає за роботу з базою даних; View обробляє запити й виконує бізнес-логіку; Template генерує HTML-інтерфейс. У Django немає контролерів, замість них обробка введів користувачів виконують файли forms.py та Java Script файли.

Такий підхід забезпечує логічний розподіл відповідальностей і сприяє стабільності, масштабованості та зручності підтримки вебресурсу (рис. 2.8).

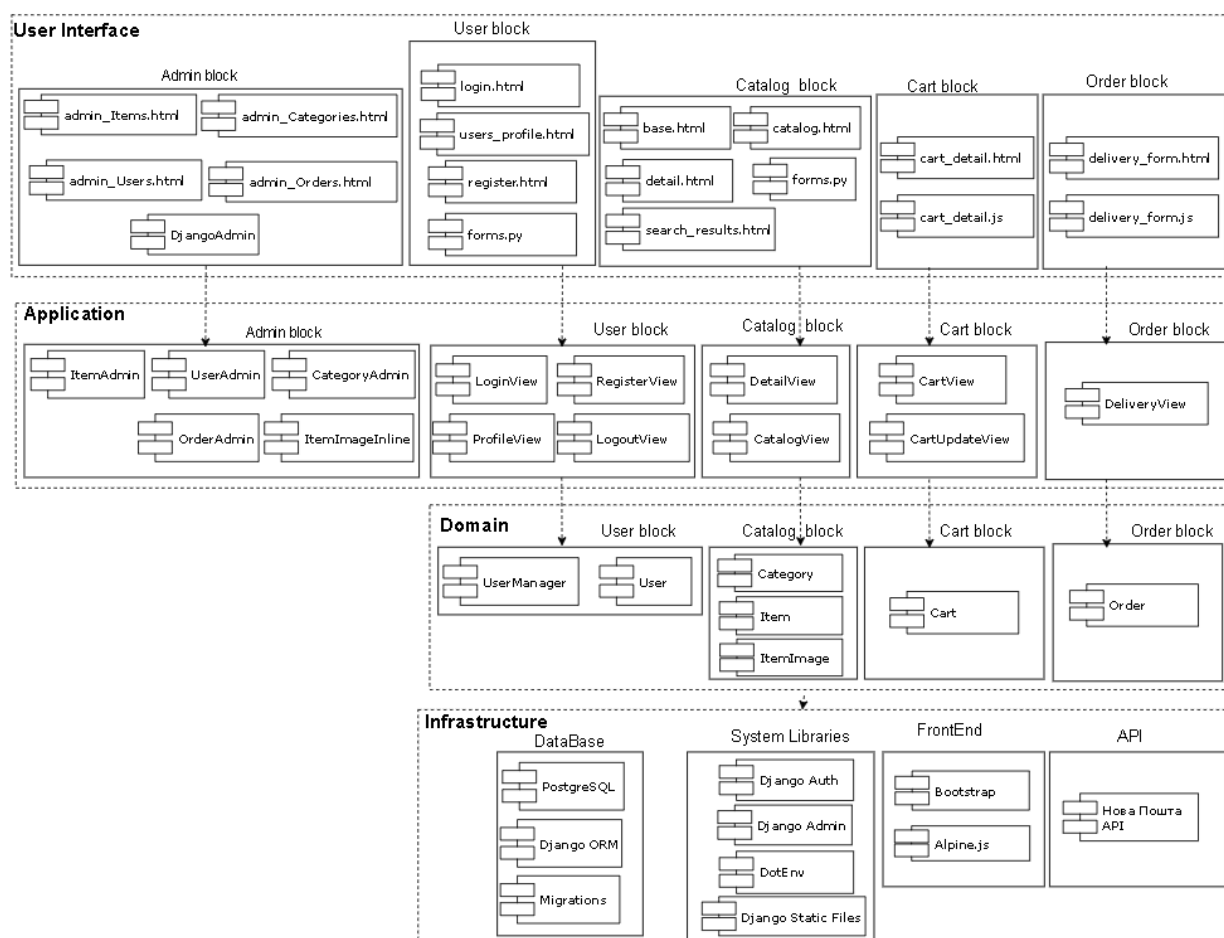


Рисунок 2.8 – Діаграма шарів вебресурсу



## 2.9 Проектування розгортання вебресурсу

Розробка діаграми розгортання вебресурсу для продажу гаджетів та аксесуарів є ще одним етапом, що забезпечує чітке уявлення про технічну інфраструктуру, необхідну для стабільного функціонування системи. Така діаграма ілюструє фізичне розміщення серверів, мережеских вузлів та інших технічних компонентів, що дозволяє визначити спосіб взаємодії різних частин вебдодатку у реальному середовищі.

Діаграма розгортання вебресурсу представлена на рис. 2.9.

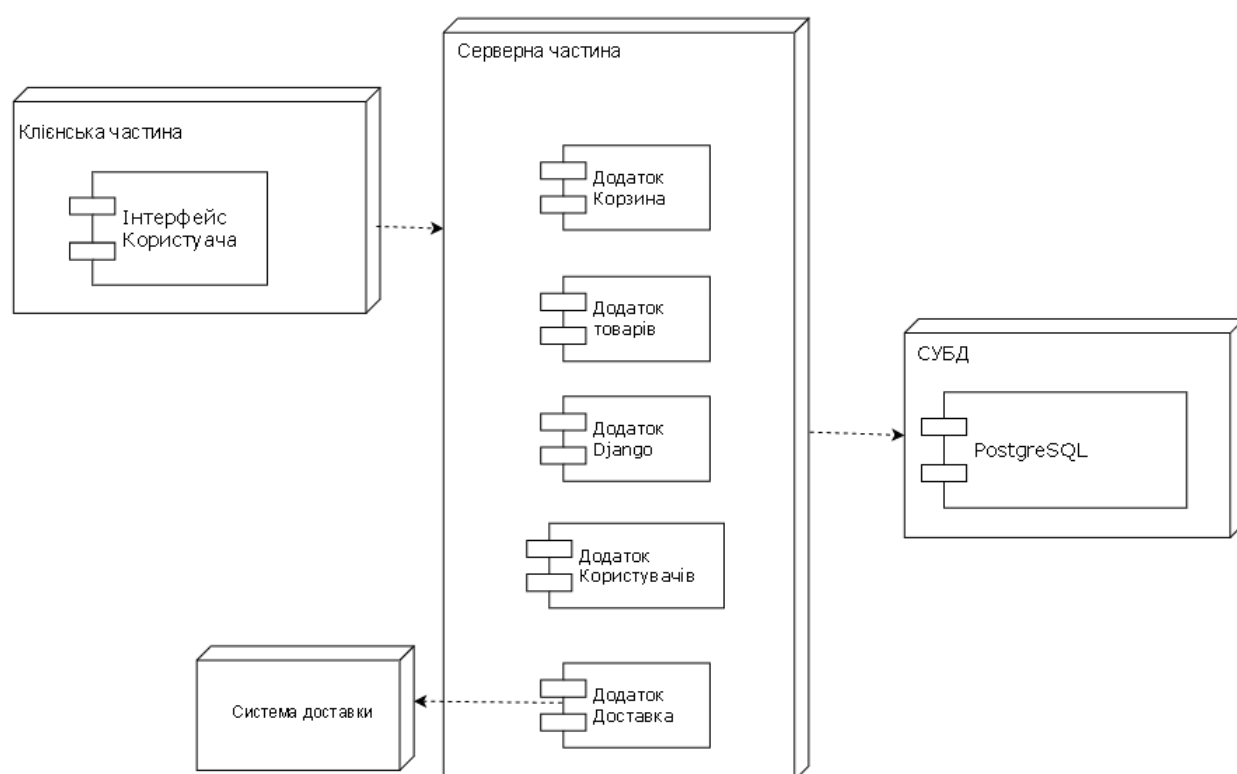


Рисунок 2.9 – Діаграма розгортання вебресурсу

## 2.10 Моделювання динамічної поведінки вебресурсу

Для відображення динамічної поведінки вебресурсу та аналізу взаємодії його компонентів під час виконання функціональних сценаріїв було створено UML-

діаграми. Зокрема, розроблено діаграми активності та послідовності, які демонструють логіку роботи системи поетапно.

Діаграма активності (Activity Diagram) – це графічне подання процесів в системі. Вона використовується для наочного відображення процесів або потоків даних через різні кроки.

На рис. 2.10 наведено діаграму активності для сценарію «Перегляд каталога і детальна інформація товару».

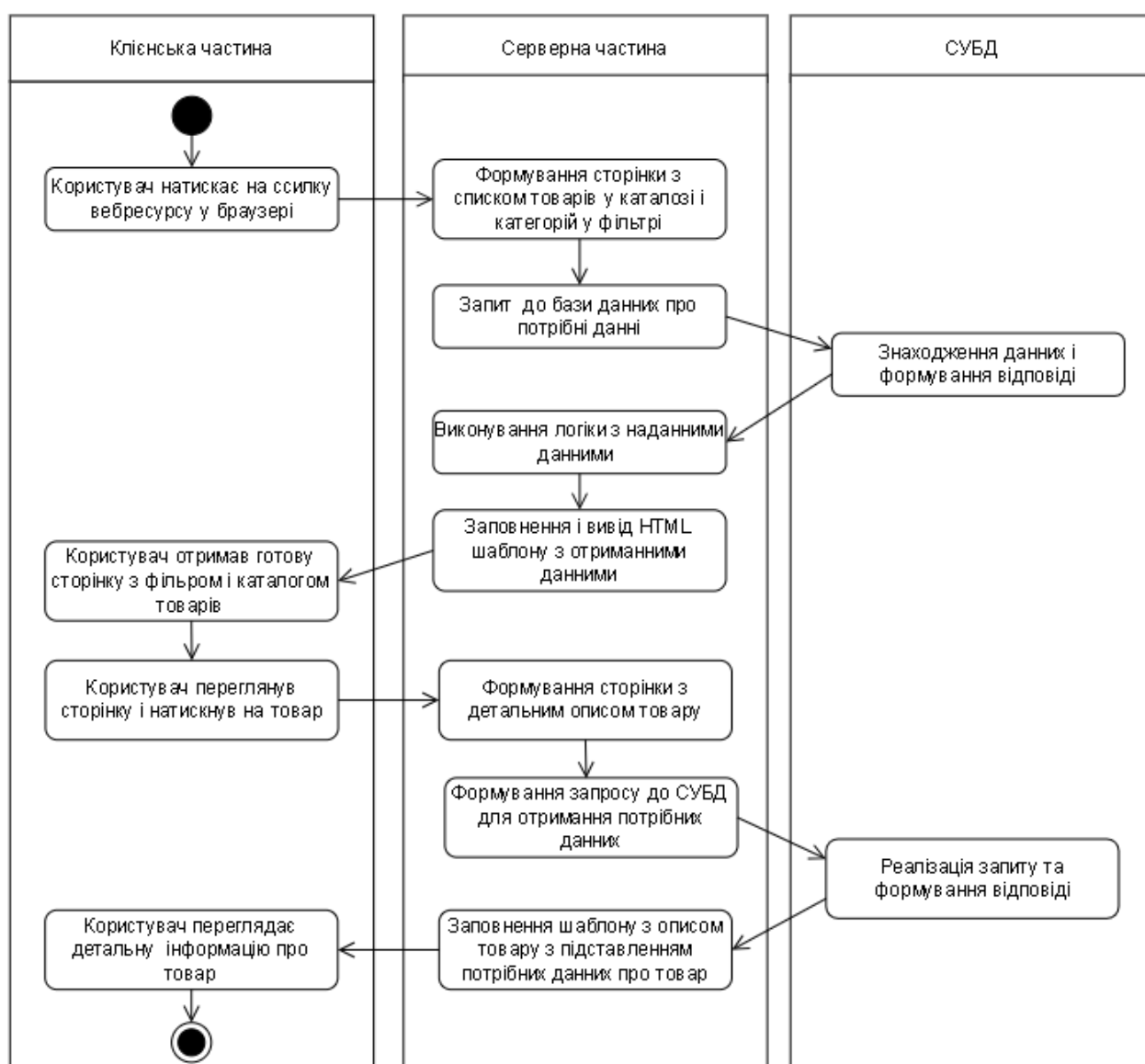


Рисунок 2.10 – Діаграма активності сценарію «Перегляд каталога і детальна інформація товару»

Діаграма послідовності (Sequence Diagram) є важливим інструментом для моделювання динаміки системи. Вона детально відображає взаємодію між об'єктами у визначеній часовій послідовності, що дозволяє наочно побачити порядок обміну даними і повідомленнями між різними компонентами під час виконання певного функціонального сценарію. Такий тип діаграм демонструє, як саме ініціюються події, як об'єкти реагують на виклики та як відбувається передача даних у процесі взаємодії.

На рис. 2.11 представлено діаграму послідовності, що ілюструє сценарії перегляду каталогу і детальної інформації про товар. Вона відображає послідовну взаємодію користувача з вебінтерфейсом.

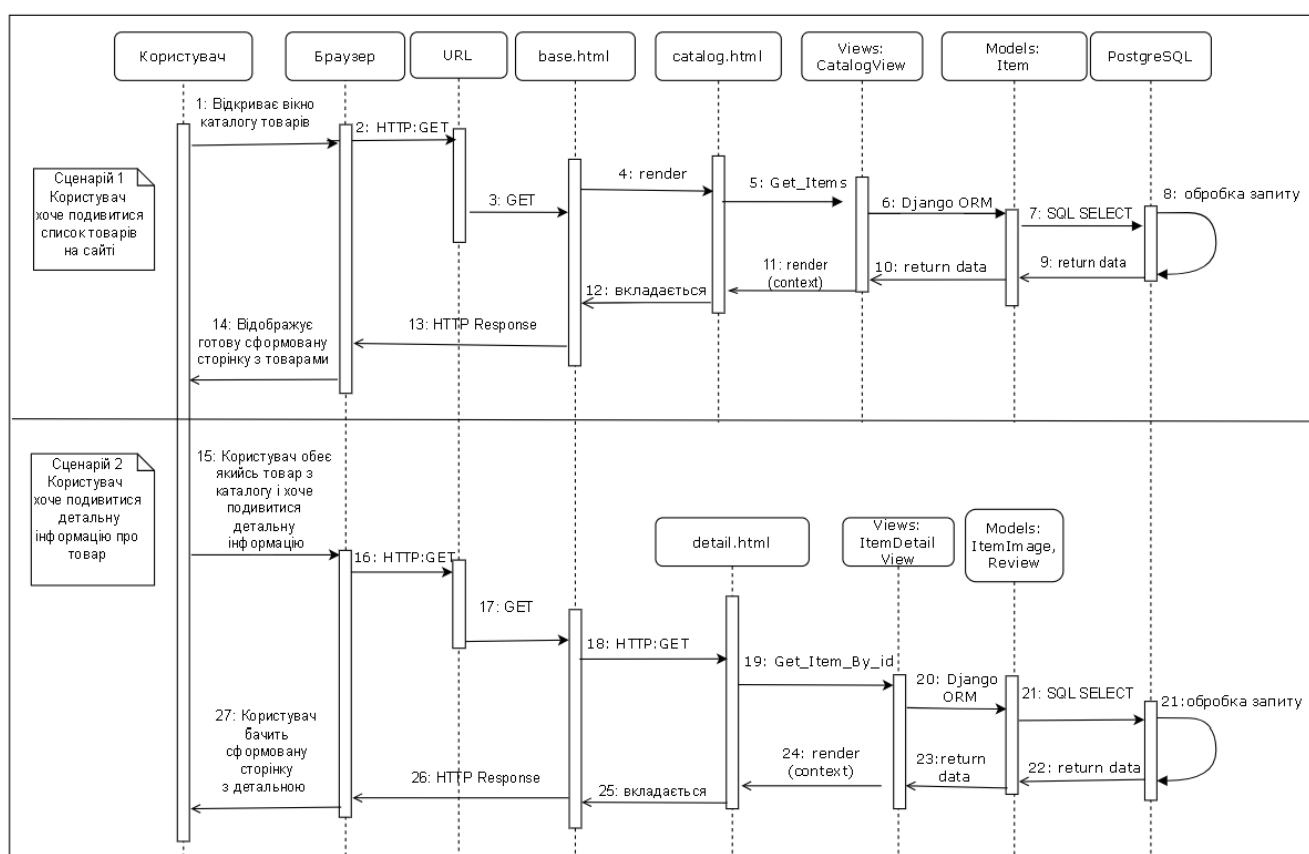


Рисунок 2.11 – Діаграма послідовності сценарію «Перегляд каталогу і товарів»

Основна користь діаграми послідовності полягає в тому, що вона допомагає зрозуміти логіку роботи системи на прикладі конкретного процесу – хто ініціює

дію, які об'єкти залучаються до її виконання та в якій послідовності це відбувається. Це особливо важливо на етапі проєктування, коли необхідно врахувати усі можливі взаємозв'язки між частинами системи та забезпечити коректну реалізацію бізнес-логіки.

## 2.11 Проєктування моделі даних вебресурсу

Проєктування структури даних вебресурсу з продажу гаджетів та аксесуарів є ключовим етапом у створенні ефективної та зручної для масштабування інформаційної системи. Грамотно спроектована схема забезпечує раціональне зберігання, швидкий доступ і обробку даних. Схема даних вебресурсу наведена на рис. 2.12.

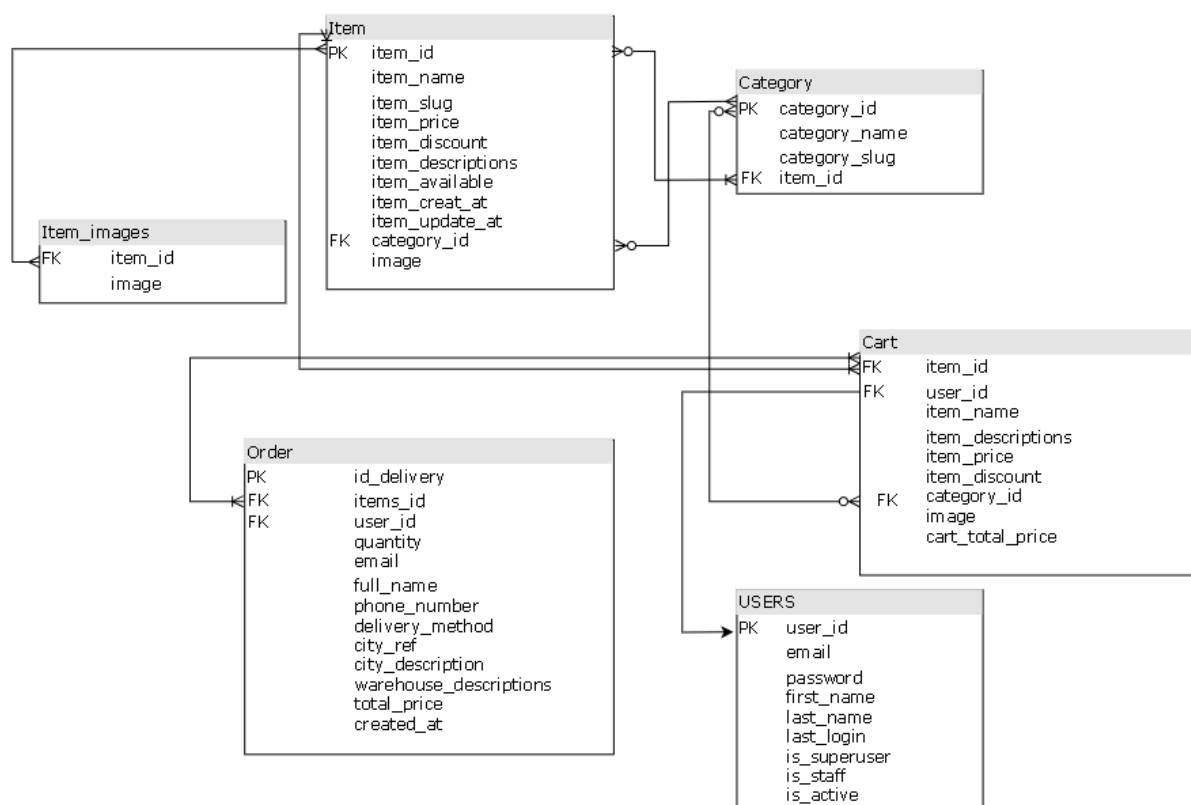


Рисунок 2.12 – Представлення схеми даних вебресурсу

До основних етапів цього процесу належать: аналіз функціональних вимог, визначення основних сутностей, їх характеристик, встановлення логічних зв'язків

між сутностями, проведення нормалізації структури та підготовка документації зі схемою бази даних.

## 2.12 Уявлення інтерфейсів вебресурсу

Користувацький інтерфейс побудований за допомогою стандартних вебтехнологій – таких як HTML і CSS, а також фреймворків для зручної верстки (наприклад, Bootstrap) та засобів для реактивної поведінки елементів Alpine.js.

На рис. 2.13 наведена діаграма уявлення інтерфейсів вебресурсу.

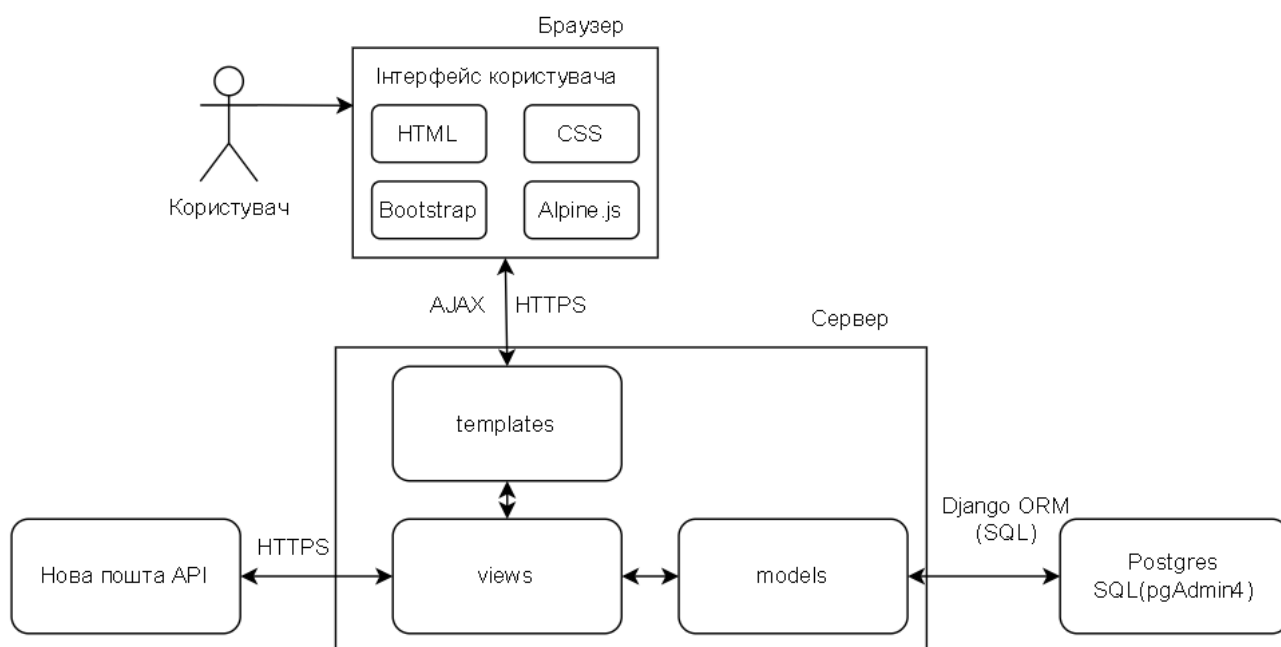


Рисунок 2.13 – Діаграма уявлення інтерфейсів вебресурсу

З боку користувача основна взаємодія відбувається через браузер, у якому функціонує клієнтська частина застосунку. Комунікація між клієнтською частиною та сервером реалізується через протокол HTTPS із використанням технології AJAX, що забезпечує асинхронну обробку запитів без повного перезавантаження сторінки.

На серверній стороні структура побудована відповідно до архітектури фреймворку Django. Шаблони (templates) відповідають за формування HTML-

сторінок, які відправляються до браузера. Контролери (views) керують логікою обробки запитів, взаємодіють з моделями даних та зовнішніми сервісами. Моделі (models), у свою чергу, використовують об'єктно-реляційне відображення (ORM), що дозволяє взаємодіяти з базою даних PostgreSQL без написання SQL-коду.

Окремим компонентом є зовнішній API служби доставки Нової Пошти, з яким серверна частина комунікує через захищений протокол HTTPS.

### **2.13 Забезпечення безпеки вебресурсу**

Безпека вебресурсу є критично важливою складовою в процесі розробки та розгортання інформаційної системи. Django – це сучасний фреймворк для веброзробки, який має вбудовані механізми захисту від поширених типів атак. Завдяки цим механізмам забезпечується базовий рівень захисту навіть без додаткової конфігурації, що робить Django одним із найнадійніших інструментів для створення безпечних вебзастосунків.

Серед основних засобів безпеки, реалізованих у Django, можна виокремити:

1) захист від міжсайтового скриптингу (XSS). Django автоматично екранує (escape) всі вихідні дані у шаблонах, запобігаючи вставці шкідливих скриптів у HTML. Це дозволяє уникнути ситуацій, коли зловмисник може вставити шкідливий JavaScript-код на сторінки, що переглядаються іншими користувачами;

2) захист від підробки міжсайтових запитів (CSRF). Для всіх POST-запитів у формах Django автоматично вимагає наявності CSRF-токена – спеціального унікального маркера, який генерується для кожного користувача. Це виключає виконання шкідливих запитів з інших сайтів від імені користувача без його відома;

3) захист від SQL-ін'єкцій. Django використовує ORM (Object-Relational Mapping), яка будує SQL-запити безпосередньо з Python-коду, автоматично екрануючи всі параметри. Таким чином, запити до бази даних не схильні до SQL-ін'єкцій навіть при обробці даних з користувацького вводу;

4) шифрування паролів. Для зберігання паролів користувачів Django застосовує хеш-функції (bcrypt, PBKDF2), що забезпечує надійний захист навіть у випадку витоку даних із бази;

5) контроль доступу та автентифікація. Django має вбудовану систему авторизації та автентифікації користувачів. Можна задавати рівні доступу до різних частин сайту за допомогою груп, прав та декораторів доступу (@login\_required, @permission\_required);

6) захист від клікджекінгу. Через налаштування HTTP-заголовків, таких як X-Frame-Options, фреймворк забороняє вбудовування вебсторінки у фрейми, що захищає від атак типу clickjacking;

7) безпечні сесії. Django реалізує зберігання сесій у захищеному вигляді та підтримує опції для шифрування cookies, заборони передачі через HTTP (HttpOnly, Secure);

8) захист від brute-force атак. Хоча базовий захист не включає обмеження кількості спроб входу, це можна реалізувати за допомогою сторонніх бібліотек (наприклад, django-axes), які інтегруються у фреймворк та фіксують підозрілу активність.

Таким чином, фреймворк Django забезпечує багаторівневий підхід до захисту вебресурсу, автоматизуючи основні аспекти безпеки та дозволяючи розробнику зосередитися на бізнес-логіці. За потреби рівень безпеки можна розширити за допомогою додаткових інструментів, таких як брандмауери, сканери вразливостей або системи виявлення вторгнень.

## **2.14 Перелік технологій вебресурсу**

У процесі розробки вебресурсу було використано сучасний набір технологій та інструментів, які забезпечили ефективну реалізацію функціональності, адаптивний інтерфейс та високий рівень безпеки. Перелік ключових технологій:

1) мова програмування Python – основна мова розробки серверної частини, обрана за її читабельність, простоту та розвинену екосистему;

2) вебфреймворк Django – високорівневий фреймворк для створення вебдодатків, що забезпечує швидку розробку, вбудовану ORM, шаблонізатор, систему автентифікації та безпеки;

3) шаблонізатор: Django Templates – система шаблонів для формування HTML-сторінок на стороні сервера з динамічними даними;

4) база даних PostgreSQL – обрана як основна система управління базами даних за її стабільність, масштабованість і підтримку складних запитів;

5) ORM: Django ORM – механізм взаємодії з базою даних, що дозволяє працювати з даними через Python-код без написання SQL;

6) клієнтські технології (frontend):

а) HTML5 – мова розмітки для структурування контенту сторінок;

б) CSS3 – стилізація та візуальне оформлення елементів інтерфейсу;

в) Bootstrap – фреймворк для створення адаптивного дизайну;

г) Alpine.js – JavaScript-фреймворк для інтерактивності на стороні клієнта;

д) AJAX – технологія для асинхронного обміну даними з сервером без перезавантаження сторінки;

7) API-зв'язки – Нова Пошта API, інтеграція з сервісом доставки для отримання інформації про відділення, формування доставок;

8) протоколи HTTPS – забезпечення захищеного з'єднання між клієнтом і сервером;

9) інструменти керування базою даних pgAdmin 4 – графічний інтерфейс для адміністрування PostgreSQL;

10) середовище розробки PyCharm 2024.3.4 – зручне середовище для розробки з підтримкою розширень;

11) Git – система контролю версій для ведення історії змін;

12) GitHub – хостинг-ресурс для зберігання та спільної роботи над проєктом.



## Висновки до розділу 2

У другому розділі виконано детальне проєктування вебресурсу для продажу гаджетів та аксесуарів. Розгляд почався з визначення мети системи та її основних функцій: каталогізація товарів, облік користувачів, оформлення замовлень із варіантами доставки, а також система оцінок і відгуків.

Проведено аналіз і візуалізацію потоків інформації, що дало змогу окреслити критичні точки взаємодії в системі, розмежувати зони відповідальності та забезпечити належну продуктивність і безпеку. Створено UML-діаграму варіантів використання, яка охоплює основні ролі: гість, зареєстрований користувач та адміністратор. Це дозволило сформулювати функціональні вимоги та врахувати особливості доступу різних категорій користувачів.

Окрім функціональних вимог, визначено також нефункціональні: адаптивність інтерфейсу, зручна навігація, захист даних, стабільність і масштабованість. На основі цього було спроектовано інтерфейси, структуру бази даних, архітектуру системи, а також макети сторінок.

Для реалізації обрано сучасні технології: Django як фреймворк, Alpine.js для клієнтської логіки, Bootstrap для адаптивного дизайну та PostgreSQL як СУБД. Такий стек забезпечує гнучкість, продуктивність і легкість у подальшій підтримці системи.

Таким чином, у розділі створено архітектурну та логічну основу вебресурсу, що дозволяє ефективно перейти до етапу реалізації. Це значно знижує ризики помилок, спрощує тестування і сприяє створенню якісного продукту, орієнтованого на користувача.

## 3 ПРОГРАМНА РЕАЛІЗАЦІЯ ВЕБРЕСУРСУ З ПІДТРИМКИ ПРОДАЖУ ГАДЖЕТІВ ТА АКСЕСУАРІВ

### 3.1 Структура проєкту вебресурсу

Вебресурс для онлайн-продажу гаджетів та аксесуарів з технічної точки зору реалізовано у вигляді вебзастосунку на основі фреймворку Django, який поєднує серверну логіку з генерацією динамічних HTML-сторінок (рис. 3.1).

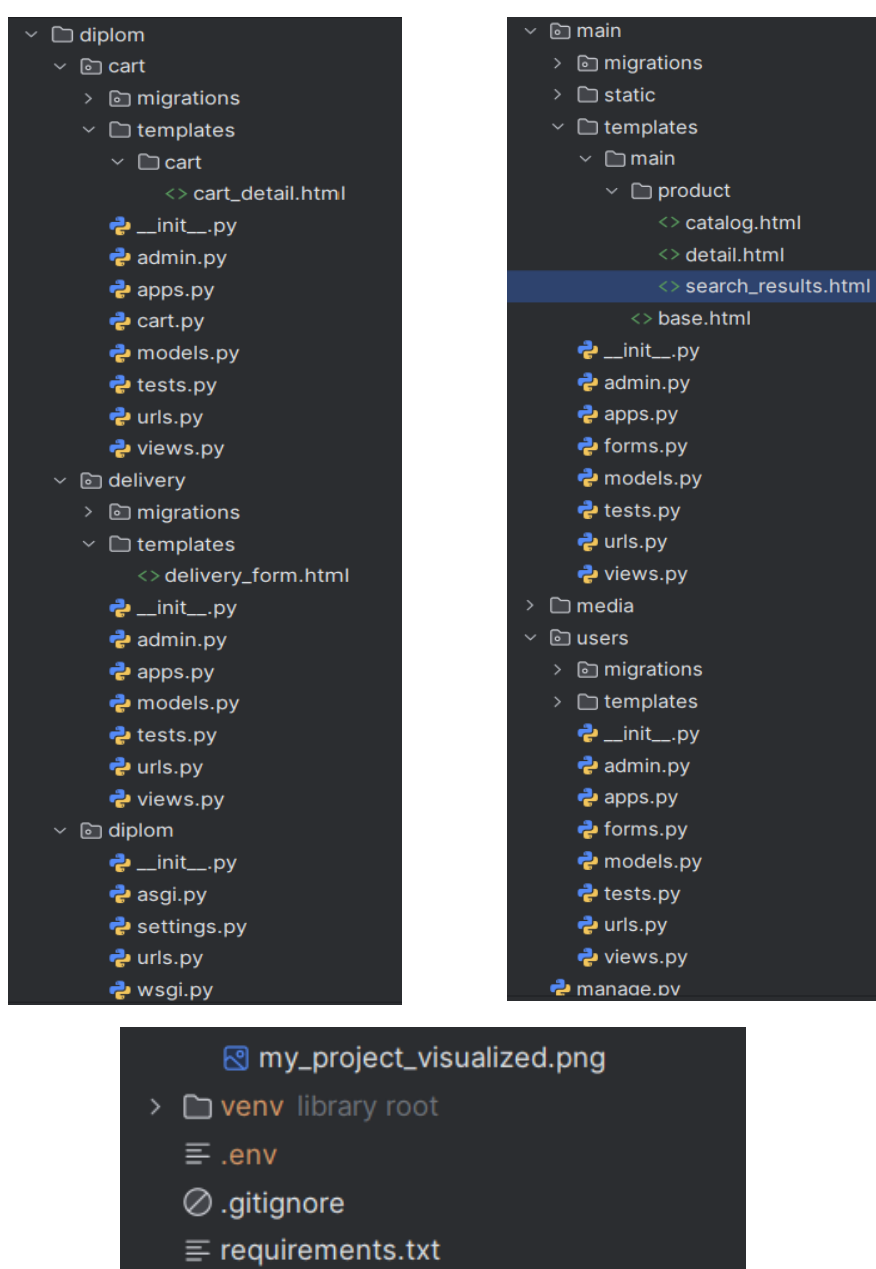


Рисунок 3.1 – Структура проєкту вебресурсу

Для створення інтерфейсу користувача застосовуються технології HTML5, CSS3, а також фреймворк Bootstrap, що забезпечує адаптивність та сучасний дизайн. Додаткова клієнтська логіка реалізована за допомогою JavaScript та Alpine.js, що дозволяє реалізовувати інтерактивність без важких JavaScript-бібліотек.

Розробка проєкту здійснювалася у середовищі PyCharm 2024.3.4 з використанням системи керування версіями Git. У якості системи керування базами даних використано PgAdmin4 для бази даних PostgreSQL, а доступ до неї реалізовано через ORM Django, що дозволяє працювати з базою на рівні Python-класів.

Для кращого розуміння структури вебресурсу нижче наводиться короткий опис основних модулів програмного коду:

1) `settings.py` – містить налаштування застосунку, зокрема конфігурацію бази даних, параметри безпеки та глобальні параметри проєкту і сервера;

2) `views` – включає обробники HTTP-запитів, які відповідають за маршрутизацію, виклик бізнес-логіки та формування відповіді користувачу. Саме тут відбувається інтеграція з шаблонами;

3) `templates` – директорія, що містить HTML-шаблони сторінок, які динамічно наповнюються даними за допомогою механізму Django Template Language;

4) `static` – включає всі статичні ресурси: CSS-файли, JavaScript-скрипти, зображення та компоненти Bootstrap/Alpine.js;

5) `models` – модулі, які описують структуру об'єктів предметної області (товари, категорії, користувачі, замовлення тощо) та автоматично створюють відповідні таблиці в базі даних;

6) `forms` – містить форми для взаємодії користувача з сайтом (реєстрація, авторизація, оформлення замовлення), які пов'язані з відповідними моделями;

7) `services` – реалізовує окремі компоненти бізнес-логіки: обробку замовлень, розрахунок вартості, роботу з API доставки тощо;

Завдяки розділенню структури на логічні модулі забезпечується зручність підтримки, масштабованість і прозорість логіки функціонування вебресурсу.

### 3.2 Уявлення про структуру класів вебресурсу

Діаграма класів дає змогу розробникам та іншим учасникам проєкту краще уявити структуру вебресурсу, зрозуміти взаємозв'язки між його компонентами та своєчасно виявити потенційні проблеми або недоліки ще на етапі проєктування. Це сприяє оптимізації архітектури, більш ефективному плануванню та забезпечує єдине бачення програмного продукту (рис. 3.2).

На цій діаграмі класів зображено основні сутності, що відповідають таблицям бази даних вебресурсу з підтримки продажу гаджетів і аксесуарів.

Сутність User містить атрибути, що характеризують користувача: id, email, ім'я, роль користувача і пароль. До цієї сутності прив'язані функції управління для адміна над іншими користувачами яка знаходиться вже у сутності UserAuth. До цієї таблички прив'язана функція з повним логуванням дій адміністратора, які зміни робив над користувачами, товарами, категоріями яка знаходиться у сутності Admin.

Сутність Main представляє собою модель Item. Це сам товар, який буде відображатися у каталозі. Він має такі атрибути як: id, категорію, наявність, метрики коли створен і коли оновлен, детальний опис товару, знижка у відсотках, головну картинку для картки, ціну і slug це автопереклад на англійську для urls. Ще є сутність Category, котра прив'язується до Item і має поля id, ім'я, slug. Також наявна модель ItemImage яку прив'язують до Item, в ній зберігаються id товару, всі детальні фото товару для його детального опису. І остання модель це Review котра прив'язана до Item яка потрібна для відображення коментарів користувачів до товару. Вона має id, коментар, ім'я користувача, id товару, рейтинг, коли створен.

Сутність Delivery має атрибут Order для оформлення доставки товару користувачу. Цей атрибут має такі поля як id, назву міста або населеного пункту і у яке відділення доставити посилку, користувач обирає це все зі списку, котрий доступний у Новій пошті. Далі атрибут Order має поле способу доставки, email користувача, його повне ім'я, id доставки, номер користувача і повну ціну замовлення.

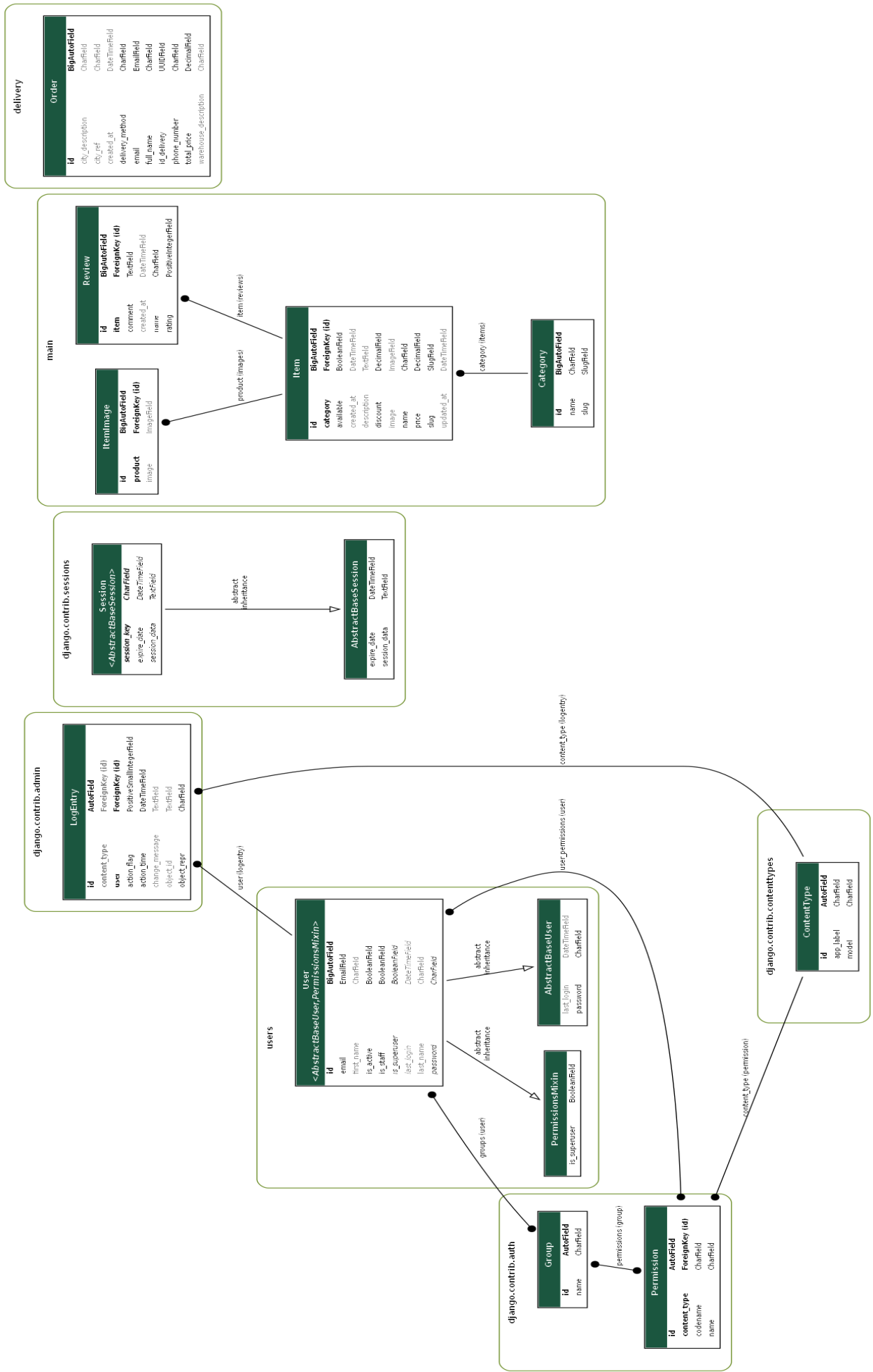


Рисунок 3.2 — Діаграма класів вебресурсу

### 3.3 Використання сервісу GitHub для організації управління програмним кодом системи

В ході розробки вебресурсу з підтримки продажу гаджетів та аксесуарів використовувалася система контролю версій Git, з репозиторієм, розміщеним на платформі GitHub. Такий підхід забезпечив ефективне управління змінами, контроль версій та можливість гнучкого командного розроблення в умовах інкрементального додавання функціональності (табл. 3.1).

Під час розробки було дотримано основних практик роботи з Git, зокрема:

- а) було заборонено прямі коміти до основної гілки main;
- б) усі зміни проходили через pull request після обов'язкової перевірки;
- в) коміти створювалися регулярно та логічно, що дозволяло легко відстежувати хід розробки;
- г) для кожного окремого модуля створювалися окремі гілки – feature branches.

Таблиця 3.1 – Метрики репозиторію вебресурсу

Коміти	Рядків коду додано	Рядків коду видалено	Відкриті Issues	Закриті Issues
45	3910	600	26	26

### 3.4 Підрахунок метрик вихідного коду вебресурсу

Аналіз метрик вихідного коду є ключовим елементом оцінки розробленого вебресурсу. Він допомагає глибше оцінити якість програмного забезпечення, що особливо важливо для платформ, які працюють із потенційно ризикованим контентом. У таких випадках першочергового значення набувають стабільність роботи системи та відповідність стандартам безпеки. Завдяки аналізу можна на ранньому етапі виявити технічні проблеми — наприклад, часті зміни в окремих

файлах або великі обсяги видаленого коду можуть свідчити про недоліки архітектури чи помилки в проектуванні. Це дозволяє завчасно реагувати на потенційні ризики та підвищити надійність кінцевого продукту.

У табл. 3.2 представлено показники метрик, що були зібрані для оцінки вихідного коду вебресурсу з продажу гаджетів та аксесуарів.

Таблиця 3.2 – Показники метрик вихідного коду вебресурсу

Метрика	Показник метрики
Загальна кількість рядків коду у проекті ІС	1755
Середня кількість рядків коду в одному класі	60
Максимальна кількість рядків коду в одному класі	67
Середня кількість рядків коду в одному методі.	14
Максимальна кількість рядків коду в одному методі.	55
Максимальна глибина дерева успадкування	4
Середня цикломатична складність методу.	1,77
Максимальна цикломатична складність методу.	5
Коментованість коду	11%

### 3.5 Список якості реалізації вебресурсу

Для перевірки відповідності реалізації вебресурсу з продажу гаджетів та аксесуарів вимогам до якості програмного забезпечення, наведено узагальнений контрольний список.

У табл. 3.3 містяться ключові твердження, які дозволяють оцінити стандарти кодування, архітектурні рішення, безпеку та гнучкість підтримки системи.

Таблиця 3.3 – Список якості реалізації ІС

Твердження	Відповідь	Пояснення у разі негативної відповіді
Використання логування (logging)	Так	—
Захист від ін'єкцій SQL	Так	—
Захист від Javascript/XSS ін'єкцій	Так	—
Використання валідації даних у всіх полях введення для користувача інтерфейсів	Так	—
Використання інсталяторів/інтернет магазинів	Так	—
Використання засобів синхронізації даних у разі багатопоточної програми	Так	—
Відсутність зашитих в програмний код конфігураційних параметрів програми.	Так	—
Чи застосовані UI паттерни під час розробки UI.	Так	—
Враховано coding style guide lines для вибраної мови програмування.	Так	—
Чи враховані рекомендовані guide lines при розробці користувача інтерфейсу для тієї чи іншої ОС.	Так	—

### 3.6 Функціональне тестування вебресурсу

Функціональне тестування являє собою процес перевірки програмного забезпечення з метою визначення відповідності його роботи встановленим функціональним вимогам. В процесі створення вебсайту для продажу гаджетів і аксесуарів даний тип тестування передбачає перевірку ключових функцій ресурсу, таких як реєстрація користувача, здійснення замовлення, пошук товарів тощо. Нижче подано опис етапів виконання тестування та перелік тест-кейсів (англ. Test Case, TC), розроблених для функціонального аналізу роботи сайту.



### ТС1. Реєстрація нового користувача.

Для реалізації цього тест-кейсу виконуються такі дії:

- 1) відкрити головну сторінку сайту з каталогом;
- 2) натискання кнопки «Реєстрація»;
- 3) заповнення обов'язкових полів реєстраційної форми;
- 4) натискання кнопки «Зареєструватися»;
- 5) перевірка появи повідомлення про успішну реєстрацію;
- 6) перевірка можливості входу з новоствореними обліковими даними.

### ТС2. Автентифікація зареєстрованого користувача.

Для реалізації необхідно:

- 1) відкрити головну сторінку сайту з каталогом;
- 2) обрати пункт «Вхід»;
- 3) ввести коректні логін і пароль;
- 4) натиснути кнопку «Увійти»;
- 5) переконатися, що вхід здійснено успішно і користувача перенаправлено

на головну сторінку.

### ТС3. Перевірка каталогу товарів.

Для цього необхідно:

- 1) відкрити головну сторінку сайту з каталогом;
- 2) переглянути перелік доступної продукції;
- 3) перевірити наявність фото, назви, оцінок, знижок та вартості кожного тов.;
- 4) протестувати функцію сортування за ціною.

### ТС4. Перевірка роботи пошуку товарів.

Для реалізації потрібно:

- 1) відкрити головну сторінку сайту з каталогом;
- 2) введення назви товару, тип простія, або частку назви у поле пошуку;
- 3) натиснути Enter;
- 4) перевірка відповідності результатів пошуковому запиту;

### ТС5. Фільтрація продукції за категоріями.

Необхідно:

- 1) відкрити головну сторінку сайту з каталогом;
- 2) обрати одну з категорій у фільтрі;
- 3) через секунду сторінка автоматично оновлюється;
- 4) перевірити, що відображаються тільки відповідні товари;
- 5) зняти фільтрацію та переконатися в коректному відображенні всіх тов.

ТС6. Оформлення замовлення.

Необхідно:

- 1) авторизуватися на сайті;
- 2) додати обраний товар до кошика;
- 3) перейти до кошика і натиснути "Оформити замовлення";
- 4) заповнити відповідні поля;
- 5) підтвердити замовлення та перевірити появу повідомлення про успішну операцію.

ТС7. Відгуки та рейтинги.

Для реалізації тест-кейсу виконується:

- 1) відкрити головну сторінку сайту з каталогом;
- 2) подивитися у каталог відображаються або ні зірочки оцінки користувачів;
- 3) авторизація користувача;
- 4) перехід до детальної сторінки з товаром;
- 5) перегляд розділу з відгуками і оцінками;
- 6) додавання власного коментаря та рейтингу;
- 7) перевірка їхньої появи на сторінці товару.

ТС8. Перевірка контактної інформації.

Слід виконати:

- 1) відкрити сторінку каталога,пошука,профіля чи кошика;
- 2) пролистати вниз сторінки;
- 3) знайти блок footer;

4) переконатися в наявності актуальної інформації (телефон , адреса, email, посилання на Соцмережі);

5) перевірити коректність посилань на соціальні мережі.

ТС9. Демонстраційні фото.

Для перевірки функціоналу слід:

1) відкрити головну сторінку сайту з каталогом;

2) перейти до детальних сторінки товарів;

3) переглянути наявні фото;

4) перевірити їх коректне відтворення;

5) перевірити їх коректне перемикавання;

5) оцінити якість та релевантність контенту.

Загалом було підготовлено 9 тест-кейсів, спрямованих на функціональну перевірку вебресурсу для продажу піротехнічної продукції. Підсумки проведеного тестування наведено у табл. 3.4.

Таблиця 3.4 – Протокол функціонального тестування вебресурсу

Номер тест-кейсу	Назва тест-кейсу	Фактичний результат	Результат
ТС1	Реєстрація нового користувача	Користувач успішно реєструється і може увійти в систему	Виконано
ТС2	Автентифікація користувача	Користувач успішно входить в систему	Виконано
ТС3	Перевірка каталогу продукції	Каталог продукції відображається коректно, кожен товар має фото, назву, скидку та ціну	Виконано
ТС4	Перевірка роботи пошуку товарів	Пошук працює коректно, товари відображаються правильні	Виконано
ТС5	Фільтрація продукції за категоріями	Фільтрація продукції працює коректно	Виконано
ТС6	Оформлення замовлення	Замовлення успішно оформлюється, користувач отримує підтвердження	Виконано
ТС7	Відгуки та рейтинги продукції	Відгуки та рейтинги додаються та відображаються коректно	Виконано
ТС8	Перевірка контактної інформації.	Контактна інформація у футері відображається коректно	Виконано
ТС9	Демонстраційні фото	Фото відображаються коректно, слайдер фото працює коректно, зміст фото коректний	Виконано

Детальна інструкція користувача вебресурсу з підтримки продажу гаджетів і аксесуарів наведена в додатку А данної роботи.

Фрагменти вихідного коду розробленого програмного забезпечення у вигляді вебресурсу з підтримки продажу гаджетів і аксесуарів наведені в додатку Б даної роботи.

### **Висновки до розділу 3**

У результаті реалізації програмної частини вебресурсу було створено вебзастосунок на основі фреймворку Django, що забезпечує сучасний, адаптивний і безпечний інтерфейс для підтримки онлайн-продажу гаджетів та аксесуарів.

Структура проекту чітко поділена на логічні модулі, що відповідають за окремі аспекти функціональності: конфігурацію, бізнес-логіку, обробку запитів, шаблони інтерфейсу, роботу з базою даних тощо. Для реалізації клієнтської частини застосовано HTML5, CSS3, Bootstrap і JavaScript-фреймворк Alpine.js, що забезпечує інтерактивний інтерфейс.

Код вебресурсу має достатній рівень коментованості та підтримує вимоги до стилю написання. Аналіз метрик програмного коду дозволив оцінити складність та структуру системи. Система контролю версій GitHub використовувалася для ефективного управління процесом розробки, що засвідчують відповідні метрики активності репозиторію.

Функціональне тестування показало працездатність основних сценаріїв використання системи: реєстрація та автентифікація користувачів, перегляд товарів, пошук, фільтрація, оформлення замовлень, а також перегляд і додавання відгуків. Результати тестування підтверджують відповідність реалізованої системи функціональним вимогам і свідчать про її готовність до подальшого впровадження та експлуатації.

Також була розроблена інструкція користувача вебресурсу та наданий лістинг фрагментів вихідного коду розробленого програмного забезпечення.

## ЗАГАЛЬНІ ВИСНОВКИ

У результаті виконання кваліфікаційної роботи було досягнуто поставленої мети – створення функціонального та безпечного вебресурсу для продажу гаджетів і аксесуарів. Проведене дослідження, проектування та реалізація системи дозволили отримати цілісне уявлення про процес розробки вебзастосунку з урахуванням актуальних вимог користувачів і технологічних тенденцій.

На першому етапі роботи було здійснено аналіз сучасного ринку продажу гаджетів, що дозволило визначити ключові тенденції, проблеми та очікування кінцевих користувачів. Було проведено аналіз існуючих рішень – інтернет-магазинів подібної тематики, що дало змогу виокремити їхні сильні та слабкі сторони. Це стало підґрунтям для формування вимог до розроблюваної системи. Було обрано технології, що забезпечують надійність, безпеку, продуктивність та зручність подальшої підтримки системи: Django, Alpine.js, Bootstrap і PostgreSQL. Вибір патерну Model-View-Template (MVT) став логічним рішенням для структурування проєкту на основі фреймворку Django.

Другий розділ був присвячений розробці логічної та архітектурної моделі вебресурсу. Було створено UML-діаграми варіантів використання, що дозволило визначити ролі користувачів та функціонал, доступний кожній з них. Окрім того, проектування охоплювало нефункціональні вимоги, що включають стабільність роботи системи, зручну навігацію та масштабованість. Було детально опрацьовано макети сторінок, структуру бази даних, визначено логіку роботи основних модулів. Таке системне бачення дозволило знизити ризики під час реалізації та забезпечити готовність продукту до впровадження в реальні умови.

У третьому розділі було безпосередньо реалізовано програмну частину проєкту. Розробка виконувалася відповідно до визначеної структури та обраного стеку технологій. Функціональність системи охоплює всі ключові аспекти онлайн-продажу: реєстрацію та автентифікацію користувачів, перегляд товарів, пошук і фільтрацію, додавання до кошика, оформлення замовлень, систему оцінок і відгуків. Код реалізовано відповідно до вимог щодо стилю програмування, з

дотриманням принципів модульності та коментування. Було проведено функціональне тестування системи, результати якого підтвердили працездатність основних сценаріїв та відповідність програмного забезпечення заданим вимогам. Розроблено інструкцію користувача, що полегшує взаємодію з системою, та надано приклади фрагментів вихідного коду.

У підсумку, реалізований вебресурс відповідає сучасним стандартам веброзробки, орієнтований на зручність користувача та може бути використаний як основа для подальшого вдосконалення або масштабування. Отримані знання та досвід у межах даного проєкту можуть бути застосовані у майбутній розробці і професійних задачах.

## СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. World of Gadgets. URL: <https://worldofgadgets.com.ua/> (дата звернення 12.04.2025).
2. KiwiStore. URL: <https://kiwistore.in.ua/> (дата звернення 12.04.2025)..
3. GadgetShop. URL: <https://gadget-shop.com.ua/ua/?srsltid=AfmBOorPaWgg0PBvkj9nIwxrihHCT2bAHKfIxPqSuw28oyUzIX6TcVkh> (дата звернення 12.04.2025)
4. Bootstrap: <https://getbootstrap.com/docs/5.3/getting-started/introduction/> (дата звернення 07. 04.2025).
5. Tailwind CSS. URL: <https://www.tailwindapp.com/about> (дата звернення 08. 04.2025).
6. Bulma. URL: <https://bulma.io/documentation/> (дата звернення 10.04.2025).
7. Angular. URL: <https://angular.dev/overview> (дата звернення 03.05.2025).
8. Vue.js. URL: <https://vuejs.org/guide/introduction.html> (дата звернення 05.05.2025).
9. Alpine.js. URL: <https://alpinejs.dev/components#integrations> (дата звернення:01.05.2025).
10. Django. URL: <https://docs.djangoproject.com/en/5.2/> (дата звернення: 15.03.2025).
11. Express.js. URL: <https://expressjs.com/en/starter/examples.html> (дата звернення: 15.04.2025).
12. Spring Boot Reference Guide. URL: <https://docs.spring.io/spring-boot/docs/current/reference/html/> (дата звернення: 01.04.2025).
13. MySQL Documentation. URL: <https://dev.mysql.com/doc> (дата звернення: 10.04.2025).
14. PostgreSQL – Документація. URL <https://www.postgresql.org/docs/> (дата звернення: 05.04.2025).
15. SQLite Features. URL: <https://www.sqlite.org/about.html> (дата звернення: 15.04.2025).

16. Azure DevOps. URL: <https://cloudcockpit.com/how-it-works> (дата звернення: 06.05.2025).
17. GitHub. URL: <https://github.com/> (дата звернення 07.03.2025).
18. GitLab. URL: <https://docs.gitlab.com/> (дата звернення: 15.03.2025).
19. MVT. URL: <https://webwizard.ie/blog/understanding-djangos-mvt-architecture-a-beginners-guide-to-models-views-and-templates/> дата звернення: 15.04.2025).
20. Decorator URL: <https://medium.com/@amirm.lavasani/design-patterns-in-python-decorator-c882c0db6501> (дата звернення: 22.04.2025).
21. Observer. URL: <https://refactoring.guru/design-patterns/observer> (дата звернення 20.03.2025).



## ДОДАТОК А

### Макети сторінок вебресурсу

Макети сторінок зображені на рис. А.1–А.3.

Logo	<input type="text" value="Search"/>	Profile	Cart
<div>Login</div> <div>Login Form</div>			
Contact Details	Contact Details	Footer	Contact Details
Contact Details	Contact Details	Contact Details	Contact Details

Рисунок А.1 – Макет сторінки «Вхід у акаунт»

Logo	<input type="text" value="Search"/>	Profile	Cart
<div>Register account</div> <div>Register Form</div>			
Contact Details	Contact Details	Footer	Contact Details
Contact Details	Contact Details	Contact Details	Contact Details

Рисунок А.2 – Макет сторінки «Реєстрація у акаунті»

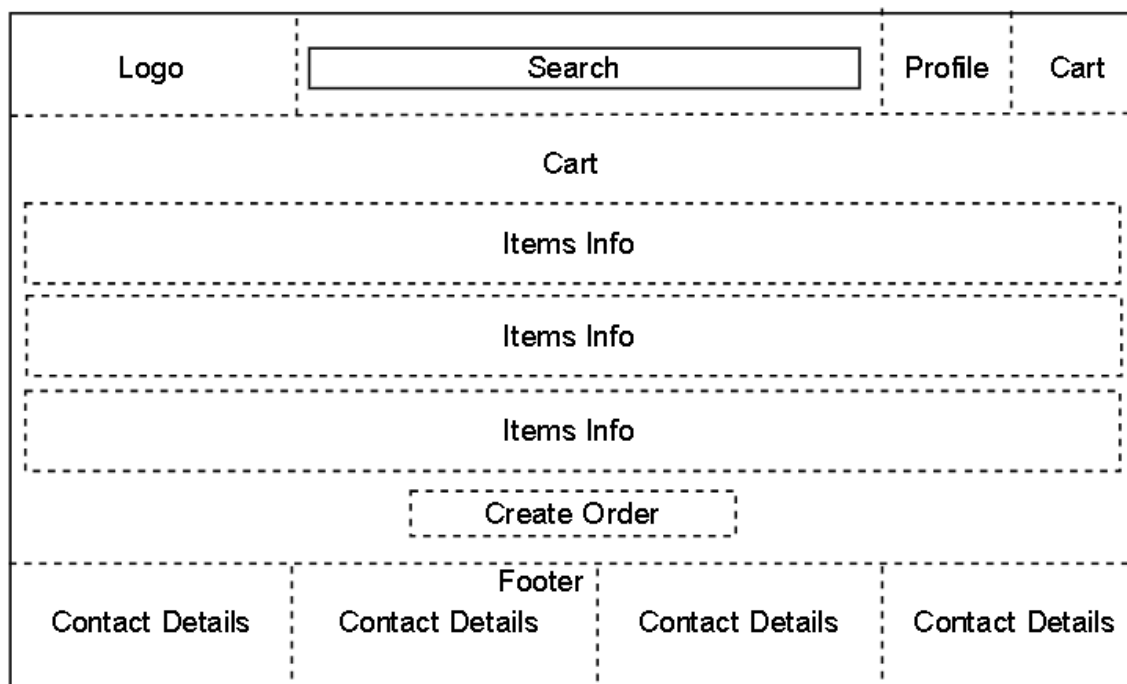


Рисунок А.3 – Макет сторінки «Корзина»

## ДОДАТОК Б

### Інструкція користувача

Інструкція користувача вебресурсу покликана поглибити розуміння сценаріїв взаємодії клієнта з вебсайтом та надати пояснення щодо придбання піротехнічних товарів.

На рис. Б.1 наведений екран головної сторінки вебресурсу з підтримки продажу гаджетів і аксесуарів. На ній ми бачимо, що користувач зараз «Гість», тому йому доступні кнопки вхід в профіль і реєстрація.

Ще на цій сторінці ми бачимо картинку логотипу, натискання на яку здійснює переадресацію на цю головку сторінку, ще з права бачимо картки з товарами, а зліва розташований фільтр, який дозволяє користувачу зручно фільтрувати товари та сортувати їх за різними критеріями і цінами. У кожній картці є зображення товару, його назва, рейтинг, знижка, ціна та кнопка додати у кошик. Щоб дізнатися більше про товар, ми просто натискаємо на картку – і переходимо на сторінку з детальною інформацією. Якщо нам щось сподобалося, ми можемо одразу додати товар до кошика через відповідну кнопку.

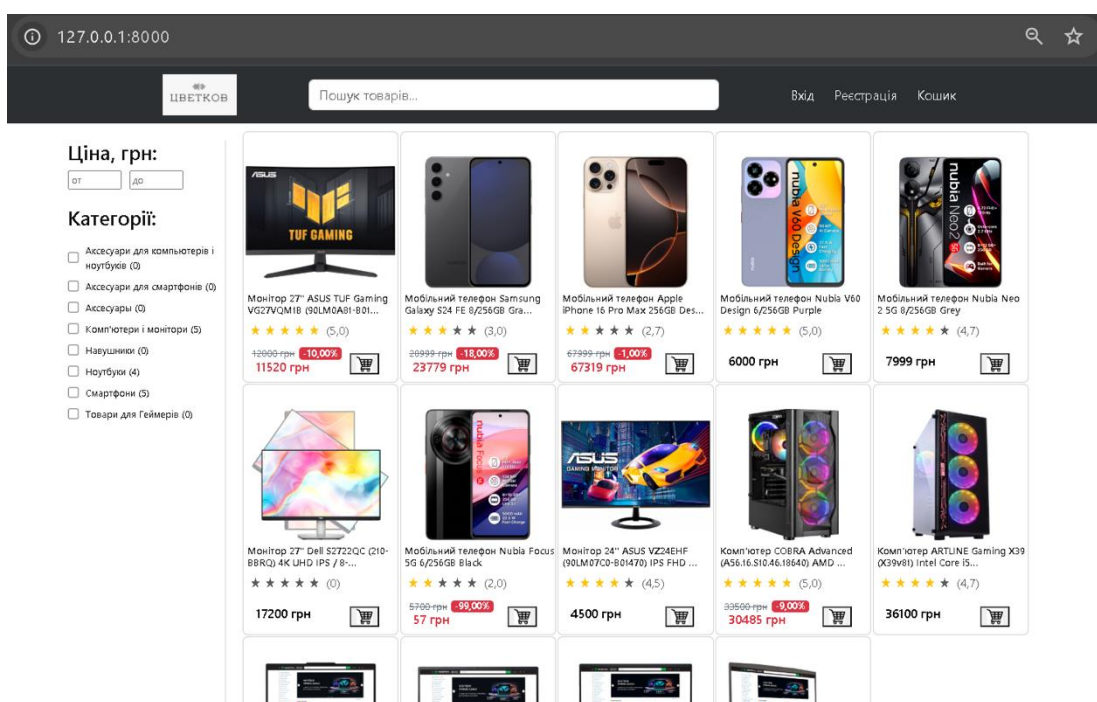


Рисунок Б.1 – Екран головної сторінки з каталогом без авторизації користувача

Далі на рис. Б.2 детальніше розглянемо реєстрацію.

Для того, щоб зареєструвати свій акаунт, користувачу потрібно ввести свій email, пароль і підтвердження пароля.

Для введенних даних є перевірка на валідність:

1) Email повинен мати в собі @ і якийсь існуючий домен (gmail.com, ukr.net або інший), якщо ввести щось інше після @ то з'явиться помилка про неправильний ввід пошти;

2) Пароль не повинен складатися тільки з цифр, тоб-то потрібно ввести і якийсь букви або символи, що забезпечує крипто стійкість і загальну безпеку акаунта користувача;

3) Підтвердження пароля має такіж потреби, як і поле «Пароль», але ще він повинен бути таким-же як і «Пароль», тоб-то мати такуж саму комбінацію чисел, букв і символів.

127.0.0.1:8000/users/register/

«» ЦВЕТКОВ Пошук товарів... Вхід Реєстрація Кошик

### Реєстрація акаунта

Email:  
alexandrtsvetkov@gmail.com  
Введіть ваш email

Пароль:  
Пароль має бути не менш 8 символів і не повинен мати тільки цифри. Введіть якісь букви чи символи.

Підтвердження пароля:  
Введіть той-же самий пароль ще раз для підтвердження.

Зареєструватися Вхід

Телефон	Адреса	Контакти	Соцмережі
+38 (066) 193-51-59	Україна, Одеса	Email: cvetkov12221420@stud.op.edu.ua	Facebook

Рисунок Б.2 – Вікно реєстрації користувача у системі

Після коректного вводу даних, і підтвердження реєстрації, користувач бачить впливаюче модальне вікно (рис. Б.3) з підтвердженням реєстрації і короткий опис отриманих можливостей. Після натискання на хрестик виконується автоматична переадресація на головне вікно з каталогом.

The screenshot shows a web interface for account registration. At the top, there is a dark header bar with a logo 'ЦВЕТКОВ' on the left, a search bar 'Пошук товарів...' in the center, and links 'Профіль' and 'Кошик' on the right. Below the header, the title 'Реєстрація акаунта' is centered. The registration form includes fields for 'Email:' (containing 'alexandrtsvetkov@gmail'), 'Введіть ваш email', 'Пароль:', and 'Підтвердження пароля:'. A modal dialog box titled 'Реєстрацію завершено' is overlaid on the form, displaying the message: 'Ви успішно зареєструвалися! Обов'язково запам'ятайте ваш пароль. Тепер ви можете робити замовлення товарів!'. At the bottom of the form are buttons 'Зареєструватися' and 'Вхід'.

Рисунок Б.3 – Вікно успішної реєстрації користувача у системі

Далі на рис. Б.4 детальніше розглянемо вхід у акаунт.

Для того, щоб здійснити вхід у свій акаунт, користувачу потрібно ввести свій email і пароль.

Для введених даних є перевірка на валідність:

1) Email повинен мати в собі @ і якийсь існуючий домен (gmail.com, ukr.net або інший), якщо ввести щось інше після @ то з'явиться помилка про неправильний ввід пошти;

2) «Пароль» не повинен складатися тільки з цифр, тоб-то потрібно ввести і якийсь букви або символи, що забезпечує крипто стійкість і загальну безпеку користувача.

The top part of the image shows a login window titled 'Вхід у акаунт'. It features a dark header bar with the same elements as Figure B.3. Below the header, there are input fields for 'Email:' and 'Пароль:'. At the bottom are buttons 'Вхід' and 'Реєстрація'. The bottom part of the image shows a table representing a user profile:

Телефон	Адреса	Контакти	Соцмережі
+38 (068) 193-51-59	Україна, Одеса	Email:	Facebook

Рисунок Б.4 – Вікно входу користувача у систему

Після коректного вводу даних, і підтвердження входу, користувач бачить впливаюче вікно (рис. Б.5) з підтвердженням входу і повідомлення про можливість замовлення товарів. Після закриття якого автоматично відбувається переадресація на головне вікно з каталогом.



Рисунок Б.5 – Вікно «Успішний вхід» користувача у систему

Далі на рис. Б.6 детальніше розглянемо сторінку з детальної інформацією про товар.

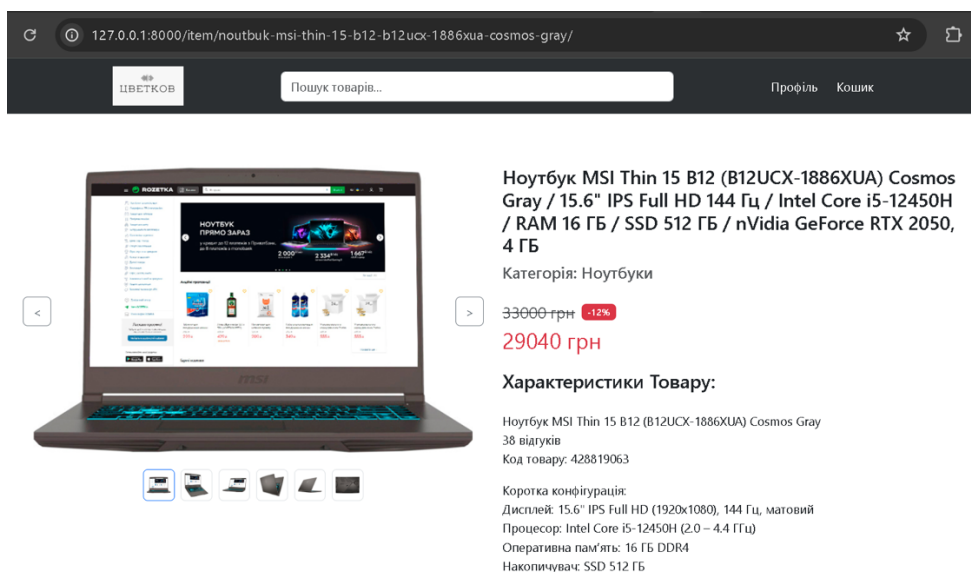


Рисунок Б.6 – Вікно детальної інформації про товар

Знаходячись на головній сторінці вебресурсу і натиснувши на карточку будь-якого товару, система автоматично переводить користувача на сторінку з обраним користувачем товару і показує детальну інформацію.

Фактично ця сторінка складається з двох контейнерів і знизу блок з відгуками.

Лівий контейнер – детальні фотографії вибраного користувачем товару. По бокам фото є перемикачі у ліво і в право на інші фотографії данного товару. Знизу детальних картинок товару додана навігація по картинкам, яка облегшує перехід на необхідне фото, замість того, щоб клацати до потрібного картинки, користувач просто натискає на зменшену копію фото і одразу переходить до нього.

Правий контейнер – назва товару, категорія, ціна до знижки, ціна зі знижкою, сама знижка, повний опис товару і знизу кнопка для додавання цього товару у кошик.

Ця сторінка спроектована так, щоб при перегляді детальної інформації та прокручуванні опису, детальні фото товару і навігація по картинкам залишалися постійно перед очима користувача.

Далі йде блок з відгуками про товар (рис. Б.7), в якому користувач може залишити власний відгук про товар і оцінити його від 1 зірочки до 5 зірочок (1-дуже поганий, а 5 це дуже гарний товар). А може і передивитися відгуки інших користувачів з їх коментарями.

Далі на рис. Б.8 детальніше розглянемо сторінку з доданими товарами у кошик. Якщо користувачу підходить товар і він готовий його купити, то знаходячись на сторінці з детальним описом товару він натискає кнопку «Додати у кошик». Його автоматично переносить на сторінку з кошиком. На цій сторінці відображен список з усіма доданими користувачем товарами. У кожного товару відображається його головна картинка, назва. Ще є кількість товару, котру можна змінювати і через секунду сторінка автоматично оновиться і покаже оновлені данні цін, кількості товарів. Біля нього відображення загальної ціни сумми кількості обраного товару, і так для кожного товару з списку. Ще для кожного товару додана кнопка з видаленням непотрібного товару зі списку кошика.

Залиште відгук про товар:

Name:

Rating:

Comment:

Надіслати

Відгуки користувачів:

- Владислав Паламарчук — ★★★★★

28.05.2025 15:02

Впринципі норм але система охолодження не така продумана як в асус ноутах, якщо любите грати довго і не сильно шумно то це не той ноут, маю асус туф, сис. охолодження по розташуванню і 2 кулера є перевагою
- Юлія Каргар — ★★★★★

28.05.2025 15:01

Вибирали сину ноут для навчання. Син вмовив цю марку і модель взяти. Тепер в стандрф шпилиться, бо ноут ігровий. Так що, батьки, якщо хочете, щоб дитина не робила уроки, а весь час грала - сміливо беріть цю модель!))) Ну а якщо без жартів, то хороший ноутбук і для гри, і для навчання. Поки всім задоволені, сподіваюсь і надалі так буде.
- Олег Гуртовий — ★★★★★

28.05.2025 14:59

Апарат потужний, на мої потреби - топ, АЛЕ гріється (навіть, в налаштуваннях вибрав режим Еко), гріється до того, що сам перезавантажується, або висвічує синій екран.

Рисунок Б.7 – Частина детальної сторінки з блоком «Коментарі»


ЦВЕТКОВ

Пошук товарів...

Профіль Кошик

Кошик


Товари



Монітор 27" ASUS TUF Gaming V627VQM1B (90LMDA81-B01170) -- FullHD / VA / 280Hz / 1 мс / FreeSync Premium / GameFast Input / Shadow Boost / Speakers 2W

34560грн


Видалити товар



Комп'ютер ARTLINE Gaming X39 (X39v81) Intel Core i5-12400F/ RAM 16ГБ / SSD 1ТБ / nVidia GeForce RTX 4060 8ГБ

36100грн

Видалити товар



Ноутбук MSI Thin 15 B12 (B12UCX-1886XUA) Cosmos Gray / 15.6" IPS Full HD 144 Гц / Intel Core i5-12450H / RAM 16 ГБ / SSD 512 ГБ / nVidia GeForce RTX 2050, 4 ГБ

29040грн

Видалити товар

Загальна ціна товарів: 99700 грн

Оформити замовлення

Телефон

Адреса

Контакти

Соцмережі

+38 (069) 193-51-59

Україна, Одеса

Email: cvetkov.12221420@stud.op.edu.ua

Facebook

Рисунок Б.8 – Сторінка зі списком всіх доданих товарів у «Кошик»



З низу цього списку відображається загальна ціна товарів, тоб-то сума всіх одиниць обраних товарів зі списку. Біля нього знаходиться кнопка з оформленням замовлення.

Далі на рис. Б.9 детальніше розглянемо сторінку з оформленням доставки товару.

Ваше замовлення:	
<p>Монітор 27" ASUS TUF Gaming VG27VQM1B (90LM0A81-801170) -- FullHD / VA / 280Hz / 1 мс / FreeSync Premium / GameFast Input / Shadow Boost / Speakers 2W <b>Кількість: 3</b></p>	34560 грн
<p>Комп'ютер ARTLINE Gaming X39 (X39v81) Intel Core i5-12400F / RAM 16ГБ / SSD 1ТБ / nVidia GeForce RTX 4060 8ГБ <b>Кількість: 1</b></p>	36100 грн
<p>Ноутбук MSI Thin 15 B12 (B12UCX-1886XUA) Cosmos Gray / 15.6" IPS Full HD 144 Гц / Intel Core i5-12450H / RAM 16 ГБ / SSD 512 ГБ / nVidia GeForce RTX 2050, 4 ГБ <b>Кількість: 1</b></p>	29040 грн
<b>Загальна вартість: 99700 грн</b>	

Оформлення доставки	
Спосіб доставки:	Нова Пошта
Ім'я та прізвище:	
Телефон:	+380
Email:	@gmail.com
Місто:	Оберіть місто
Відділення:	Оберіть відділення
Загальна вартість (грн):	99700
<b>Оформити замовлення</b>	

Рисунок Б.9 – Сторінка оформлення способу доставки товару до користувача

На цьому рисунку бачимо, що сторінка складається з двох частин.

У лівій частині відображається список з усіма обраними товарами з кошика. Кожен товар має картинку, назву, сумму цін за кожну одиницю товару і виділену жирним шрифтом для наглядності обрану користувачем кількість кожного товару. Знизу пишеться сума всього замовлення.

На правій частині відображається форма з оформленням доставки. Спочатку йде «Спосіб доставки». У цьому пункті користувач може обрати «Нова Пошта», або «Самовивіз» з м. Одеси.

Далі йде поле з ім'ям і фамілією користувача, яке потрібно при відправленні і отриманні замовлення у поштовому відділенні. Введені данні у це поле користувачем проходить перевірку. В це поле не можна ввести англійські букви і цифри, тільки букви з кирилиці.

Далі користувачу потрібно ввести номер телефона, заготовка з кодом країни України вже додана. У полі введені вже додані 3 числа, та користувач повинен додати ще 9 цифр, тоб-то ввести тільки український номер телефона, котрий складається з 12 символів.

Далі йде поле де потрібно заповнити Email користувача для можливості листування. Email повинен мати в собі @ і якийсь існуючий домен (gmail.com, ukr.net або інший), якщо ввести щось інше після @ то з'явиться помилка про неправильний ввід пошти.

Далі йде вибір доступних у Новій Пошті міста доставки, в якому користувач хоче отримати своє замовлення. Потім йде вибір локального поштового відділення, найзручнішого для отримання замовлення користувачем. І останній пункт – це сама загальна сума вартості товарів, які обрав і хоче замовити користувач, це поле не можна змінити.

На рис. Б.10 зображується приклад автоматичної перевірки введених користувачем даних.

### Оформлення доставки

Спосіб доставки:

Нова Пошта

Ім'я та прізвище:

Alex Tsvetkov

Введіть коректне ім'я (укр/рус)

Телефон:

+38012312312

Телефон має бути у форматі +380XXXXXXXXX

Email:

123123wefewfesfsf@gmail.com

Місто:

Абазівка (Полтавський р-н, Полтавська обл)

Відділення:

Пункт приймання-видачі (до 30 кг): вул. Білоуська, 2в

Загальна вартість (грн):

99700

Оформити замовлення

Рисунок Б.10 – Зразок проходження автоматичної перевірки

Після натискання користувачем кнопки Оформити замовлення, воно зберігається і приходить сповіщення про успішне Прийняття замовлення (рис. Б.11). Після цього з користувачем зв'яжеться співробітник магазину, який уточнить правильне введення персональних даних і правильність списку товарів обраних у замовленні.

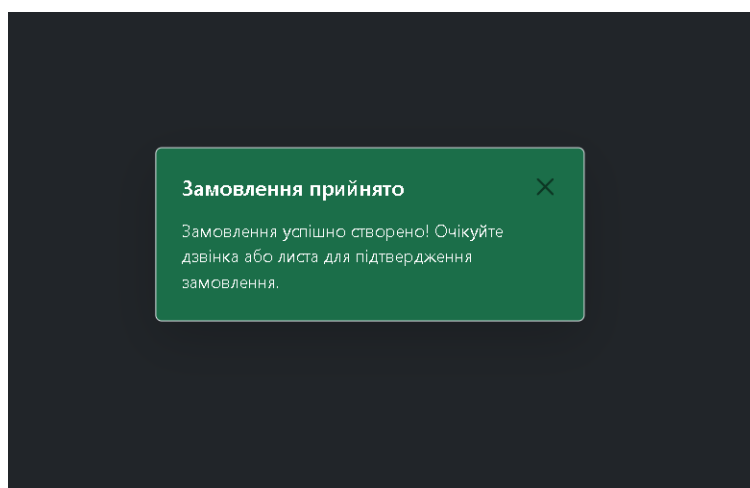


Рисунок Б.11 – Вікно успішного створення замовлення користувачем

Після натискання на хрестик Користувача автоматично переносить на головний екран з каталогом товарів.

## ДОДАТОК В

### Фрагменти програмного коду вебресурсу

#### В.1 Код з файлу `views.py` додатку `main`

```

from django.views.generic import ListView, DetailView
from .models import Item, Category
from django.db.models import Q
from django.shortcuts import render
from django.http import JsonResponse
from .forms import ReviewForm
from django.shortcuts import redirect

class CatalogView(ListView):
    model = Item
    # указываем расположение шаблона
    template_name = 'main/product/catalog.html'
    context_object_name = 'items'
    # собираем сортировки
    def get_queryset(self):
        queryset = super().get_queryset()
        category_slugs = self.request.GET.getlist('category')
        min_price = self.request.GET.get('min_price')
        max_price = self.request.GET.get('max_price')
        if category_slugs:
            queryset = queryset.filter(category__slug__in=category_slugs)
        if min_price:
            queryset = queryset.filter(price__gte=min_price)
        if max_price:
            queryset = queryset.filter(price__lte=max_price)
        return queryset
    # передаем контекст
    def get_context_data(self, **kwargs):
        context = super().get_context_data(**kwargs)
        context['categories'] = Category.objects.all()
        context['selected_categories'] = self.request.GET.getlist('category')
        context['min_price'] = self.request.GET.get('min_price', '')
        context['max_price'] = self.request.GET.get('max_price', '')
        return context
# страница товара
class ItemDetailView(DetailView):

```

```

model = Item
template_name = 'main/product/detail.html'
context_object_name = 'item'
slug_field = 'slug'
slug_url_kwarg = 'slug'
def get_context_data(self, **kwargs):
    context = super().get_context_data(**kwargs)
    item = self.object
    context['form'] = ReviewForm()
    context['reviews'] = item.reviews.order_by('-created_at')
    return context

def post(self, request, *args, **kwargs):
    self.object = self.get_object()
    form = ReviewForm(request.POST)
    if form.is_valid():
        review = form.save(commit=False)
        review.item = self.object
        review.save()
        return redirect(self.request.path_info)
    context = self.get_context_data()
    context['form'] = form
    return self.render_to_response(context)

# Поиск товаров по запросу
def item_search(request):
    query = request.GET.get('q', '')
    items = Item.objects.filter(name__icontains=query) if query else []

    categories = Category.objects.all()
    selected_categories = request.GET.getlist('category')

    return render(request, 'main/product/search_results.html', {
        'products': items,
        'query': query,
        'categories': categories,
        'selected_categories': selected_categories,
    })

def item_autocomplete(request):
    query = request.GET.get('q', '')
    if query:
        items = Item.objects.filter(name__icontains=query)[:10]

```

```

        data = list(items.values('id', 'name'))
    else:
        data = []
    return JsonResponse({'results': data})

```

## В.2 Код з файлу `delivery_form.html` додатку `delivery`

```

{% extends 'main/base.html' %}
{% load static %}

{% block title %}{{ item.name }}{% endblock title %}

<link rel="stylesheet" href="{% static 'css/main.css' %}">

{% block content %}
    <div class="container my-4" x-data="productDetail()">
        <div class="row">
            <!-- Изображения -->
            <div class="col-md-6">
                <div class="sticky-top" style="top: 80px;">
                    <div class="position-relative">
                        <template x-for="(image, index) in images" :key="index">
                            
                        </template>
                    </div>
                    <!-- Кнопки навигации -->
                    <button
                        class="btn btn-outline-secondary position-absolute
top-50 start-0 translate-middle-y"
                        @click="prev"
                        style="z-index: 10;"
                        :disabled="images.length <= 1"
                    ><
                    </button>

                    <button
                        class="btn btn-outline-secondary position-absolute
top-50 end-0 translate-middle-y"
                        @click="next"
                        style="z-index: 10;"
                        :disabled="images.length <= 1"
                    >>
                    </button>
                </div>
            </div>
            <!-- Миниатюрные фотки для переключалки-->
            <div class="d-flex justify-content-center mt-3 gap-2">
                <template x-for="(image, index) in images" :key="index">
                    
                                </template>
                                </div>
                                </div>
</div>
<!-- Описание и детали -->
<div class="col-md-6 d-flex flex-column">

    <h4>{{ item.name }}</h4>
    <h5 class="text-muted">Категорія: {{ item.category.name }}</h5>

    <div class="mt-3 d-flex flex-column">
        {% if item.discount %}
            <div class="d-flex align-items-center gap-2">
                <span class="text-decoration-line-through text-muted
fs-5">{{ item.price|floatformat:0 }} грн</span>
                <span class="badge bg-danger w-auto">-{{
item.discount|floatformat:0 }}%</span>
            </div>
            <div>
                <span class="fs-3 text-danger">{{
item.get_price_with_discount|floatformat:0 }} грн</span>
            </div>
            {% else %}
                <div class="fs-3">{{ item.price|floatformat:0 }} грн</div>
            {% endif %}
        </div>

        <h4 class="mt-3 "> Характеристики Товару: </h4>
        <p >{{ item.description|linebreaks }}</p>

        <form action="{% url 'cart:cart_add' item.id %}" method="post">
            {% csrf_token %}
            <button type="submit" class="btn btn-success mt-3">Додати у
кошик</button>
        </form>

        <template x-if="addedToCart">
            <div class="alert alert-success mt-3" role="alert">
                Товар додан у кошик!
            </div>
        </template>
    </div>
</div>
</div>

<hr class="my-5">

<!-- ОСТАВИТЬ ОТЗЫВ -->
<div class="mt-5">
    <h4>Залиште відгук про товар:</h4>
    <form method="post" class="mt-3">
        {% csrf_token %}
        <div class="mb-3">
            {{ form.name.label_tag }}{{ form.name }}
        </div>
        <div class="mb-3">

```

```

        {{ form.rating.label_tag }}{{ form.rating }}
    </div>
    <div class="mb-3">
        {{ form.comment.label_tag }}{{ form.comment }}
    </div>
    <button type="submit" class="btn btn-primary">Надіслати</button>
</form>
</div>

<!-- Отзывы на товары-->
<div class="mt-5">
    <h4>Відгуки користувачів:</h4>
    {% for review in reviews %}
        <div class="border rounded p-3 mb-3">
            <strong>{{ review.name }}</strong> -
            <span>
                {% for i in "12345"|make_list %}
                    {% if forloop.counter <= review.rating %}
                        <span class="text-warning fs-4">★</span> {# Золотая звезда для
выбранных#}
                    {% else %}
                        <span class="text-muted fs-5">★</span> {# Серая звезда для
невывбранных#}
                    {% endif %}
                {% endfor %}
            </span>
            <div class="text-muted small">{{ review.created_at|date:"d.m.Y
H:i" }}</div>
            <p class="mt-2 mb-0">{{ review.comment }}</p>
        </div>
        {% empty %}
            <p>Поки нема відгуків. Залиште відгук першим!</p>
        {% endfor %}
    </div>

<script>
    function productDetail() {
        return {
            currentIndex: 0,
            images: [
                {% for image in item.images.all %}
                    '{{ image.image.url }}' {% if not forloop.last %}, {% endif
%}
                {% endfor %}
            ],
            addedToCart: false,
            prev() {
                this.currentIndex = (this.currentIndex - 1 +
this.images.length) % this.images.length;
            },
            next() {
                this.currentIndex = (this.currentIndex + 1) %
this.images.length;
            },
            addToCart() {
                this.addedToCart = true;
                setTimeout(() => this.addedToCart = false, 3000);
                $el.querySelector('form').submit();
            }
        }
    }
</script>
{% endblock content %}

```



### В.3 Код з файлу detail.html додатку main

```

<!DOCTYPE html>
<html lang="uk" x-data="deliveryApp({{ total_price|safe }})" x-init="init()">
<head>
  <meta charset="UTF-8">
  <title>Оформлення доставки</title>

  <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css"
rel="stylesheet">

  <style>
    [x-cloak] {
      display: none !important;
    }
  </style>
</head>
<body class="p-4 bg-light">

<script src="https://cdn.jsdelivr.net/npm/alpinejs@3.x.x/dist/cdn.min.js"
defer></script>
<div class="container bg-white p-4 rounded shadow">
  <div class="row">
    <!-- Лівая колонка: товари с корзиноі -->
    <div class="col-md-7 border-end pe-4">
      <h5 class="mb-3 fs-3">Ваше замовлення:</h5>
      <template x-if="cart.length === 0">
        <p>Кошик порожній</p>
      </template>

      {% for item in cart %}
        <div class="mb-3 p-2 border rounded d-flex">
          <div class="me-3">
            
          </div>
          <div class="flex-grow-1">
            <div class="d-flex justify-content-between ">
              <div class="item-name-delivery" style="width:
310px;word-wrap: break-word; ">
                <span>{{ item.item.name }}</span><br>
                <strong>Кількість: {{ item.quantity }}</strong>
              </div>
              <div>
                <strong class="fw-semibold">{{
item.total_price|floatformat:0 }} грн</strong>
              </div>
            </div>
          </div>
        </div>
      {% endfor %}

      <div class="mt-4 fs-5" style="padding-left: 100px ">
        <strong>Загальна вартість: {{ total_price|floatformat:0 }}
грн</strong>
      </div></div>

```

```

<!-- Правая колонка: форма доставки товара -->
<div class="col-md-5 ps-4">
  <h3 class="mb-4">Оформлення доставки</h3>

  <!-- Выбор способа доставки -->
  <div class="mb-3">
    <label class="form-label">Спосіб доставки:</label>
    <select class="form-select" x-model="deliveryMethod">
      <option value="pickup">Самовивіз з м.Одеса проспект Шевченка,
1</option>
      <option value="np">Нова Пошта</option>
    </select>
  </div>

  <!-- Поля для Самовівоза и Новой Почтой -->
  <div class="mb-3">
    <label>Ім'я та прізвище:</label>
    <input class="form-control" x-model="form.full_name">
    <div class="text-danger" x-text="errors.full_name" x-
show="errors.full_name"></div>

  </div>

  <div class="mb-3">
    <label>Телефон:</label>
    <input class="form-control" x-model="form.phone_number">
    <div class="text-danger" x-text="errors.phone_number" x-
show="errors.phone_number"></div>
  </div>

  <div class="mb-3">
    <label>Email:</label>
    <input class="form-control" x-model="form.email" type="email">
    <div class="text-danger" x-text="errors.email" x-
show="errors.email"></div>
  </div>

  <!-- Только если выбрана доставка Новой Почтой -->
  <template x-if="deliveryMethod === 'np'">
    <div>
      <div class="mb-3">
        <label>Micro:</label>
        <select class="form-select" x-model="selectedCity"
@change="loadWarehouses">
          <option value="">Оберіть micro</option>
          <template x-for="city in cities" :key="city.Ref">
            <option :value="city.Ref" x-
text="city.Description"></option>
            <div class="text-danger" x-text="errors.city" x-
show="errors.city"></div>
          </template>
        </select>
      </div>

      <div class="mb-3">
        <label>Відділення:</label>
        <select class="form-select" x-model="form.warehouse">
          <option value="">Оберіть відділення</option>
          <template x-for="w in warehouses" :key="w.Ref">
            <option :value="w.Description" x-
text="w.Description"></option>
            <div class="text-danger" x-text="errors.warehouse"
x-show="errors.warehouse"></div>

```

```

        </template>
      </select>
    </div>
  </div>
</template>

<!-- Стоимость -->
<div class="mb-3">
  <label>Загальна вартість (грн):</label>
  <input class="form-control" type="number" x-
model="form.total_price" readonly>
</div>

<button class="btn btn-success" @click="submitOrder">Оформити
заовлення</button>

<!-- Модальное окно ,если успешно создан заказ пользователем -->
<div x-show="orderSuccess" x-transition
  class="position-fixed top-50 start-50 translate-middle border
rounded shadow-lg p-4 bg-success bg-opacity-75 text-white"
  style="z-index: 1055; width: 90%; max-width: 400px;" x-cloak>

  <div class="d-flex justify-content-between align-items-center mb-
3">
    <h5 class="m-0">Замовлення прийнято</h5>
    <button @click="orderSuccess = false;
window.location.href='/'" type="button"
      class="btn-close" aria-label="Закрити"></button>
  </div>
  <p class="mb-0">Замовлення успішно створено! Очікуйте дзвінка або
листа для підтвердження
    замовлення.</p>
</div>
</div>

<!-- Тёмный фон при успешном создании и объявление пользователю -->
<div x-show="orderSuccess" x-transition.opacity
  class="position-fixed top-0 start-0 w-100 h-100 bg-dark bg-opacity-
80"
  style="z-index: 1050;" x-cloak>
</div>

<script>
function deliveryApp(totalPrice) {
  return {
    deliveryMethod: 'np',
    cities: [],
    warehouses: [],
    selectedCity: '',
    orderSuccess: false,
    errors: {},
    form: {
      full_name: '',
      phone_number: '+380',
      email: '@gmail.com',
      warehouse: '',
      total_price: totalPrice
    },
    init() {
      fetch('/delivery/api/get_cities/')
        .then(res => res.json())
        .then(data => this.cities = data.data);
    }
  };
}

```

```

    },
    loadWarehouses() {

fetch(`/delivery/api/get_warehouses/?cityRef=${this.selectedCity}`)
    .then(res => res.json())
    .then(data => this.warehouses = data.data);
    },
    validateForm() {
        this.errors = {};
        const nameRegex = /^[A-Яa-яІіїіЄєІі\ś'-]{2,}$/u;
        const phoneRegex = /^\\+380\\d{9}$/;
        const emailRegex = /^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\\. [a-
zA-Z]{2,}$/;

        if (!nameRegex.test(this.form.full_name))
            this.errors.full_name = 'Введіть коректне ім'я
(укр/рус)';

        if (!phoneRegex.test(this.form.phone_number))
            this.errors.phone_number = 'Телефон має бути у форматі
+380XXXXXXXXXX';

        if (!emailRegex.test(this.form.email))
            this.errors.email = 'Невірний формат email';
        if (this.deliveryMethod === 'np' && !this.selectedCity)
            this.errors.city = 'Оберіть місто';
        if (this.deliveryMethod === 'np' && !this.form.warehouse)
            this.errors.warehouse = 'Оберіть відділення';

        return Object.keys(this.errors).length === 0;
    },
    submitOrder() {
        if (!this.validateForm()) return;

        const csrfToken = document.querySelector('meta[name="csrf-
token"]').getAttribute('content');
        const payload = {
            ...this.form,
            city: this.selectedCity,
            delivery_method: this.deliveryMethod
        };

        fetch('/delivery/api/submit_order/', {
            method: 'POST',
            headers: {
                'Content-Type': 'application/json',
                'X-CSRFToken': csrfToken
            },
            credentials: 'same-origin',
            body: JSON.stringify(payload)
        })
            .then(res => {
                if (!res.ok) throw new Error('Щось пішло не так');
                return res.json();
            })
            .then(data => {
                if (data.success) this.orderSuccess = true;
            })
            .catch(error => {
                console.error('Помилка при відправці замовлення:',
error);
            });
    }
});
    }
}

</script>
</body></html>

```