

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

ПОЯСНЮВАЛЬНА ЗАПИСКА

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

на тему:

«Розробка додатку інтернет магазину Case Space»

Виконав студент групи ЗПЗк-23-1
МАРЧУК Максим Дмитрович

Керівник роботи:
ЄФРЕМОВ Юрій Миколайович

Рецензент:
ВЛАСЕНКО Олег Васильович

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

«ЗАТВЕРДЖУЮ»

Зав. кафедри інженерії
програмного забезпечення
Тетяна Вакалюк

«26» лютого 2025 р.

ЗАВДАННЯ
на кваліфікаційну роботу

Здобувач вищої освіти: **Марчук Максим Дмитрович**

Керівник роботи: **Єфремов Юрій Миколайович**

Тема роботи: **«Розробка додатку інтернет магазину Case Space»**,
затверджена Наказом закладу вищої освіти від **«14» лютого 2025 р., №62/с**

Вихідні дані для роботи: **методи та технології створення веб-додатків для електронної комерції, аналіз, синтез, порівняння, моделювання.**

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

Розділ	Консультант	Завдання видав	Завдання прийняв
1	Єфремов Ю.М.	27.02.2025р.	27.02.2025р.
2	Єфремов Ю.М.	27.02.2025р.	27.02.2025р.
3	Єфремов Ю.М.	27.02.2025р.	27.02.2025р.

РЕФЕРАТ

Випускна кваліфікаційна робота бакалавра складається з розробки додатку інтернет магазину «Case Space» та пояснювальної записки. Пояснювальна записка до випускної роботи містить 78 сторінок, 59 ілюстрацій, 2 таблиці та 30 використаних джерел.

Метою роботи є розробка інтернет-магазину чохлів для телефонів «Case Space», який забезпечує зручний доступ до асортименту, швидке оформлення замовлень, використовуючи сучасний технологічний стек.

Архітектура програмного продукту побудована на сучасних веб-технологіях, а саме Django, DRF, React, Zustand та Mantine. Технічні можливості фреймворку Django в поєднанні з Django REST Framework дозволяють втілити просунуту систему фільтрів для надання користувачу можливості швидкого пошуку товару за його вподобанням.

Програмний продукт орієнтується на забезпечення швидкого, зручного та сучасного користувацького досвіду використання додатку за допомогою використання можливостей фреймворку Mantine.

КЛЮЧОВІ СЛОВА: ВЕБДОДАТОК, API, ІНТЕРНЕТ-МАГАЗИН, REST, REACT.

					ІПЗ.КР.Б – 121 – 25 – ПЗ				
Змн.	Арк.	№ докум.	Підпис	Дата					
Розроб.		Марчук М.Д.			Розробка додатку інтернет магазину Case Space	Літ.		Арк.	Аркушів
Керівник		Єфремов Ю.М.						3	78
						Житомирська політехніка, група ЗІПЗк-23-1			
Н. Контр.		Тичина О.П.							
Зав. каф.		Вакалюк Т.А.							

ABSTRACT

The bachelor's thesis consists of the development of the Case Space online store application and an explanatory note. The explanatory note to the graduation thesis contains 78 pages, 59 illustrations, 2 tables and 30 references.

The aim of the work is to develop an online store of phone cases “Case Space”, which provides convenient access to the assortment, fast ordering, using a modern technological stack.

The architecture of the software product is based on modern web technologies, namely Django, DRF, React, Zustand, and Mantine. The technical capabilities of the Django framework in combination with the Django REST Framework allow implementing an advanced filter system to provide the user with the ability to quickly search for a product of his preference.

The software product is focused on providing a fast, convenient and modern user experience of using the application by using the capabilities of the Mantine framework.

KEYWORDS: WEB APPLICATION, API, E-COMMERCE, REST, REACT.

					ІІЗ.КР.Б – 121 – 25 – ІІЗ	Арк.
						4
Змн.	Арк.	№ докум.	Підпис	Дата		

ЗМІСТ

РЕФЕРАТ.....	3
ABSTRACT	4
ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ	6
ВСТУП.....	7
РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ.....	9
1.1. Визначення предметної області та її особливості	9
1.2. Огляд аналогів	12
Висновок до розділу 1	18
РОЗДІЛ 2. ТЕХНІЧНЕ ЗАБЕЗПЕЧЕННЯ ТА ПРОЄКТУВАННЯ СИСТЕМИ.....	19
2.1. Обґрунтування вибору мови програмування	19
2.2. Обґрунтування вибору React, Vite, Zustand та Mantine для фронтенду	24
2.3. Розробка архітектури системи.....	28
Висновок до розділу 2	35
РОЗДІЛ 3. ВПРОВАДЖЕННЯ ТА ОЦІНКА РОБОТИ ІНТЕРНЕТ-МАГАЗИНУ ЧОХЛІВ ДЛЯ ТЕЛЕФОНІВ CASE SPACE	37
3.1. Опис фрагментів коду розробки інтернет-магазину	37
3.2. Демонстрація результатів роботи розробленого інтернет-магазину Case Space	67
Висновок до розділу 3	72
ВИСНОВКИ	74
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	76
ДОДАТКИ.....	79

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

API – прикладний програмний інтерфейс

REST – Representational State Transfer

DRF – Django REST Framework

JSON – JavaScript Object Notation

JWT – JSON Web Token

URL – Уніфікований локатор ресурсів (Uniform Resource Locator)

HTTP – Hypertext Transfer Protocol

ORM – Об'єктно-реляційне відображення (Object-Relational Mapping)

CSRF – Міжсайтова підробка запиту (Cross-Site Request Forgery)

XSS – Міжсайтовий скриптинг (Cross-Site Scripting)

SQL – Structured Query Language

UI – Дизайн інтерфейсу користувача (User Interface)

OAuth2 – Open Authorization v2.0

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						6
Змн.	Арк.	№ докум.	Підпис	Дата		

ВСТУП

Актуальність теми. Сучасний розвиток інформаційних технологій та зростання популярності електронної комерції роблять актуальним створення зручних і функціональних інтернет-магазинів. Ринок аксесуарів для мобільних телефонів, зокрема чохлів, демонструє стабільне зростання завдяки широкому розповсюдженню смартфонів. За даними аналітичних звітів, в 2024 році об’єм світового ринку аксесуарів для телефонів перевищив 40 мільярдів доларів США, значну частку якого займають чохла. В Україні попит на такі товари також зростає, що пояснюється прагненням користувачів захистити свої пристрої та персоналізувати їх за допомогою стильних та унікальних аксесуарів.

Інтернет-магазини є основним каналом продажу чохлів, оскільки забезпечують зручність вибору, доступність асортименту та швидкість доставки. Проте багато існуючих платформ мають недоліки, такі як незручна навігація, обмежені можливості фільтрації або недостатня підтримка персоналізації. В цьому контексті розробка інтернет-магазину «Case Space», орієнтованого на продаж чохлів для телефонів, є актуальною. Такий проєкт дозволяє поєднати сучасні технології веб-розробки з потребами ринку, пропонуючи користувачам інтуїтивно зрозумілий інтерфейс, широкий асортимент та підтримку трендів, таких як екологічність матеріалів і кастомізація дизайну. Крім того, створення локального інтернет-магазину сприяє розвитку українського ринку аксесуарів, пропонуючи конкурентоспроможні рішення з швидкою доставкою та адаптацією до потреб місцевих споживачів.

Метою кваліфікаційної роботи є розробка інтернет-магазину чохлів для телефонів «Case Space», який забезпечує зручний доступ до асортименту, швидке оформлення замовлень, використовуючи сучасний технологічний стек на основі Python (Django/DRF), PostgreSQL, React, Vite, Zustand і Mantine.

Для досягнення поставленої мети визначено такі завдання:

- 1. Провести аналіз предметної області, включаючи дослідження ринку чохлів для телефонів і огляд існуючих інтернет-магазинів.

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						7
Змн.	Арк.	№ докум.	Підпис	Дата		

2. Обґрунтувати вибір мови програмування Python і фреймворків Django/DRF для бекенду, а також React, Vite, Zustand і Mantine для фронтенду.
3. Розробити архітектуру системи з використанням UML-діаграм для моделювання її функціоналу та структури.
4. Реалізувати серверну частину інтернет-магазину з використанням Django/DRF і PostgreSQL.
5. Створити клієнтську частину з сучасним інтерфейсом, використовуючи React і бібліотеки Zustand і Mantine.
6. Протестувати функціонал системи.

Об’єктом дослідження є процес розробки інтернет-магазинів.

Предметом дослідження є методи та технології створення веб-додатків для електронної комерції.

Методами дослідження є аналіз, синтез, порівняння, моделювання.

Структура роботи складається із вступу, трьох основних розділів роботи, висновків, списку використаної літератури, додатків.

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						8
Змн.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1. Визначення предметної області та її особливості

Предметна область дослідження охоплює сферу електронної комерції, зокрема діяльність інтернет-магазинів, які займаються продажем різноманітних товарів, включаючи аксесуари для мобільних телефонів, такі як чохли. Інтернет-магазини стали ключовим елементом сучасної торгівлі, забезпечуючи зручний доступ до широкого асортименту продукції, можливість порівняння цін та швидке оформлення замовлень. Зростання популярності смартфонів і попиту на пов'язані з ними аксесуари підкреслює важливість аналізу інтернет-магазинів як платформ для реалізації таких товарів [1].

Інтернет-магазини відіграють центральну роль в сучасній економіці, дозволяючи споживачам купувати товари без необхідності відвідувати фізичні торговельні точки. Вони забезпечують доступ до різноманітної продукції, від побутових товарів до спеціалізованих аксесуарів, таких як чохли для телефонів, в будь-який час і з будь-якого місця. За даними аналітичних звітів, в 2024 році об'єм світового ринку електронної комерції перевищив 6 трильйонів доларів США, що свідчить про стабільне зростання цього сектору. В Україні електронна комерція також набирає обертів завдяки широкому проникненню інтернету та зростанню кількості користувачів смартфонів, що сприяє попиту на аксесуари, зокрема чохли. Ці платформи пропонують зручність, широкий вибір і можливість швидкої доставки, що робить їх привабливими для сучасних споживачів [3].

Інтернет-магазини, які продають різноманітні товари, можна умовно поділити на кілька категорій, кожна з яких має свої особливості. Перша категорія – це маркетплейси, такі як Rozetka в Україні чи AliExpress на міжнародному ринку. Вони об'єднують продукцію від численних продавців, пропонуючи широкий асортимент товарів, включаючи чохли для телефонів, електроніку, одяг та побутові товари. Такі платформи приваблюють споживачів завдяки

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						9
Змн.	Арк.	№ докум.	Підпис	Дата		

різноманітності пропозицій і конкурентним цінам. Однак через велику кількість товарів інтерфейс може бути перевантаженим, що ускладнює пошук специфічної продукції, наприклад, чохлів для певної моделі телефону. Крім того, якість товарів на маркетплейсах може варіюватися через різницю в надійності продавців, а міжнародні платформи, такі як AliExpress, часто стикаються з проблемою тривалої доставки.

Друга категорія – спеціалізовані інтернет-магазини, які зосереджені на певних групах товарів, наприклад, аксесуарах для телефонів. Прикладами є Spigen або Case-Mate, які пропонують чохли, захисні плівки та інші аксесуари. Такі магазини вирізняються високою якістю продукції та детальними описами товарів, що полегшує вибір для споживачів, які шукають надійні рішення. Однак їхній асортимент зазвичай обмежений порівняно з маркетплейсами, а ціни можуть бути вищими через фокус на преміум-продукцію. Крім того, такі платформи рідко пропонують можливості для персоналізації, що може бути недоліком для користувачів, які хочуть унікальний дизайн [4].

Третя категорія – магазини брендів, такі як офіційні сайти Samsung чи Apple, які продають аксесуари, сумісні з їх продукцією. Ці платформи гарантують оригінальність товарів і високу сумісність з пристроями бренду, що викликає довіру в споживачів. Проте асортимент таких магазинів обмежений продукцією одного виробника, а ціни часто є вищими через преміум-позиціонування. Крім того, вони рідко пропонують товари з індивідуальним дизайном, що може не задовольняти потреби користувачів, які шукають унікальність.

Четверта категорія – нішові платформи, які зосереджені на унікальних або персоналізованих товарах, наприклад, чохлах із індивідуальним дизайном або екологічних матеріалах. Такі магазини приваблюють споживачів, які цінують оригінальність, але їхній асортимент зазвичай вузький, ніж в маркетплейсах, а просування таких платформ потребує значних зусиль через меншу впізнаваність бренду.

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						10
Змн.	Арк.	№ докум.	Підпис	Дата		

Цільова аудиторія інтернет-магазинів, що продають чохли для телефонів, є різноманітною і включає кілька ключових сегментів. Молодь віком від 16 до 30 років віддає перевагу яскравим і модним дизайнам, часто шукаючи можливість персоналізації через унікальні малюнки чи написи. Ця група активно використовує соціальні мережі для пошуку трендових товарів і цінує швидкий доступ до асортименту. Дорослі користувачі віком від 30 до 50 років зазвичай обирають практичні чохли, які забезпечують надійний захист і довговічність. Бізнес-аудиторія віддає перевагу елегантним чохлам з преміум-матеріалів, таких як шкіра чи метал, що відповідають діловому стилю. Екосвідомі споживачі, незалежно від віку, шукають продукцію з біорозкладних матеріалів, таких як бамбук чи перероблений пластик, що відображає сучасний тренд на екологічність. Ці різні потреби аудиторії формують вимоги до асортименту та функціоналу інтернет-магазинів, які мають бути гнучкими, щоб задовольнити різні смаки [7].

Ринок електронної комерції зазнає впливу кількох ключових тенденцій, які впливають на розвиток інтернет-магазинів. Однією з основних є зростання мобільної комерції, коли значна частка покупок здійснюється через смартфони. Це підкреслює важливість адаптивного дизайну, швидкого завантаження сторінок і зручної навігації на мобільних пристроях. Ще однією тенденцією є попит на персоналізацію, коли споживачі прагнуть купувати товари з унікальним дизайном, наприклад, чохли з власними малюнками чи написами. Екологічність також набуває значення, оскільки дедалі більше покупців обирають продукцію з перероблених або біорозкладних матеріалів. Інтеграція з сучасними технологіями, такими як безконтактні платежі через Apple Pay чи Google Pay, стає стандартом, що полегшує процес оплати. В контексті українського ринку швидка доставка через співпрацю з локальними службами, такими як «Нова Пошта» чи «Укрпошта», є ключовим фактором конкурентоспроможності.

Конкуренція в сфері електронної комерції є високою. Великі маркетплейси, такі як Rozetka чи AliExpress, домінують завдяки широкому асортименту та низьким цінам, але стикаються з проблемами перевантаженого

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						11
Змн.	Арк.	№ докум.	Підпис	Дата		

інтерфейсу та нерівномірної якості товарів. Спеціалізовані магазини, такі як Spigen чи Case-Mate, пропонують якісну продукцію, але обмежені в асортименті та можливостях персоналізації. Магазини брендів, такі як офіційні сайти Apple чи Samsung, забезпечують довіру, але їх ціни та обмежений вибір можуть відштовхувати частину споживачів. Нішові платформи приваблюють унікальними пропозиціями, але потребують значних зусиль для залучення аудиторії. Для успіху спеціалізованого магазину, такого як «Case Space», необхідно поєднати широкий асортимент, швидку доставку, локальну адаптацію та унікальні функції, такі як персоналізація, щоб вирізнитися на тлі конкурентів.

1.2. Огляд аналогів

Сфера електронної комерції в 2024 році залишається одним із найдинамічніших секторів економіки, забезпечуючи продаж різноманітних товарів, від побутової техніки до аксесуарів для мобільних телефонів, таких як чохли. Інтернет-магазини, як в Україні, так і в світі, є основним каналом торгівлі, що пропонує зручність, широкий асортимент і швидку доставку. Детальний огляд їхнього функціоналу, сильних і слабких сторін дозволяє зрозуміти сучасні стандарти електронної комерції та визначити ключові аспекти для створення конкурентоспроможного інтернет-магазину «Case Space» [6].

Світовий ринок електронної комерції в 2024 році досяг обсягу 6.3 трильйона доларів США, що на 8% більше порівняно з 2023 роком, за даними Statista. В Україні, попри воєнні виклики, сектор онлайн-торгівлі зріс на 25%, з загальним обсягом продажів у 239 мільярдів гривень, що становить 10% від загальної роздрібної торгівлі. Кількість онлайн-покупців в Україні сягнула 11 мільйонів, з середньою частотою покупок 17 разів на рік на суму 1300 гривень. Аксесуари для телефонів, зокрема чохли, є однією з популярних категорій завдяки попиту серед молоді, ділових користувачів і екосвідомих споживачів. Аналіз інтернет-магазинів, що продають такі товари, розкриває їхні особливості та допомагає сформулювати стратегію для «Case Space».

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						12
Змн.	Арк.	№ докум.	Підпис	Дата		

Український ринок електронної комерції представлений кількома ключовими гравцями, серед яких Rozetka є найбільшим маркетплейсом. Rozetka пропонує широкий асортимент товарів, від електроніки до чохлів для телефонів, об'єднуючи продукцію від тисяч продавців. Платформа забезпечує зручний пошук з фільтрами за брендом телефону, матеріалом чи ціною, а також детальні описи товарів із відгуками користувачів. Користувачі можуть оформити замовлення через кошик, вибрати спосіб доставки («Нова Пошта», «Укрпошта») і оплатити через LiqPay або карткою. Перевагою Rozetka є швидка доставка в межах України, часто протягом 1-2 днів, і конкурентні ціни завдяки великому обсягу продажів. Проте інтерфейс платформи може бути перевантаженим через величезну кількість товарів, що ускладнює пошук специфічних чохлів. Крім того, якість товарів залежить від продавця, що може призводити до розчарувань в покупців, які очікують стабільної якості [26].

Інший український маркетплейс, Prom.ua, також об'єднує товари від різних продавців, але більше орієнтований на малий і середній бізнес. Платформа пропонує чохли для телефонів в різних цінових сегментах, з можливістю порівняння пропозицій від кількох постачальників. Prom.ua забезпечує базові функції, такі як пошук, фільтри та оформлення замовлень, але його інтерфейс менш сучасний порівняно з Rozetka. Перевагою є можливість знайти унікальні товари від локальних виробників, наприклад, чохли ручної роботи. Однак платформа страждає від нерівномірної якості обслуговування, оскільки продавці мають різний рівень професіоналізму. Доставка може займати більше часу, якщо продавець розташований у віддаленому регіоні, що знижує конкурентоспроможність порівняно з Rozetka [27].

На міжнародному рівні Amazon є провідним маркетплейсом, що пропонує мільйони товарів, включаючи чохли для телефонів від відомих брендів, таких як Spigen, OtterBox, і бюджетних виробників. Платформа вирізняється потужною системою пошуку з розширеними фільтрами, рекомендаціями на основі попередніх покупок та детальними відгуками. Amazon забезпечує швидку доставку в багатьох країнах, інтеграцію з платіжними системами (Visa, PayPal) і

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						13
Змн.	Арк.	№ докум.	Підпис	Дата		

підтримку клієнтів через чат. Перевагою є глобальна присутність і довіра до бренду, а також програма Amazon Prime, яка пропонує безкоштовну доставку для передплатників. Однак для українських користувачів доставка з Amazon може займати 10-20 днів і супроводжуватися додатковими митними зборами, що знижує привабливість платформи. Крім того, величезний асортимент може ускладнювати вибір для користувачів, які шукають конкретний товар [9].

AliExpress, ще один міжнародний маркетплейс, популярний в Україні завдяки низьким цінам і широкому вибору чохлів для телефонів, від бюджетних силіконових до персоналізованих з унікальними дизайнами. Платформа пропонує базові функції пошуку, фільтрації та оформлення замовлень, а також відгуки з фотографіями від покупців, що допомагають оцінити якість. Перевагою AliExpress є доступність товарів, які важко знайти в локальних магазинах, і можливість персоналізації через сторонніх продавців. Однак доставка з AliExpress часто займає 2-6 тижнів, а якість товарів може бути непередбачуваною через низький контроль продавців. Платформа також має обмежену підтримку клієнтів, що ускладнює вирішення спорів [11].

Спеціалізовані магазини, такі як Spigen, фокусуються на аксесуарах для телефонів, пропонуючи чохли преміум-класу, відомі своєю міцністю та дизайном. Сайт Spigen має зручний інтерфейс з фільтрами за моделлю телефону, типом чохла (захисний, протиударний) і матеріалом. Кожен товар супроводжується детальним описом, технічними характеристиками та якісними зображеннями. Перевагою є висока якість продукції та сумісність з конкретними моделями, що викликає довіру в користувачів, які цінують надійність. Проте асортимент Spigen обмежений порівняно з маркетплейсами, а ціни вищі, що може відштовхувати економних покупців. Доставка в Україну може займати до двох тижнів, а персоналізація чохлів не підтримується, що є недоліком для молодшої аудиторії.

Case-Mate, ще один спеціалізований магазин, пропонує стильні чохли з акцентом на модний дизайн, часто з декоративними елементами, такими як стрази чи металеві вставки. Сайт має адаптивний дизайн, зручний пошук і

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						14
Змн.	Арк.	№ докум.	Підпис	Дата		

функцію оформлення замовлень з підтримкою міжнародних платіжних систем. Перевагою є інтеграція з соціальними мережами, що дозволяє ділитися дизайнами, і підтримка персоналізації для окремих моделей. Однак ціни на Case-Mate вищі за середні, а асортимент обмежений декоративними чохлами, що не відповідає потребам користувачів, які шукають протиударні моделі [2]. Доставка в Україну є дорогою та тривалою, що знижує привабливість для локальних споживачів. Порівняння інтернет-магазинів узагальнено в таблиці 1.1

Таблиця 1.1

Порівняння функціоналу та характеристик інтернет-магазинів

Інтернет-магазин	Функціонал	Переваги	Недоліки	Частка ринку (%)
Rozetka	Пошук, фільтри, відгуки, швидка доставка	Широкий асортимент, швидка доставка	Перевантажений інтерфейс, нерівна якість	45 (Україна)
Prom.ua	Пошук, порівняння продавців, фільтри	Унікальні товари, низькі ціни	Нерівна якість, повільна доставка	20 (Україна)
Amazon	Розширені фільтри, рекомендації, чат	Глобальна присутність, якість	Довга доставка в Україну, мито	30 (світ)
AliExpress	Пошук, відгуки з фото, персоналізація	Низькі ціни, унікальні товари	Довга доставка, низька якість	15 (світ)
Spigen	Фільтри за моделлю, детальні описи	Висока якість, сумісність	Високі ціни, обмежений асортимент	5 (світ)
Case-Mate	Пошук, персоналізація, соцмережі	Стильний дизайн, персоналізація	Високі ціни, довга доставка	3 (світ)

Ця таблиця порівнює ключові інтернет-магазини за їхнім функціоналом, перевагами, недоліками та часткою ринку. Rozetka домінує в Україні завдяки швидкій доставці та широкому асортименту, тоді як Amazon лідирує на світовому рівні через глобальну присутність. Спеціалізовані магазини, як Spigen і Case-Mate, мають меншу частку, але приваблюють якістю та дизайном.

Переваги інтернет-магазинів значною мірою залежать від їх типу. Rozetka виграє завдяки швидкій доставці та широкому асортименту, що відповідає потребам українських споживачів, які цінують локальну адаптацію. Prom.ua приваблює економних покупців і тих, хто шукає унікальні товари від малих виробників. Amazon забезпечує глобальну доступність і довіру, що робить його вибором для користувачів в розвинених країнах. AliExpress пропонує низькі ціни та персоналізацію, що приваблює молодь. Spigen і Case-Mate вирізняються якістю та дизайном, але їх аудиторія обмежена тими, хто готовий платити більше за преміум-продукцію [8].

Недоліки також варіюються. Rozetka і Prom.ua стикаються з проблемами нерівномірної якості через різних продавців, а їх інтерфейси можуть бути складними для швидкого пошуку. Amazon і AliExpress мають тривалу доставку в Україну, що знижує їх привабливість для локальних користувачів. Spigen і Case-Mate обмежені в асортименті та не пропонують достатньої персоналізації, що важливо для сучасного ринку.

Ринкові тенденції 2024 року формують вимоги до інтернет-магазинів. Мобільна комерція домінує, з 70% покупок через смартфони, що вимагає адаптивного дизайну та швидкого завантаження. Попит на персоналізацію, наприклад, чохли з унікальними дизайнами, зростає серед молоді. Екологічність набуває популярності, з попитом на товари з перероблених матеріалів. Інтеграція з соціальними мережами, як Instagram, посилює продажі через інфлюенс-маркетинг. В Україні швидка доставка через локальні служби є ключовою конкурентною перевагою. Ринкові показники інтернет-магазинів наведено в таблиці 1.2

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						16
Змн.	Арк.	№ докум.	Підпис	Дата		

Таблиця 1.2

Ринкові показники інтернет-магазинів

Інтернет-магазин	Обсяг продажів (млрд)	Кількість користувачів (млн)	Частота покупок на рік
Rozetka	108 (грн)	8	20
Prom.ua	48 (грн)	5	15
Amazon	2800 (дол. США)	310	25
AliExpress	900 (дол. США)	200	18
Spigen	50 (дол. США)	10	10
Case-Mate	30 (дол. США)	5	8

Представлена частка ринку інтернет-магазинів в Україні (рис. 1.1)

Частка ринку інтернет-магазинів в Україні (2024)

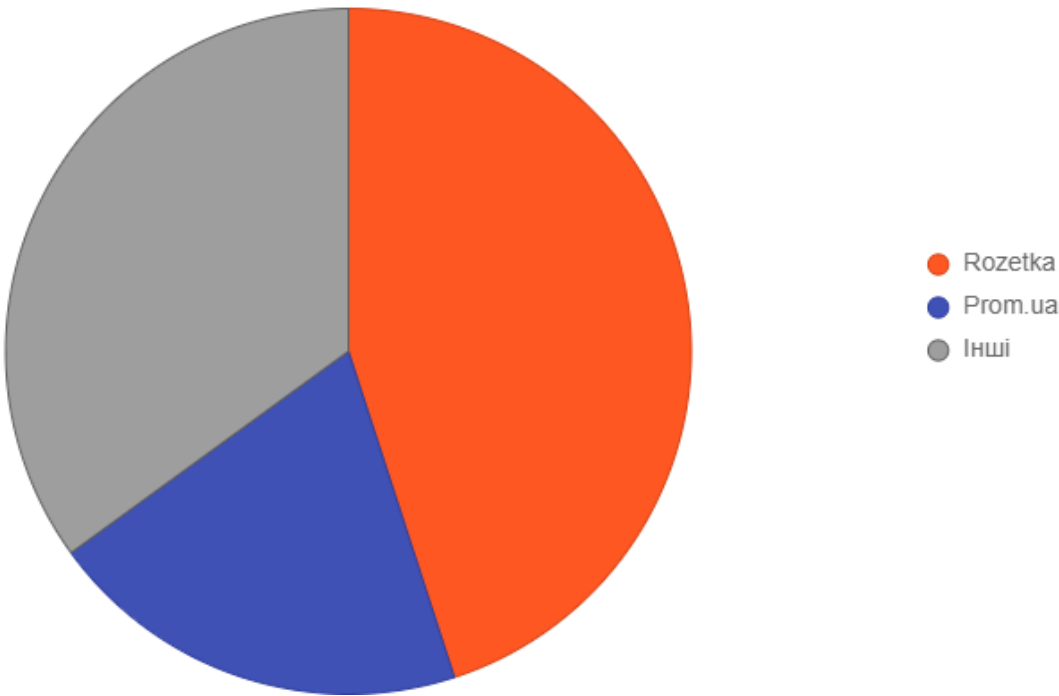


Рис. 1.1 – Частка ринку інтернет-магазинів

Ця діаграма показує розподіл частки ринку в Україні: Rozetka займає 45%, Prom.ua – 20%, інші платформи – 35%. Це підкреслює домінування Rozetka.

Висновок до розділу 1

Проведений аналіз предметної області та огляд інтернет-магазинів України і світу дозволив визначити ключові особливості сфери електронної комерції, зокрема в контексті продажу аксесуарів для мобільних телефонів, таких як чохли. Предметна область охоплює діяльність інтернет-магазинів, які забезпечують торгівлю різноманітними товарами, від електроніки до персоналізованих аксесуарів. Аналіз показав, що маркетплейси, спеціалізовані магазини, брендові та нішові платформи мають свої сильні та слабкі сторони, які впливають на їх привабливість для споживачів. Маркетплейси, такі як Rozetka чи Amazon, домінують завдяки широкому асортименту та конкурентним цінам, але стикаються з перевантаженими інтерфейсами та нерівномірною якістю. Спеціалізовані магазини, як Spigen і Case-Mate, пропонують якісну продукцію, але обмежені в асортименті та персоналізації. Нішові платформи приваблюють унікальними пропозиціями, але потребують значних зусиль для просування.

Ринкові тенденції 2024 року, зокрема зростання мобільної комерції, попит на персоналізацію та екологічність, формують сучасні вимоги до інтернет-магазинів. В Україні швидка доставка та локальна адаптація є ключовими факторами успіху. Цільова аудиторія, що включає молодь, ділових користувачів і екосвідомих споживачів, потребує гнучкого асортименту та зручного функціоналу. Отримані дані створюють основу для розробки інтернет-магазину «Case Space», який має поєднувати широкий вибір чохлів, швидку доставку, можливості персоналізації та адаптивний інтерфейс, щоб ефективно конкурувати на ринку електронної комерції.

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						18
Змн.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 2. ТЕХНІЧНЕ ЗАБЕЗПЕЧЕННЯ ТА ПРОЄКТУВАННЯ СИСТЕМИ

2.1. Обґрунтування вибору мови програмування

Розробка інтернет-магазину «Case Space» для продажу чохлів для телефонів потребує ретельного вибору технологій, які забезпечують швидкість створення, надійність, безпеку та можливість масштабування системи. Для реалізації серверної частини проєкту обрано мову програмування Python в поєднанні з фреймворками Django і Django REST Framework (DRF). Цей вибір ґрунтується на універсальності Python, його зрозумілому синтаксисі, потужній екосистемі бібліотек і розширених можливостях Django/DRF для створення сучасних веб-додатків [17].

Python в 2025 році залишається однією з найпопулярніших мов програмування, посідаючи перші позиції в рейтингах, таких як TIOBE Index і Stack Overflow Developer Survey. Його популярність пояснюється лаконічним і читабельним синтаксисом, який дозволяє розробникам створювати зрозумілий код із мінімальними зусиллями. Для проєкту «Case Space» це має вирішальне значення, оскільки інтернет-магазин потребує швидкої реалізації складних компонентів, таких як каталог товарів, система управління замовленнями, автентифікація користувачів та інтеграція з платіжними системами. Читабельність Python полегшує командну розробку, дозволяючи кільком програмістам ефективно працювати над кодом, а також спрощує подальшу підтримку та внесення змін в систему. Наприклад, якщо в майбутньому знадобиться додати нові функції, такі як рекомендаційна система для персоналізованих пропозицій чохлів, Python забезпечує гнучкість для швидкої адаптації.

Екосистема Python включає широкий набір бібліотек, які значно прискорюють розробку. Для веб-додатків доступні інструменти для роботи з

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						19
Змн.	Арк.	№ докум.	Підпис	Дата		

базами даних, обробки HTTP-запитів, інтеграції з зовнішніми API та забезпечення безпеки.

Для бекенду інтернет-магазину «Case Space» обрано фреймворк Django, який є одним із найпотужніших інструментів для веб-розробки на Python. Django базується на принципі «батареї в комплекті», що означає наявність вбудованих компонентів для вирішення типових завдань, таких як автентифікація користувачів, керування базами даних і захист від поширених атак, наприклад, CSRF, XSS чи SQL-ін'єкцій. Для інтернет-магазину, де безпека даних користувачів і транзакцій є критично важливою, ці можливості забезпечують надійний захист без необхідності розробки додаткових механізмів. Наприклад, Django автоматично обробляє захист від CSRF-атак, що знижує ризик несанкціонованого доступу до даних під час оформлення замовлень. Крім того, Django ORM (Object-Relational Mapping) спрощує взаємодію з базою даних PostgreSQL, обраною для проєкту через її надійність і здатність обробляти складні запити. Це дозволяє легко створювати моделі даних для товарів, замовлень і профілів користувачів, використовуючи об'єкти Python замість написання складних SQL-запитів [18].

Django REST Framework (DRF) доповнює Django, надаючи інструменти для створення RESTful API, необхідного для зв'язку бекенду з фронтендом, який в «Case Space» реалізовано на React із використанням Vite, Zustand і Mantine. DRF дозволяє швидко створювати API-ендпоінти для таких операцій, як отримання списку чохлів, додавання товару до кошика чи автентифікація користувача. Його можливості, такі як серіалізація даних, підтримка JWT (JSON Web Tokens) для безпечної автентифікації та автоматична генерація документації API, значно спрощують розробку та інтеграцію. Наприклад, в «Case Space» DRF може використовуватися для створення ендпоінту `/api/products/`, який повертає список товарів з фільтрами за брендом телефону чи матеріалом, що забезпечує швидке відображення каталогу на фронтенді. Гнучкість DRF дозволяє легко масштабувати API, додаючи нові ендпоінти для майбутніх функцій, таких як відстеження статусу доставки чи персоналізовані рекомендації.

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						20
Змн.	Арк.	№ докум.	Підпис	Дата		

Порівняння Python і Django/DRF з альтернативними технологіями підтверджує їх доцільність для проєкту. Однією з альтернатив є JavaScript з фреймворком Node.js, який вирізняється високою швидкістю обробки асинхронних запитів, що корисно для додатків з великою кількістю одночасних користувачів. Наприклад, Node.js ефективний для чатів чи потокового контенту, але його складніший синтаксис і менш зріла екосистема порівняно з Python ускладнюють швидку розробку складних систем, таких як інтернет-магазин. Node.js потребує додаткових бібліотек для реалізації таких функцій, як автентифікація чи захист від атак, тоді як Django пропонує їх «з коробки». Окрім того, асинхронна модель Node.js може бути надмірною для «Case Space», де більшість операцій, таких як обробка замовлень, є послідовними.

Іншою альтернативою є PHP із фреймворком Laravel, який також підходить для веб-розробки. Laravel пропонує схожі до Django можливості, такі як ORM і захист від атак, але PHP втрачає популярність в 2025 році через меншу гнучкість порівняно з Python, особливо для інтеграції з сучасними технологіями, такими як машинне навчання чи API платіжних систем. Наприклад, створення рекомендаційної системи для «Case Space» на основі аналізу вподобань користувачів легше реалізувати на Python завдяки бібліотекам, таким як TensorFlow чи scikit-learn, тоді як PHP потребує складніших рішень. Крім того, синтаксис PHP менш інтуїтивний, що може уповільнити розробку та ускладнити залучення нових розробників [19].

Ще одна альтернатива – Java з фреймворком Spring Boot, який відомий своєю надійністю та масштабованістю в корпоративних системах. Однак Java є більш громіздкою мовою, що потребує більше часу на написання коду порівняно з Python. Наприклад, створення моделі даних для каталогу чохлів в Spring Boot вимагає більше коду, ніж у Django ORM. Крім того, Java менш популярна серед стартапів і невеликих проєктів, таких як «Case Space», через вищі вимоги до ресурсів і складність підтримки.

Python і Django/DRF також відповідають сучасним ринковим тенденціям електронної комерції. Попит на персоналізацію, наприклад, можливість

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						21
Змн.	Арк.	№ докум.	Підпис	Дата		

замовлення чохлів з індивідуальним дизайном, потребує гнучкої серверної логіки, яку легко реалізувати через Django. Наприклад, модель даних для кастомізованих чохлів може включати поля для зберігання зображень чи тексту, які користувач завантажує через фронтенд, а DRF забезпечує передачу цих даних через API. Інтеграція з платіжними системами, такими як Way For Pay, спрощується завдяки бібліотекам Python, які дозволяють безпечно обробляти транзакції та перевіряти їхній статус. В контексті українського ринку інтеграція з логістичними службами, такими як «Нова Пошта», також підтримується через API, які легко підключити за допомогою DRF.

Безпека є ще одним важливим аспектом, де Django/DRF демонструють переваги. Django забезпечує захист від типових вразливостей, таких як SQL-ін'єкції, завдяки ORM, який автоматично екранує запити. DRF підтримує безпечну автентифікацію через JWT, що дозволяє користувачам «Case Space» безпечно входити в систему та керувати своїм профілем. Наприклад, автентифікація через JWT забезпечує, що лише авторизовані користувачі можуть переглядати історію замовлень чи змінювати налаштування. Django також пропонує вбудовані інструменти для захисту від CSRF-атак, що важливо для безпечної обробки форм оплати чи оформлення замовлень.

Модульність і тестування є додатковими перевагами Python і Django/DRF. Django включає інструменти для створення юніт-тестів, що дозволяють перевіряти коректність роботи таких функцій, як додавання товару до кошика чи обробка платежу. Наприклад, тестовий сценарій може перевірити, чи правильно оновлюється статус замовлення після успішної оплати. Python також підтримує модульну структуру коду, що полегшує розподіл проєкту на окремі компоненти, такі як керування каталогом, замовленнями чи профілями користувачів. Це забезпечує гнучкість для майбутнього масштабування, наприклад, додавання функції відстеження доставки чи аналітики продажів.

Відповідність Python і Django/DRF локальним вимогам українського ринку також є важливою. Швидка доставка через локальні служби, такі як «Нова Пошта», потребує інтеграції з їх API, що легко реалізувати через Python-

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						22
Змн.	Арк.	№ докум.	Підпис	Дата		

бібліотеки, такі як requests. Крім того, підтримка української мови в інтерфейсі адмін-панелі Django спрощує керування контентом для адміністраторів магазину. Наприклад, адмін-панель дозволяє швидко додавати нові чохли до каталогу чи редагувати описи, що важливо для оперативного оновлення асортименту.

Для оцінки ефективності вибору Python і Django/DRF було проаналізовано їх продуктивність в реальних проєктах. Багато успішних інтернет-магазинів, таких як платформи на базі Shopify чи власних рішень, використовують Python для бекенду завдяки його універсальності. Наприклад, Django застосовується в проєктах, таких як Instagram чи Pinterest, що підтверджує його здатність обробляти великі обсяги даних і запитів. Для «Case Space» це означає, що обраний стек може впоратися з потенційним зростанням кількості користувачів і замовлень в майбутньому.

Порівняння з іншими технологіями також враховує витрати на розробку та підтримку. Python і Django/DRF зменшують час розробки завдяки готовим компонентам, що знижує витрати порівняно з Java, яка потребує більше ресурсів. Крім того, велика кількість Python-розробників на ринку праці в Україні полегшує залучення спеціалістів для підтримки проєкту. Наприклад, за даними DOU.ua за 2024 рік, Python є однією з топ-3 мов програмування за кількістю вакансій в Україні, що забезпечує доступність кваліфікованих кадрів.

Отже, вибір Python з фреймворками Django і DRF для бекенду інтернет-магазину «Case Space» є обґрунтованим завдяки простоті синтаксису, потужній екосистемі, вбудованим інструментам безпеки та можливостям масштабування. Порівняння з альтернативами, такими як Node.js, Laravel і Spring Boot, показує, що Python і Django/DRF забезпечують оптимальний баланс між швидкістю розробки, безпекою та гнучкістю. Ці технології дозволяють ефективно реалізувати необхідний функціонал, від каталогу товарів до інтеграції з платіжними системами, і відповідають сучасним ринковим тенденціям, включаючи персоналізацію, безпеку та локальну адаптацію для українського ринку.

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						23
Змн.	Арк.	№ докум.	Підпис	Дата		

2.2. Обґрунтування вибору React, Vite, Zustand та Mantine для фронтенду

Розробка клієнтської частини інтернет-магазину «Case Space» вимагає створення зручного, швидкого та адаптивного інтерфейсу, який забезпечує комфортну взаємодію користувачів з каталогом товарів, кошиком, системою персоналізації чохлів і оформленням замовлень. Для реалізації фронтенду обрано технологічний стек, що включає бібліотеку React, інструмент для збирання Vite, бібліотеку керування станом Zustand і бібліотеку компонентів Mantine. Цей вибір зумовлений їхньою ефективністю, гнучкістю, сучасними можливостями та відповідністю вимогам проєкту, таким як адаптивність, швидкість завантаження та підтримка персоналізації [20].

React, розроблений компанією Meta, є однією з найпопулярніших бібліотек для створення інтерфейсів веб-додатків в 2024 році, що підтверджується даними Stack Overflow Developer Survey, де він займає провідні позиції серед фронтед-технологій. React базується на компонентному підході, що дозволяє розбити інтерфейс на модульні частини, такі як картка товару, фільтри каталогу чи форма оформлення замовлення. Для «Case Space» це забезпечує гнучкість в розробці, дозволяючи створювати перевикористовувані компоненти, наприклад, для відображення чохлів з різними характеристиками чи динамічного кошика. Компонентний підхід також спрощує підтримку та оновлення інтерфейсу, що важливо для додавання нових функцій, таких як конструктор дизайну чохлів. React використовує віртуальний DOM, який оптимізує оновлення інтерфейсу, зменшуючи кількість операцій із реальним DOM, що забезпечує високу швидкість відображення сторінок, критичну для мобільних користувачів, які становлять до 70% аудиторії інтернет-магазинів у 2024 році.

Vite обрано як інструмент для збирання фронтенду завдяки його швидкості та сучасному підходу до розробки. На відміну від традиційного Create React App (CRA), який використовує Webpack, Vite застосовує нативні ES-модулі браузера, що значно прискорює час збирання та перезавантаження сторінок під час

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						24
Змн.	Арк.	№ докум.	Підпис	Дата		

розробки. Для «Case Space» це дозволяє розробникам швидко тестувати зміни в інтерфейсі, наприклад, оновлення стилів карток товарів чи фільтрів. Vite також оптимізує кінцевий бандл, зменшуючи розмір файлів JavaScript і CSS, що скорочує час завантаження сторінок для користувачів. Наприклад, каталог з сотнями чохлів завантажується швидше завдяки ефективному кешуванню та модульній структурі, що створюється Vite. Окрім того, Vite підтримує сучасні стандарти JavaScript, такі як ES Modules, що спрощує інтеграцію з бібліотеками, такими як React і Mantine.

Zustand обрано для керування станом фронтенду через його легкість, мінімалістичний підхід і високу продуктивність. На відміну від Redux, який часто використовується в React-додатках, Zustand має простіший API і не вимагає складної структури редюсерів чи мідлварів. Для «Case Space» Zustand ідеально підходить для керування такими станами, як вміст кошика, фільтри каталогу чи налаштування персоналізованих чохлів. Наприклад, користувач може додавати чохли до кошика, а Zustand миттєво оновлює стан, відображаючи актуальну кількість товарів і загальну суму. Легкість Zustand дозволяє уникнути надмірної складності коду, що прискорює розробку та полегшує підтримку. Також, Zustand підтримує асинхронні операції, що корисно для інтеграції з API бекенду, наприклад, для отримання даних про товари чи статус замовлення [15].

Mantine обрано як бібліотеку компонентів для створення стильного та адаптивного інтерфейсу. Mantine пропонує набір готових компонентів, таких як кнопки, модальні вікна, форми та таблиці, які легко налаштовуються під дизайн «Case Space». Наприклад, компонент Mantine для фільтрів дозволяє створити зручний інтерфейс для вибору чохлів за брендом телефону, матеріалом чи ціною, що відповідає потребам користувачів. Mantine підтримує адаптивний дизайн, що забезпечує коректне відображення на смартфонах, планшетах і ПК, що є критично важливим, враховуючи зростання мобільної комерції. Бібліотека інтегрується з Tailwind CSS, що дозволяє швидко стилізувати компоненти, наприклад, для створення привабливих карток товарів із анімаціями чи ефектами

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						25
Змн.	Арк.	№ докум.	Підпис	Дата		

наведення. Mantine також пропонує темну тему, що відповідає сучасним трендам дизайну та підвищує комфорт для користувачів.

Порівняння обраного стеку з альтернативами підтверджує його доцільність. Наприклад, замість React можна було б обрати Vue.js, який також підтримує компонентний підхід і має простіший синтаксис. Однак React має більшу спільноту, більше бібліотек і ширшу підтримку в екосистемі, що полегшує пошук рішень для специфічних завдань, таких як інтеграція з платіжними системами чи конструктором дизайну. Vue.js менш популярний в Україні, що може ускладнити залучення розробників для підтримки проєкту. Angular, інша альтернатива, є більш громіздким через свою складну архітектуру, що збільшує час розробки для проєктів, таких як «Case Space», де потрібна швидка реалізація.

Для збирання фронтенду альтернативою Vite є Webpack, який використовується в Create React App. Webpack пропонує більше можливостей для кастомізації, але його збирання повільніше, що уповільнює розробку, особливо під час тестування змін в реальному часі. Vite, навпаки, забезпечує швидке перезавантаження сторінок і оптимізацію бандлів, що критично для проєкту з великою кількістю динамічних компонентів, таких як каталог товарів чи форма персоналізації. Інша альтернатива, Parcel, є простішою в налаштуванні, але менш гнучкою для складних проєктів, що потребують інтеграції з численними бібліотеками [13].

Для керування станом альтернативою Zustand є Redux або MobX. Redux, хоча й потужний, додає складності через необхідність створення редюсерів, дій і мідлварів, що може бути надмірним для «Case Space», де стани відносно прості, наприклад, керування кошиком чи фільтрами. MobX пропонує реактивний підхід, але його складніша модель може ускладнити підтримку коду. Zustand, завдяки мінімалістичному API, забезпечує швидке налаштування та ефективне керування станом, що ідеально для потреб проєкту.

Щодо бібліотеки компонентів, альтернативою Mantine є Material-UI або Ant Design. Material-UI пропонує широкий набір компонентів, але його дизайн

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						26
Змн.	Арк.	№ докум.	Підпис	Дата		

прив'язаний до Material Design від Google, що може обмежувати кастомізацію під унікальний стиль «Case Space». Ant Design більше підходить для корпоративних додатків, але його компоненти менш гнучкі для створення сучасних інтерфейсів з анімаціями, які приваблюють молодшу аудиторію. Mantine забезпечує баланс між готовим функціоналом і можливостями кастомізації, що відповідає потребам адаптивного та стильного дизайну.

Обраний стек відповідає сучасним ринковим тенденціям електронної комерції. Зростання мобільної комерції, коли 70% покупок в 2024 році здійснюються через смартфони, вимагає швидкого та адаптивного інтерфейсу. React із Vite забезпечує швидке завантаження сторінок, а Mantine гарантує коректне відображення на різних пристроях. Попит на персоналізацію, наприклад, створення чохлів з власним дизайном, підтримується через компоненти Mantine для форм і Zustand для керування станом, що дозволяє користувачам динамічно завантажувати зображення чи текст. Інтеграція з соціальними мережами, наприклад, для поширення дизайну чохлів, спрощується завдяки модульній структурі React. В контексті українського ринку, де швидка доставка та локалізація є важливими, адаптивний інтерфейс із підтримкою української мови, реалізований через Mantine, підвищує зручність для користувачів.

React, Vite, Zustand і Mantine також забезпечують ефективність розробки та підтримки. React дозволяє створювати перевикористовувані компоненти, що зменшує обсяг коду для таких елементів, як картки товарів чи форми замовлення. Vite прискорює процес розробки завдяки швидкому збиранню, а Zustand мінімізує накладні витрати на керування станом. Mantine скорочує час на створення інтерфейсу завдяки готовим компонентам, які легко стилізувати під бренд «Case Space». Наприклад, компонент Mantine для модального вікна може використовуватися для відображення конструктора дизайну чохлів, де користувач обирає кольори чи завантажує зображення [13].

Для оцінки ефективності стеку було проаналізовано його використання в інших проєктах. Багато сучасних інтернет-магазинів, таких як платформи на базі

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						27
Змн.	Арк.	№ докум.	Підпис	Дата		

Shopify чи власних рішень, використовують React для фронтенду через його гнучкість та продуктивність. Vite застосовується в проєктах, де потрібна швидка розробка, наприклад, в стартапах. Mantine і Zustand популярні серед невеликих команд, які прагнуть швидкої реалізації без втрати якості. Наприклад, подібний стек використовується в нішевих магазинах, що пропонують персоналізовані товари, що підтверджує його придатність для «Case Space».

Порівняння з альтернативами також враховує витрати на розробку. React, Vite, Zustand і Mantine є безкоштовними та open-source, що знижує витрати порівняно з комерційними рішеннями, такими як Shopify. Велика кількість React-розробників в Україні, за даними DOU.ua за 2024 рік, полегшує залучення спеціалістів для підтримки проєкту. Наприклад, React є другою за популярністю технологією серед фронтед-розробників в Україні, що забезпечує доступність кадрів.

Отже, вибір React, Vite, Zustand і Mantine для фронтенду "Case Space" є обґрунтованим завдяки їхній продуктивності, гнучкості та відповідності сучасним вимогам електронної комерції. React забезпечує компонентний підхід і швидке оновлення інтерфейсу, Vite прискорює розробку та оптимізує завантаження, Zustand спрощує керування станом, а Mantine пропонує стильні та адаптивні компоненти. Порівняння з альтернативами, такими як Vue.js, Angular, Webpack, Redux і Material-UI, підтверджує оптимальність обраного стеку для створення зручного, швидкого та конкурентоспроможного інтернет-магазину.

2.3. Розробка архітектури системи

Розробка інтернет-магазину «Case Space» для продажу чохлів для телефонів вимагає створення продуманої архітектури системи, яка забезпечує її функціональність, безпеку, продуктивність і можливість масштабування. Архітектура визначає структуру компонентів, принципи їх взаємодії та способи обробки даних, що є основою для реалізації проєкту.

Архітектура «Case Space» базується на клієнт-серверній моделі, що є стандартом для веб-додатків електронної комерції. Клієнтська частина, реалізована на React із використанням Vite, Zustand і Mantine, відповідає за створення адаптивного інтерфейсу, обробку взаємодії користувача та надсилення запитів до сервера. Серверна частина, розроблена на Python із фреймворками Django і Django REST Framework (DRF), обробляє запити, керує даними та інтегрується з базою даних PostgreSQL і зовнішніми сервісами. Така структура забезпечує чіткий розподіл обов'язків, що полегшує розробку, тестування та масштабування. Наприклад, клієнтська частина може динамічно оновлювати кошик в реальному часі, тоді як серверна частина перевіряє наявність товару, обробляє оплату та надсилає дані до логістичної служби.

Основними компонентами системи є клієнтський інтерфейс, API-сервер, база даних і зовнішні сервіси. Клієнтський інтерфейс включає модулі для відображення каталогу чохлів, фільтрів, кошика, форми персоналізації та профілю користувача. API-сервер, побудований на DRF, надає RESTful ендпоінти для таких операцій, як отримання списку товарів, створення замовлення чи автентифікація. База даних PostgreSQL зберігає інформацію про товари, замовлення, користувачів та персоналізовані дизайни. Зовнішні сервіси, зокрема Way For Pay для обробки платежів і «Нова Пошта» для доставки, інтегруються через API, забезпечуючи безпечні транзакції та відстеження відправлень. Взаємодія між компонентами здійснюється через HTTP-запити у форматі JSON, що відповідає принципам REST-архітектури.

REST-архітектура дозволяє кожному ресурсу, такому як товар чи замовлення, мати унікальний ідентифікатор (URI) і підтримувати стандартні методи (GET, POST, PUT, DELETE). Наприклад, запит GET до ендпоінту /api/products/ повертає список чохлів з фільтрами за брендом телефону, матеріалом чи ціною, тоді як POST до /api/orders/ створює нове замовлення. Для забезпечення безпеки застосовується JWT-автентифікація, яка захищає доступ до ендпоінтів, таких як /api/profile/, дозволяючи лише авторизованим користувачам переглядати історію замовлень чи редагувати профіль. Django

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						29
Змн.	Арк.	№ докум.	Підпис	Дата		

забезпечує захист від типових атак, таких як CSRF, XSS і SQL-ін'єкції, що є критично важливим для безпеки транзакцій і даних користувачів. Продуктивність досягається через оптимізацію запитів до бази даних за допомогою Django ORM і кешування з бібліотекою Django Redis, що зменшує час відповіді на запити, наприклад, при завантаженні великого каталогу.

База даних PostgreSQL обрана завдяки її надійності, підтримці складних запитів і можливості масштабування. Схема бази даних включає таблиці для товарів, замовлень, користувачів, категорій і персоналізованих дизайнів. Таблиця товарів містить поля для назви, ціни, бренду телефону, матеріалу та URL зображень, тоді як таблиця замовлень пов'язує користувача, товари, статус доставки та дату створення. Зв'язки між таблицями, такі як зовнішні ключі, забезпечують цілісність даних. Наприклад, при видаленні товару з каталогу пов'язані замовлення автоматично оновлюються. Django ORM спрощує взаємодію з базою даних, дозволяючи розробникам працювати з об'єктами Python замість SQL-запитів, що прискорює створення таких функцій, як фільтрація чохлів чи підрахунок суми замовлення.

Інтеграція з зовнішніми сервісами є ключовим елементом архітектури. Платіжна система Way For Pay обробляє транзакції через API, який робить виклики до бекенду. Наприклад, при оформленні замовлення Way For Pay надсилає POST-запит до серверу із даними про суму, статус транзакції, підпис продавця (інформація зашифрована в MD-5 із секретним ключем продавця) та дату і час транзакції.

Архітектура системи враховує вимоги безпеки, продуктивності та масштабування. Безпека досягається через JWT-автентифікацію, HTTPS для шифрування даних і захист Django від атак. Наприклад, ендпоінт `/api/orders/` доступний лише авторизованим користувачам, що запобігає несанкціонованим діям. Продуктивність забезпечується через індексацію бази даних для швидких запитів, кешування популярних сторінок, таких як каталог, і оптимізацію API. Масштабування можливе завдяки розгортанню системи в хмарних сервісах, таких як AWS, з використанням балансувальників навантаження та

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						30
Змн.	Арк.	№ докум.	Підпис	Дата		

горизонтального масштабування серверів для обробки зростаючої кількості користувачів.

Порівняння архітектури з альтернативами підтверджує доцільність вибору. Монолітна архітектура, обрана для «Case Space», спрощує розробку та розгортання для невеликого проєкту, тоді як мікросервісна архітектура була б надмірно складною через необхідність координації сервісів. Наприклад, моноліт дозволяє централізовано керувати каталогом і замовленнями, що знижує витрати порівняно з мікросервісами, де кожен модуль потребує окремого розгортання. Serverless-архітектура, наприклад, на AWS Lambda, могла б зменшити витрати на інфраструктуру, але ускладнила б інтеграцію з PostgreSQL і управління станом. Event-driven архітектура, що використовує черги повідомлень, була б корисною для асинхронної обробки, але надмірною для поточних потреб «Case Space», де більшість операцій є послідовними.

Діаграма варіантів використання

Діаграма варіантів використання ілюструє взаємодію акторів з системою «Case Space». Користувач виконує дії, такі як перегляд каталогу, фільтрація чохлів за брендом чи матеріалом, додавання товарів до кошика, оформлення замовлення, створення персоналізованих дизайнів, відстеження доставки та редагування профілю. Адміністратор керує каталогом, додаючи чи редагуючи товари, обробляє замовлення, аналізує статистику продажів і керує користувачами. Платіжна система обробляє транзакції, а логістична служба оновлює статус доставки. Відносини включення показують, що оформлення замовлення залежить від обробки транзакцій і оновлення статусу доставки, тоді як відношення розширення вказують, що керування каталогом може включати додавання товару, а оформлення замовлення – редагування профілю. Ця діаграма визначає ключові сценарії використання, що є основою для реалізації функціоналу.

Зображено діаграму варіантів використання (рис. 2.1)

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						31
Змн.	Арк.	№ докум.	Підпис	Дата		

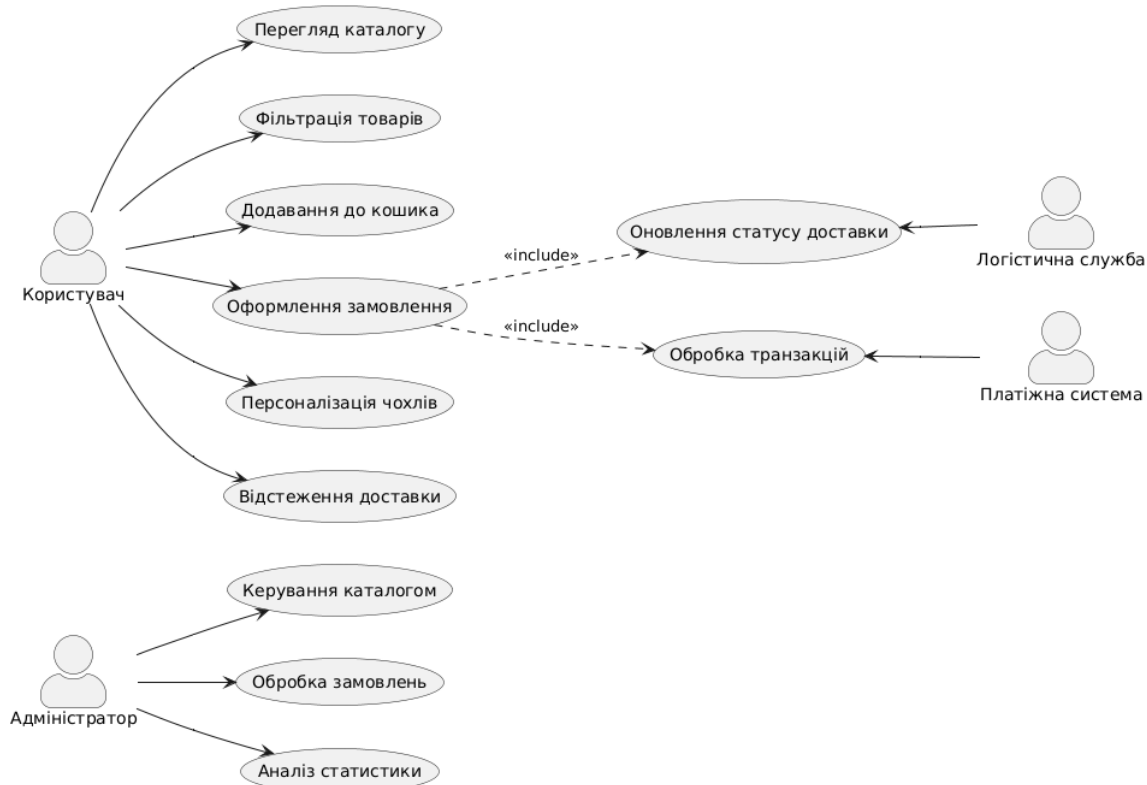


Рис. 2.1 – Діаграма варіантів використання

Діаграма класів

Діаграма класів описує статичну структуру системи, показуючи основні класи та їхні зв'язки. Клас User представляє користувача з атрибутами, такими як ідентифікатор, ім'я, електронна пошта, пароль, телефон і адреса, та методами для реєстрації, входу, оновлення профілю та отримання замовлень. Клас Product описує товар з атрибутами, такими як назва, ціна, бренд телефону, матеріал, зображення та кількість на складі, і методами для отримання деталей, оновлення запасів і застосування знижок. Клас Order представляє замовлення з атрибутами, такими як статус, сума, дата створення та адреса доставки, і методами для створення, скасування та оновлення статусу. Клас Category пов'язує товари за категоріями, а CustomDesign описує персоналізовані дизайни з атрибутами для зображень, тексту та шрифту. Зв'язки показують, що один користувач може мати багато замовлень і дизайнів, а замовлення включає кілька товарів. Ця діаграма є основою для реалізації моделей даних у Django ORM.

Зображено діаграму класів (рис. 2.2)

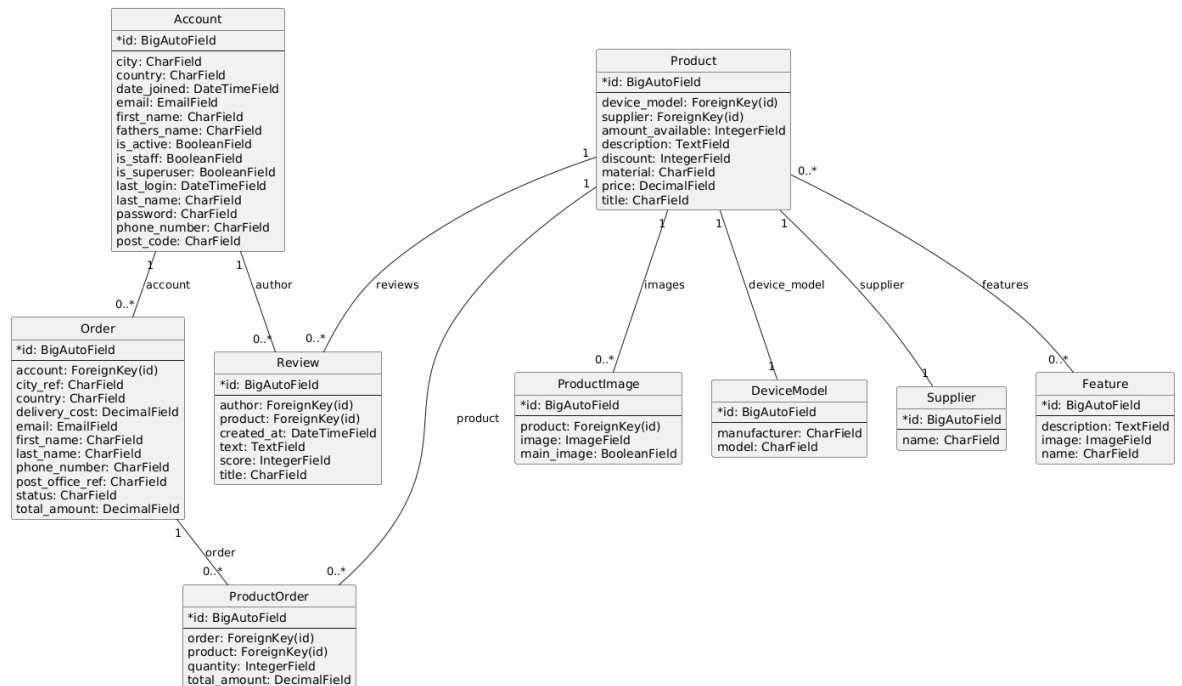


Рис. 2.2 – Діаграма класів

Діаграма послідовності

Діаграма послідовності ілюструє процес оформлення замовлення. Користувач натискає кнопку «Оформити замовлення» у клієнтському інтерфейсі, який надсилає POST-запит до ендпоінту `/api/orders/` із JWT-токеном і даними замовлення. API-сервер перевіряє токен, запитує дані про товари та користувача з бази даних, перевіряє їх правильність (ціна та знижка кожного товару та сума замовлення), та надсилає відповідь зі створеним замовленням до клієнтського інтерфейсу, після чого відбувається POST-запит до ендпоінту `/payment/initiate/` для отримання даних для форми оплати. Форма оплати після отримання даних переводить користувача на сторінку оплати. В свою чергу Way For Pay надсилає POST-запит до серверу на ендпоінт `/wfp/callback/` для обробки транзакції та підтвердження статусу замовлення. Якщо запит від Way For Pay був коректним після перевірки вмісту, то інтерфейс відображає повідомлення про успішне оформлення. Ця діаграма показує послідовність операцій і точки інтеграції, допомагаючи виявити потенційні затримки, наприклад, при обробці платежів чи перевірці даних замовлення.

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						33
Змн.	Арк.	№ докум.	Підпис	Дата		

Зображено діаграму послідовності (рис. 2.3)

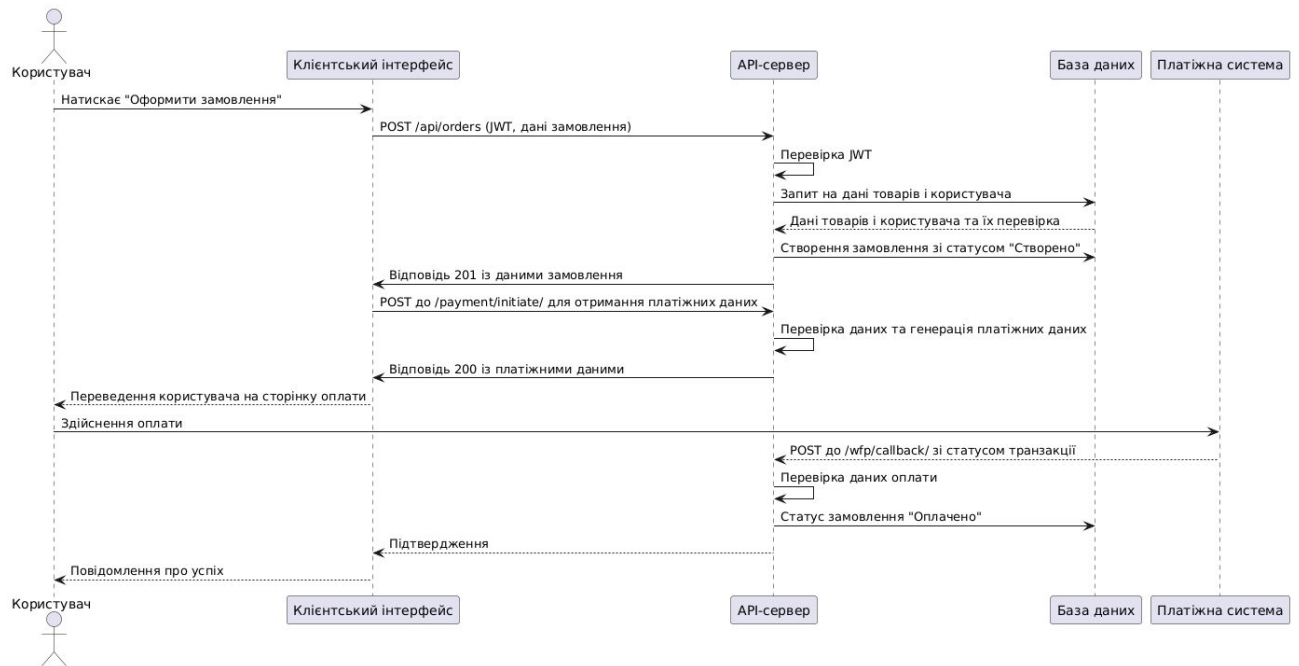


Рис. 2.3 – Діаграма послідовності

Діаграма компонентів

Діаграма компонентів ілюструє високорівневу структуру системи, показуючи основні компоненти та їх зв'язки. Клієнтська частина включає React UI (інтерфейс), API Client (модуль для HTTP-запитів), Zustand (керування станом) і Mantine (компоненти). Серверна частина складається з API-сервера (DRF), Django Models (моделі даних) і PostgreSQL (база даних). Зовнішні сервіси представлені Way For Pay і «Новою Поштою». React UI взаємодіє з API-сервером через API Client, який створено за допомогою бібліотеки «axios» надсилаючи HTTP-запити у форматі JSON. API-сервер інтегрується з PostgreSQL для зберігання даних і з зовнішнім сервісом для обробки платежів. Ця діаграма допомагає зрозуміти загальну архітектуру та взаємодію модулів.

Зображено діаграму компонентів зображена (рис. 2.4)

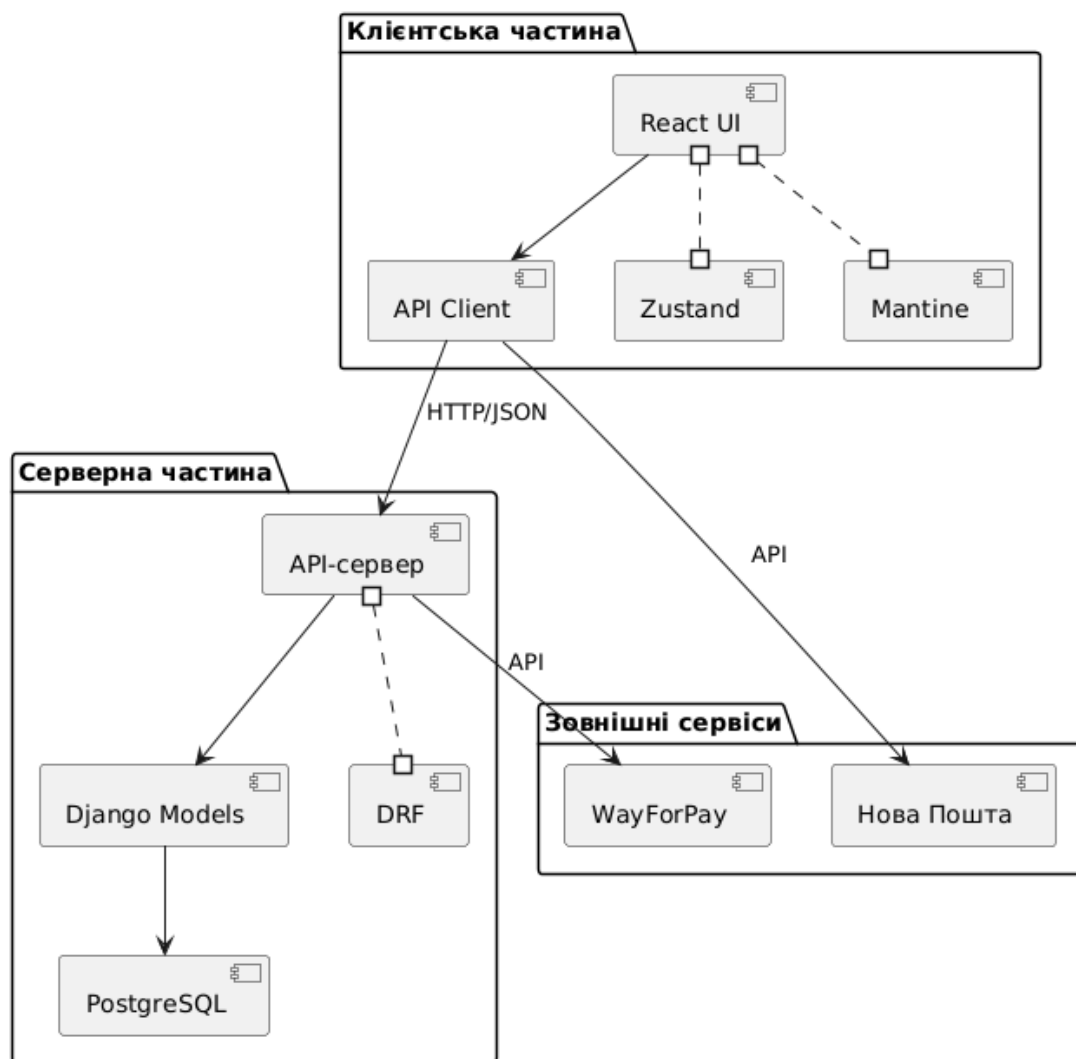


Рис. 2.4 – Діаграма компонентів

Архітектура "Case Space" базується на клієнт-серверній моделі з REST-архітектурою, що забезпечує ефективну взаємодію компонентів. UML-діаграми, створені в PlantUML, моделюють функціонал, структуру, поведінку та компоненти системи, сприяючи її розробці. Архітектура враховує вимоги безпеки, продуктивності, масштабування, інтеграції з зовнішніми сервісами та локальні особливості українського ринку, забезпечуючи конкурентоспроможність інтернет-магазину.

Висновок до розділу 2

Другий розділ присвячено обґрунтуванню технологічних рішень і розробці архітектури системи для інтернет-магазину «Case Space». Проведений аналіз

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						35
Змн.	Арк.	№ докум.	Підпис	Дата		

дозволив визначити оптимальний технологічний стек і структуру системи, які забезпечують її функціональність, безпеку, продуктивність і можливість масштабування. Для серверної частини обрано мову програмування Python із фреймворками Django та Django REST Framework завдяки їхній простоті, потужній екосистемі та вбудованим інструментам безпеки, що ідеально відповідають потребам обробки даних, автентифікації та інтеграції з платіжними й логістичними сервісами. Клієнтська частина реалізована на React з використанням Vite, Zustand і Mantine, що забезпечує швидкий, адаптивний і зручний інтерфейс, підтримуючи сучасні вимоги мобільної комерції та персоналізації.

Архітектура системи базується на клієнт-серверній моделі з REST-архітектурою, що гарантує ефективну взаємодію між компонентами. Використання бази даних PostgreSQL забезпечує надійне зберігання даних, а інтеграція з зовнішніми сервісами, такими як Way For Pay і «Нова Пошта», підтримує швидку обробку платежів і доставку. Моделювання системи за допомогою UML-діаграм, дозволило чітко візуалізувати функціонал, структуру та поведінку системи, полегшуючи її розробку та документування. Архітектура враховує локальні особливості українського ринку, зокрема попит на швидку доставку та локалізований інтерфейс. Запропоновані технологічні рішення та архітектура відповідають сучасним тенденціям електронної комерції, включаючи зростання мобільних покупок, попит на персоналізацію та екологічність.

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						36
Змн.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 3. ВПРОВАДЖЕННЯ ТА ОЦІНКА РОБОТИ ІНТЕРНЕТ-МАГАЗИНУ ЧОХЛІВ ДЛЯ ТЕЛЕФОНІВ CASE SPACE

3.1. Опис фрагментів коду розробки інтернет-магазину

Опис фрагментів коду розробки є одним із ключових етапів розробки інтернет-магазину, який дає можливість зрозуміти логіку роботи магазину, його особливості та інші важливі аспекти.

Система автентифікації та управління користувачами

Кастомізація адаптерів для автентифікації

Цей фрагмент коду, розташований у папці accounts, відповідає за модифікацію стандартної поведінки бібліотеки django-allauth для автентифікації користувачів. Він містить два класи, які змінюють логіку збереження користувачів і пов'язування соціальних акаунтів. Перший клас дозволяє створювати акаунти без паролів для користувачів, які реєструються через соціальні мережі, що спрощує процес входу. Другий клас забезпечує автоматичне пов'язування соціального акаунта, наприклад, Google, із локальним акаунтом, якщо їхні верифіковані електронні адреси збігаються, запобігаючи дублюванню профілів. Це підвищує зручність для користувачів «Case Space», дозволяючи використовувати єдиний акаунт для всіх способів входу.

Фрагмент коду виглядає наступним чином (рис. 3.1)

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						37
Змн.	Арк.	№ докум.	Підпис	Дата		

```

from allauth.account.models import EmailAddress
from allauth.socialaccount.adapter import DefaultSocialAccountAdapter
from allauth.account.adapter import DefaultAccountAdapter

class NoPasswordOverrideAdapter(DefaultAccountAdapter):
    def save_user(self, request, user, form, commit=True):
        user = super().save_user(request, user, form, commit=False)

        # Only set password if user has no password yet
        if not user.has_usable_password():
            user.set_unusable_password()

        if commit:
            user.save()
        return user

class EnhancedSocialAccountAdapter(DefaultSocialAccountAdapter):
    # Solution to connect local users to social accounts
    # with enforcing check for email_validated field
    # coming from Google data
    def pre_social_login(self, request, sociallogin):
        DefaultSocialAccountAdapter

```

Рис. 3.1 – Кастомізація адаптерів для автентифікації

Реєстрація моделі в адмін-панелі

Цей код, відповідає за інтеграцію моделі Account із адміністративною панеллю Django. Він забезпечує можливість керування об'єктами моделі, такими як створення, редагування чи видалення користувачів, через стандартний інтерфейс адмінки. Для "Case Space" це дозволяє адміністраторам ефективно управляти профілями користувачів, наприклад, перевіряти їхні дані чи змінювати статуси.

Фрагмент коду виглядає наступним чином (рис. 3.2):

```

from django.contrib import admin
from .models import Account

admin.site.register(Account)

```

Рис. 3.2 – Реєстрація моделі в адмін-панелі

Конфігурація додатка accounts

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						38
Змн.	Арк.	№ докум.	Підпис	Дата		

Цей фрагмент визначає конфігурацію Django-додатка accounts, який є частиною backend-структури «Case Space». Код задає тип автоінкрементного поля для моделей і вказує назву додатка. Це необхідно для правильної ініціалізації додатка в проєкті, забезпечуючи коректну роботу всіх його компонентів, пов'язаних із автентифікацією та управлінням користувачами.

Фрагмент коду виглядає наступним чином (рис. 3.3):

```
from django.apps import AppConfig

class AccountsConfig(AppConfig):
    default_auto_field = 'django.db.models.BigAutoField'
    name = 'accounts'
```

Рис. 3.3 – Конфігурація додатка accounts

Кастомний клієнт для OAuth2

Розташований у папці accounts, цей код створює кастомний клієнт для OAuth2-авторизації в рамках django-allauth. Він модифікує стандартний клієнт, дозволяючи обробляти специфічні параметри авторизації, такі як область доступу, без їх використання в батьківському класі. В контексті «Case Space» цей клієнт застосовується для автентифікації через соціальні мережі, зокрема Google, забезпечуючи коректну взаємодію з API провайдера.

Фрагмент коду виглядає наступним чином (рис. 3.4):

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						39
Змн.	Арк.	№ докум.	Підпис	Дата		

```

from allauth.socialaccount.providers.oauth2.client import OAuth2Client

class FixedOAuth2Client(OAuth2Client):
    def __init__(
        self,
        request,
        consumer_key,
        consumer_secret,
        access_token_method,
        access_token_url,
        callback_url,
        _scope,
        scope_delimiter=" ",
        headers=None,
        basic_auth=False,
    ):
        super().__init__(
            request,
            consumer_key,
            consumer_secret,
            access_token_method,
            access_token_url,
            callback_url,
            scope_delimiter,
            headers,
            basic_auth,
        )

```

Рис. 3.4 – Кастомний клієнт для OAuth2

Кастомна модель користувача та її менеджер

Цей код визначає кастомну модель користувача та менеджер для неї. Менеджер забезпечує створення звичайних користувачів і суперкористувачів із email як основним ідентифікатором, нормалізуючи адресу та встановлюючи пароль. Модель розширює стандартну модель Django, замінюючи username на email і додаючи поля для імені, прізвища, по батькові, телефону, міста, країни та поштового коду. В «Case Spase» це дозволяє зберігати детальну інформацію про користувачів, необхідну для оформлення замовлень і доставки.

Фрагмент коду виглядає наступним чином (рис. 3.5):

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						40
Змн.	Арк.	№ докум.	Підпис	Дата		

```

from django.contrib.auth.models import AbstractUser
from django.db import models
from django.contrib.auth.models import BaseUserManager

class AccountManager(BaseUserManager):
    def create_user(self, email, password=None, **extra_fields):
        if not email:
            raise ValueError("The Email field is required")
        email = self.normalize_email(email)
        user = self.model(email=email, **extra_fields)
        user.set_password(password)
        user.save(using=self._db)
        return user

    def create_superuser(self, email, password=None, **extra_fields):
        extra_fields.setdefault("is_staff", True)
        extra_fields.setdefault("is_superuser", True)

        if not extra_fields.get("is_staff"):
            raise ValueError("Superuser must have is_staff=True.")
        if not extra_fields.get("is_superuser"):
            raise ValueError("Superuser must have is_superuser=True.")

        return self.create_user(email, password, **extra_fields)

```

Рис. 3.5 – Кастомна модель користувача та її менеджер

Серіалізатори для моделі користувача

Цей фрагмент містить два серіалізатори для моделі Account. Перший серіалізатор обробляє дані для створення користувача через API, перевіряючи унікальність email і зберігаючи пароль та особисті дані. Другий серіалізатор використовується для відображення імені та прізвища користувача у відгуках. В «Case Space» ці серіалізатори забезпечують безпечну реєстрацію через API та відображення даних у відгуках про чохли.

Фрагмент коду виглядає наступним чином (рис. 3.6):

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						41
Змн.	Арк.	№ докум.	Підпис	Дата		

```

from rest_framework import serializers
from django.contrib.auth import get_user_model
from .models import Account

User = get_user_model()

class AccountSerializer(serializers.ModelSerializer):
    class Meta:
        model = Account
        fields = [
            "email",
            "first_name",
            "last_name",
            "fathers_name",
            "phone_number",
            "password",
        ]
        extra_kwargs = {"password": {"write_only": True}}

    def validate_email(self, email):
        if User.objects.filter(email=email).exists():
            raise serializers.ValidationError(
                "Користувач з такою адресою електронної пошти вже існує"
            )
        return email

```

Рис. 3.6 – Серіалізатори для моделі користувача

Шаблон для тестів

Цей код є порожнім шаблоном для модуля тестів Django. Він призначений для написання тестів, які перевіряють функціонал додатка accounts, наприклад, створення користувачів чи автентифікацію, але в цьому фрагменті конкретні тести відсутні. В «Case Space» цей файл може використовуватися для забезпечення якості коду через автоматизоване тестування.

Фрагмент коду виглядає наступним чином (рис. 3.7):

```

from django.test import TestCase

# Create your tests here.

```

Рис. 3.7 – Шаблон для тестів

Налаштування URL-маршрутів

Цей фрагмент визначає маршрути для API-ендпоінтів, пов'язаних з автентифікацією та управлінням користувачами. Він включає маршрути для

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						42
Змн.	Арк.	№ докум.	Підпис	Дата		

отримання, оновлення та перевірки JWT-токенів, а також для реєстрації та отримання даних користувача. В «Case Space» ці маршрути забезпечують доступ до функцій автентифікації через API, наприклад, для входу чи створення акаунта.

Фрагмент коду виглядає наступним чином (рис. 3.8):

```
from django.urls import path, include
from rest_framework_simplejwt.views import (
    TokenObtainPairView,
    TokenRefreshView,
    TokenVerifyView,
)
from .views import RegisterView, GetAccount

urlpatterns = [
    path("token/", TokenObtainPairView.as_view(), name="token_obtain_pair"),
    path("token/refresh/", TokenRefreshView.as_view(), name="token_refresh"),
    path("token/verify/", TokenVerifyView.as_view(), name="token_verify"),
    path("register/", RegisterView.as_view()),
    path("get/", GetAccount.as_view()),
]
```

Рис. 3.8 – Налаштування URL-маршрутів додатку accounts

API-представлення для автентифікації

Цей код, містить три API-представлення для автентифікації та управління користувачами. Перше представлення повертає дані автентифікованого користувача, друге обробляє реєстрацію через API, а третє забезпечує вхід через Google із додаванням JWT-токенів у відповідь. В «Case Space» ці представлення дозволяють користувачам реєструватися, входити через соціальні мережі та отримувати інформацію про свій профіль через API.

Фрагмент коду виглядає наступним чином (рис. 3.9):

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						43
Змн.	Арк.	№ докум.	Підпис	Дата		

```

from rest_framework.views import APIView
from rest_framework.permissions import IsAuthenticated
from rest_framework.response import Response
from rest_framework import status
from rest_framework_simplejwt.tokens import RefreshToken
from dj_rest_auth.registration.views import SocialLoginView
from allauth.socialaccount.providers.google.views import GoogleOAuth2Adapter

from .serializers import AccountSerializer
from .clients import FixedOAuth2Client


class GetAccount(APIView):
    permission_classes = [IsAuthenticated]

    def get(self, request):
        user = request.user

        return Response(
            {
                "email": user.email,
                "first_name": user.first_name,
                "last_name": user.last_name,
                "fathers_name": user.fathers_name,
                "phone_number": user.phone_number,
                "city": user.city,
                "country": user.country,
                "post_code": user.post_code,
            },
            status=status.HTTP_200_OK,
        )

```

Рис. 3.9 – API-представлення для автентифікації

1. Базові фрагменти системи

Конфігурація ASGI для проєкту

Цей фрагмент визначає точку входу для асинхронного серверного шлюзу (ASGI) проєкту «Case Space». Він налаштовує ASGI-додаток, який дозволяє обробляти асинхронні запити, що є важливим для сучасних веб-додатків із підтримкою WebSocket або інших асинхронних функцій. Код встановлює змінну середовища для модуля налаштувань і створює ASGI-додаток, який інтегрується з Django. Для "Case Space" це забезпечує можливість розгортання проєкту на асинхронних серверах, таких як Uvicorn, що підвищує продуктивність при обробці великої кількості запитів.

Фрагмент коду виглядає наступним чином (рис. 3.10):

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						44
Змн.	Арк.	№ докум.	Підпис	Дата		

```

"""
ASGI config for case_space project.

It exposes the ASGI callable as a module-level variable named ``application``.

For more information on this file, see
https://docs.djangoproject.com/en/5.2/howto/deployment/asgi/
"""

import os

from django.core.asgi import get_asgi_application

os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'case_space.settings')

application = get_asgi_application()

```

Рис. 3.10 – Конфігурація ASGI для проєкту

Налаштування Django-проєкту

Цей код є основним файлом налаштувань для Django-проєкту «Case Space». Він визначає ключові параметри системи, такі як секретний ключ, налаштування бази даних PostgreSQL, підключення додатків, middleware, конфігурацію автентифікації через JWT і django-allauth, а також інтеграцію з платіжною системою WayForPay. Код забезпечує підтримку соціальної автентифікації через Google, CORS для фронтенду на локальному сервері, управління статичними та медіафайлами, а також локалізацію. Для «Case Space» ці налаштування створюють основу для роботи всіх компонентів, від автентифікації користувачів до обробки замовлень і платежів, адаптованих до українського ринку.

Фрагмент коду виглядає наступним чином (рис. 3.11):

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						45
Змн.	Арк.	№ докум.	Підпис	Дата		

```

"""
Django settings for case_space project.

Generated by 'django-admin startproject' using Django 5.2.1.

For more information on this file, see
https://docs.djangoproject.com/en/5.2/topics/settings/

For the full list of settings and their values, see
https://docs.djangoproject.com/en/5.2/ref/settings/
"""

import os
from pathlib import Path
from dotenv import load_dotenv
from datetime import timedelta

load_dotenv()

# Build paths inside the project like this: BASE_DIR / 'subdir'.
BASE_DIR = Path(__file__).resolve().parent.parent

# Quick-start development settings - unsuitable for production
# See https://docs.djangoproject.com/en/5.2/howto/deployment/checklist/

# SECURITY WARNING: keep the secret key used in production secret!
SECRET_KEY = "django-insecure-^fmy5ekr$@y=#acwcj@tl95%0320wowop#89%q$^&pz9e$2yp!"

# SECURITY WARNING: don't run with debug turned on in production!
DEBUG = True

```

Рис. 3.11 – Налаштування Django-проєкту

Налаштування URL-маршрутів проєкту

Цей фрагмент визначає основну маршрутизацію URL для проєкту «Case Space». Він налаштовує маршрути для API-ендпоінтів, адміністративної панелі, автентифікації та інтеграції з платіжними системами. Код використовує роутер Django REST Framework для автоматичної генерації маршрутів для ресурсів, таких як замовлення, продукти, постачальники та відгуки, а також включає специфічні маршрути для ініціації платежів, обробки callback від WayForPay і соціальної автентифікації через Google. В режимі налагодження додається підтримка медіафайлів. Для «Case Space» це забезпечує централізований доступ до всіх функціональних компонентів системи через API.

Фрагмент коду виглядає наступним чином (рис. 3.12):

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						46
Змн.	Арк.	№ докум.	Підпис	Дата		

```

"""
URL configuration for case_space project.

The `urlpatterns` list routes URLs to views. For more information please see:
    https://docs.djangoproject.com/en/5.2/topics/http/urls/
Examples:
Function views
    1. Add an import:  from my_app import views
    2. Add a URL to urlpatterns:  path('', views.home, name='home')
Class-based views
    1. Add an import:  from other_app.views import Home
    2. Add a URL to urlpatterns:  path('', Home.as_view(), name='home')
Including another URLconf
    1. Import the include() function: from django.urls import include, path
    2. Add a URL to urlpatterns:  path('blog/', include('blog.urls'))
"""

from django.contrib import admin
from django.urls import path, include
from rest_framework.routers import DefaultRouter
from orders.views import OrderViewSet, ProductOrderViewSet
from products.views import (
    DeviceModelViewSet,
    ProductViewSet,
    ProductImageViewSet,
    GetMaterialsView,
)
from suppliers.views import SupplierViewSet
from reviews.views import ReviewsViewSet
from payments.views import InitiatePaymentView, WayForPayCallbackView
from accounts.views import GoogleLogin

```

Рис. 3.12 – Налаштування URL-маршрутів проєкту

Конфігурація WSGI для проєкту

Цей код налаштовує точку входу для WSGI-сервера (Web Server Gateway Interface) проєкту «Case Space». Він створює WSGI-додаток, який дозволяє розгортати проєкт на синхронних серверах, таких як Gunicorn, для обробки HTTP-запитів. Код встановлює змінну середовища для модуля налаштувань і ініціалізує додаток Django. Для «Case Space» це забезпечує сумісність з традиційними серверами для стабільної роботи веб-додатка.

Фрагмент коду виглядає наступним чином (рис. 3.13):

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						47
Змн.	Арк.	№ докум.	Підпис	Дата		

```

"""
WSGI config for case_space project.

It exposes the WSGI callable as a module-level variable named ``application``.

For more information on this file, see
https://docs.djangoproject.com/en/5.2/howto/deployment/wsgi/
"""

import os

from django.core.wsgi import get_wsgi_application

os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'case_space.settings')

application = get_wsgi_application()

```

Рис. 3.13 – Конфігурація WSGI для проєкту

2. Управління замовленнями в інтернет-магазині

Реєстрація моделей замовлень в адмін-панелі

Цей фрагмент забезпечує інтеграцію моделей Order і ProductOrder із адміністративною панеллю Django. Він дозволяє адміністраторам інтернет-магазину «Case Space» керувати замовленнями та їхніми складовими через стандартний інтерфейс адмінки, включаючи перегляд, редагування чи видалення записів. Для магазину це спрощує контроль над процесом обробки замовлень, наприклад, зміну статусу доставки чи перевірку деталей конкретного товару в замовленні.

Фрагмент коду виглядає наступним чином (рис. 3.14):

```

from django.contrib import admin
from .models import Order, ProductOrder

admin.site.register(Order)
admin.site.register(ProductOrder)

```

Рис. 3.14 – Реєстрація моделей замовлень в адмін-панелі

Конфігурація додатка orders

Цей код визначає конфігурацію Django-дodatка orders для проєкту «Case Space». Він встановлює тип автоінкрементного поля для моделей і задає назву додатку, що необхідно для правильної ініціалізації модуля в системі. В контексті

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						48
Змн.	Арк.	№ докум.	Підпис	Дата		

магазину цей фрагмент забезпечує коректну роботу компонентів, пов'язаних з обробкою замовлень, таких як збереження даних про покупки чи зв'язок з продуктами.

Фрагмент коду виглядає наступним чином (рис. 3.15):

```
from django.apps import AppConfig

class OrdersConfig(AppConfig):
    default_auto_field = 'django.db.models.BigAutoField'
    name = 'orders'
```

Рис. 3.15 – Конфігурація додатку orders

Моделі для замовлень

Цей фрагмент, описує моделі даних для управління замовленнями в «Case Space». Модель Order зберігає інформацію про замовлення, включаючи статус, загальну суму, вартість доставки, дані для логістики та контактну інформацію клієнта, з можливістю пов'язування з акаунтом користувача. Модель ProductOrder визначає товари в замовленні, їх кількість та суму, встановлюючи зв'язок з продуктом і замовленням. Для магазину ці моделі є основою для збереження та обробки даних про покупки, наприклад, відстеження статусу доставки чи підрахунку вартості чохлів.

Фрагмент коду виглядає наступним чином (рис. 3.16):

```
from django.db import models
from django.conf import settings
from products.models import Product

class Order(models.Model):
    class OrderStatus(models.TextChoices):
        CREATED = "created", "Створено"
        PAID = "paid", "Оплачено"
        PENDING = "pending", "Очікує підтвердження"
        CONFIRMED = "confirmed", "Підтверджено"
        SHIPPED = "shipped", "Відправлено"
        DELIVERED = "delivered", "Доставлено"
        CANCELLED = "cancelled", "Скасовано"
```

Рис. 3.16 – Моделі для замовлень

Серіалізатори для замовлень

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						49
Змн.	Арк.	№ докум.	Підпис	Дата		

Цей код містить серіалізатори для моделей ProductOrder і Order, які використовуються для обробки даних через API в «Case Space». Серіалізатор ProductOrderSerializer перевіряє коректність суми товару з урахуванням знижок та кількості, забезпечуючи відповідність ціни з базою даних. Серіалізатор OrderSerializer обробляє створення замовлення, включаючи товари, перевіряє унікальність email для неавторизованих користувачів та гарантує правильність загальної суми з урахуванням доставки. В магазині ці серіалізатори забезпечують безпечне оформлення замовлень через API, наприклад, для покупки чохлів із персоналізацією.

Фрагмент коду виглядає наступним чином (рис. 3.17):

```
from django.contrib.auth import get_user_model
from django.db import transaction
from rest_framework.serializers import (
    ModelSerializer,
    ValidationError,
    PrimaryKeyRelatedField,
    CharField,
)
from decimal import Decimal, ROUND_HALF_UP
from .models import ProductOrder, Order
from products.models import Product
from products.serializers import ProductSerializer, MinifiedProductSerializer

Account = get_user_model()

class ProductOrderSerializer(ModelSerializer):
    product_id = PrimaryKeyRelatedField(
        queryset=Product.objects.all(), source="product"
    )
    product = MinifiedProductSerializer(read_only=True)

    class Meta:
        model = ProductOrder
        fields = ["product_id", "product", "quantity", "total_amount"]
```

Рис. 3.17 – Серіалізатори для замовлень

Шаблон для тестів замовлень

Цей фрагмент є порожнім шаблоном для модуля тестів Django, призначеним для перевірки функціоналу додатка orders. Він створює основу для написання тестів, які можуть перевіряти створення замовлень, зміну статусів чи

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						50
Змн.	Арк.	№ докум.	Підпис	Дата		

правильність підрахунку суми. В «Case Space» цей файл дозволяє забезпечити якість коду шляхом автоматизованого тестування логіки замовлень, хоча конкретні тести в цьому фрагменті ще не реалізовані.

Фрагмент коду виглядає наступним чином (рис. 3.18):

```
from django.test import TestCase

# Create your tests here.
```

Рис. 3.18 – Шаблон для тестів замовлень

API-представлення для замовлень

Цей код містить два класи ViewSet для роботи з замовленнями через API в «Case Space». Перший клас обробляє операції з моделлю Order, дозволяючи створювати замовлення без автентифікації, але обмежуючи перегляд і редагування лише авторизованим користувачам або адміністраторам. Другий клас керує моделлю ProductOrder, забезпечуючи API-доступ до товарів в замовленнях. Для магазину ці представлення реалізують функціонал оформлення та перегляду замовлень через API, наприклад, додавання чохлів до кошика чи перевірки історії покупок.

Фрагмент коду виглядає наступним чином (рис. 3.19):

```

from rest_framework.permissions import IsAuthenticated, AllowAny
from rest_framework.viewsets import ModelViewSet
from .serializers import ProductOrderSerializer, OrderSerializer
from .models import Order, ProductOrder

class OrderViewSet(ModelViewSet):
    serializer_class = OrderSerializer
    permission_classes = [IsAuthenticated]
    queryset = Order.objects.all()

    def get_permissions(self):
        if self.action == "create":
            return [AllowAny()]
        elif self.action in ["retrieve", "list"]:
            return [IsAuthenticated()]
        return [IsAuthenticated()]

    def get_queryset(self):
        user = self.request.user
        if user.is_staff:
            return Order.objects.all()
        elif user.is_authenticated:
            return Order.objects.filter(account=user)
        return Order.objects.none()

```

Рис. 3.19 – API-представлення для замовлень

4. Обробка платежів

Шаблон для адмін-панелі платежів

Цей фрагмент є порожнім шаблоном для реєстрації моделей у адміністративній панелі Django. Він призначений для майбутньої інтеграції моделей, пов'язаних із платежами, щоб адміністратори «Case Space» могли керувати транзакціями через інтерфейс адмінки. Наразі код не містить активної функціональності, але створює основу для додавання моделей, наприклад, для зберігання історії платежів.

Фрагмент коду виглядає наступним чином (рис. 3.20):

```

from django.contrib import admin

# Register your models here.

```

Рис. 3.20 – Шаблон для адмін-панелі платежів

Конфігурація додатку payments

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						52
Змн.	Арк.	№ докум.	Підпис	Дата		

Цей код визначає конфігурацію Django-додатка payments для проєкту «Case Space». Він установлює тип автоінкрементного поля для моделей і задає назву додатка, що забезпечує правильну ініціалізацію модуля в системі. Для магазину цей фрагмент є основою для роботи компонентів, пов'язаних з обробкою платежів, таких як інтеграція з WayForPay, забезпечуючи їхню коректну взаємодію з іншими частинами проєкту.

Фрагмент коду виглядає наступним чином (рис. 3.21):

```
from django.apps import AppConfig

class PaymentsConfig(AppConfig):
    default_auto_field = 'django.db.models.BigAutoField'
    name = 'payments'
```

Рис. 3.21 – Конфігурація додатку payments

Шаблон для моделей платежів

Цей фрагмент є порожнім шаблоном для визначення моделей даних, пов'язаних з платежами в «Case Space». Він призначений для майбутнього створення моделей, які могли б зберігати інформацію про транзакції, статуси платежів чи пов'язані дані. Наразі код не містить активних моделей, але створює основу для їхньої реалізації, наприклад, для відстеження платежів через WayForPay.

Фрагмент коду виглядає наступним чином (рис. 3.22):

```
from django.db import models

# Create your models here.
```

Рис. 3.22 – Шаблон для моделей платежів

Шаблон для тестів платежів

Цей код є порожнім шаблоном для модуля тестів Django, призначеним для перевірки функціоналу додатка payments. Він створює основу для написання тестів, які могли б перевіряти генерацію підписів, обробку callback-запитів чи ініціацію платежів. В «Case Space» цей файл дозволяє забезпечити якість коду

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						53
Змн.	Арк.	№ докум.	Підпис	Дата		

шляхом автоматизованого тестування логіки платежів, хоча конкретні тести ще не реалізовані.

Фрагмент коду виглядає наступним чином (рис. 3.23):

```
from django.test import TestCase

# Create your tests here.
```

Рис. 3.23 – Шаблон для тестів платежів

Утиліти для генерації підписів

Цей фрагмент містить функції для створення цифрових підписів, необхідних для безпечної взаємодії з платіжною системою WayForPay в «Case Space». Перша функція генерує підпис для ініціації платежу, об'єднуючи дані в рядок і використовуючи HMAC-MD5 з секретним ключем. Друга функція створює підпис для підтвердження транзакції, включаючи референс замовлення, статус і час. Ці функції забезпечують захист від підробки даних, що є критично важливим для безпеки платежів у магазині.

Фрагмент коду виглядає наступним чином (рис. 3.24):

```
import hmac
import hashlib
from django.utils.encoding import force_bytes

def generate_merchant_signature(data_list, secret_key):
    concatenated_string = ";".join(str(i) for i in data_list)
    signature = hmac.new(
        key=secret_key.encode("utf-8"),
        msg=concatenated_string.encode("utf-8"),
        digestmod=hashlib.md5,
    ).hexdigest()
    return signature

def generate_accept_signature(order_reference, status, timestamp, secret_key):
    data_str = f"{order_reference};{status};{timestamp}"
    return hmac.new(
        force_bytes(secret_key), force_bytes(data_str), hashlib.md5
    ).hexdigest()
```

Рис. 3.24 – Утиліти для генерації підписів

API-представлення для обробки платежів

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						54
Змн.	Арк.	№ докум.	Підпис	Дата		

Цей код містить два API-представлення для інтеграції з платіжною системою WayForPay в «Case Space». Перше представлення обробляє callback-запити від WayForPay, перевіряючи цифровий підпис, оновлюючи статус замовлення до «оплачено» при успішній транзакції, зменшуючи кількість товару в наявності та видаляючи замовлення при відхиленні. Воно повертає підписану відповідь для підтвердження. Друге представлення ініціює платіж, перевіряючи валідність замовлення, формуючи дані для WayForPay з підписом і повертаючи URL для оплати. Для магазину ці представлення забезпечують безпечну обробку платежів, наприклад, для оплати чохлів через картку.

Фрагмент коду виглядає наступним чином (рис. 3.25):

```
import json
import time
import hmac
import hashlib
import logging
from django.conf import settings
from django.utils.encoding import force_bytes
from django.db import transaction
from rest_framework.views import APIView
from rest_framework.response import Response
from rest_framework.permissions import AllowAny
from rest_framework.exceptions import NotFound, PermissionDenied, NotAcceptable
from datetime import datetime
from payments.utils import generate_merchant_signature, generate_accept_signature
from orders.models import Order

logger = logging.getLogger(__name__)

def decode_json(raw_data):
    return json.loads(raw_data)
```

Рис. 3.25 – API-представлення для обробки платежів

5. Управління продуктами в інтернет-магазині

Реєстрація моделей продуктів у адмін-панелі

Цей код забезпечує інтеграцію моделей Product, ProductImage, DeviceModel і Feature з адміністративною панеллю Django. Він дозволяє адміністраторам «Case Space» керувати продуктами, їх зображеннями, характеристиками та моделями через стандартний інтерфейс адмін.-панелі, включаючи додавання нових чохлів, зміну цін чи завантаження зображень. Для

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						55
Змн.	Арк.	№ докум.	Підпис	Дата		

магазину це спрощує управління асортиментом і підтримку актуального каталогу.

Фрагмент коду виглядає наступним чином (рис. 3.26):

```
from django.contrib import admin
from .models import Product, ProductImage, DeviceModel, Feature

admin.site.register(Product)
admin.site.register(ProductImage)
admin.site.register(DeviceModel)
admin.site.register(Feature)
```

Рис. 3.26 – Реєстрація моделей продуктів у адмін-панелі

Конфігурація додатку products

Цей фрагмент визначає конфігурацію Django-дodatка products для проєкту «Case Space». Він встановлює тип автоінкрементного поля для моделей і задає назву додатка, що забезпечує коректну ініціалізацію модуля в системі проєкту. В контексті магазину цей код підтримує роботу компонентів, пов'язаних з каталогом, таких як відображення чохлів чи фільтрація за характеристиками.

Фрагмент коду виглядає наступним чином (рис. 3.27):

```
from django.apps import AppConfig

class ProductsConfig(AppConfig):
    default_auto_field = 'django.db.models.BigAutoField'
    name = 'products'
```

Рис. 3.27 – Конфігурація додатку products

Фільтрація продуктів

Цей код реалізує фільтр для моделі Product через бібліотеку django_filters. Він дозволяє здійснювати пошук і фільтрацію чохлів за назвою, ціною, зниженою ціною, виробником, моделлю пристрою, матеріалом чи постачальником. Для «Case Space» цей фрагмент забезпечує зручну навігацію по каталогу, дозволяючи клієнтам швидко знаходити чохли, наприклад, для конкретного смартфона чи з певного матеріалу, через API-запити.

Фрагмент коду виглядає наступним чином (рис. 3.28):

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						56
Змн.	Арк.	№ докум.	Підпис	Дата		

```

from django_filters import rest_framework as filters
from .models import Product

class ProductFilter(filters.FilterSet):
    title = filters.CharFilter(field_name="title", lookup_expr="icontains")
    price = filters.RangeFilter()
    discounted_price = filters.RangeFilter(field_name="discounted_price")
    manufacturer = filters.CharFilter(
        field_name="device_model__manufacturer", lookup_expr="icontains"
    )
    model = filters.CharFilter(
        field_name="device_model__model", lookup_expr="icontains"
    )
    material = filters.CharFilter()
    supplier = filters.NumberFilter(field_name="supplier")

class Meta:
    model = Product
    fields = [
        "title",
        "model",
        "manufacturer",
        "price",
        "material",
        "supplier",
        "discounted_price",
    ]

```

Рис. 3.28 – Фільтрація продуктів

Моделі даних для продуктів

Цей фрагмент описує моделі даних для управління продуктами в «Case Space». Модель MaterialChoices задає варіанти матеріалів для чохів, таких як силікон чи шкіра. Модель Feature зберігає характеристики, наприклад, ударостійкість. Модель DeviceModel описує моделі пристроїв, для яких призначені чохла. Модель Product містить інформацію про чохол, включаючи назву, постачальника, матеріал, ціну, знижку та доступну кількість, а ProductImage зберігає зображення продукту. Для магазину ці моделі є основою каталогу, дозволяючи деталізувати асортимент.

Фрагмент коду виглядає наступним чином (рис. 3.29):

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						57
Змн.	Арк.	№ докум.	Підпис	Дата		

```

from django.db import models
from django.core.validators import MinValueValidator, MaxValueValidator
from suppliers.models import Supplier

class MaterialChoices(models.TextChoices):
    SILICONE = "silicone", "Силікон"
    TPU = "tpu", "TPU (термополіуретан)"
    PLASTIC = "plastic", "Пластик"
    LEATHER = "leather", "Шкіра"
    FABRIC = "fabric", "Тканина"
    METAL = "metal", "Метал"
    OTHER = "other", "Інший"

class Feature(models.Model):
    name = models.CharField(max_length=100, unique=True)
    description = models.TextField()
    image = models.ImageField(upload_to="media/features_images/", blank=True)

    def __str__(self):
        return self.name

class DeviceModel(models.Model):
    manufacturer = models.CharField(max_length=255)
    model = models.CharField(max_length=255)

    def __str__(self):
        return f"{self.manufacturer} {self.model}"

```

Рис. 3.29 – Моделі даних для продуктів

Серіалізатори для продуктів

Цей код містить серіалізатори для моделей Product, ProductImage, DeviceModel і Feature, які використовуються для обробки даних через API в «Case Space». Серіалізатор ProductSerializer формує детальне представлення продукту, включаючи модель пристрою, зображення, відгуки, середній рейтинг, матеріал і постачальника. Серіалізатор MinifiedProductSerializer надає скорочене представлення для списків, наприклад, в кошику. Інші серіалізатори обробляють зображення, характеристики та моделі пристроїв. Для магазину ці серіалізатори забезпечують відображення чохлів в каталозі через API.

Фрагмент коду виглядає наступним чином (рис. 3.30):

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						58
Змн.	Арк.	№ докум.	Підпис	Дата		

```

from rest_framework.serializers import (
    ModelSerializer,
    FloatField,
    SerializerMethodField,
    IntegerField
)
from .models import Product, ProductImage, DeviceModel, Feature
from reviews.serializers import ReviewSerializer
from suppliers.serializers import ProductSupplierSerializer

class ProductImageSerializer(ModelSerializer):
    class Meta:
        model = ProductImage
        fields = ["id", "image"]

class DeviceModelSerializer(ModelSerializer):
    class Meta:
        model = DeviceModel
        fields = "__all__"

class FeatureSerializer(ModelSerializer):
    class Meta:
        model = Feature
        fields = ["id", "name", "description", "image"]

```

Рис. 3.30 – Серіалізатори для продуктів

Шаблон для тестів продуктів

Цей фрагмент є порожнім шаблоном для модуля тестів Django, призначеним для перевірки функціоналу додатка products. Він створює основу для написання тестів, які могли б перевіряти створення продуктів, фільтрацію чи обчислення знижок. В «Case Space» цей файл дозволяє забезпечити якість коду шляхом автоматизованого тестування логіки каталогу, хоча конкретні тести ще не реалізовані.

Фрагмент коду виглядає наступним чином (рис. 3.31):

```

from django.test import TestCase

# Create your tests here.

```

Рис. 3.31 – Шаблон для тестів продуктів

API-представлення для продуктів

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						59
Змн.	Арк.	№ докум.	Підпис	Дата		

Цей код містить чотири API-представлення для роботи з продуктами через API в «Case Space». Перше представлення керує продуктами, підтримуючи фільтрацію, сортування за рейтингом чи ціною та обчислення знижених цін. Друге представлення обробляє зображення продуктів, дозволяючи додавати чи видаляти картинки. Третє представлення керує моделями пристроїв, включаючи унікальний ендпоінт для отримання списку виробників. Четверте представлення повертає список доступних матеріалів. Для магазину ці представлення забезпечують функціонал каталогу, наприклад, перегляд чохлів чи вибір матеріалів через API.

Фрагмент коду виглядає наступним чином (рис. 3.32):

```
from decimal import Decimal
from django.db.models import Avg, Value, F, ExpressionWrapper, DecimalField, Func, Count
from django.db.models.functions import Coalesce
from rest_framework.permissions import IsAuthenticatedOrReadOnly
from rest_framework.viewsets import ModelViewSet
from rest_framework.views import APIView
from rest_framework.decorators import action
from rest_framework.response import Response
from .serializers import (
    ProductImageSerializer,
    ProductSerializer,
    DeviceModelSerializer,
)
from .models import Product, ProductImage, DeviceModel, MaterialChoices
from .filters import ProductFilter

class ProductViewSet(ModelViewSet):
    queryset = Product.objects.all()
    serializer_class = ProductSerializer
    permission_classes = [IsAuthenticatedOrReadOnly]
    filterset_class = ProductFilter
    ordering_fields = [
        "average_score",
        "title",
        "price",
        "amount_available",
        "discount",
        "discounted_price",
    ]
    ordering = ["-average_score"]
```

Рис. 3.32 – Шаблон для тестів продуктів

6. Управління відгуками в інтернет-магазині

Реєстрація моделі відгуків у адмін-панелі

Цей код забезпечує інтеграцію моделі Review з адміністративною панеллю

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						60
Змн.	Арк.	№ докум.	Підпис	Дата		

Django. Він дозволяє адміністраторам «Case Space» керувати відгуками про продукти через стандартний інтерфейс адмін-панелі, включаючи перегляд, редагування чи видалення коментарів користувачів. Для магазину це спрощує модерацію відгуків, наприклад, для перевірки їхньої відповідності правилам або видалення некоректних оцінок чохлів.

Фрагмент коду виглядає наступним чином (рис. 3.33):

```
from django.contrib import admin
from .models import Review

admin.site.register(Review)
```

Рис. 3.33 – Реєстрація моделі відгуків у адмін-панелі

Конфігурація додатку reviews

Цей фрагмент визначає конфігурацію Django-дodatка reviews для проєкту "Case Space". Він установлює тип автоінкрементного поля для моделей і задає назву додатка, що забезпечує правильну ініціалізацію модуля в системі проєкту. У контексті магазину цей код підтримує роботу компонентів, пов'язаних із відгуками, таких як збереження оцінок чи відображення коментарів на сторінці продукту.

Фрагмент коду виглядає наступним чином (рис. 3.34):

```
from django.contrib import admin
from .models import Review

admin.site.register(Review)
```

Рис. 3.34 – Конфігурація додатку reviews

Модель відгуків

Цей код описує модель Review для зберігання відгуків в «Case Space». Модель містить інформацію про автора, продукт, дату створення, заголовок, текст відгуку та оцінку від 0 до 5. Вона пов'язує відгук з користувачем і продуктом, забезпечуючи каскадне видалення при видаленні пов'язаних об'єктів. Для магазину ця модель є основою для збереження оцінок чохлів, дозволяючи клієнтам ділитися думками та впливати на рейтинг продуктів.

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						61
Змн.	Арк.	№ докум.	Підпис	Дата		

Фрагмент коду виглядає наступним чином (рис. 3.35):

```
from django.apps import AppConfig

class ReviewsConfig(AppConfig):
    default_auto_field = 'django.db.models.BigAutoField'
    name = 'reviews'
```

Рис. 3.35 – Модель відгуків

Серіалізатор для відгуків

Цей фрагмент містить серіалізатор для моделі Review, який використовується для обробки даних відгуків через API в «Case Space». Серіалізатор включає поля для дати створення, автора, заголовка, тексту, оцінки та продукту, при цьому автор і дата є лише для читання. Він використовує серіалізатор з модуля accounts для відображення імені та прізвища автора. Для магазину цей серіалізатор забезпечує відображення відгуків на сторінці продукту та створення нових коментарів через API.

Фрагмент коду виглядає наступним чином (рис. 3.36):

```
from rest_framework.serializers import ModelSerializer
from .models import Review
from accounts.serializers import ReviewAccountSerializer

class ReviewSerializer(ModelSerializer):
    author = ReviewAccountSerializer(read_only=True)

    class Meta:
        model = Review
        fields = ["created_at", "author", "title", "description", "score", "product"]
        read_only_fields = ["created_at", "author"]
```

Рис. 3.36 – Серіалізатор для відгуків

Шаблон для тестів відгуків

Цей код є порожнім шаблоном для модуля тестів Django, призначеним для перевірки функціоналу додатка reviews. Він створює основу для написання тестів, які могли б перевіряти створення відгуків, валідацію оцінок чи відображення коментарів. В «Case Space» цей файл дозволяє забезпечити якість

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						62
Змн.	Арк.	№ докум.	Підпис	Дата		

коду шляхом автоматизованого тестування логіки відгуків, хоча конкретні тести ще не реалізовані.

Фрагмент коду виглядає наступним чином (рис. 3.37):

```
from django.test import TestCase

# Create your tests here.
```

Рис. 3.37 – Шаблон для тестів відгуків

API-представлення для відгуків

Цей фрагмент містить API-представлення для роботи з відгуками через Django REST Framework в «Case Space». Воно дозволяє створювати, переглядати, редагувати чи видаляти відгуки, з обмеженням на редагування лише для авторизованих користувачів, тоді як перегляд доступний усім. Під час створення відгуку автоматично встановлюється поточний користувач як автор. Для магазину це представлення забезпечує функціонал додавання відгуків про чохли та їх відображення через API.

Фрагмент коду виглядає наступним чином (рис. 3.38):

```
from rest_framework.permissions import IsAuthenticatedOrReadOnly
from rest_framework.viewsets import ModelViewSet
from .models import Review
from .serializers import ReviewSerializer

class ReviewsViewSet(ModelViewSet):
    permission_classes = [IsAuthenticatedOrReadOnly]
    queryset = Review.objects.all()
    serializer_class = ReviewSerializer

    def perform_create(self, serializer):
        serializer.save(author=self.request.user)
```

Рис. 3.38 – Шаблон для тестів відгуків

7. Управління постачальниками в інтернет-магазині

Реєстрація моделі постачальника в адмін-панелі

Цей код забезпечує інтеграцію моделі Supplier з адміністративною панеллю Django. Він дозволяє адміністраторам «Case Space» керувати даними про постачальників через стандартний інтерфейс адмін-панелі, включаючи

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						63
Змн.	Арк.	№ докум.	Підпис	Дата		

створення, редагування чи видалення записів. Для магазину це спрощує управління інформацією про компанії, які постачають чохли, забезпечуючи контроль над асортиментом і логістикою.

Фрагмент коду виглядає наступним чином (рис. 3.39):

```
from django.contrib import admin
from .models import Supplier

admin.site.register(Supplier)
```

Рис. 3.39 – Реєстрація моделі постачальника в адмін-панелі

Конфігурація додатку suppliers

Цей фрагмент визначає конфігурацію Django-дodatка suppliers для проєкту «Case Space». Він встановлює тип автоінкрементного поля для моделей і задає назву додатку, що забезпечує коректну ініціалізацію модуля в системі. В контексті магазину цей код підтримує функціонал, пов'язаний з постачальниками, наприклад, відображення їх у каталозі продуктів чи зв'язок із чохлами.

Фрагмент коду виглядає наступним чином (рис. 3.40):

```
from django.apps import AppConfig

class SuppliersConfig(AppConfig):
    default_auto_field = 'django.db.models.BigAutoField'
    name = 'suppliers'
```

Рис. 3.40 – Конфігурація додатку suppliers

Модель постачальника

Цей код описує модель Supplier для зберігання інформації про постачальників в «Case Space». Модель містить єдине поле для назви компанії, що дозволяє ідентифікувати постачальника. Вона є основою для зв'язку з продуктами, забезпечуючи відстеження походження чохлів. Для магазину ця модель спрощує організацію даних про партнерів, які постачають товари, підтримуючи логістичні процеси.

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						64
Змн.	Арк.	№ докум.	Підпис	Дата		

Фрагмент коду виглядає наступним чином (рис. 3.41):

```
from django.db import models

class Supplier(models.Model):
    name = models.CharField(max_length=255)

    def __str__(self):
        return self.name
```

Рис. 3.41 – Модель постачальника

Серіалізатори для постачальників

Цей фрагмент містить два серіалізатори для моделі Supplier, які використовуються для обробки даних через API в «Case Space». Перший серіалізатор включає всі поля моделі, забезпечуючи повне представлення постачальника для API-запитів. Другий серіалізатор є скороченим і відображає лише назву постачальника, що використовується для вбудованих даних в серіалізаторах продуктів. Для магазину ці серіалізатори дозволяють відображати інформацію про постачальників у каталозі чи адмін-панелі через API.

Фрагмент коду виглядає наступним чином (рис. 3.42):

```
from rest_framework.serializers import ModelSerializer
from .models import Supplier

class SupplierSerializer(ModelSerializer):
    class Meta:
        model = Supplier
        fields = "__all__"

class ProductSupplierSerializer(ModelSerializer):
    class Meta:
        model = Supplier
        fields = ["name"]
```

Рис. 3.42 – Серіалізатори для постачальників

Шаблон для тестів постачальників

Цей код є порожнім шаблоном для модуля тестів Django, призначеним для перевірки функціоналу додатку suppliers. Він створює основу для написання тестів, які могли б перевіряти створення постачальників чи їхній зв'язок із

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						65
Змн.	Арк.	№ докум.	Підпис	Дата		

продуктами. В «Case Space» цей файл забезпечує можливість автоматизованого тестування для гарантії якості коду, хоча конкретні тести ще не реалізовані.

Фрагмент коду виглядає наступним чином (рис. 3.43):

```
from django.test import TestCase

# Create your tests here.
```

Рис. 3.43 – Серіалізатори для постачальників

API-представлення для постачальників

Цей фрагмент містить API-представлення для роботи з постачальниками через Django REST Framework в «Case Space». Воно дозволяє створювати, переглядати, редагувати чи видаляти постачальників, з обмеженням на редагування лише для авторизованих користувачів, тоді як перегляд доступний всім. Без пагінації, воно повертає повний список постачальників. Для магазину це представлення забезпечує функціонал управління даними постачальників через API, наприклад, для додавання нових партнерів.

Фрагмент коду виглядає наступним чином (рис. 3.44):

```
from rest_framework.permissions import IsAuthenticatedOrReadOnly
from rest_framework.viewsets import ModelViewSet
from .serializers import SupplierSerializer
from .models import Supplier

class SupplierViewSet(ModelViewSet):
    serializer_class = SupplierSerializer
    permission_classes = [IsAuthenticatedOrReadOnly]
    pagination_class = None
    queryset = Supplier.objects.all()
```

Рис. 3.44 – API-представлення для постачальників

Наданий опис дає можливість оцінити розробку інтернет-магазину Case Space та в подальшому продемонструвати його роботу.

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						66
Змн.	Арк.	№ докум.	Підпис	Дата		

3.2. Демонстрація результатів роботи розробленого інтернет-магазину Case Space

Після проведення опису фрагментів коду розробки інтернет-магазину потрібно провести тестування розробленого продукту.

Користувач, зайшовши на сайт інтернет-магазину, переходить на головну сторінку яка має наступний вигляд (рис. 3.45):

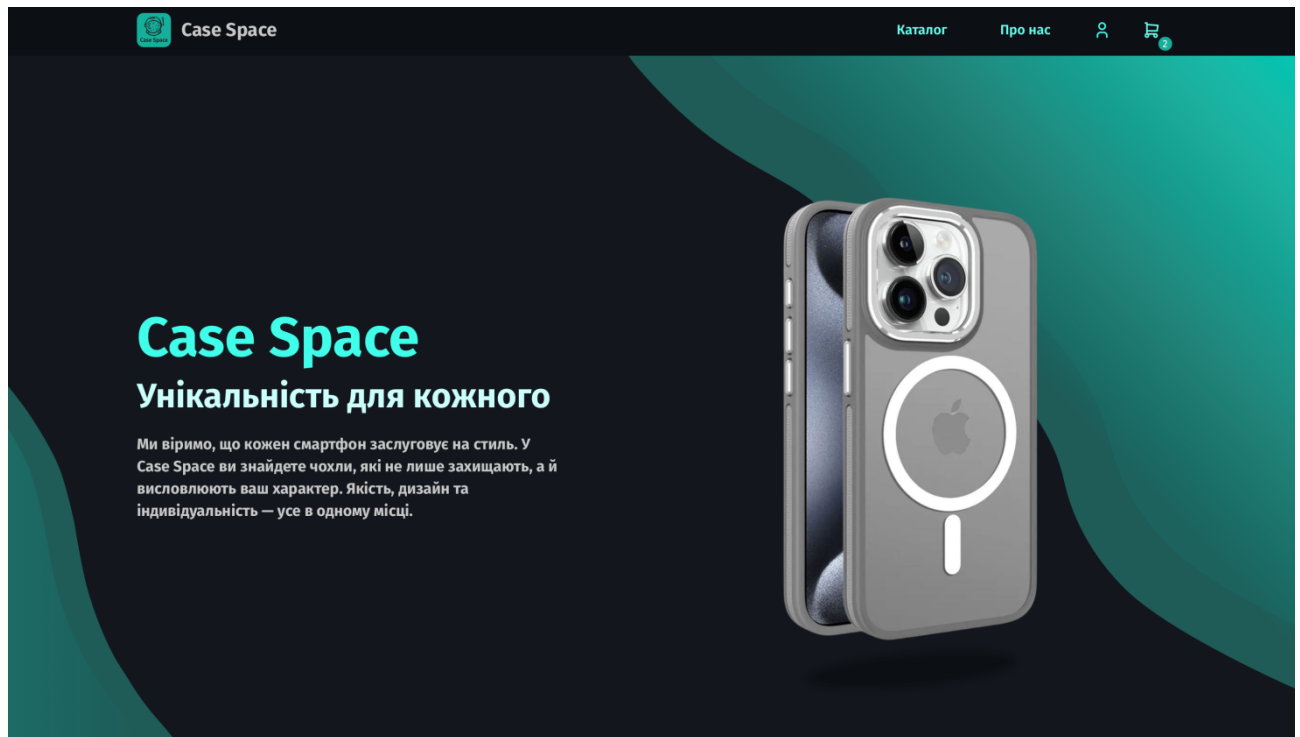


Рис. 3.45 – Головна сторінка інтернет-магазину Case Space

Натиснувши на кнопку «Каталог», користувач переходить до каталогу товарів, який виглядає наступним чином (рис. 3.46):

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						67
Змн.	Арк.	№ докум.	Підпис	Дата		

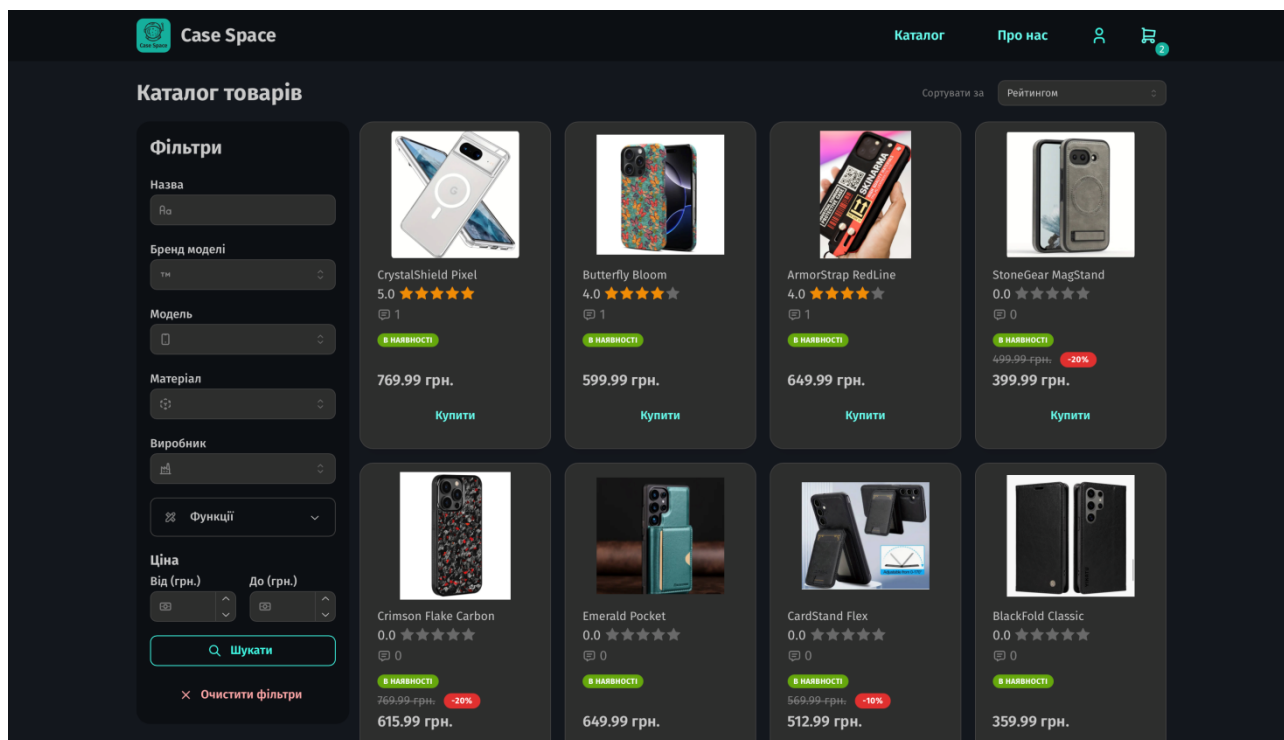


Рис. 3.46 – Каталог товарів інтернет-магазину Case Space

Використавши фільтр, користувач обирає потрібний йому чохол, наприклад (рис. 3.47):

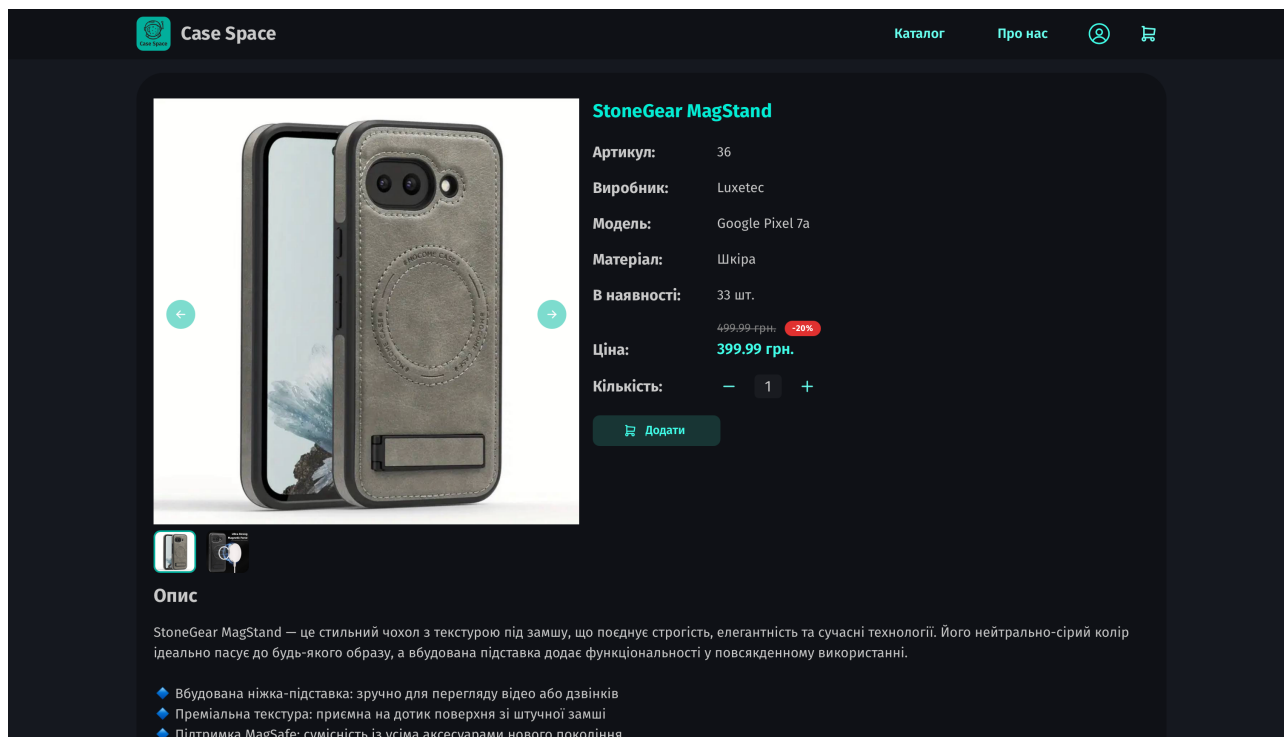


Рис. 3.47 – Характеристики обраного товару

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						68
Змн.	Арк.	№ докум.	Підпис	Дата		

Після ознайомлення з характеристиками обраного товару, користувач додає товари до кошику (рис. 3.48):

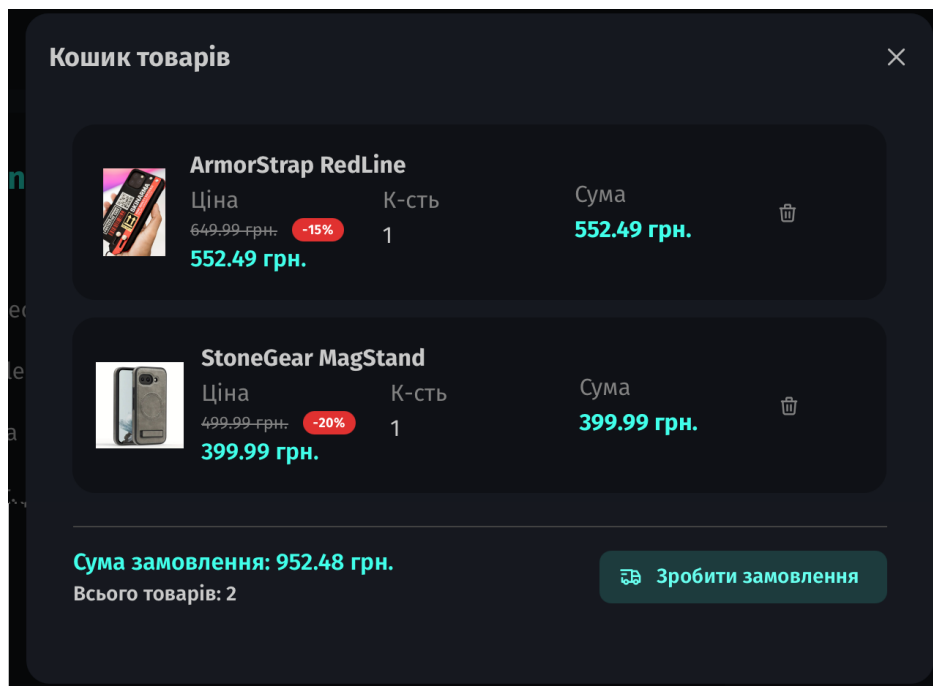


Рис. 3.48 – Обрані товари в кошику

Далі, користувачу пропонується оформити замовлення, після переходу на відповідну сторінку (рис. 3.49):

Рис. 3.49 – Сторінка оформлення замовлення

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						69
Змн.	Арк.	№ докум.	Підпис	Дата		

Після заповнення особистих даних, користувачу пропонується обрати місце, куди потрібно доставити товар (рис. 3.50):

The screenshot shows a dark-themed mobile application interface. At the top, there is a section titled "Населений пункт" (Settlement) with a dropdown menu showing "Житомир" (Zhytomyr). Below this is a section titled "Відділення" (Branch) with a dropdown menu showing "Оберіть відділення" (Select branch). A list of delivery branches is displayed below the dropdown, each with its address: "Відділення №1: вул. Гранітна, 16", "Відділення №2 (до 30 кг на одне місце): вул. Небесної Сотні, 52", "Відділення №3 (до 30 кг на одне місце): майдан Перемоги, 10", "Відділення №4 (до 30 кг): вул. Чуднівська, 113А", "Відділення №5 (до 30 кг): вул. Б. Тена, 9", "Відділення №6 (до 30 кг): вул. Київська, 104", and "Відділення №8: вул. Металістів, 2 (заїзд з вул. Сергія Параджанова 66)".

Рис. 3.50 – Вибір відділення Нової Пошти для доставки чохлів

Після заповнення всіх необхідних даних, замовник переходить до оплати замовлення (рис. 3.51):

The screenshot shows a two-panel online payment interface. The left panel, titled "case-space.com.ua", lists the items: "ArmorStrap RedLine (1.00)" and "StoneGear MagStand (1.00)". It shows the total amount to be paid: "До сплати: 2 UAH". Below this is a QR code and a button labeled "Оплатити з мобільного" (Pay with mobile). The right panel offers payment options: "Оплатити з Apple Pay" and "Оплатити через Google Pay". Below these are logos for VISA, MasterCard, and Google Pay. A form for card payment is also present, with fields for "Номер картки" (Card number), "ММ / РР" (MM / RR), "ПІБ власника" (Cardholder name), and "CVV2". A "Телефон" (Phone) field is also visible. A blue "Оплатити" (Pay) button is at the bottom.

Рис. 3.51 – Сторінка online-оплати замовлення

Після оплати, замовник отримує підтвердження про те, що його замовлення сформовано та оплачено (рис. 3.52):

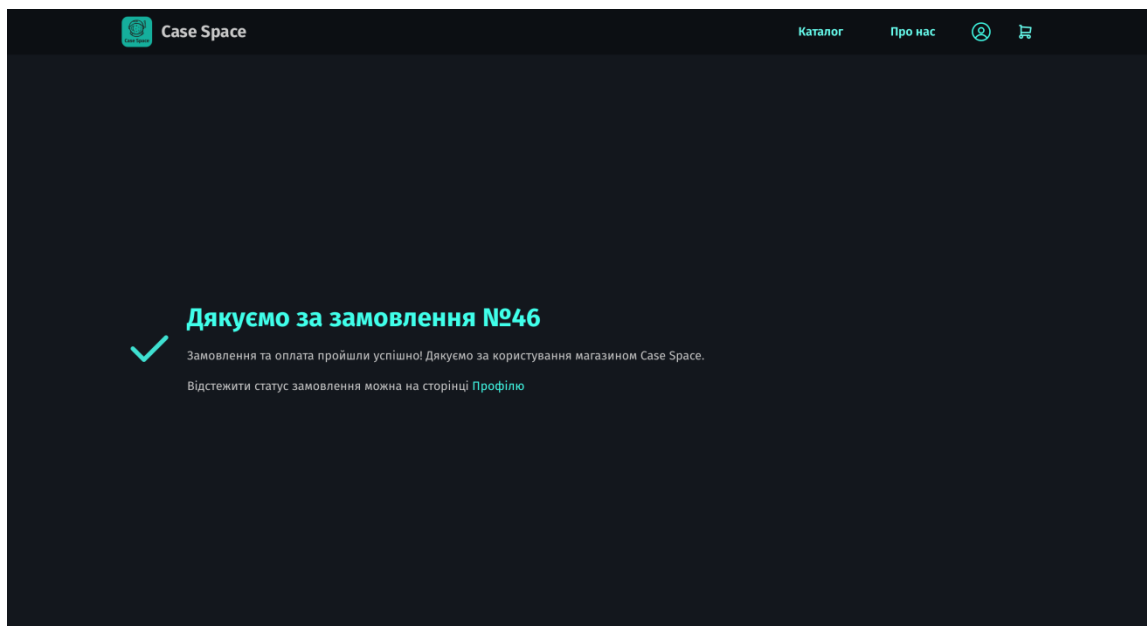


Рис. 3.52 – Підтвердження замовлення

Після того, як замовник отримав інформацію про те, що замовлення успішно сформовано та оплачено, він може відстежити статус замовлення (рис. 3.53):

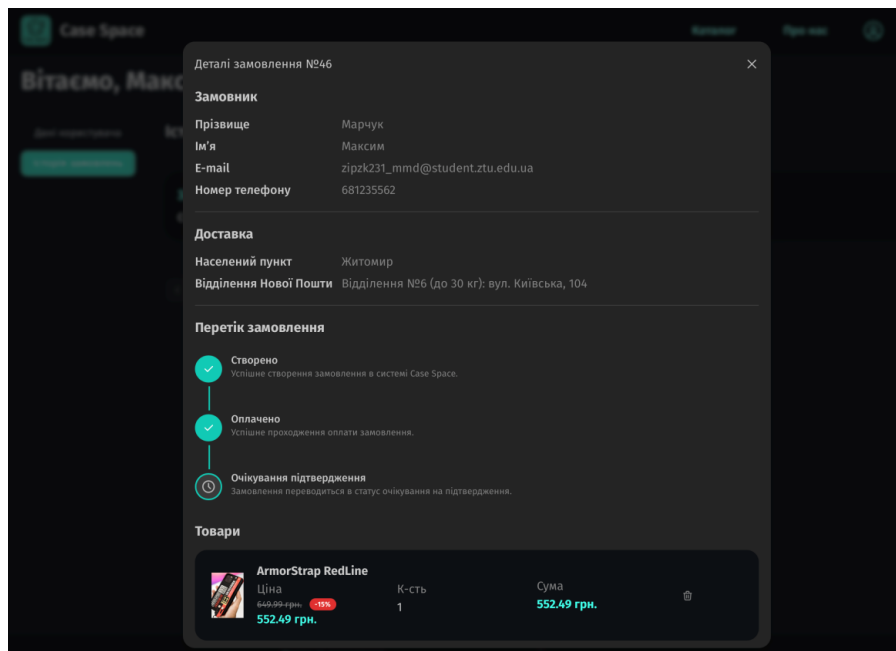


Рис. 3.53 – Відстежування статусу замовлення

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						71
Змн.	Арк.	№ докум.	Підпис	Дата		

Отримавши своє замовлення, замовник може залишити відгук про отриманий чохол (рис. 3.54):

Відгуки користувачів

Залишити відгук

Оцінка ★★★★★

Заголовок

Відгук

Надіслати

Всього відгуків: 1

Оцінка від користувачів: 4.0/5 ★★★★★

Антон Панасенко
 Чудова якість за ці кошти
 09.06.2025, 12:31:58
 ★★★★★
 Купував чохол для сина в подарунок, все дуже сподобалось. Дійсно якісний матеріал який приємно тримати в руці, також має зручний карабін і що важливо чохол має приємну вагу та гарно сидить в руці за формою.

Рис. 3.54 – Відгук про отриманий товар

Таким чином, отримані результати свідчать про те, що розроблений інтернет-магазин Case Space є повністю функціональним, та відповідає всім сучасним вимогам в сфері електронної комерції.

Висновок до розділу 3

В третьому розділі виконано детальний аналіз структури та функціоналу backend-частини інтернет-магазину «Case Space», орієнтованого на продаж чохлів для телефонів. Через опис фрагментів коду розкрито ключові аспекти реалізації системи. Модуль accounts забезпечує автентифікацію користувачів через Email і Google, з кастомною моделлю для зберігання особистих даних. Модуль orders відповідає за оформлення замовлень, їх валідацію та управління статусами, включаючи інтеграцію з логістикою. Модуль payments реалізує безпечну обробку транзакцій через WayForPay, використовуючи цифрові підписи для захисту даних. Модуль products формує каталог товарів з

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						72
Змн.	Арк.	№ докум.	Підпис	Дата		

підтримкою фільтрації, знижок і характеристик, тоді як reviews дозволяє користувачам залишати відгуки, впливаючи на рейтинг продуктів. Модуль suppliers організує дані про постачальників, забезпечуючи зв'язок з асортиментом. Всі модулі побудовані на Django з REST Framework, що гарантує модульність, безпеку та масштабованість.

Демонстрація роботи магазину на прикладі здійснення замовлення показала, як користувач може зареєструватися, обрати чохол, оформити замовлення, оплатити його через WayForPay і залишити відгук. Цей приклад підтвердив інтеграцію всіх компонентів системи, від автентифікації до обробки платежів, і довів її практичну застосовність для українського ринку. Таким чином, розроблена система відповідає сучасним вимогам до e-commerce, забезпечуючи зручність для користувачів і ефективність для адміністраторів.

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						73
Змн.	Арк.	№ докум.	Підпис	Дата		

ВИСНОВКИ

Під час виконання кваліфікаційної роботи було розроблено інтернет-магазин «Case Space» для продажу чохлів для телефонів, який є адаптованим до потреб українського ринку електронної комерції. В процесі виконання роботи здійснено комплексне дослідження та практичну реалізацію системи, що охоплює аналіз, проектування та втілення.

Предметна область інтернет-магазинів, зокрема в сегменті аксесуарів для мобільних пристроїв, була детально проаналізована, визначено її особливості та проведено порівняння з аналогами. Це дозволило сформулювати унікальні характеристики «Case Space», такі як інтеграція з локальною платіжною системою Way For Pay і підтримка соціальної автентифікації через Google, що забезпечує зручність і надійність для користувачів.

Для технічної реалізації обрано Python із фреймворком Django для backend завдяки їхній надійності, безпеці та підтримці REST API, що ідеально відповідає вимогам e-commerce. Фронтенд розроблено на React з використанням Vite для швидкості, Zustand для управління станом і Mantine для сучасного інтерфейсу. Архітектура системи, побудована на монолітному підході, забезпечує цілісність, спрощену розробку та єдину точку управління для всіх компонентів, таких як автентифікація, каталог, замовлення, платежі, відгуки та управління постачальниками.

Процес розробки «Case Space» охопив створення модулів для автентифікації користувачів, обробки замовлень, безпечних платежів, управління каталогом із фільтрацією, відгуками та постачальниками. Аналіз фрагментів коду продемонстрував їх функціональність та взаємодію. Практична демонстрація роботи магазину на прикладі оформлення замовлення підтвердила коректну роботу всіх компонентів: від реєстрації користувача до оплати та залишення відгуку, підкреслюючи зручність, безпеку та ефективність системи.

Розроблений інтернет-магазин «Case Space» є конкурентоспроможним рішенням, яке поєднує сучасні технології та орієнтацію на користувача.

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						74
Змн.	Арк.	№ докум.	Підпис	Дата		

Результати роботи можуть бути основою для подальшого розвитку, наприклад, через додавання функцій персоналізації чохлів чи інтеграцію з іншими платіжними та логістичними постачальниками.

Таким чином, кваліфікаційна робота досягла мети створення практичного та функціонального рішення для онлайн-торгівлі, що відповідає сучасним стандартам e-commerce.

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						75
Змн.	Арк.	№ докум.	Підпис	Дата		

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Alam, F. Integrating Payment Gateways in Django: A Guide for E-Commerce Projects. Medium, 2024
2. Alotaibi, M. B. E-Commerce Adoption in Saudi Arabia: An Empirical Study. Journal of Advanced Research in Business and Management Studies, 2021, 25(1), 12-20
3. Asad, M. M., Khan, M. A. Blockchain Technology in E-Commerce: A Case Study of Supply Chain Transparency. International Journal of Business Information Systems, 2023, 42(2), 189-204
4. Bhatti, A., Akram, H. The Moderating Role of Social Media in E-Commerce: A Study on Consumer Purchase Intention. Journal of Retailing and Consumer Services, 2022, 65, 102886
5. Chen, Y., Zhang, L. Mobile Payment Systems and Their Impact on E-Commerce Efficiency. Electronic Commerce Research, 2024, 24(3), 567-582
6. Chmielarz, W., Zborowski, M. Trends in E-Commerce Development in Poland: A Consumer Perspective. Journal of Theoretical and Applied Electronic Commerce Research, 2023, 18(1), 234-249
7. Dastane, O., Goh, M. L. Factors Influencing Consumer Trust in E-Commerce Platforms. Asia Pacific Journal of Marketing and Logistics, 2021, 33(10), 2125-2142
8. Emerging Blockchain Integrated E-Commerce using Django. International Journal of Scientific Research in Science and Technology, 2024, 11(3), 45-52
9. Gupta, S., Kumar, V. The Role of AI in Enhancing E-Commerce Customer Experience. Journal of Business Research, 2023, 155, 113456
10. Hasan, M., Popp, J. The Impact of Digital Transformation on E-Commerce Performance. Sustainability, 2022, 14(5), 2756

					ІІЗ.КР.Б – 121 – 25 – ІІЗ	Арк.
						76
Змн.	Арк.	№ докум.	Підпис	Дата		

11. Kim, J., Lee, H. C. User Experience Design in E-Commerce Websites: A Case Study of UX Optimization. International Journal of Human-Computer Interaction, 2024, 40(4), 987-1002
12. Li, X., Wang, Y. Security Challenges in E-Commerce Payment Systems: A Review. Computers and Security, 2023, 126, 103089
13. Lim, W. M., Yap, S. F. The Impact of Mobile Commerce on Retail: Trends and Challenges. Journal of Retailing and Consumer Services, 2021, 63, 102742
14. Liu, Z., Zhang, Q. The Role of APIs in Modern E-Commerce Architectures. Journal of Systems and Software, 2024, 199, 111632
15. Patel, R., Sharma, S. Scalable Web Development with Django and React: A Case Study. International Journal of Web Engineering and Technology, 2023, 18(2), 134-150
16. Shaleva, O. I. Trends in the Development of E-Commerce in Ukraine. Springer Proceedings in Business and Economics, 2024, 89-102
17. Singh, A., Rana, N. P. Consumer Behavior in Online Retail: A Study of Trust and Loyalty. Electronic Markets, 2022, 32(3), 1567-1582
18. Wang, Q., Zhao, H. Integrating Payment Gateways in Web Applications: Challenges and Solutions. Journal of Web Engineering, 2023, 22(4), 501-518
19. Xu, J., Zhang, Y. The Role of Microservices in E-Commerce Scalability. IEEE Transactions on Software Engineering, 2024, 50(6), 1234-1249
20. Zhang, L., Chen, H. Enhancing E-Commerce with React and State Management Tools. Journal of Software Engineering Research and Development, 2023, 11(1), 45-60
21. Бойко, О. В., Яценко, О. М. Електронна комерція в Україні: сучасні тенденції та перспективи розвитку. Вісник Київського національного університету імені Тараса Шевченка. Економіка, 2023, 2(215), 45-52
22. Гнатів, І. М. Розвиток платіжних систем в умовах цифровізації економіки України. Фінансово-кредитна діяльність: проблеми теорії та практики, 2022, 4(41), 78-86

					ІІЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						77
Змн.	Арк.	№ докум.	Підпис	Дата		

23. Дорошенко, А. О. Тенденції розвитку цифрових платіжних систем в Україні. Фінанси України, 2022, 7, 89-97
24. Кравець, В. М., Шевчук, О. А. Розвиток електронної комерції в умовах цифровізації економіки. Економічний простір, 2023, 184, 112-120
25. Литвиненко, А. О. Вплив цифрових технологій на розвиток електронної комерції в Україні. Економіка та держава, 2021, 12, 54-60
26. Мельник, Т. Г. Сучасні тенденції розвитку ринку електронної комерції в Україні. Вісник Хмельницького національного університету, 2022, 3(305), 98-104
27. Синявська, О. Є. Електронна комерція в Україні: тенденції та перспективи розвитку. Економічний вісник, 2021, 4(18), 123-130
28. Ткачук, В. О. Інтернет-магазини в Україні: правові аспекти та перспективи розвитку. Юридичний вісник, 2023, 2(67), 145-152
29. Шкригун, Ю. О. Е-бізнес, е-комерція та електронна торгівля: відмінності та особливості. Економіка та суспільство, 2022, 38, 1-8
30. Яворська, Н. П. Цифрові платіжні системи в електронній комерції: український контекст. Вісник Львівського університету. Серія економічна, 2023, 62, 78-85

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						78
Змн.	Арк.	№ докум.	Підпис	Дата		

ДОДАТКИ

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						79
Змн.	Арк.	№ докум.	Підпис	Дата		

Кастомна модель користувача та її менеджер

```

from django.contrib.auth.models import AbstractUser
from django.db import models
from django.contrib.auth.models import BaseUserManager

class AccountManager(BaseUserManager):
    def create_user(self, email, password=None, **extra_fields):
        if not email:
            raise ValueError("The Email field is required")
        email = self.normalize_email(email)
        user = self.model(email=email, **extra_fields)
        user.set_password(password)
        user.save(using=self._db)
        return user

    def create_superuser(self, email, password=None, **extra_fields):
        extra_fields.setdefault("is_staff", True)
        extra_fields.setdefault("is_superuser", True)

        if not extra_fields.get("is_staff"):
            raise ValueError("Superuser must have is_staff=True.")
        if not extra_fields.get("is_superuser"):
            raise ValueError("Superuser must have is_superuser=True.")

        return self.create_user(email, password, **extra_fields)

class Account(AbstractUser):
    email = models.EmailField(unique=True)
    first_name = models.CharField(max_length=255)
    last_name = models.CharField(max_length=255)
    fathers_name = models.CharField(max_length=255)
    phone_number = models.CharField(max_length=20)
    city = models.CharField(max_length=255, blank=True, null=True)
    country = models.CharField(max_length=255, blank=True, null=True)
    post_code = models.CharField(max_length=25, blank=True, null=True)
    username = None

    USERNAME_FIELD = "email"
    REQUIRED_FIELDS = []

    objects = AccountManager()

    def __str__(self):
        return self.email

```

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						80
Змн.	Арк.	№ докум.	Підпис	Дата		

API-представлення для автентифікації

```

from rest_framework.views import APIView
from rest_framework.permissions import IsAuthenticated
from rest_framework.response import Response
from rest_framework import status
from rest_framework_simplejwt.tokens import RefreshToken
from dj_rest_auth.registration.views import SocialLoginView
from allauth.socialaccount.providers.google.views import GoogleOAuth2Adapter

from .serializers import AccountSerializer
from .clients import FixedOAuth2Client

class GetAccount(APIView):
    permission_classes = [IsAuthenticated]

    def get(self, request):
        user = request.user

        return Response(
            {
                "email": user.email,
                "first_name": user.first_name,
                "last_name": user.last_name,
                "fathers_name": user.fathers_name,
                "phone_number": user.phone_number,
                "city": user.city,
                "country": user.country,
                "post_code": user.post_code,
            },
            status=status.HTTP_200_OK,
        )

class RegisterView(APIView):
    def post(self, request):
        serializer = AccountSerializer(data=request.data)

        if serializer.is_valid():
            # DBG ONLY
            if "test" not in request.data["email"]:
                serializer.save()

            return Response(
                {"message": "Account created successfully"},
                status=status.HTTP_201_CREATED,
            )
        return Response(serializer.errors, status=status.HTTP_400_BAD_REQUEST)

class GoogleLogin(SocialLoginView):
    adapter_class = GoogleOAuth2Adapter
    client_class = FixedOAuth2Client
    callback_url = "postmessage"

    def get_response(self):
        response = super().get_response()
        user = self.user
        refresh = RefreshToken.for_user(user)

```

					ІІЗ.КР.Б – 121 – 25 – ІІЗ	Арк.
						81
Змн.	Арк.	№ докум.	Підпис	Дата		

```

response.data["access"] = str(refresh.access_token)
response.data["refresh"] = str(refresh)
return response

```

Додаток В

Налаштування Django-проєкту

```

import os
from pathlib import Path
from dotenv import load_dotenv
from datetime import timedelta

load_dotenv()

# Build paths inside the project like this: BASE_DIR / 'subdir'.
BASE_DIR = Path(__file__).resolve().parent.parent

# Quick-start development settings - unsuitable for production
# See https://docs.djangoproject.com/en/5.2/howto/deployment/checklist/

# SECURITY WARNING: keep the secret key used in production secret!
SECRET_KEY = "django-insecure-
^fmy5ekr$@y=#acwcj@tl95%0320wowop#89%q$^&pz9e$2yp!"

# SECURITY WARNING: don't run with debug turned on in production!
DEBUG = True

ALLOWED_HOSTS = []

# Application definition
INSTALLED_APPS = [
    # Django core
    "django.contrib.admin",
    "django.contrib.auth",
    "django.contrib.contenttypes",
    "django.contrib.sessions",
    "django.contrib.messages",
    "django.contrib.staticfiles",
    "django.contrib.sites",
    # Django REST framework
    "rest_framework",
    "rest_framework_simplejwt.token_blacklist",
    # Django auth
    "allauth",
    "allauth.account",
    "allauth.socialaccount",
    "allauth.socialaccount.providers.google",
    "dj_rest_auth",
    "dj_rest_auth.registration",
    "corsheaders",
    # Project's apps
    "accounts",
    "orders",
    "products",
    "suppliers",
    "django_filters",
    "reviews",
    "payments",

```

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						82
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        "django_extensions",
    ]

MIDDLEWARE = [
    "django.middleware.security.SecurityMiddleware",
    "django.contrib.sessions.middleware.SessionMiddleware",
    "corsheaders.middleware.CorsMiddleware",
    "django.middleware.common.CommonMiddleware",
    "django.middleware.csrf.CsrfViewMiddleware",
    "django.contrib.auth.middleware.AuthenticationMiddleware",
    "django.contrib.messages.middleware.MessageMiddleware",
    "django.middleware.clickjacking.XFrameOptionsMiddleware",
    "allauth.account.middleware.AccountMiddleware",
]

ROOT_URLCONF = "case_space.urls"

if DEBUG:
    CORS_ALLOWED_ORIGINS = ["http://localhost:5173"]
    CORS_ALLOW_ALL_ORIGINS = True

# Way for Pay
WFP_MERCHANT_DOMAIN_NAME = os.getenv("WAY_FOR_PAY_MERCHANT_DOMAIN")
WFP_SECRET_KEY = os.getenv("WAY_FOR_PAY_MERCHANT_SECRET_KEY")
WFP_MERCHANT_LOGIN = os.getenv("WAY_FOR_PAY_MERCHANT_LOGIN")
WFP_MERCHANT_PASSWORD = os.getenv("WAY_FOR_PAY_MERCHANT_PASSWORD")

TEMPLATES = [
    {
        "BACKEND": "django.template.backends.django.DjangoTemplates",
        "DIRS": [],
        "APP_DIRS": True,
        "OPTIONS": {
            "context_processors": [
                "django.template.context_processors.request",
                "django.contrib.auth.context_processors.auth",
                "django.contrib.messages.context_processors.messages",
            ],
        },
    },
]

WSGI_APPLICATION = "case_space.wsgi.application"

# Database
# https://docs.djangoproject.com/en/5.2/ref/settings/#databases

DATABASES = {
    "default": {
        "ENGINE": "django.db.backends.postgresql",
        "NAME": "casespace",
        "USER": os.getenv("DB_USER"),
        "PASSWORD": os.getenv("DB_PASSWORD"),
        "HOST": "127.0.0.1",
        "PORT": "5432",
    }
}

# JWT Token config
SIMPLE_JWT = {
    "ACCESS_TOKEN_LIFETIME": timedelta(minutes=200),
    "REFRESH_TOKEN_LIFETIME": timedelta(days=1),
}

```

					ІІЗ.КР.Б – 121 – 25 – ІІЗ	Арк.
						83
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    "ROTATE_REFRESH_TOKENS": True,
    "BLACKLIST_AFTER_ROTATION": True,
    "AUTH_HEADER_TYPES": ("Bearer",),
}

# dj-rest-auth config
USE_JWT = True
REST_AUTH = {"USE_JWT": True, "TOKEN_MODEL": None}

# Django allauth
ACCOUNT_USER_MODEL_USERNAME_FIELD = None
ACCOUNT_SIGNUP_FIELDS = ["email*", "password1*", "password2*"]
ACCOUNT_LOGIN_METHODS = {"email"}
ACCOUNT_EMAIL_REQUIRED = True
ACCOUNT_USERNAME_REQUIRED = False
ACCOUNT_AUTHENTICATION_METHOD = "email"
ACCOUNT_UNIQUE_EMAIL = True
SOCIALACCOUNT_QUERY_EMAIL = True
ACCOUNT_EMAIL_VERIFICATION = "none"
SOCIALACCOUNT_EMAIL_AUTHENTICATION = True

# Custom allauth adapters to fix the issues
SOCIALACCOUNT_ADAPTER = "accounts.adapters.EnhancedSocialAccountAdapter"
ACCOUNT_ADAPTER = "accounts.adapters.NoPasswordOverrideAdapter"

SITE_ID = 1

SOCIALACCOUNT_PROVIDERS = {
    "google": {
        "FETCH_USERINFO": False,
        "SOCIALACCOUNT_EMAIL_AUTHENTICATION_AUTO_CONNECT": True,
        "EMAIL_AUTHENTICATION": True,
        "SCOPE": [
            "profile",
            "email",
        ],
        "AUTH_PARAMS": {
            "access_type": "online",
        },
    },
}

# Password validation
# https://docs.djangoproject.com/en/5.2/ref/settings/#auth-password-validators

AUTH_USER_MODEL = "accounts.Account"

AUTH_PASSWORD_VALIDATORS = [
    {
        "NAME":
        "django.contrib.auth.password_validation.UserAttributeSimilarityValidator",
    },
    {
        "NAME":
        "django.contrib.auth.password_validation.MinimumLengthValidator",
    },
    {
        "NAME":
        "django.contrib.auth.password_validation.CommonPasswordValidator",
    },
    {
        "NAME":
        "django.contrib.auth.password_validation.NumericPasswordValidator",
    },
]

```

					ІІЗ.КР.Б – 121 – 25 – ІІЗ	Арк.
						84
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    },
]

AUTHENTICATION_BACKENDS = [
    "django.contrib.auth.backends.ModelBackend",
    "allauth.account.auth_backends.AuthenticationBackend",
]

REST_FRAMEWORK = {
    "DEFAULT_AUTHENTICATION_CLASSES": (
        "rest_framework_simplejwt.authentication.JWTAuthentication",
    ),
    "DEFAULT_PAGINATION_CLASS":
"rest_framework.pagination.PageNumberPagination",
    "DEFAULT_FILTER_BACKENDS": [
        "django_filters.rest_framework.DjangoFilterBackend",
        "rest_framework.filters.OrderingFilter",
    ],
    "PAGE_SIZE": 12,
}

# Internationalization
# https://docs.djangoproject.com/en/5.2/topics/i18n/

LANGUAGE_CODE = "en-us"

TIME_ZONE = "UTC"

USE_I18N = True

USE_TZ = True

# Static files (CSS, JavaScript, Images)
# https://docs.djangoproject.com/en/5.2/howto/static-files/

STATIC_URL = "static/"
MEDIA_URL = "/media/"
MEDIA_ROOT = os.path.join(BASE_DIR, "media")

# Default primary key field type
# https://docs.djangoproject.com/en/5.2/ref/settings/#default-auto-field

DEFAULT_AUTO_FIELD = "django.db.models.BigAutoField"

```

					ІІЗ.КР.Б – 121 – 25 – ІІЗ	Арк.
						85
Змн.	Арк.	№ докум.	Підпис	Дата		

Налаштування URL-маршрутів проєкту

```

from django.contrib import admin
from django.urls import path, include
from rest_framework.routers import DefaultRouter
from orders.views import OrderViewSet, ProductOrderViewSet
from products.views import (
    DeviceModelViewSet,
    ProductViewSet,
    ProductImageViewSet,
    GetMaterialsView,
)
from suppliers.views import SupplierViewSet
from reviews.views import ReviewsViewSet
from payments.views import InitiatePaymentView, WayForPayCallbackView
from accounts.views import GoogleLogin

# DBG ONLY
from django.conf import settings
from django.conf.urls.static import static

router = DefaultRouter()
router.register(r"orders", OrderViewSet)
router.register(r"orders/product", ProductOrderViewSet)
router.register(r"products", ProductViewSet)
router.register(r"product-images", ProductImageViewSet)
router.register(r"models", DeviceModelViewSet)
router.register(r"suppliers", SupplierViewSet)
router.register(r"reviews", ReviewsViewSet)

urlpatterns = [
    path("", include(router.urls)),
    path("admin/", admin.site.urls),
    path("accounts/", include("accounts.urls")),
    path("materials/", GetMaterialsView.as_view()),
    path("payment/initiate/", InitiatePaymentView.as_view()),
    path("wfp/callback/", WayForPayCallbackView.as_view()),
    # social auth
    path("auth/registration/", include("dj_rest_auth.registration.urls")),
    path("auth/social/google/", GoogleLogin.as_view(), name="google_login"),
]

if settings.DEBUG:
    urlpatterns += static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)

```

					ІІЗ.КР.Б – 121 – 25 – ІІЗ	Арк.
						86
Змн.	Арк.	№ докум.	Підпис	Дата		

Моделі даних для продуктів інтернет-магазину

```

from django.db import models
from django.core.validators import MinValueValidator, MaxValueValidator
from suppliers.models import Supplier

class MaterialChoices(models.TextChoices):
    SILICONE = "silicone", "Силікон"
    TPU = "tpu", "TPU (термополіуретан)"
    PLASTIC = "plastic", "Пластик"
    LEATHER = "leather", "Шкіра"
    FABRIC = "fabric", "Тканина"
    METAL = "metal", "Метал"
    OTHER = "other", "Інший"

class Feature(models.Model):
    name = models.CharField(max_length=100, unique=True)
    description = models.TextField()
    image = models.ImageField(upload_to="media/features_images/", blank=True)

    def __str__(self):
        return self.name

class DeviceModel(models.Model):
    manufacturer = models.CharField(max_length=255)
    model = models.CharField(max_length=255)

    def __str__(self):
        return f"{self.manufacturer} {self.model}"

class Product(models.Model):
    title = models.CharField(max_length=255)
    supplier = models.ForeignKey(Supplier, on_delete=models.CASCADE)
    device_model = models.ForeignKey(
        DeviceModel, on_delete=models.CASCADE, related_name="product"
    )
    material = models.CharField(
        max_length=20, choices=MaterialChoices.choices,
        default=MaterialChoices.OTHER
    )
    features = models.ManyToManyField(Feature, related_name="products",
        blank=True)
    price = models.DecimalField(max_digits=10, decimal_places=2)
    amount_available = models.IntegerField()
    description = models.TextField(null=True, blank=True, default="")
    discount = models.IntegerField(default=0, validators=[
        MinValueValidator(0), MaxValueValidator(100)
    ])

    def __str__(self):
        return self.title

class ProductImage(models.Model):
    product = models.ForeignKey(
        Product, related_name="images", on_delete=models.CASCADE
    )
    image = models.ImageField(upload_to="media/product_images/")
    main_image = models.BooleanField(default=False)

```

					ІІЗ.КР.Б – 121 – 25 – ІІЗ	Арк.
						87
Змн.	Арк.	№ докум.	Підпис	Дата		

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

Додаток Е

Серіалізатори для продуктів інтернет-магазину

```
from rest_framework.serializers import (
    ModelSerializer,
    FloatField,
    SerializerMethodField,
    IntegerField
)
from .models import Product, ProductImage, DeviceModel, Feature
from reviews.serializers import ReviewSerializer
from suppliers.serializers import ProductSupplierSerializer

class ProductImageSerializer(ModelSerializer):
    class Meta:
        model = ProductImage
        fields = ["id", "image"]

class DeviceModelSerializer(ModelSerializer):
    class Meta:
        model = DeviceModel
        fields = "__all__"

class FeatureSerializer(ModelSerializer):
    class Meta:
        model = Feature
        fields = ["id", "name", "description", "image"]

class ProductSerializer(ModelSerializer):
    device_model = DeviceModelSerializer(read_only=True)
    images = ProductImageSerializer(many=True, read_only=True)
    reviews = ReviewSerializer(many=True, read_only=True)
    reviews_count = IntegerField(read_only=True)
    features = FeatureSerializer(many=True, read_only=True)
    average_score = FloatField(read_only=True)
    material = SerializerMethodField()
    supplier = ProductSupplierSerializer(read_only=True)

    def get_material(self, obj):
        return {"value": obj.material, "label": obj.get_material_display()}

    class Meta:
        model = Product
        fields = [
            "id",
            "title",
            "device_model",
            "price",
            "amount_available",
            "description",
            "images",
            "reviews",
            "average_score",
            "material",
            "features",
            "supplier",
```

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						88
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        "discount",
        "reviews_count"
    ]

    def get_average_score(self, obj):
        return round(obj.average_score, 1) if obj.average_score is not None else
0.0

class MinifiedProductSerializer(ModelSerializer):
    images = ProductImageSerializer(many=True, read_only=True)

    class Meta:
        model = Product
        fields = ["id", "title", "price", "images", "discount"]

```

Додаток Ж

API-представлення для продуктів

```

from decimal import Decimal
from django.db.models import Avg, Value, F, ExpressionWrapper, DecimalField,
Func, Count
from django.db.models.functions import Coalesce
from rest_framework.permissions import IsAuthenticatedOrReadOnly
from rest_framework.viewsets import ModelViewSet
from rest_framework.views import APIView
from rest_framework.decorators import action
from rest_framework.response import Response
from .serializers import (
    ProductImageSerializer,
    ProductSerializer,
    DeviceModelSerializer,
)
from .models import Product, ProductImage, DeviceModel, MaterialChoices
from .filters import ProductFilter

class ProductViewSet(ModelViewSet):
    queryset = Product.objects.all()
    serializer_class = ProductSerializer
    permission_classes = [IsAuthenticatedOrReadOnly]
    filterset_class = ProductFilter
    ordering_fields = [
        "average_score",
        "title",
        "price",
        "amount_available",
        "discount",
        "discounted_price",
    ]
    ordering = ["-average_score"]

    def get_queryset(self):
        return (
            Product.objects.select_related("device_model")
            .prefetch_related("images", "reviews", "features")
            .annotate(
                average_score=Coalesce(Avg("reviews__score"), Value(0.0)),
                discounted_price=ExpressionWrapper(
                    Func(
                        F("price")

```

					ІІЗ.КР.Б – 121 – 25 – ІІЗ	Арк.
						89
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        * (Value(1.00) - (F("discount") /
Value(Decimal("100.00")))),
        Value(2), # round to 2 decimal places
        function="ROUND",
    ),
    output_field=DecimalField(max_digits=10, decimal_places=2),
),
reviews_count=Count("reviews"),
)
)

```

```

class ProductImageViewSet(ModelViewSet):
    serializer_class = ProductImageSerializer
    permission_classes = [IsAuthenticatedOrReadOnly]
    queryset = ProductImage.objects.all()

```

```

class DeviceModelViewSet(ModelViewSet):
    serializer_class = DeviceModelSerializer
    permission_classes = [IsAuthenticatedOrReadOnly]
    queryset = DeviceModel.objects.all()
    filterset_fields = ["manufacturer"]
    pagination_class = None

    @action(detail=False, methods=["get"])
    def manufacturers(self, request):
        manufacturers = DeviceModel.objects.values_list(
            "manufacturer", flat=True
        ).distinct()
        return Response(manufacturers)

```

```

class GetMaterialsView(APIView):
    def get(self, request):
        data = [
            {"value": choice.value, "label": choice.label} for choice in
MaterialChoices
        ]
        return Response(data)

```

					ІІЗ.КР.Б – 121 – 25 – ІІЗ	Арк.
						90
Змн.	Арк.	№ докум.	Підпис	Дата		