

БАКАЛАВРСЬКА РОБОТА

БР. ІП - 07.00.00.000 ІЗ

ІП-22-4

Кликайло Вікторія

2026

Івано-Франківський національний технічний університет нафти і газу

Факультет інформаційних технологій

Кафедра інженерії програмного забезпечення

Кликайло Вікторія Ігорівна

(прізвище, ім'я, по батькові)

УДК 004.942

(індекс)

БАКАЛАВРСЬКА РОБОТА

Реалізація Full Stack застосунку електронної комерції з аналітичними модулями

(назва роботи)

Інженерія програмного забезпечення

(назва освітньої програми)

121– Інженерія програмного забезпечення

(шифр і назва спеціальності)

Робота містить результати власних досліджень, використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело:

Здобувач освітнього ступеня _____ Кликайло В.І.
(підпис, ініціали та прізвище здобувача)

Науковий керівник _____ Дмитрик Тарас Богданович, асистент
(підпис, прізвище, ім'я, по батькові, науковий ступінь, вчене звання керівника)

Допущено до захисту
Завідувач кафедри

доц. _____ Бандура В.В.
(посада) (підпис) (дата) (ініціали та прізвище)

Івано-Франківськ – 2026

Інститут, факультет інформаційних технологій

Кафедра інженерії програмного забезпечення

Ступінь вищої освіти бакалавр

Спеціальність 121 – Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ:

Зав. кафедрою ІПЗ

доц. В.В. Бандура

“ ” 2026 р.

ЗАВДАННЯ

НА БАКАЛАВРСЬКУ РОБОТУ СТУДЕНТОВІ

Кликайло Вікторії Ігорівні

(прізвище, ім'я, по-батькові)

1. Тема проекту (роботи) “ Реалізація Full Stack застосунку електронної комерції з аналітичними модулями”

керівник проекту (роботи) Дмитрик Тарас Богданович, асист.

затверджені наказом закладу вищої освіти від “ 21 ” квітня 2026 р. № 179/7

2. Строк подання студентом проекту (роботи) 10 червня 2026 р.

3. Вихідні дані до проекту (роботи) Результати і матеріали отримані під час проходження переддипломної практики

4. Зміст розрахунково - пояснювальної записки (перелік питань, які потрібно розробити)

1. Аналіз предметної області та існуючих рішень в сфері електронної комерції

2. Формування функціональних та нефункціональних вимог

3. Проектування архітектури веб-застосунку

4. Реалізація користувацької та адміністративної частин застосунку

5. Реалізація аналітичних модулів

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)

1. Діаграма варіантів використання системи інтернет-магазину сумок (рис. 2.1, с. 28)

2. Діаграма взаємодії адміністратора з адміністративною панеллю (рис. 2.2, с. 29)

3. Діаграма діяльності процесу оформлення замовлення (рис. 2.3, с. 34)

4. Архітектура веб-застосунку інтернет-магазину (рис. 3.1, с. 36)

5. Структура взаємодії компонентів за шаблоном MVC (рис. 3.2, с. 37)

6. Консультанти розділів проекту (роботи)

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання 10 січня 2026 року

Керівник

(підпис)

(розшифровка підпису)

Завдання прийняв до виконання

(підпис)

(розшифровка підпису)

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проекту (роботи)	Строк виконання етапів проекту	Примітка
1	Визначення та обґрунтування теми дипломної роботи	15.01.2026	Виконано
2	Аналіз предметної області електронної комерції та існуючих рішень	29.01.2026	Виконано
3	Проектування архітектури веб-застосунку, структури бази даних та інтерфейсу	23.02.2026	Виконано
4	Реалізація функціоналу веб-застосунку та адміністративної панелі	07.03.2026	Виконано
5	Розробка аналітичних модулів та тестування системи	21.03.2026	Виконано
6	Оформлення пояснювальної записки бакалаврської роботи	23.04.2026	Виконано

Студент – дипломник

(підпис)

Кликайло В. І

(розшифровка підпису)

Керівник роботи

(підпис)

Дмитрик Т. Б

(розшифровка підпису)

АНОТАЦІЯ

Бакалаврська робота містить 103 сторінки, 45 рисунків, список використаних джерел із 35 найменувань, 3 додатки.

Метою роботи є розробка Full Stack веб-застосунку електронної комерції, який забезпечує зручну взаємодію користувачів із системою, управління товарами та замовленнями, а також аналіз показників діяльності за допомогою модулів.

Об'єкт дослідження: процеси розробки та функціонування веб-застосунків електронної комерції.

Предмет дослідження: методи та технології розробки веб-застосунку, включаючи проєктування архітектури, структури бази даних, користувацького інтерфейсу, реалізацію бізнес-логіки та аналітичних модулів.

Результати дослідження: розроблено веб-застосунок на базі Ruby on Rails із використанням архітектури MVC, що забезпечує функціонал перегляду каталогу товарів, оформлення замовлень, управління даними через адміністративну панель та аналіз діяльності системи.

У першому розділі проведено аналіз предметної області електронної комерції та програмних рішень.

У другому розділі визначено функціональні та нефункціональні вимоги й постановку задачі.

У третьому розділі спроектовано архітектуру, базу даних та інтерфейс застосунку.

У четвертому розділі реалізовано програмний застосунок, включаючи користувацьку частину, адміністративну панель, бізнес-логіку й аналітичні модулі.

У п'ятому розділі проведено тестування веб-застосунку.

Висновок: розроблений веб-застосунок відповідає поставленій меті, забезпечує функціонал інтернет-магазину та аналіз діяльності системи.

КЛЮЧОВІ СЛОВА: ЕЛЕКТРОННА КОМЕРЦІЯ, ВЕБ-ЗАСТОСУНОК, ІНТЕРНЕТ-МАГАЗИН, RUBY ON RAILS, MVC, FULL STACK, АНАЛІТИКА, АДМІНІСТРАТИВНА ПАНЕЛЬ.

ABSTRACT

The bachelor's thesis consists of 103 pages, 45 figures, a bibliography of 35 references, and 3 appendices.

The aim of the thesis is to develop a full-stack e-commerce web application that ensures convenient user interaction with the system, management of products and orders, as well as analysis of performance metrics using analytical modules.

Subject of research: the processes of development and operation of e-commerce web applications.

Research subject: methods and technologies for web application development, including the design of architecture, database structure, user interface, and the implementation of business logic and analytical modules.

Research results: a web application based on Ruby on Rails was developed using the MVC architecture, providing functionality for browsing the product catalog, placing orders, managing data via the administrative panel, and analyzing system performance.

The first chapter analyzes the domain of e-commerce and software solutions.

The second chapter defines the functional and non-functional requirements and the problem statement.

The third chapter designs the architecture, database, and application interface.

The fourth chapter implements the software application, including the user interface, administrative panel, business logic, and analytical modules.

The fifth chapter describes the testing of the web application.

Conclusion: The developed web application meets the set objectives, providing the functionality of an online store and the ability to analyze system performance.

KEYWORDS: E-COMMERCE, WEB APPLICATION, ONLINE STORE, RUBY ON RAILS, MVC, FULL STACK, ANALYTICS, ADMIN PANEL.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ	9
ВСТУП.....	10
РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ІСНУЮЧИХ РІШЕНЬ В СФЕРІ ЕЛЕКТРОННОЇ КОМЕРЦІЇ	12
1.1 Аналіз предметної області електронної комерції	12
1.2 Сучасний стан розвитку електронної комерції	15
1.3 Аналіз існуючих програмних рішень та аналогів.....	18
1.4 Обґрунтування вибору технологій та засобів розробки	20
1.5 Висновки по розділу	23
РОЗДІЛ 2. АНАЛІЗ ВИМОГ ТА ПОСТАНОВКА ЗАДАЧІ.....	24
2.1 Постановка задачі	24
2.2 Формування функціональних вимог	25
2.3 Формування нефункціональних вимог	27
2.4 Моделювання користувачів та сценаріїв взаємодії	29
2.5 Висновки по розділу	31
РОЗДІЛ 3. ПРОЄКТУВАННЯ АРХІТЕКТУРИ ТА СТРУКТУРИ СИСТЕМИ	32
3.1 Проєктування архітектури веб-застосунку.....	32
3.2 Проєктування структури бази даних.....	35
3.3 Проєктування користувацького та адміністративного інтерфейсів	37
3.4 Структура програмного забезпечення	40
3.5 Висновки по розділу	42
РОЗДІЛ 4. РЕАЛІЗАЦІЯ FULL STACK ЗАСТОСУНКУ ЕЛЕКТРОННОЇ КОМЕРЦІЇ.....	43
4.1 Реалізація користувацької частини застосунку	43
4.2 Реалізація адміністративної панелі	57
4.3 Реалізація бізнес-логіки системи	68
4.4 Реалізація аналітичних модулів.....	70

					БР. ІІІ – 07.00.00.000 ПЗ			
Змн.	Арк.	№ докум.	Підпис	Дата	Реалізація Full Stack застосунку електронної комерції з аналітичними модулями Пояснювальна записка	Літ.	Арк.	Аркушів
Розроб.		Кликайло В. І.						
Перевір.		Дмитрик Т. Б.					7	103
Реценз.		Корнута В.А..				ІФНТУНГ, ІІІ-22-4		
Н. Контр.		Піх М. М.						
Затверд.		Бандура В. В.						

4.5 Висновки по розділу.....	72
РОЗДІЛ 5. ТЕСТУВАННЯ ВЕБ-ЗАСТОСУНКУ.....	73
5.1 Тестування користувацької частини веб-застосунку	73
5.2 Тестування адміністративної частини	75
5.3 Результати тестування та оцінка роботи системи	77
5.4 Висновки по розділу	78
ВИСНОВОК.....	79
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	80
ДОДАТКИ	
БІБЛІОГРАФІЧНА ДОВІДКА	

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

MVC – Model – View – Controller, шаблон побудови архітектурного програмного забезпечення.

CRUD – створити, прочитати, оновити, вилучити.

API – Application Programming Interface, інтерфейс програмування застосунків.

DB – база даних.

UI – User Interface, інтерфейс користувача.

UX – User Experience, досвід взаємодії.

CI – Continuous Integration, безперервна інтеграція.

CD – Continuous Delivery, безперервна доставка.

KPI – Key Performance Indicators, ключові показники ефективності

					БР.ІП - 07.00.00.000 ПЗ	Арк.
						9
Змн.	Арк.	№ докум.	Підпис	Дата		

ВСТУП

У наш час, коли інформаційні технології розвиваються з неймовірною швидкістю, електронна комерція займає одне з ключових місць у світі бізнесу. Інтернет-магазини, завдяки своїм веб-застосункам, надають миттєвий доступ до товарів і послуг, спрощують процес покупки і дозволяють автоматизувати основні бізнес-процеси. Зростаюча популярність онлайн-торгівлі, розвиток цифрових сервісів і зміна поведінки споживачів підвищують вимоги до якості, функціональності та зручності використання веб-застосунків.

Особливо важливою стає інтеграція аналітичних можливостей у веб-застосунки електронної комерції. Аналіз даних про користувачів, замовлення та товари допомагає підвищити ефективність управлінських рішень, оптимізувати бізнес-процеси і покращити користувацький досвід. У сучасних умовах конкуренції на ринку електронної комерції наявність аналітичних інструментів стає важливим фактором успіху будь-якого програмного продукту.

Актуальність теми обумовлена потребою в створенні сучасних веб-застосунків електронної комерції, які поєднують зручний інтерфейс, ефективну бізнес-логіку та потужні аналітичні інструменти. Хоча існує чимало рішень, багато з них мають обмежені можливості аналітики або недостатню гнучкість у налаштуванні, що підкреслює потребу в нових підходах до розробки Full Stack застосунків.

Метою роботи є створення Full Stack веб-застосунку для електронної комерції з аналітичними модулями, який забезпечить ефективне управління товарами та замовленнями, а також можливість аналізу ключових показників діяльності системи.

Для досягнення мети потрібно вирішити такі завдання: провести аналіз сфери електронної комерції та існуючих програмних рішень; визначити функціональні та нефункціональні вимоги до застосунку; спроектувати архітектуру системи, структуру бази даних і користувацький інтерфейс; реалізувати веб-застосунок із використанням сучасних технологій; розробити

					БР.ІІІ - 07.00.00.000 ПЗ	Арк.
						10
Змн.	Арк.	№ докум.	Підпис	Дата		

адміністративну панель для управління даними; реалізувати аналітичні модулі для обробки та візуалізації даних; провести тестування розробленого застосунку.

Об’єктом дослідження є процеси розробки та функціонування веб-застосунків електронної комерції, які охоплюють аналіз вимог, проєктування, реалізацію та тестування програмного забезпечення.

Предметом дослідження є методи, технології та інструменти розробки веб-застосунків електронної комерції, зокрема архітектурні підходи, моделювання структури бази даних, розробка інтерфейсу користувача, реалізація бізнес-логіки та аналітичних модулів.

У процесі дослідження використовувалися методи аналізу та узагальнення для вивчення предметної області та існуючих рішень, методи моделювання для проєктування архітектури системи, а також методи програмування для реалізації функціоналу веб-застосунку.

У процесі розробки застосунку особливу увагу приділено використанню сучасних підходів до створення веб-систем, зокрема принципів Full Stack розробки, що передбачають інтеграцію клієнтської та серверної частин. Застосування архітектурного шаблону MVC дозволяє забезпечити чітке розділення логіки, даних і представлення, що підвищує масштабованість і зручність супроводу системи. Використання фреймворку Ruby on Rails та системи керування базами даних PostgreSQL сприяє ефективній реалізації функціоналу, а застосування засобів візуалізації даних дозволяє наочно представити результати аналітичної обробки.

Практичне значення отриманих результатів полягає в створенні повнофункціонального веб-застосунку електронної комерції, який може бути використаний для організації роботи інтернет-магазину, автоматизації бізнес-процесів та аналізу діяльності системи. У роботі послідовно розглядаються аналіз предметної області, проєктування системи та реалізація застосунку з подальшим тестуванням.

Бакалаврська робота містить 103 сторінки, 45 рисунків, 5 розділів, список використаних джерел із 35 найменувань, 3 додатки.

					БР.ІІІ - 07.00.00.000 ІІЗ	Арк.
						11
Змн.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ІСНУЮЧИХ РІШЕНЬ В СФЕРІ ЕЛЕКТРОННОЇ КОМЕРЦІЇ

Електронна комерція – це одна з найбільш динамічних галузей, де інформаційні технології грають ключову роль. Вона охоплює все: від продажу товарів і послуг до обробки замовлень, взаємодії з користувачами та аналізу роботи системи [1]. Щоб такі веб-додатки працювали ефективно, важливо мати зручний інтерфейс, продуману бізнес-логіку, надійну структуру даних і сучасні аналітичні інструменти [4]. Тому ретельний аналіз предметної області та наявних рішень допомагає визначити основні підходи до створення систем електронної комерції, виявити їхні особливості та обґрунтувати вибір технологій для подальшої розробки.

1.1. Аналіз предметної області електронної комерції

Електронна комерція зараз фактично стала частиною цифрової економіки, яка активно розвивається завдяки ІТ-технологіям. Вона охоплює купівлю та продаж товарів і послуг через Інтернет, а також різні пов'язані процеси: обробку замовлень, роботу з даними і взаємодію з користувачами. Через те, що онлайн-торгівля набирає популярності, зростає потреба у зручних, функціональних і стабільних веб-застосунках [1].

Аналіз сучасних рішень показує, що більшість інтернет-магазинів мають базовий функціонал, але досить часто виникають проблеми з навігацією, гнучкістю управління товарами та слабкою аналітикою [2].

Тому є потреба створювати веб-застосунки, які поєднують стандартний функціонал із розширеними можливостями управління, аналітики та більш зручним користувацьким досвідом [4].

1.1.1 Загальна характеристика електронної комерції

Електронна комерція загалом охоплює діяльність, пов'язану з обміном товарами та послугами через інформаційно-комунікаційні технології. До неї

					БР.ІІІ - 07.00.00.000 ПЗ	Арк.
						12
Змн.	Арк.	№ докум.	Підпис	Дата		

входять продажі, маркетинг, логістика, обробка платежів і післяпродажне обслуговування. Основна мета таких систем — зробити взаємодію між продавцем і покупцем простішою та ефективнішою за рахунок автоматизації процесів, що дозволяє швидше обробляти замовлення і зменшувати кількість помилок [1].

Серед важливих характеристик сучасних веб-застосунків залишаються зручний інтерфейс, достатня швидкість роботи, аналітичні можливості та зрозуміле управління даними. Водночас частина систем обмежується лише базовим функціоналом, через що взаємодія користувача із застосунком інколи стає складнішою [2].

Аналіз таких рішень дозволяє побачити їхні плюси й мінуси, що потім можна використати як основу для створення більш гнучкого та зручного веб-застосунку [5].

1.1.2 Функціональні можливості веб-застосунків інтернет-магазинів

Веб-застосунки інтернет-магазинів охоплюють повний цикл взаємодії з користувачем — від пошуку товарів до їх покупки. Основні можливості включають перегляд і пошук у каталозі, фільтрацію, відображення інформації про товар, формування кошика та оформлення замовлення. Сучасні системи також використовують персоналізовані рекомендації та відгуки користувачів [1].

Окрім базового функціоналу, важливими залишаються зручність інтерфейсу, швидкість роботи та наявність аналітичних інструментів [6]. Недостатня увага до цих аспектів може призводити до проблем із навігацією та продуктивністю системи [2].

Доступ до історії замовлень дозволяє користувачам відстежувати взаємодію із системою, а адміністративний інтерфейс — ефективно керувати інформацією [5].

1.1.3 Адміністративна складова систем електронної комерції

Адміністративний розділ веб-застосунку відповідає за роботу всієї системи, надаючи інструменти для управління товарами, замовленнями та користувачами. Це дозволяє відстежувати інформацію, керувати асортиментом і обробляти

					БР.ІІІ - 07.00.00.000 ПЗ	Арк.
						13
Змн.	Арк.	№ докум.	Підпис	Дата		

замовлення, а також досить швидко реагувати на зміни та підтримувати актуальність даних [4].

Водночас у багатьох існуючих рішеннях адміністративна частина реалізована досить обмежено або не зовсім зручно, через що керування системою стає складнішим і знижується її ефективність [2].

Важливою частиною є аналітичні модулі, які дають змогу оцінювати продуктивність системи, відстежувати ключові показники та приймати більш обґрунтовані управлінські рішення [1].

1.1.4 Використання аналітичних модулів

Аналітичні модулі зазвичай використовуються в системах електронної комерції, оскільки відповідають за збір, обробку та аналіз даних. Вони дають інформацію про продажі, поведінку користувачів, популярність товарів і ефективність маркетингових кампаній [1].

На основі цих даних можна визначати тенденції, прогнозувати попит і приймати більш обґрунтовані управлінські рішення, що в цілому допомагає оптимізувати бізнес-процеси і підвищити ефективність системи [2].

Водночас не всі веб-застосунки мають достатній рівень аналітики або подають її у зручному форматі, через що можливості аналізу трохи обмежуються. Використання графіків і діаграм значно спрощує сприйняття та інтерпретацію даних [15].

1.1.5 Вимоги до зручності використання веб-застосунків

Зручність використання зараз вважається одним із основних критеріїв якості веб-застосунків. Користувачі зазвичай хочуть бачити інтуїтивно зрозумілий інтерфейс, нормальну швидкість роботи системи та можливість отримати потрібний результат з мінімальною кількістю дій [6].

Основними аспектами є логічно продумана структура інтерфейсу, зручна навігація, адаптивність під різні пристрої та стабільність роботи системи [13].

Водночас багато веб-застосунків мають певні недоліки, пов'язані зі

					БР.ІІІ - 07.00.00.000 ПЗ	Арк.
						14
Змн.	Арк.	№ докум.	Підпис	Дата		

складною навігацією або перевантаженим інтерфейсом, що ускладнює взаємодію користувача із системою. Юзабіліті тісно пов'язане з технічними характеристиками, зокрема надійністю та продуктивністю, і в цілому впливає на ефективність веб-застосунку [27].

1.1.6 Вимоги до надійності, продуктивності та безпеки

Системи електронної комерції повинні відповідати вимогам надійності, продуктивності та безпеки. Вони мають стабільно працювати навіть при досить великому навантаженні і забезпечувати обробку великих обсягів даних. Особливу увагу зазвичай приділяють захисту особистих і фінансових даних користувачів, оскільки порушення безпеки може призвести до втрати довіри [1].

Для цього використовують механізми автентифікації та авторизації, шифрування даних і захист від типових веб-загроз [10]. Якщо рівень безпеки недостатній або продуктивність низька, це може негативно впливати на стабільність системи та користувацький досвід [12].

Дотримання цих вимог є необхідною умовою для нормальної роботи веб-застосунку та збереження довіри користувачів [20].

1.2. Сучасний стан розвитку електронної комерції

Завдяки розвитку цифрових технологій і тому, що веб-застосунки активно використовуються в бізнесі, електронна комерція зараз швидко розвивається. Інтернет-торгівля стала звичною частиною економіки, оскільки дозволяє надавати товари та послуги незалежно від того, де знаходиться користувач. Розвиток мобільних технологій, платіжних систем і зростання кількості інтернет-користувачів ще більше розширюють можливості веб-застосунків [2].

Водночас вимоги до таких систем постійно ростуть, особливо щодо швидкості роботи, зручності використання та рівня безпеки. Через це з'являється потреба постійно вдосконалювати програмні рішення і застосовувати сучасні підходи до розробки, які впливають на структуру та функціональність веб-застосунків [4].

					БР.ІІІ - 07.00.00.000 ПЗ	Арк.
						15
Змн.	Арк.	№ докум.	Підпис	Дата		

1.2.1 Трансформація веб-застосунків електронної комерції

Веб-застосунки електронної комерції з часом пройшли шлях від простих каталогів товарів до складних інформаційних систем. Сучасні рішення поєднують оплату і доставку, обробку замовлень, аналітичні інструменти та елементи персоналізації [4].

Однією з помітних тенденцій є інтеграція сторонніх сервісів, що дозволяє розширювати функціональність без значного ускладнення архітектури [5].

Водночас велика кількість інтеграцій може ускладнювати підтримку та впливати на стабільність системи, тому вибір технологій і архітектурних рішень має бути обґрунтованим [20].

1.2.2 Орієнтація на користувацький досвід

Сучасні веб-застосунки в основному роблять акцент на тому, щоб користувачу було зручно з ними працювати. У конкурентному середовищі важливими залишаються інтерфейс, швидкість роботи та доступність системи. До ключових аспектів належать адаптивний дизайн, швидке завантаження сторінок і швидку реакцію на дії користувача [13].

Персоналізація також має значення, бо дозволяє враховувати індивідуальні потреби користувачів і робить взаємодію більш зручною. Водночас недостатня оптимізація інтерфейсу або занадто велика кількість функцій можуть ускладнювати використання веб-застосунку та негативно впливати на користувацький досвід [6].

1.2.3 Роль аналітики в сучасних системах

Аналітичні інструменти обробляють дані про поведінку користувачів, обсяги продажів і продуктивність системи. Це дозволяє визначати популярні товари, оцінювати ефективність маркетингових кампаній і знаходити проблемні місця в роботі веб-застосунку [1]. Також аналітика допомагає прогнозувати попит і планувати подальший розвиток системи.

Отримані результати використовують для оптимізації бізнес-процесів і

					БР.ІП - 07.00.00.000 ПЗ	Арк.
						16
Змн.	Арк.	№ докум.	Підпис	Дата		

покращення якості обслуговування. При цьому ефективність аналітики залежить від того, як подаються дані, тому використання графіків і діаграм спрощує їх сприйняття і допомагає приймати обґрунтовані управлінські рішення [15].

1.2.4 Автоматизація бізнес-процесів

Автоматизація охоплює обробку замовлень, оновлення інформації про товари, управління складськими запасами та взаємодію з користувачем. Вона зменшує потребу в ручній роботі, пришвидшує обробку даних і знижує ймовірність помилок.

Рівень автоматизації безпосередньо впливає на ефективність системи, особливо при зростанні обсягів даних. Недостатня автоматизація може призводити до затримок у виконанні операцій і робити управління процесами складнішим.

Впровадження автоматизованих механізмів дозволяє зробити бізнес-процеси більш узгодженими та ефективніше використовувати ресурси, що важливо для масштабованих веб-застосунків [4].

1.2.5 Забезпечення безпеки та захисту даних

Сучасні веб-застосунки електронної комерції повинні забезпечувати належний рівень безпеки та захисту даних користувачів. Це особливо важливо для збереження персональної інформації, а також даних, які пов'язані з оформленням замовлень і оплатою.

Для цього використовують механізми автентифікації та авторизації, шифрування даних і різні способи захисту від поширених веб-загроз [10]. Також важливо обмежувати доступ до функцій системи відповідно до ролей користувачів [12].

Достатній рівень безпеки підвищує довіру користувачів і допомагає забезпечити стабільну роботу веб-застосунку [20].

1.2.6 Тенденції розвитку онлайн-ринку

Ринок електронної комерції постійно розвивається, через що зростає

					БР.ІІІ - 07.00.00.000 ПЗ	Арк.
						17
Змн.	Арк.	№ докум.	Підпис	Дата		

кількість веб-застосунків і конкуренція між ними. З'являється більше вузькоспеціалізованих інтернет-магазинів, орієнтованих на конкретні потреби користувачів [2].

У таких умовах важливими залишаються швидкість роботи, зручність інтерфейсу та можливість персоналізації. Користувачі очікують простий процес покупки, швидкий доступ до інформації та адаптацію сервісу під свої потреби [6].

Таким чином, сучасні веб-застосунки мають бути гнучкими, адаптивними та орієнтованими на користувача [4].

1.3. Аналіз існуючих програмних рішень та аналогів

Сучасні веб-застосунки електронної комерції пропонують різні рішення для організації продажів, обробки замовлень, управління товарами та взаємодії з користувачами, що сприяє автоматизації процесів [1].

Різні платформи відрізняються за функціональністю, гнучкістю та можливостями налаштування. Деякі рішення мають обмеження в кастомізації або є складними у впровадженні [2].

Таким чином, аналіз існуючих рішень дозволяє визначити їх переваги та недоліки й сформулювати підхід до розробки власного веб-застосунку [4].

1.3.1 Великі платформи електронної комерції

Серед відомих платформ можна виділити такі, як Rozetka, Amazon та eBay, які забезпечують взаємодію між великою кількістю продавців і покупців. Вони мають широкий функціонал: пошук і фільтрацію товарів, систему відгуків та інтеграцію з платіжними і логістичними сервісами [3].

Основною перевагою є те, що вони можуть працювати з великими обсягами даних і обслуговувати велику кількість користувачів [4]. Але через складну архітектуру такі платформи не дуже гнучкі, і підлаштувати їх під конкретні потреби буває складно [5].

Тому такі рішення не завжди підходять для проєктів, де потрібна індивідуальна логіка або нестандартний функціонал.

					БР.ІІІ - 07.00.00.000 ПЗ	Арк.
						18
Змн.	Арк.	№ докум.	Підпис	Дата		

1.3.2 Готові рішення для створення інтернет-магазинів

До цієї категорії належать Shopify, WooCommerce та OpenCart, які дозволяють швидко створити інтернет-магазин без глибоких знань програмування. Вони мають готові шаблони, модулі та інтеграції з платіжними й аналітичними сервісами [2].

Основною перевагою є те, що ними просто користуватись і можна швидко запустити проєкт, тому вони добре підходять для малого та середнього бізнесу [6].

Водночас такі рішення мають обмеження у гнучкості, і зробити нестандартний функціонал часто буває складно або взагалі неможливо. Це створює проблеми, коли система стає більш складною [4].

1.3.3 Індивідуальна розробка веб-застосунків

Індивідуальна розробка передбачає створення веб-застосунку під конкретні потреби бізнесу. Такий підхід дає можливість реалізувати унікальний функціонал, повністю контролювати архітектуру системи та підключати потрібні сервіси [4].

Серед основних плюсів — гнучкість, масштабованість і можливість адаптуватись під змінні вимоги, що дозволяє ефективно реалізовувати складні бізнес-процеси та організовувати роботу з даними [5].

Водночас індивідуальна розробка потребує досить великих ресурсів, зокрема часу, коштів і технічних знань, що ускладнює її використання для невеликих проєктів [20].

Таким чином, цей підхід доцільний для створення повноцінних і масштабованих систем, орієнтованих на конкретні вимоги [21].

1.3.4 Використання аналітичних інструментів у сучасних системах

Більшість сучасних платформ електронної комерції використовують аналітичні інструменти для оцінки ефективності роботи. Вони дають базову інформацію про продажі, поведінку користувачів і популярність товарів [1].

Водночас можливості глибшого аналізу та візуалізації даних часто обмежені або потребують підключення додаткових сервісів, через що складніше отримати

					БР.ІІІ - 07.00.00.000 ПЗ	Арк.
						19
Змн.	Арк.	№ докум.	Підпис	Дата		

повну картину роботи системи і це впливає на якість прийняття рішень [15].

Тому є сенс використовувати розширені аналітичні модулі, які дають змогу зручніше аналізувати дані і показувати їх у більш наочному вигляді [28].

1.3.5 Порівняльний аналіз та формування вимог до системи

Проведений аналіз показує, що кожен підхід має свої особливості. Великі платформи дають широкий функціонал, але їх складніше підлаштувати під конкретні потреби. Готові рішення зручні у використанні, проте мають обмежену гнучкість, тоді як індивідуальна розробка дозволяє реалізувати потрібний функціонал, але потребує більше ресурсів [4].

Вибір підходу залежить від масштабу проєкту та вимог до системи: для невеликих проєктів частіше використовують готові рішення, а для складніших систем - індивідуальну розробку [20].

З урахуванням цього доцільно створити веб-застосунок, який поєднує гнучкість індивідуальної розробки, достатній рівень функціональності та аналітичні можливості системи [1]. Отримані результати формують основу для визначення вимог до системи на наступних етапах проєктування [5].

1.4 Обґрунтування вибору технологій та засобів розробки

Вибір технологій для веб-застосунку електронної комерції має значення, оскільки впливає на ефективність роботи системи, її розвиток і подальшу підтримку. При цьому враховують функціональність, обробку даних та реалізацію бізнес-логіки [4].

Під час формування технологічного стеку основну увагу зазвичай приділяють продуктивності, безпеці, масштабованості, зручності розробки та підтримки, а також можливостям інтеграції і подальшого розширення системи [20].

1.4.1 Вибір серверної частини застосунку

Для серверної частини було обрано фреймворк Ruby on Rails, який базується на архітектурі MVC. Такий підхід дозволяє розділити обробку даних, відображення

					БР.ІІІ - 07.00.00.000 ПЗ	Арк.
						20
Змн.	Арк.	№ докум.	Підпис	Дата		

і керування застосунком, що спрощує підтримку коду [5].

Ruby on Rails дає можливість швидко реалізувати функціонал завдяки принципам «convention over configuration» та «don't repeat yourself». Фреймворк також має вбудовані інструменти для роботи з базою даних, маршрутами і запитами [7].

У порівнянні з Node.js або Django, Ruby on Rails дозволяє швидше почати розробку і має багато готових рішень для задач електронної комерції [8].

Окрім цього, фреймворк має вбудовані механізми захисту від поширених веб-загроз, що важливо при роботі з персональними і фінансовими даними [10].

Таким чином, вибір Ruby on Rails є виправданим завдяки швидкості розробки, зручності підтримки та достатньому рівню безпеки [31].

1.4.2 Вибір системи керування базами даних

Для зберігання даних було обрано PostgreSQL, яка відзначається стабільністю, надійністю та високою продуктивністю. Вона підтримує складні запити, транзакції та забезпечує цілісність даних, що важливо для систем електронної комерції [14].

Якщо порівнювати з іншими рішеннями, такими як MySQL або NoSQL, PostgreSQL краще підходить для роботи зі складними структурами даних та аналітичними запитами. Підтримка JSON дає більше гнучкості у зберіганні інформації, а транзакції забезпечують більш надійну обробку даних [11].

Такий вибір дозволяє ефективно працювати з великими обсягами даних, забезпечує швидкий доступ до них і можливість масштабування системи [20].

1.4.3 Реалізація користувацького інтерфейсу

Користувацький інтерфейс реалізується з використанням HTML, CSS та JavaScript, що забезпечує гнучкість розробки та зручність використання. Ці технології є стандартом у веб-розробці й коректно працюють у різних браузерах та на різних пристроях [13]. Адаптивний дизайн забезпечує нормальне відображення інтерфейсу на мобільних пристроях [25].

					БР.ІІІ - 07.00.00.000 ПЗ	Арк.
						21
Змн.	Арк.	№ докум.	Підпис	Дата		

JavaScript використовується для реалізації динамічних елементів і підвищення інтерактивності без постійного перезавантаження сторінок [22]. Відмова від складних фронтенд-фреймворків спрощує розробку та інтеграцію клієнтської і серверної частин, а також дозволяє уникати зайвого ускладнення структури коду.

1.4.4 Організація взаємодії клієнтської та серверної частин

Взаємодія між клієнтом і сервером відбувається через HTTP-запити, що дозволяє забезпечити нормальну роботу всієї системи. Такий підхід дає змогу розділити обробку даних і їх відображення, що спрощує розробку та подальшу підтримку [4]. Клієнтська частина відповідає за відображення інтерфейсу та взаємодію з користувачем, тоді як сервер обробляє запити і працює з даними [5]. Це дозволяє більш чітко розподілити функціональність між частинами системи.

Організований обмін даними зменшує затримки, підвищує швидкість обробки запитів і загальну продуктивність системи, особливо коли користувачів багато. Крім цього, використання стандартних протоколів дозволяє забезпечити сумісність із різними клієнтами і сервісами [13]. Розділення логіки клієнта і сервера спрощує подальший розвиток системи [20].

1.4.5 Реалізація аналітичних можливостей

Аналітика реалізується через обробку даних на сервері з подальшим їх відображенням у вигляді графіків і діаграм. Для цього використовуються інструменти візуалізації, які дозволяють зручно показувати інформацію у зрозумілому вигляді [15]. Дані для аналізу збираються під час роботи системи та зберігаються в базі даних для подальшої обробки [14].

У системі враховуються показники обсягу продажів, кількості замовлень і активності користувачів, що допомагає помічати певні тенденції та приймати більш обґрунтовані рішення [1]. Такий підхід спрощує аналіз інформації і робить його більш наочним.

Такий підхід дає більш повне уявлення про дані та дозволяє ефективніше їх

					БР.ІП - 07.00.00.000 ПЗ	Арк.
						22
Змн.	Арк.	№ докум.	Підпис	Дата		

використовувати порівняно з базовими аналітичними рішеннями [28].

1.4.6 Використання інструментів розробки та підтримки

Під час розробки використовуються інструменти контролю версій і автоматизації, що дозволяє відстежувати зміни, організовувати командну роботу та покращувати якість коду [30]. Системи контролю версій дають можливість зберігати історію змін і спрощують роботу кількох розробників над одним застосунком.

Автоматизація окремих процесів зменшує кількість ручних дій, спрощує тестування, розгортання та подальшу підтримку системи [20]. Вона також допомагає швидше знаходити проблеми і позитивно впливає на стабільність роботи веб-застосунку [17].

Таким чином, обрані технології та інструменти забезпечують продуктивність, зручність використання та можливість подальшого розвитку системи [4].

1. 5 Висновки по розділу:

Аналіз предметної області електронної комерції дозволив визначити основні принципи роботи веб-застосунків інтернет-магазинів і сформулювати вимоги до системи. Проведений аналіз показав, що наявні рішення мають обмеження в гнучкості, кастомізації та аналітиці, і це враховувалось під час розробки системи. Вибір технологій, зокрема Ruby on Rails, пояснюється тим, що з ним можна швидко розробляти, зручніше підтримувати систему та мати достатній рівень безпеки.

					БР.ІІІ - 07.00.00.000 ПЗ	Арк.
						23
Змн.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 2. АНАЛІЗ ВИМОГ ТА ПОСТАНОВКА ЗАДАЧІ

Формування вимог до веб-застосунку є важливим етапом розробки, оскільки саме воно визначає його функціональні можливості, обмеження та подальший розвиток [4]. Чітко прописані вимоги допомагають краще зрозуміти, якою має бути система, а також зменшують ризик помилок на наступних етапах.

На основі аналізу предметної області та існуючих рішень визначаються основні характеристики системи, її функціональність і структура [5]. Важливо врахувати різні типи користувачів і сценарії їх взаємодії із застосунком, адже це напряду впливає на подальше проєктування системи [6].

2.1 Постановка задачі

Постановка задачі визначає мету створення веб-застосунку та основні кроки, необхідні для її досягнення [4]. Вона базується на результатах аналізу предметної області та допомагає сформулювати загальні вимоги до системи.

2.1.1 Мета створення системи

Метою проєкту є створення веб-застосунку електронної комерції, який дозволяє користувачам переглядати товари, додавати їх до кошика, оформлювати замовлення та взаємодіяти із системою через зручний інтерфейс [1].

Крім цього, система повинна стабільно працювати, забезпечувати захист даних користувачів і мати можливість подальшого розширення функціоналу [12]. Окремо увага приділяється зручності взаємодії користувача із системою та підтримці основних бізнес-процесів інтернет-магазину [6].

2.1.2 Основні завдання розробки

Для досягнення поставленої мети необхідно виконати такі завдання:

- визначити функціональні та нефункціональні вимоги до системи;
- сформулювати модель користувачів і сценарії їх взаємодії;
- спроектувати структуру веб-застосунку;

					БР.ІІІ - 07.00.00.000 ПЗ	Арк.
						24
Змн.	Арк.	№ докум.	Підпис	Дата		

- розробити структуру бази даних;
- спроектувати користувацький та адміністративний інтерфейси;
- визначити структуру програмного забезпечення.

Виконання цих завдань дозволяє сформувати логічну структуру системи та забезпечити її відповідність поставленим вимогам [4].

2.1.3 Очікувані результати

Результатом є веб-застосунок електронної комерції, який відповідає визначеним вимогам і забезпечує ефективну взаємодію користувачів із системою.

Передбачається створення структурованої, масштабованої та зручної у використанні системи, яка підтримує основні бізнес-процеси інтернет-магазину, забезпечує обробку даних і може бути розширена відповідно до подальших потреб [5].

2.2 Формування функціональних вимог

Функціональні вимоги окреслюють можливості веб-застосунку та детально описують дії, які можуть виконувати користувачі в системі. Вони становлять основу для реалізації ключових процесів електронної комерції [4].

У системі передбачено кілька ролей: гість, зареєстрований користувач і адміністратор. Кожна з цих ролей має свій набір прав доступу та функцій, що дозволяє чітко розмежувати доступ до даних і можливостей системи [5]. Такий підхід забезпечує зручність і безпеку взаємодії користувачів із застосунком [12].

2.2.1 Функціональні вимоги з боку користувача

Веб-застосунок має надавати такі функціональні можливості:

- перегляд головної сторінки та ознайомлення з асортиментом товарів;
- доступ до каталогу з категоріями та можливістю сортування;
- перегляд детальної інформації про товар (опис, ціна, зображення, характеристики);
- реєстрацію та авторизацію користувачів;

					БР.ІП - 07.00.00.000 ПЗ	Арк.
						25
Змн.	Арк.	№ докум.	Підпис	Дата		

- можливість залишення відгуків;
- додавання товарів до кошика, зміна кількості та видалення позицій;
- збереження стану кошика та його об'єднання після авторизації;
- оформлення замовлення з вибором способу доставки та оплати;
- перегляд історії замовлень і їх деталей;
- можливість скасування замовлення залежно від його статусу;
- перегляд і редагування профілю;
- надсилання повідомлень через контактну форму.

Ці функціональні можливості забезпечують повний цикл взаємодії користувача із системою — від вибору товару до оформлення замовлення та подальшого супроводу [1]. Взаємодія користувача із системою ілюструється у вигляді діаграми варіантів використання (рис. 2.1).

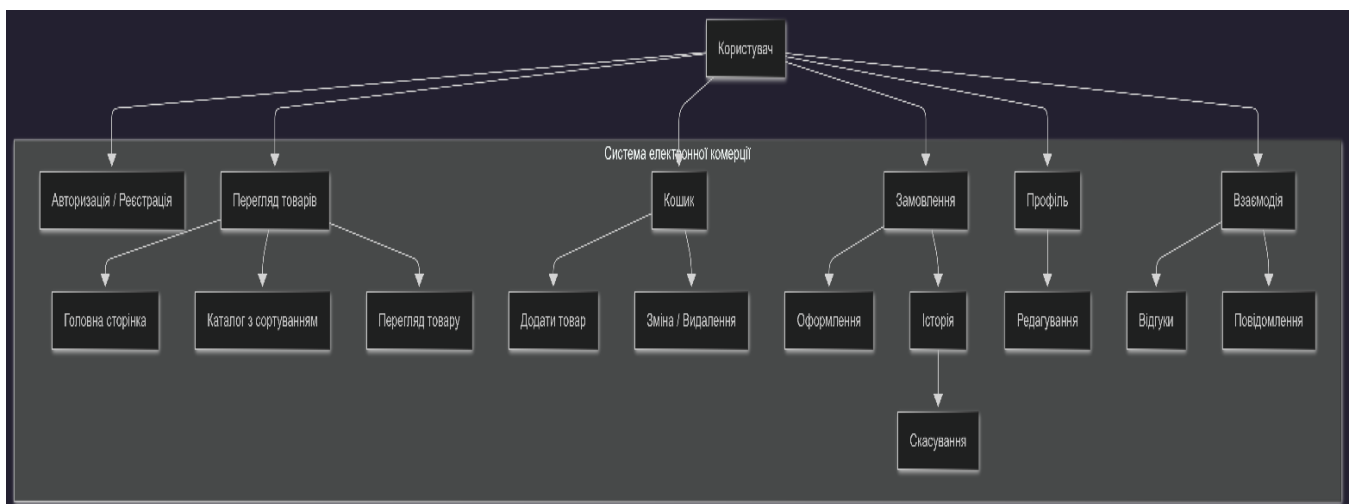


Рисунок 2.1 – Діаграма варіантів використання системи інтернет-магазину сумок

2.2.2 Функціональні вимоги з боку адміністратора

Адміністративна частина веб-застосунку має забезпечувати такі функціональні можливості:

- авторизацію адміністратора і доступ до адмін-панелі;
- управління товарами (додавання, редагування, видалення, налаштування характеристик і завантаження зображень);

- перегляд і обробку замовлень зі зміною їх статусів відповідно до бізнес-логіки;
- управління користувачами та їх повідомленнями (перегляд, редагування, видалення, призначення ролей, надання відповідей);
- перегляд аналітичної інформації (дохід, кількість замовлень, популярні товари, активність користувачів) з можливістю фільтрації за періодами та візуалізації;
- обмеження доступу до адміністративного функціоналу для неавторизованих користувачів.

Ці можливості дозволяють централізовано керувати системою і контролювати її роботу, а також дивитись ефективність через аналітику [1].

Функціональні можливості адміністратора показані у вигляді діаграми взаємодії з адміністративною панеллю (рис. 2.2).

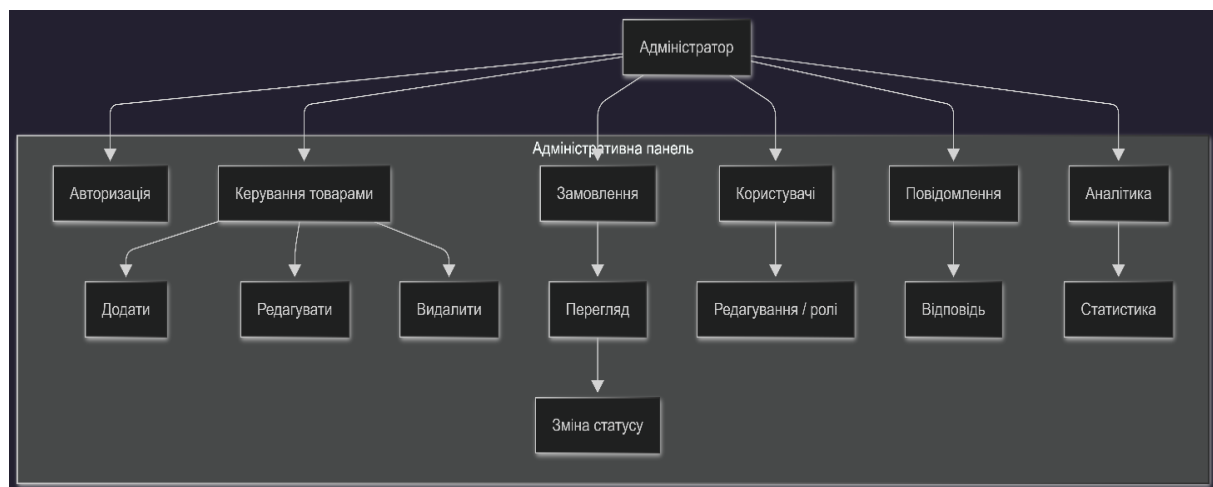


Рисунок 2.2 – Діаграма взаємодії адміністратора з адміністративною панеллю

2.3 Формування нефункціональних вимог

Нефункціональні вимоги характеризують якісні аспекти веб-застосунку та умови його функціонування. Вони не зосереджуються на тому, що система виконує, а на тому, як ефективно вона це робить. До таких вимог зазвичай належать продуктивність, безпека, масштабованість, зручність використання та надійність [4]. Визначення цих вимог є критично важливим для систем електронної комерції,

					БР.ІІІ - 07.00.00.000 ПЗ	Арк.
						27
Змн.	Арк.	№ докум.	Підпис	Дата		

оскільки вони обслуговують велику кількість користувачів і обробляють значні обсяги даних [4].

2.3.1 Продуктивність

Веб-застосунок має забезпечувати швидке завантаження сторінок і стабільну роботу навіть за високого навантаження. Це особливо важливо для операцій, пов'язаних із переглядом каталогу, управлінням кошиком та оформленням замовлення [20]. Необхідний рівень продуктивності досягається оптимізацією роботи з базою даних, ефективною обробкою запитів, зменшенням кількості звернень до сервера та використанням кешування, що знижує час відповіді системи і підвищує зручність роботи з нею [14].

2.3.2 Безпека

Система повинна забезпечувати захист персональних даних користувачів і обмеження доступу до адміністративних функцій. Дані облікових записів зберігаються у зашифрованому вигляді, а доступ до системи реалізується через механізми автентифікації та авторизації [10].

Також враховується захист від поширених веб-загроз, зокрема SQL-ін'єкцій, XSS та CSRF, що забезпечується перевіркою даних, контролем доступу та використанням вбудованих можливостей фреймворку [12].

Доступ до адміністративної частини застосунку мають лише авторизовані адміністратори. Додатково передбачено безпечне зберігання паролів і захист даних під час передачі запитів [20].

2.3.3 Масштабованість

Архітектура веб-застосунку повинна дозволяти розширення функціоналу без значних змін у структурі системи [5]. Це забезпечує можливість адаптації до зростання кількості користувачів і обсягів даних. Використання модульного підходу та поділу системи на компоненти робить її більш гнучкою, спрощуючи подальший розвиток і інтеграцію нових сервісів [4].

					БР.ІІІ - 07.00.00.000 ПЗ	Арк.
						28
Змн.	Арк.	№ докум.	Підпис	Дата		

2.3.4 Зручність використання (юзабіліті)

Інтерфейс веб-застосунку має бути зрозумілим і логічно структурованим. Навігація повинна забезпечувати швидкий доступ до основних функцій і зменшувати кількість дій, які потрібно виконати користувачу [6]. Важливим аспектом є адаптивний дизайн, що забезпечує коректне відображення застосунку на різних пристроях, підвищуючи зручність користування та покращуючи загальний досвід користувачів [13].

2.3.5 Надійність

Веб-застосунок повинен стабільно функціонувати в різних умовах і коректно обробляти помилки, що допомагає уникнути збоїв і зберегти цілісність даних. Особливу увагу слід приділити виконанню критичних операцій, зокрема оформленню замовлення [11]. Використання механізмів контролю та обробки помилок підвищує загальну стабільність системи. Таким чином, нефункціональні вимоги визначають умови ефективної, безпечної та стабільної роботи веб-застосунку, доповнюючи функціональні вимоги і формуючи основу для подальшого проектування [4].

2.4 Моделювання користувачів та сценаріїв взаємодії

Моделювання користувачів є важливим етапом у процесі проектування веб-застосунку, оскільки дозволяє визначити основні ролі, які взаємодіють із системою, та краще зрозуміти їх поведінку [5].

Важливим елементом також є опис сценаріїв взаємодії, які відображають послідовність дій користувачів під час роботи із системою. Такий підхід дозволяє виявити ключові процеси та зробити структуру веб-застосунку більш зрозумілою [4].

2.4.1 Типи користувачів системи

У веб-застосунку виділяються три основні категорії користувачів: гість, зареєстрований користувач і адміністратор. Кожна з цих ролей має свій рівень

					БР.ІІІ - 07.00.00.000 ПЗ	Арк.
						29
Змн.	Арк.	№ докум.	Підпис	Дата		

доступу та набір функцій, що дозволяє більш-менш чітко розмежувати права в системі [12]. Такий поділ допомагає уникнути зайвого доступу до функціоналу.

Гість не має облікового запису і може тільки переглядати каталог товарів та їх детальну інформацію, що дає змогу ознайомитися з асортиментом перед реєстрацією.

Зареєстрований користувач має більше можливостей: може додавати товари до кошика, оформлювати замовлення, переглядати історію покупок, редагувати профіль і залишати відгуки [1]. Його взаємодія охоплює весь процес покупки.

Адміністратор має найвищий рівень доступу та відповідає за управління товарами, замовленнями, користувачами, повідомленнями та аналітикою через адміністративну панель. Також він контролює роботу системи та може досить швидко вносити зміни [4].

Чітке розмежування ролей підвищує безпеку системи та допомагає краще організувати її роботу.

2.4.2 Сценарії взаємодії користувачів із системою

Сценарії взаємодії детально описують послідовність дій користувачів та ілюструють загальну логіку функціонування системи. Вони сприяють визначенню ключових процесів, що відбуваються у веб-застосунку, та забезпечують глибше розуміння його архітектури [5].

Ключовим сценарієм є процес оформлення замовлення, що включає перегляд товарів, їх додавання до кошика, введення необхідних даних, створення замовлення, а також вибір способу оплати та доставки [1]. Цей процес є центральним, оскільки безпосередньо пов'язаний з основною метою системи — здійсненням покупки.

Крім того, існують допоміжні сценарії, такі як реєстрація, авторизація, управління профілем і кошиком, які забезпечують повноцінну взаємодію користувача із системою [4].

Опис сценаріїв взаємодії також дозволяє краще визначити послідовність виконання окремих операцій у системі та виявити можливі проблеми ще на етапі

					БР.ІІІ - 07.00.00.000 ПЗ	Арк.
						30
Змн.	Арк.	№ докум.	Підпис	Дата		

проектування. Це особливо важливо для процесів, пов'язаних з оформленням замовлення, зміною статусів або взаємодією користувача з кошиком, оскільки такі операції прямо впливають на коректність роботи веб-застосунку і загальне враження користувача від системи [20].

Процес оформлення замовлення представлено у вигляді діаграми діяльності (рис. 2.3), що демонструє послідовність дій та зв'язки між окремими етапами.

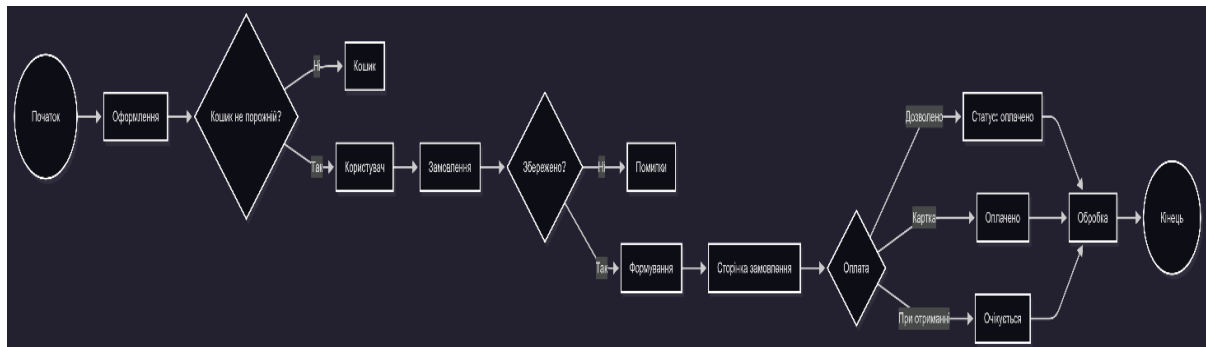


Рисунок 2.3 – Діаграма діяльності процесу оформлення замовлення

Таким чином, моделювання користувачів і сценаріїв взаємодії сприяє розумінню логіки роботи системи, виокремленню основних процесів та формуванню основи для подальшого проектування веб-застосунку [5].

2.5 Висновки по розділу:

Сформовані функціональні та нефункціональні вимоги дозволили визначити основні характеристики веб-застосунку електронної комерції, його структуру та особливості взаємодії користувачів із системою. Визначення ролей користувачів і сценаріїв роботи допомогло краще зрозуміти логіку виконання основних процесів і сформуванню основи для подальшого проектування архітектури застосунку.

Врахування вимог до продуктивності, безпеки, масштабованості та зручності використання дозволяє забезпечити стабільну роботу системи та зробити взаємодію користувачів із веб-застосунком більш зручною. Це також спрощує подальший розвиток системи.

РОЗДІЛ 3. ПРОЄКТУВАННЯ АРХІТЕКТУРИ ТА СТРУКТУРИ СИСТЕМИ

Проектування архітектури та структури системи є одним із основних етапів розробки веб-застосунку, під час якого визначається організація компонентів, структура даних і принципи взаємодії між частинами системи [4].

Під час проектування враховуються функціональні та нефункціональні вимоги системи. Окрема увага приділяється архітектурі веб-застосунку, структурі бази даних, організації інтерфейсів і загальній структурі програмного забезпечення [20].

3.1 Проектування архітектури веб-застосунку

Проектування архітектури веб-застосунку — це важливий етап, бо саме тут визначається структура системи, як між собою взаємодіють компоненти і як обробляються дані [5]. Від того, яку архітектуру обрати, залежить продуктивність, масштабованість, надійність і те, наскільки легко потім буде підтримувати систему [4]. Це особливо актуально для електронної комерції, адже такі застосунки працюють з великими обсягами даних і часто мають навантаження.

У системі планується використовувати клієнт-серверний підхід разом із архітектурним шаблоном MVC, що дозволяє чітко розділити функціональність [5].

3.1.1 Загальна архітектура системи

Веб-застосунок проектується на основі клієнт-серверної архітектури з трьома основними рівнями: клієнтським, серверним і рівнем зберігання даних [4].

Клієнтський рівень працює через веб-браузер і забезпечує взаємодію користувача із системою. Серверний рівень відповідає за обробку запитів, реалізацію бізнес-логіки та взаємодію з базою даних за допомогою фреймворку Ruby on Rails [7]. Рівень зберігання даних реалізований на основі PostgreSQL [14].

Така структура дозволяє розділити відповідальність між компонентами системи, спрощує підтримку та забезпечує можливість масштабування веб-застосунку (рис. 3.1).

					БР.ІІІ - 07.00.00.000 ПЗ	Арк.
						32
Змн.	Арк.	№ докум.	Підпис	Дата		

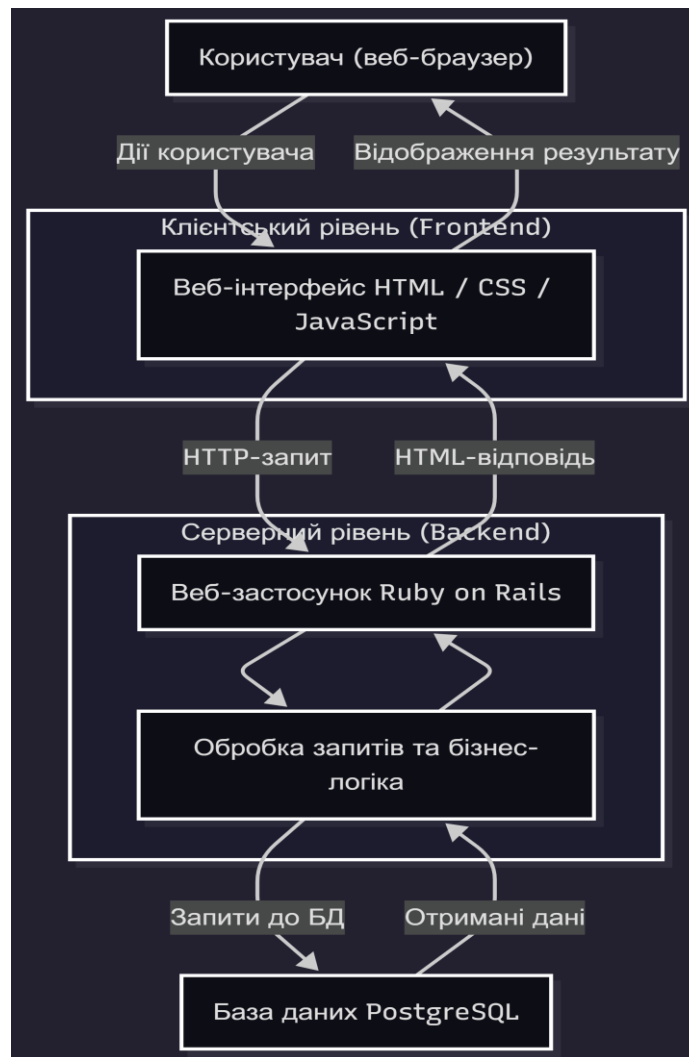


Рисунок 3.1 – Архітектура веб-застосунку інтернет-магазину

3.1.2 Архітектурний підхід та шаблон MVC

У веб-застосунку планується використовувати архітектурний шаблон Model–View–Controller (MVC), який передбачає поділ системи на три основні частини [5]. Модель (Model) відповідає за обробку даних, роботу з базою даних і реалізацію бізнес-логіки. Представлення (View) відповідає за відображення інформації для користувача. Контролер (Controller) обробляє запити, взаємодіє з моделлю та передає результати у представлення [7].

Такий підхід робить систему більш модульною, спрощує тестування і підтримку, а також дозволяє простіше вносити зміни в майбутньому [4]. Взаємодію компонентів показано на рисунку (рис. 3.2).

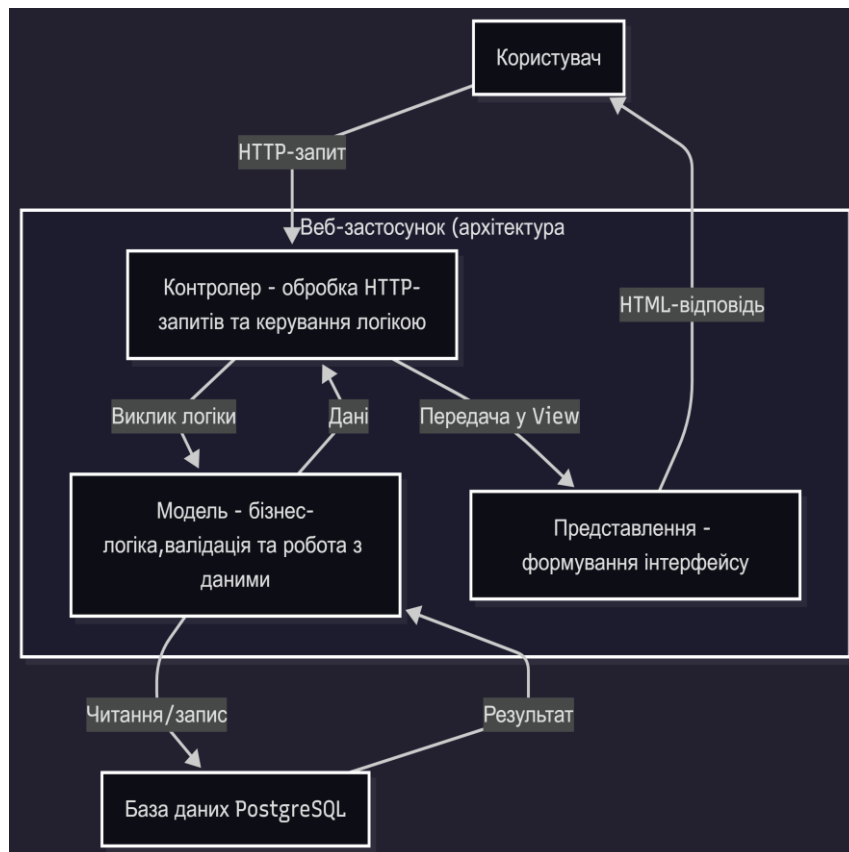


Рисунок 3.2 – Структура взаємодії компонентів за шаблоном MVC

3.1.3 Компонентна структура системи

Архітектура веб-застосунку передбачає поділ системи на окремі функціональні компоненти, де кожен відповідає за свою частину логіки [5]. Основними компонентами є модулі користувачів, товарів, кошика, замовлень і аналітики [1].

Модуль користувачів відповідає за авторизацію та роботу з профілями, модуль товарів - за зберігання інформації про продукцію, а модулі кошика й замовлень - за оформлення покупок і обробку замовлень.

Такий підхід спрощує підтримку системи та дозволяє змінювати окремі компоненти незалежно один від одного [4].

3.1.4 Взаємодія між компонентами

Взаємодія між компонентами відбувається через обробку HTTP-запитів [13]. Користувач відправляє запит через інтерфейс, після чого він переходить до

контролера. Контролер обробляє цей запит, звертається до моделі, щоб отримати або змінити дані, і передає результат у представлення [5]. Представлення формує відповідь у вигляді веб-сторінки і відображає її користувачу.

Така структура допомагає правильно розподілити відповідальність між компонентами, зменшити їх залежність один від одного і зробити роботу системи більш ефективною [4].

3.2 Проєктування структури бази даних

Проєктування структури бази даних є важливим етапом розробки, оскільки від правильності її побудови залежить продуктивність застосунку, швидкість виконання запитів, цілісність даних і можливість подальшого розширення функціоналу [4].

У системах електронної комерції база даних є центральним елементом, де зберігається інформація про користувачів, товари, замовлення, кошик та інші сутності. Її структура формується з урахуванням функціональних вимог і бізнес-логіки системи [5].

3.2.1 Вибір моделі даних та СУБД

Для реалізації системи планується використання реляційної моделі даних, яка дозволяє зберігати інформацію у вигляді пов'язаних між собою таблиць і підтримувати цілісність даних [4].

Як систему керування базами даних планується використовувати PostgreSQL, яка забезпечує транзакції, індексацію та ефективну роботу з великими обсягами інформації, що важливо для систем електронної комерції [14].

3.2.2 Опис структури бази даних

Структура бази даних проєктується з урахуванням логіки предметної області та функціональних можливостей системи [5]. Кожна сутність представлена окремою таблицею, що робить роботу з даними більш зручною. Основними

сутностями є користувачі, товари, категорії, замовлення, елементи замовлення, кошик і повідомлення. Така структура дозволяє коректно відобразити основні бізнес-процеси і робить систему більш гнучкою [1].

3.2.3 Опис основних таблиць і зв'язків

Таблиця користувачів міститиме облікові дані та ролі користувачів у системі. Один користувач зможе створювати кілька замовлень, що відповідає зв'язку типу «один-до-багатьох».

Таблиці товарів і категорій використовуватимуться для зберігання інформації про продукцію, а таблиця замовлень — для даних про покупки користувачів.

Таблиця елементів замовлення реалізуватиме зв'язок «багато-до-багатьох» між товарами і замовленнями.

Кошик використовуватиметься для тимчасового зберігання товарів, а таблиця повідомлень — для зворотного зв'язку з користувачами. Зв'язки між таблицями реалізовані за допомогою зовнішніх ключів, що забезпечує цілісність даних [14].

3.2.4 Логічна модель бази даних

Логічна модель бази даних подається у вигляді ER-діаграми, яка показує основні сутності системи та зв'язки між ними. Це допомагає краще зрозуміти структуру даних і загалом робить її більш наочною.

У моделі відображені таблиці користувачів, товарів, категорій, замовлень, елементів замовлення, кошика та повідомлень. Для кожної сутності визначені основні атрибути, які використовуються під час роботи системи. Також на ER-діаграмі показані зв'язки між таблицями, що забезпечують цілісність даних і коректну взаємодію між компонентами веб-застосунку. Окремо відображені зв'язки між користувачами та замовленнями, а також між товарами й елементами замовлення, які є основою бізнес-логіки системи [14].

ER-діаграма наведена на рисунку (рис. 3.3).

					БР.ІІІ - 07.00.00.000 ПЗ	Арк.
						36
Змн.	Арк.	№ докум.	Підпис	Дата		

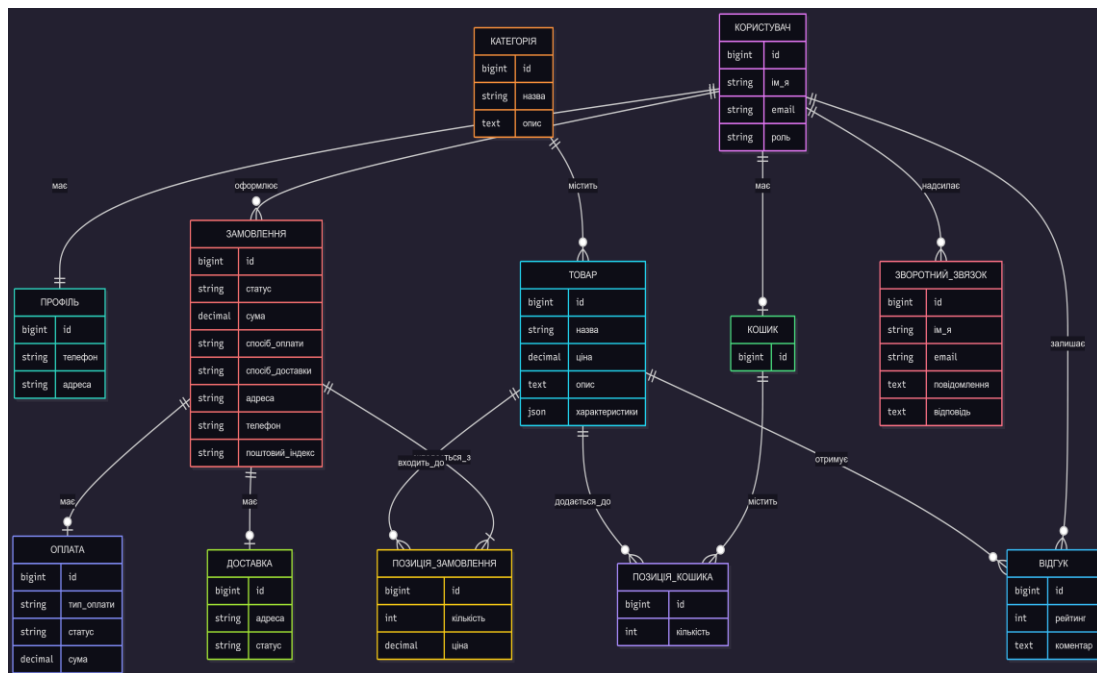


Рисунок 3.3 – ER-діаграма бази даних веб-застосунку

3.2.5 Забезпечення цілісності та оптимізації даних

Під час проєктування бази даних основний акцент робиться на цілісності даних і продуктивності системи. Для цього використовуються нормалізація, обмеження, індексація та транзакції [14].

Структура бази даних відповідає третій нормальній формі (3NF), що дозволяє уникнути дублювання даних [4]. Цілісність забезпечується за рахунок первинних і зовнішніх ключів, а також обмежень типу NOT NULL.

Для покращення продуктивності використовується індексація полів, які найчастіше застосовуються у запитах. Транзакції забезпечують коректне виконання важливих операцій, зокрема оформлення замовлень.

Спроектована структура бази даних забезпечує цілісність даних і стабільну роботу системи [20].

3.3 Проектування користувацького та адміністративного інтерфейсів

Проектування інтерфейсу є важливим етапом розробки веб-застосунку, оскільки саме через нього користувач взаємодіє із системою [6]. У сучасних веб-

застосунках електронної комерції основна увага приділяється логічній структурі, зручній навігації та адаптивності інтерфейсу [13].

Система передбачає два типи інтерфейсів: користувацький та адміністративний, кожен з яких орієнтований на свою цільову аудиторію [5].

3.3.1 Проектування користувацького інтерфейсу

Користувацький інтерфейс розробляється з урахуванням потреб гостей і зареєстрованих користувачів [6]. Основна мета полягає в тому, щоб користувач міг швидко знаходити товари, переглядати інформацію та оформляти покупку.

Структура інтерфейсу формується з урахуванням логіки сторінок і навігації між ними [5]. Основні сторінки включають головну, каталог товарів, сторінку товару, кошик, оформлення замовлення, профіль та історію замовлень. Зв'язки між ними відображені на діаграмі навігації (рис. 3.4).

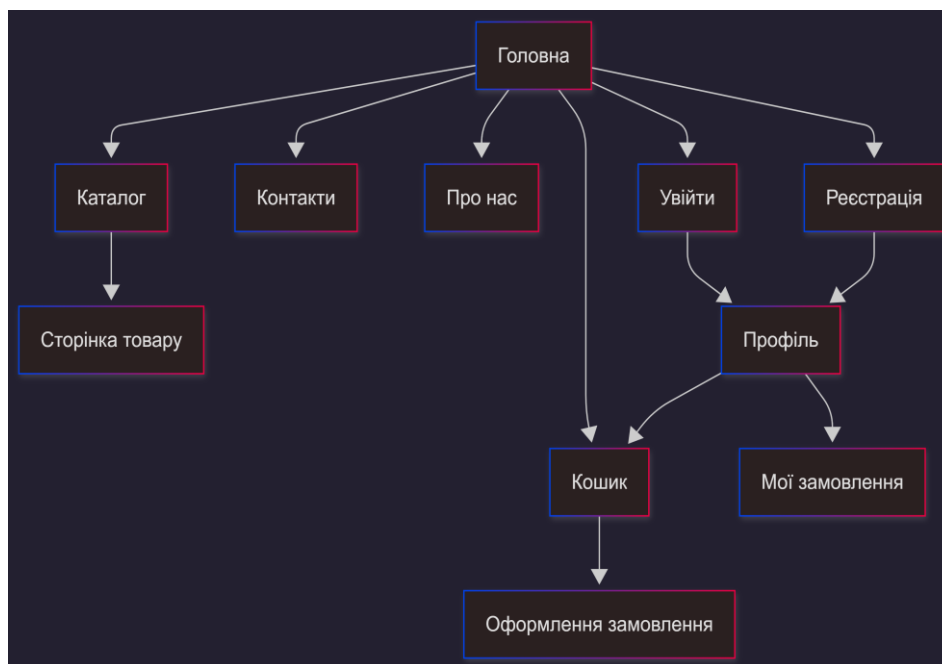


Рисунок 3.4 – Структура користувацького інтерфейсу веб-застосунку

Головна сторінка забезпечує доступ до основних розділів, каталог — перегляд товарів із пошуком і сортуванням [1], а сторінка товару містить детальну інформацію та можливість додавання до кошика.

Кошик використовується для керування вибраними товарами, а сторінка оформлення замовлення — для введення даних і підтвердження покупки.

3.3.2 Проєктування адміністративного інтерфейсу

Адміністративний інтерфейс розробляється для керування системою та контролю основних процесів [4]. Доступ до нього мають лише авторизовані користувачі з відповідними правами [12].

Структура інтерфейсу включає розділи для роботи з товарами, категоріями, замовленнями, користувачами та повідомленнями. Структуру адміністративного інтерфейсу наведено на рисунку 3.5.

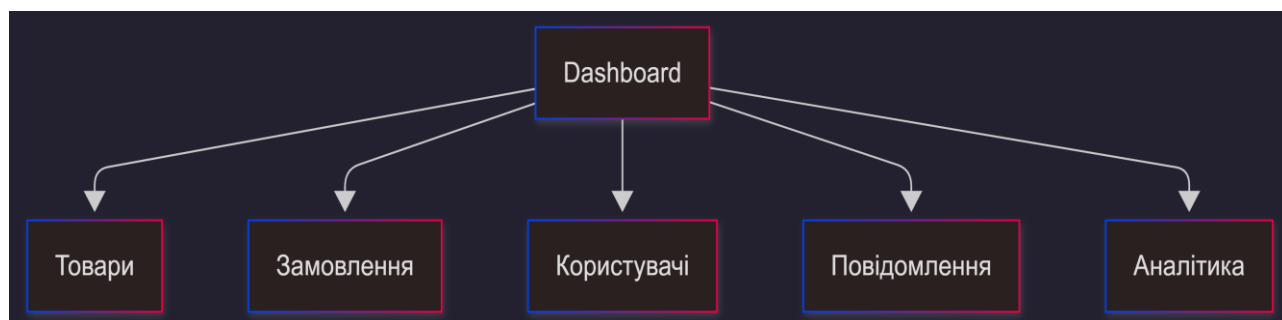


Рисунок 3.5 – Структура адміністративного інтерфейсу веб-застосунку

Інтерфейс організований у вигляді робочого середовища з навігаційним меню, таблицями та формами для роботи з даними. Передбачені функції пошуку, фільтрації, сортування та зміни статусів замовлень відповідно до бізнес-логіки [1].

3.3.3 Принципи проєктування інтерфейсу

Під час проєктування інтерфейсів враховуються основні принципи UX/UI, зокрема інтуїтивність, логічна структура, узгодженість стилю та адаптивність [6].

Єдиний стиль оформлення та зручна навігація спрощують взаємодію користувача із системою. Зменшення когнітивного навантаження досягається за рахунок зрозумілої ієрархії елементів і обмеження кількості одночасних дій [13].

Адаптивність інтерфейсу забезпечує коректне відображення на різних

пристроях, що важливо для сучасних веб-застосунків [25].

3.4 Структура програмного забезпечення

Відповідно до обраної архітектури та структури бази даних формується загальна організація програмного забезпечення [5]. Чітка структура проєкту спрощує розробку, підтримку та масштабування системи, а також дозволяє логічно розділити функціональні частини застосунку й зменшити залежність між модулями [7].

Розподіл компонентів за їх призначенням спрощує внесення змін, тестування та контроль взаємодії між окремими модулями веб-застосунку [20].

3.4.1 Загальна структура проєкту

Веб-застосунок розробляється на основі архітектури Ruby on Rails, де проєкт поділено на окремі каталоги відповідно до їх призначення [7].

Каталоги відповідають за обробку запитів, управління даними, відображення інтерфейсу та налаштування системи [5].

Загальна структура проєкту веб-застосунку ілюструється на рисунку 3.6.

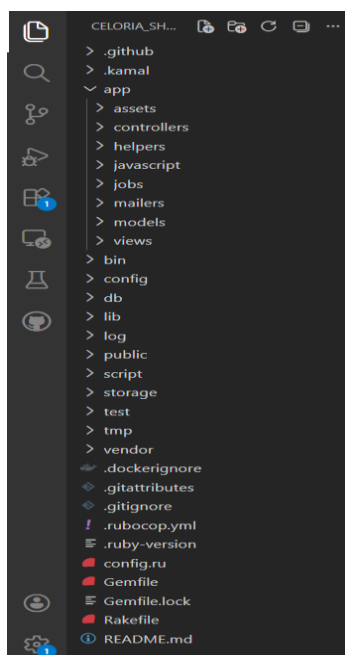


Рисунок 3.6 – Структура веб-застосунку

					БР.ІІІ - 07.00.00.000 ПЗ	Арк.
						40
Змн.	Арк.	№ докум.	Підпис	Дата		

3.4.2 Основні компоненти структури

У таблиці 3.1 представлені основні компоненти структури проєкту та їх функціональне призначення.

Таблиця 3.1

Компоненти структури проєкту

Компонент	Призначення
controllers	Обробка запитів від користувачів та впровадження логіки взаємодії.
models	Обробка даних та впровадження бізнес-логіки
views	Розробка користувацького інтерфейсу
assets	Зберігання стилів та зображень
config	Налаштування робочих параметрів застосунку
db	Структура бази даних та міграції
javascript	Динаміка поведінки застосунку

Кожен компонент виконує специфічну функцію в рамках системи та взаємодіє з іншими відповідно до архітектурних принципів [4]. Цей підхід забезпечує модульність і спрощує внесення змін у процесі еволюції системи.

Розподіл функціоналу між компонентами дозволяє краще організувати структуру проєкту та спрощує підтримку коду. Така організація також допомагає швидше знаходити потрібні частини системи під час розробки й тестування.

3.4.3 Відповідність архітектурі MVC

Структура проєкту ґрунтується на шаблоні Model–View–Controller: моделі відповідають за обробку даних, представлення — за візуалізацію інтерфейсу, а контролери — за управління запитами та взаємодію між компонентами системи [5].

Такий підхід сприяє чіткішому розподілу відповідальності, зменшенню залежностей між елементами та спрощенню процесу підтримки системи [4].

Використання шаблону MVC також спрощує розробку окремих компонентів системи, оскільки зміни в одній частині застосунку менше впливають на інші компоненти. Це робить структуру проєкту більш зрозумілою і спрощує подальшу підтримку веб-застосунку [7].

Додатковою перевагою є можливість окремо працювати з логікою,

					БР.ІІІ - 07.00.00.000 ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		41

інтерфейсом і даними без порушення роботи інших частин системи. Такий підхід також спрощує тестування та подальше розширення функціоналу веб-застосунку.

3.4.4 Переваги обраної структури

Структурований підхід до організації програмного забезпечення має чимало плюсів: зрозумілий розподіл відповідальності, простіший процес розробки, а також зручність у підтримці та тестуванні [20].

Окремою перевагою є можливість більш зручно організувати командну роботу над проектом, оскільки різні частини системи можуть розроблятися незалежно одна від одної. Це допомагає зменшити кількість конфліктів у коді та спрощує внесення змін [30].

Обрана структура дає змогу без особливих труднощів масштабувати систему, додавати нові функції і підключати додаткові сервіси без серйозних змін [5].

Крім цього, чітка структура програмного забезпечення позитивно впливає на читабельність коду та спрощує пошук можливих помилок. Це дозволяє зробити систему більш зрозумілою не тільки під час розробки, а й під час подальшої підтримки та оновлення функціоналу [4].

3.5 Висновки по розділу:

Спроектowana архітектура веб-застосунку дозволила визначити основні компоненти системи та організувати їх взаємодію відповідно до клієнт-серверного підходу і шаблону MVC. Використання такого підходу спрощує підтримку системи, розподіл функціональності та подальше розширення веб-застосунку.

Сформована структура бази даних забезпечує цілісність і коректне зберігання інформації, а проектування користувацького та адміністративного інтерфейсів дозволило зробити взаємодію із системою більш зрозумілою та зручною. Визначена структура програмного забезпечення спрощує підтримку коду і створює основу для подальшого розвитку функціоналу веб-застосунку.

					БР.ІІІ - 07.00.00.000 ПЗ	Арк.
						42
Змн.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 4. РЕАЛІЗАЦІЯ FULL STACK ЗАСТОСУНКУ ЕЛЕКТРОННОЇ КОМЕРЦІЇ

Розробка веб-застосунку електронної комерції передбачає реалізацію клієнтської частини, серверної логіки та організацію роботи з даними [4]. Під час створення системи важливо забезпечити зручну взаємодію користувача з інтерфейсом, швидку обробку запитів і надійне збереження інформації [14].

Використання full stack підходу дозволяє об'єднати всі рівні застосунку в межах єдиної структури, що спрощує підтримку та подальший розвиток системи [7].

4.1 Реалізація користувацької частини застосунку

Користувацька частина веб-застосунку забезпечує взаємодію користувача із системою та формує основний інтерфейс роботи [6]. Реалізація виконана за допомогою серверного рендерингу з використанням шаблонів ERB і JavaScript-модулів для обробки подій та взаємодії з елементами сторінки [7].

Користувацька частина охоплює роботу з каталогом товарів, кошиком, оформленням замовлення та автентифікацією користувачів [13].

4.1.1 Структура користувацької частини веб-застосунку

Клієнтська частина застосунку побудована на основі архітектури Ruby on Rails і містить контролери та шаблони представлення для обробки запитів і формування інтерфейсу [7].

Структура користувацької частини організована таким чином, щоб розділити логіку обробки даних, відображення сторінок та взаємодію з користувачем. Окремі контролери відповідають за роботу з товарами, кошиком, замовленнями та автентифікацією користувачів. Такий підхід спрощує підтримку коду та внесення змін до окремих компонентів системи [4].

Контролери забезпечують обробку запитів і передачу даних у представлення. Їх структуру наведено на рисунку 4.1.

					БР.ІІІ - 07.00.00.000 ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		43

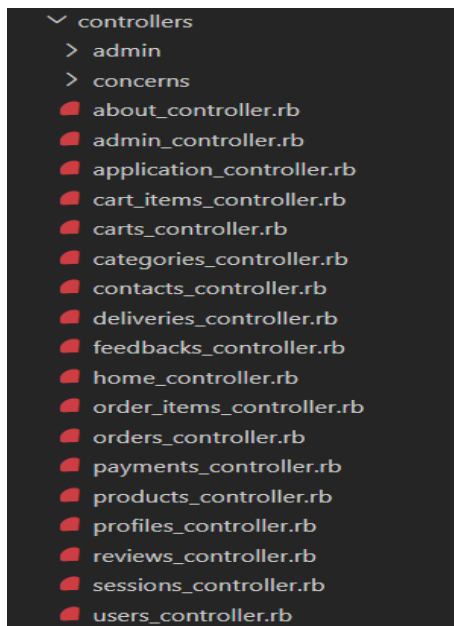


Рисунок 4.1 – Структура контролерів клієнтської частини

Шаблони представлення використовуються для формування сторінок веб-застосунку на основі отриманих даних. Їх структуру показано на рисунку 4.2.

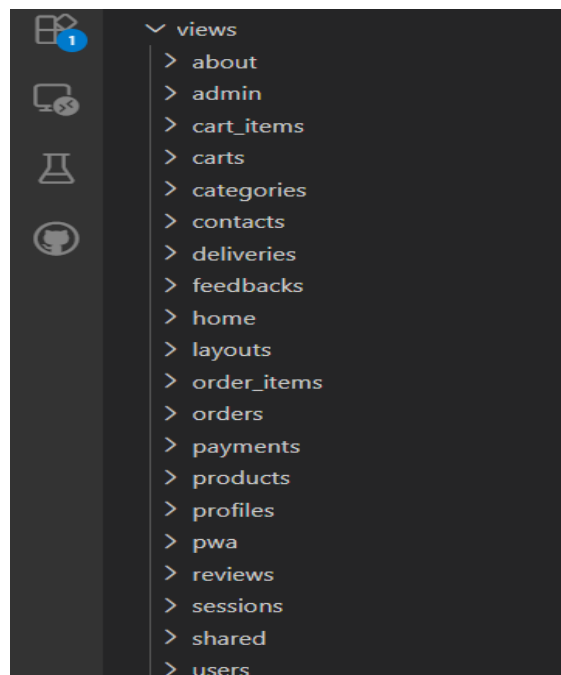


Рисунок 4.2 – Структура шаблонів представлення (views)

Реалізована структура забезпечує розподіл функціоналу між компонентами системи та спрощує подальшу підтримку веб-застосунку.

					БР.ІІІ - 07.00.00.000 ПЗ	Арк.
						44
Змн.	Арк.	№ докум.	Підпис	Дата		

4.1.2 Реалізація головної сторінки

Головна сторінка забезпечує доступ до основних розділів веб-застосунку та каталогу товарів [6].

Отримання товарів виконується через метод `index`, який формує список продукції та передає його у представлення [7].

```
def index
```

```
    @products = Product.order(created_at: :desc)
```

```
End
```

У верхній частині сторінки реалізовано hero-блок із заголовком і кнопкою переходу до каталогу. Його реалізацію показано на рисунку 4.3.

```
<div class="hero">
```

```
    <h1>Celoria — стиль у кожному кроці</h1>
```

```
    <%= link_to "Перейти до каталогу", products_path %>
```

```
</div>
```

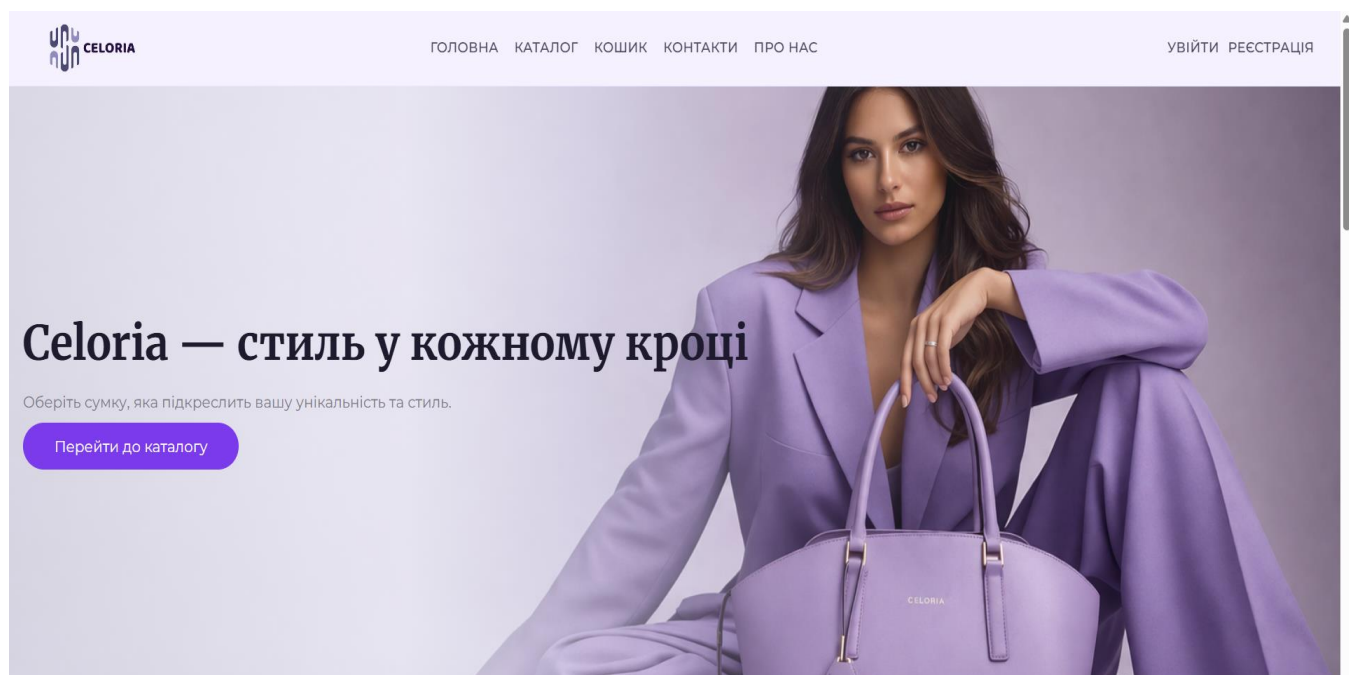


Рисунок 4.3 – Hero-блок головної сторінки

					БР.ІІІ - 07.00.00.000 ПЗ	Арк.
						45
Змн.	Арк.	№ докум.	Підпис	Дата		

Нижче розміщено блок товарів, де відображаються останні додані позиції. Для виведення використовується перебір отриманих даних із обмеженням кількості елементів [13]. Реалізацію цього блоку показано на рисунку 4.4.

```
<div class="products-grid">
  <% @products.limit(4).each do |product| %>
    <%= image_tag product.images.first %>
    <h3><%= product.name %></h3>
    <p>£<%= product.price %></p>
  <% end %>
</div>
```

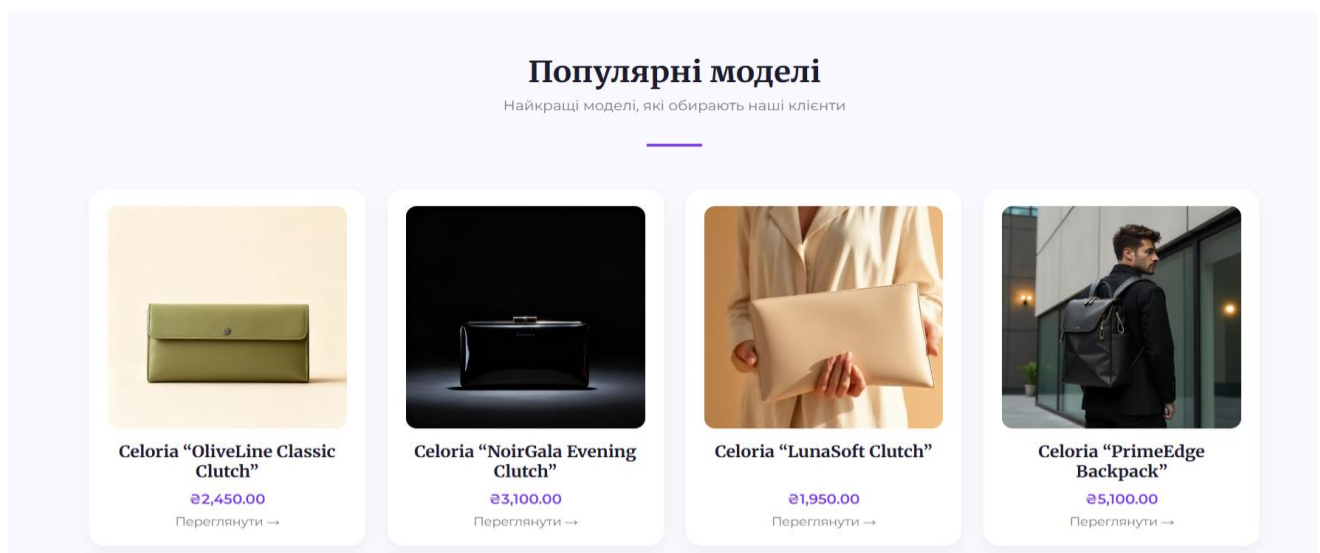


Рисунок 4.4 – Відображення товарів на головній сторінці

Реалізована структура головної сторінки забезпечує зручний перегляд основної інформації та навігацію між розділами системи [9].

4.1.3 Реалізація каталогу товарів

Каталог товарів забезпечує перегляд продукції, фільтрацію, сортування та посторінкову навігацію, що спрощує пошук потрібних позицій [6].

					БР.ІІІ - 07.00.00.000 ПЗ	Арк.
						46
Змн.	Арк.	№ докум.	Підпис	Дата		

Обробка даних виконується через контролер ProductsController. Метод index застосовує фільтрацію за категоріями, сортування та пагінацію [7].

```
def index
```

```
  @products = Product.all
```

```
  @products = @products.where(category_id: params[:category_id]) if  
  params[:category_id].present?
```

```
  @products = @products.order(created_at: :desc)
```

```
  @products = @products.page(params[:page]).per(8)
```

```
End
```

Для керування відображенням товарів реалізовано блок фільтрації та сортування, який показано на рисунку 4.5.



Рисунок 4.5 – Блок фільтрації та сортування товарів

Товари відображаються у вигляді карток із зображенням, назвою та ціною. Реалізацію цього блоку наведено нижче, а результат роботи показано на рисунку 4.6.

```
<div class="products-grid">  
  <% @products.each do |product| %>  
    <%= image_tag product.images.first %>
```

					БР.ІП - 07.00.00.000 ПЗ	Арк.
						47
Змн.	Арк.	№ докум.	Підпис	Дата		

```

<h3><%= product.name %></h3>

<p><%= product.price %></p>

<% end %>

</div>

```

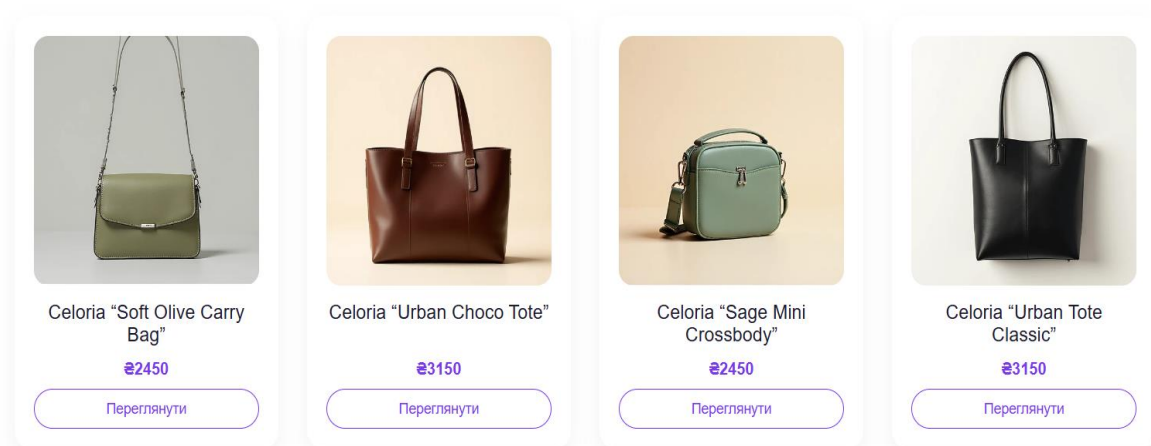


Рисунок 4.6 – Відображення товарів у каталозі

Реалізований каталог забезпечує зручний перегляд товарів і навігацію між сторінками веб-застосунку [9].

4.1.4 Реалізація сторінки товару

Сторінка товару використовується для детального перегляду інформації про продукцію перед оформленням покупки [6].

На сторінці відображаються зображення товару, опис, характеристики та відгуки користувачів.

Отримання даних виконується через контролер ProductsController, де метод show знаходить товар за його ідентифікатором [7].

```

def show
  @product = Product.find(params[:id])
End

```

Для перегляду зображень реалізовано галерею товару. Її реалізацію показано на рисунку 4.7.

```
<div class="product-thumbnails">
  <% @product.images.each do |photo| %>
    <%= image_tag photo %>
  <% end %>
</div>
```



Рисунок 4.7 – Галерея зображень товару

Основний блок сторінки містить назву товару, ціну та кнопку додавання до кошика. Приклад реалізації наведено нижче, а інтерфейс показано на рисунку 4.8.

```
<h1><%= @product.name %></h1>
<p>£<%= @product.price %></p>
```

					БР.ІП - 07.00.00.000 ПЗ	Арк.
						49
Змн.	Арк.	№ докум.	Підпис	Дата		

```
<%= button_to "Додати до кошика",  
  add_to_cart_path(product_id: @product.id) %>
```

Celoria “Urban Slim Tote”

Ціна: €3,250.00

Опис: Urban Slim Tote — стильна мінімалістична сумка для міста. Вона тонка, структурна, універсальна, підходить як для роботи, так і для університету. Легко вміщує ноутбук, документи та всі щоденні дрібниці.

Категорія: Сумки

Додати до кошика

Рисунок 4.8 – Інформаційний блок товару

На сторінці також реалізовано систему відгуків, яка дозволяє користувачам переглядати коментарі та залишати оцінки товару [13]. Інтерфейс блоку відгуків показано на рисунку 4.9.

★ Відгуки

Поки що відгуків немає. Будьте першим 😊

👍 Залишити відгук

Оцінка (1-5):

Ваш коментар:

Надіслати

🏠 Назад до каталогу

Рисунок 4.9 – Блок з відгуками про товар та формою для їх додавання

Реалізована сторінка товару забезпечує перегляд основної інформації про продукцію та взаємодію користувача із системою [8].

					БР.ІП - 07.00.00.000 ПЗ	Арк.
						50
Змн.	Арк.	№ докум.	Підпис	Дата		

4.1.5 Реалізація кошика

Кошик використовується для збереження вибраних товарів і формування замовлення перед оформленням покупки [4]. Реалізація виконана через сесію, що дозволяє зберігати товари між запитами користувача [11].

Основна логіка включає ініціалізацію кошика та додавання товарів. Приклад реалізації наведено нижче, а результат роботи показано на рисунку 4.10.

```
def load_cart
  session[:cart] ||= {}
  key = current_user ? current_user.id.to_s : "guest"
  session[:cart][key] ||= {}
  @cart = session[:cart][key]
end

def add
  product_id = params[:product_id].to_s
  @cart[product_id] ||= 0
  @cart[product_id] += 1
  save_cart
  redirect_to cart_path
end
```

Категорія: Сумки

Додати до кошика

Рисунок 4.10 – Додавання товару до кошика

Для зміни кількості товарів використовується окремий метод, який оновлює дані кошика.

					БР.ІП - 07.00.00.000 ПЗ	Арк.
						51
Змн.	Арк.	№ докум.	Підпис	Дата		

```
def update
  qty = params[:quantity].to_i
  if qty > 0
    @cart[params[:product_id].to_s] = qty
  else
    @cart.delete(params[:product_id].to_s)
  end
  save_cart
end
```

Відображення товарів у кошику реалізовано у вигляді списку з назвою товару, ціною та кількістю. Фрагмент реалізації наведено нижче, а інтерфейс кошика показано на рисунку 4.11.

```
<% @cart.each do |product_id, quantity| %>
  <% product = Product.find_by(id: product_id) %>
  <%= product.name %>
  <%= product.price %>
<% end %>
```

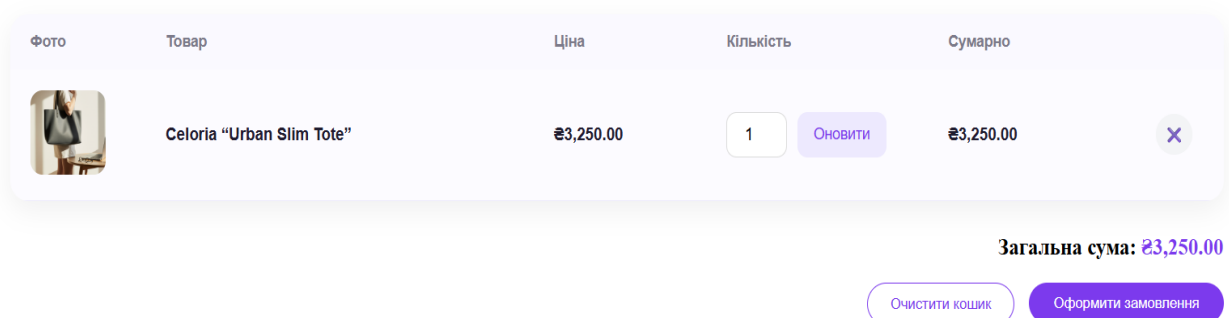


Рисунок 4.11 – Відображення товарів та загальної суми в кошику

Реалізація кошика забезпечує збереження вибраних товарів і підтримку стану між запитами користувача [11].

					БР.ІП - 07.00.00.000 ПЗ	Арк.
						52
Змн.	Арк.	№ докум.	Підпис	Дата		

4.1.6 Реалізація оформлення замовлення

Оформлення замовлення використовується для введення контактних даних користувача та створення замовлення на основі товарів із кошика [10].

Форма оформлення реалізована у шаблоні представлення та містить поля для введення персональних даних, доставки й оплати. Фрагмент реалізації наведено нижче, а інтерфейс показано на рисунку 4.12.

```
<%= form_with model: @order do |f| %>
  <%= f.text_field :first_name %>
  <%= f.telephone_field :phone %>
  <%= f.email_field :email %>
  <%= f.select :delivery_method, [["Нова Пошта", "nova_poshta"]] %>
  <%= f.submit "Підтвердити" %>
<% end %>
```

Оформлення замовлення

Ім'я

Прізвище

Телефон
Формат: +380XXXXXXXX

Email

Доставка

Адреса доставки

Оплата

Номер картки

MM/YY CVV

Підтвердити замовлення

Рисунок 4.12 – Форма оформлення замовлення

					БР.ІП - 07.00.00.000 ПЗ	Арк.
						53
Змн.	Арк.	№ докум.	Підпис	Дата		

Після підтвердження даних створюється запис замовлення та додаються товари з кошика. Основна логіка реалізації наведена нижче, а результат виконання показано на рисунку 4.13.

```
def create
  @order = Order.create(order_params.merge(
    total: calculate_total
  ))
  @cart.each do |id, qty|
    product = Product.find_by(id: id)
    @order.order_items.create(
      product: product,
      quantity: qty
    )
  end
  session[:cart] = {}
end
```

Рисунок 4.13 – Створення замовлення

					БР.ІП - 07.00.00.000 ПЗ	Арк.
						54
Змн.	Арк.	№ докум.	Підпис	Дата		

Інформація про оформлене замовлення відображається на окремій сторінці зі списком товарів і загальною вартістю. Інтерфейс сторінки показано на рисунку 4.14.

<h1>Замовлення №<%= @order.id %></h1>

<p>Разом: ₴<%= @order.total %></p>

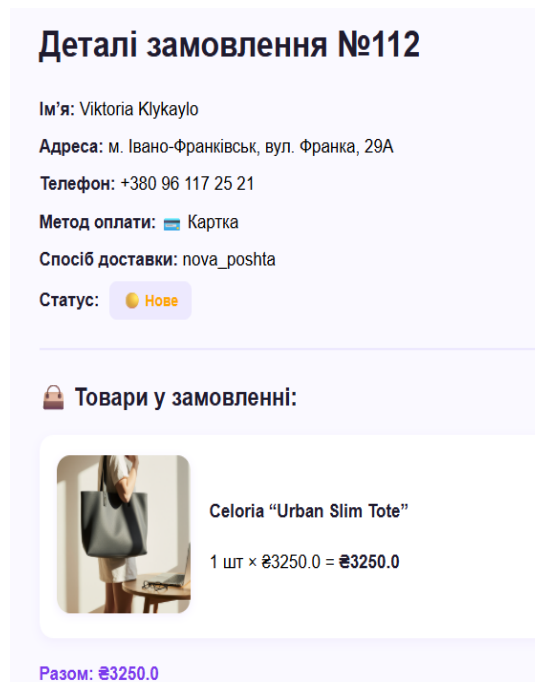


Рисунок 4.14 – Відображення інформації про замовлення

Реалізація оформлення замовлення забезпечує створення замовлення та збереження інформації про вибрані товари [11].

4.1.7 Реалізація автентифікації користувачів

Автентифікація використовується для визначення користувача та контролю доступу до функцій системи [10].

Процес входу реалізовано через контролер SessionsController. Метод create перевіряє облікові дані користувача та зберігає його ідентифікатор у сесії [7]. Фрагмент реалізації наведено нижче, а інтерфейс входу показано на рисунку 4.15.

					БР.ІІІ - 07.00.00.000 ПЗ	Арк.
						55
Змн.	Арк.	№ докум.	Підпис	Дата		

```

def create
  user = User.find_by(email: params[:email])

  if user&.authenticate(params[:password])
    session[:user_id] = user.id
  else
    render :new
  end
end
end

```

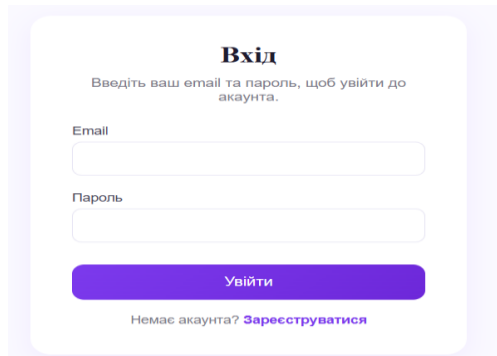


Рисунок 4.15 – Форма входу користувача

Для визначення поточного користувача та обмеження доступу до окремих дій використовуються методи `current_user` і `require_login`. Приклад реалізації показано нижче, а результат роботи наведено на рисунку 4.16.

```

def current_user
  @current_user ||= User.find_by(id: session[:user_id])
end

def require_login
  redirect_to login_path unless current_user
end

```

Рисунок 4.16 - Обмеження доступу для неавторизованого користувача

Реалізація автентифікації забезпечує контроль доступу до функціоналу веб-застосунку [12].

4.2 Реалізація адміністративної панелі

Адміністративна панель використовується для керування товарами, замовленнями, користувачами та іншими даними веб-застосунку [4]. Доступ до адміністративної частини надається лише авторизованим користувачам із відповідними правами доступу [12].

4.2.1 Структура адміністративної частини застосунку

Адміністративна частина реалізована як окремий модуль із власними контролерами та представленнями [7].

Контролери адміністративної панелі організовані у просторі імен `admin`. Їх структуру показано на рисунку 4.17.

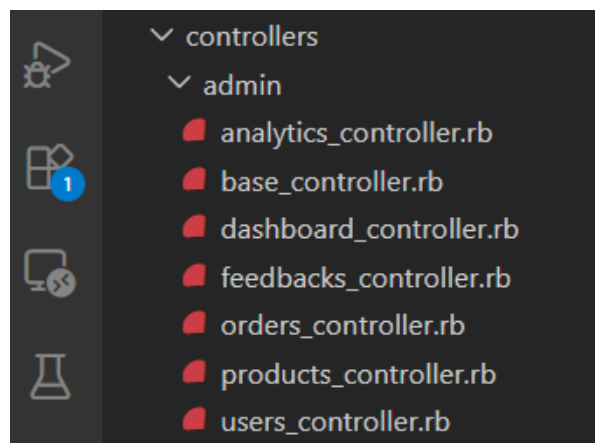


Рисунок 4.17 – Структура контролерів адміністративної частини

Представлення адміністративної панелі використовують окремі шаблони та `layout` для формування інтерфейсу. Їх структуру наведено на рисунку 4.18.

					БР.ІП - 07.00.00.000 ПЗ	Арк.
						57
Змн.	Арк.	№ докум.	Підпис	Дата		

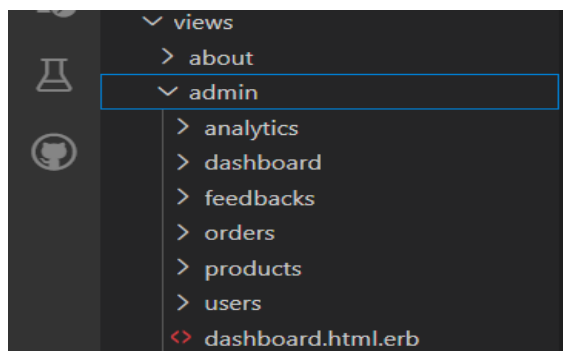


Рисунок 4.18 – Структура представлень адміністративної частини

Така організація адміністративної частини спрощує підтримку та подальший розвиток веб-застосунку [4].

4.2.2 Реалізація доступу до адміністративної панелі

Доступ до адміністративної панелі обмежений для звичайних користувачів і надається лише адміністраторам [12]. Контроль доступу реалізовано через метод `require_admin`, який перевіряє роль користувача. Фрагмент реалізації наведено нижче, а приклад обмеження доступу показано на рисунку 4.19.

```
def require_admin
  redirect_to root_path unless current_user.admin?
end
```

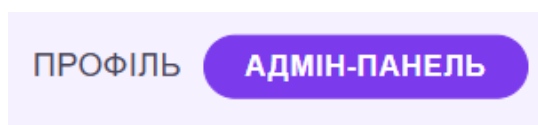


Рисунок 4.19 – Відображення доступу до адміністративної панелі

Перевірка доступу винесена в базовий контролер `Admin::BaseController`, від якого наслідуються адміністративні контролери [7]. Фрагмент реалізації наведено нижче, а інтерфейс адміністративної панелі показано на рисунку 4.20.

```
class Admin::BaseController < ApplicationController
  layout "admin"
```

```

before_action :require_admin
end

```

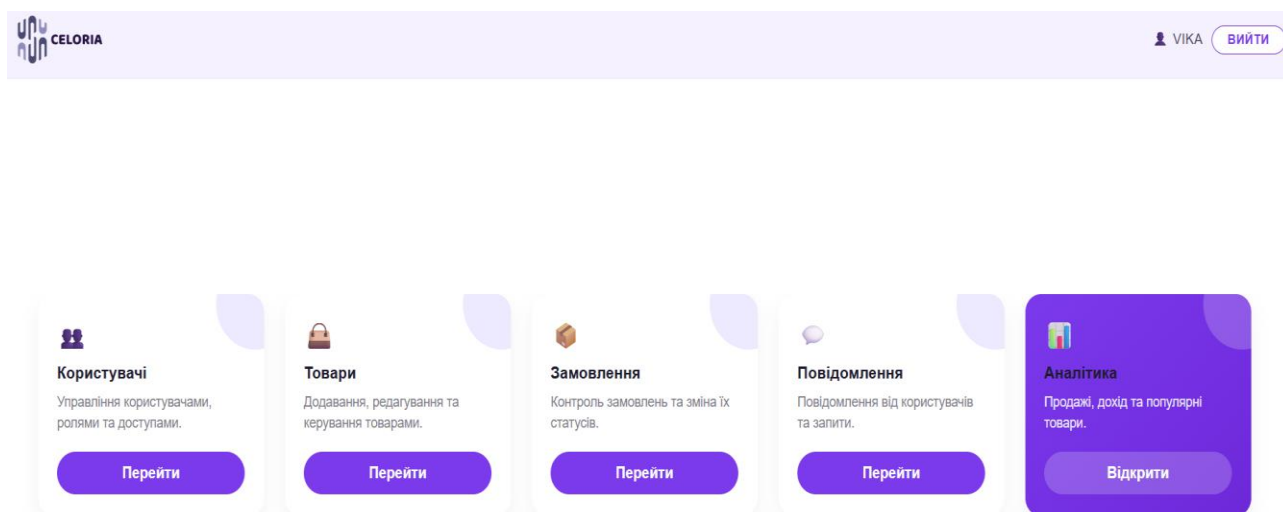


Рисунок 4.20 – Інтерфейс адміністративної панелі

Реалізований підхід забезпечує централізований контроль доступу до адміністративної частини веб-застосунку [10].

4.2.3 Управління користувачами

Модуль управління користувачами забезпечує перегляд, пошук і редагування облікових записів [4]. Реалізація виконана через контролер Admin::UsersController.

Отримання списку користувачів і фільтрація виконуються в методі index. Фрагмент реалізації наведено нижче.

```

def index
  @users = User.order(created_at: :desc)
  @users = @users.where(role: params[:role]) if params[:role].present?
End

```

Інтерфейс списку користувачів із пошуком і фільтрацією показано на рисунку 4.21.

[← Назад](#)

Користувачі

Всі ролі ▾

Пошук

Рисунок 4.21 – Список користувачів з пошуком та фільтрацією

Редагування користувача виконується через форму зі зміною основних даних та ролі користувача. Приклад реалізації наведено нижче, а інтерфейс редагування показано на рисунку 4.22.

```
<%= form_with model: [:admin, @user] do |f| %>
  <%= f.text_field :name %>

  <%= f.select :role,
    [["Admin", "admin"], ["User", "user"]] %>

  <%= f.submit "Зберегти" %>
<% end %>
```

[← Назад](#)

Редагування користувача

Ім'я

Мар'яна

Email

marianatest@gmail.com

Роль

User ▾

Зберегти

Рисунок 4.22 – Форма редагування користувача

Реалізований модуль дозволяє адмініструвати облікові записи користувачів і керувати правами доступу в системі [7].

4.2.4 Управління товарами

Модуль управління товарами забезпечує додавання, редагування, перегляд і пошук товарів у системі [4]. Реалізація виконана через контролер Admin::ProductsController.

Отримання списку товарів і фільтрація за категоріями виконуються в методі index. Фрагмент реалізації наведено нижче, а інтерфейс списку товарів показано на рисунку 4.23.

```
def index
  @products = Product.order(created_at: :desc)
  if params[:category_id].present?
    @products = @products.where(category_id: params[:category_id])
  end
end
```



Рисунок 4.23 – Список товарів з пошуком та фільтрацією

Створення товару виконується методом create, який формує об'єкт товару та зберігає його в базі даних.

```
def create
  build_characteristics
  @product = Product.new(product_params)
```

					БР.ІП - 07.00.00.000 ПЗ	Арк.
						61
Змн.	Арк.	№ докум.	Підпис	Дата		

```

@product.characteristics = @characteristics_hash
@product.save
End

```

Додавання товару виконується через форму адміністративної панелі з основними полями та завантаженням зображень. Приклад реалізації наведено нижче, а інтерфейс форми показано на рисунку 4.24.

Рисунок 4.24 – Форма додавання товару

Для роботи з характеристиками товару використовується структура у форматі «ключ–значення». Формування характеристик виконується окремим методом. Фрагмент реалізації наведено нижче, а інтерфейс додавання характеристик показано на рисунку 4.25.

```

def build_characteristics
  keys = params[:product][:characteristics_keys] || []
  values = params[:product][:characteristics_values] || []
  @characteristics_hash = keys.zip(values).to_h

```

					БР.ІІІ - 07.00.00.000 ІІЗ	Арк.
						62
Змн.	Арк.	№ докум.	Підпис	Дата		

end

Характеристики товару

Назва	Значення	-
Назва	Значення	-
Назва	Значення	-
Назва	Значення	-
+ Додати характеристику		
Створити товар		

Рисунок 4.25 – Додавання характеристик товару

Редагування товару виконується методом `update`, який дозволяє змінювати наявні дані товару. Фрагмент реалізації наведено нижче, а інтерфейс редагування показано на рисунку 4.26.

```
def update
  build_characteristics
  @product.characteristics = @characteristics_hash
  if @product.update(product_params)
    redirect_to admin_products_path
  else
    render :edit
  end
end
```

← Повернутись назад

Редагування товару

Основна інформація

Назва товару

Celoria "OliveLine Classic Clutch"

Опис

OliveLine – універсальний клатч у трендовому оливковому відтінку. Підходить для щоденного використання: у місто, на роботу, в кафе. Лаконічний дизайн і базовий колір роблять його must-have аксесуаром.

Ціна (грн)

2450,0

Категорія

Клатчі

Рисунок 4.26 – Форма редагування товару

Реалізований модуль забезпечує керування товарами та підтримку структури каталогу веб-застосунку [7].

4.2.5 Управління замовленнями

Модуль управління замовленнями забезпечує перегляд, фільтрацію та зміну статусів замовлень [4]. Реалізація виконана через контролер Admin::OrdersController.

Отримання списку замовлень і фільтрація за статусом виконуються в методі index. Фрагмент реалізації наведено нижче, а інтерфейс списку замовлень показано на рисунку 4.27.

```
def index
  @orders = Order.order(created_at: :desc)
  if params[:status].present?
    @orders = @orders.where(status: params[:status])
  end
end
```

					БР.ІП - 07.00.00.000 ПЗ	Арк.
						64
Змн.	Арк.	№ докум.	Підпис	Дата		

end

[← Назад](#)

Замовлення

Всі статуси ▾

Рисунок 4.27 – Список замовлень з фільтрацією

Для перегляду детальної інформації про замовлення використовується окрема сторінка адміністративної панелі, де відображаються дані користувача, список товарів і загальна сума замовлення. Інтерфейс сторінки показано на рисунку 4.28.

[← Повернутись назад](#)

Замовлення №112

Ім'я клієнта: Viktoria Klykaylo

Телефон: +380 96 117 25 21

Адреса: м. Івано-Франківськ, вул. Франка, 29А

Сума замовлення: ₴3,250.00

Спосіб оплати:  Картка

Поточний статус: New



Товари у замовленні:

Назва	Кількість	Ціна	Сума
Celoria "Urban Slim Tote"	1 шт.	₴3,250.00	₴3,250.00

Рисунок 4.28 – Деталі замовлення

Зміна статусу замовлення виконується методом `update_status`, який перевіряє допустимість переходу між станами відповідно до бізнес-логіки системи [11]. Фрагмент реалізації наведено нижче, а інтерфейс зміни статусу показано на рисунку 4.29.

```
def update_status
```

```
    new_status = params[:order][:status]
```

```
    return unless @order.can_transition_to?(new_status)
```

					БР.ІП - 07.00.00.000 ПЗ	Арк.
						65
Змн.	Арк.	№ докум.	Підпис	Дата		

```
@order.update(status: new_status)
```

```
End
```

Змінити статус

Cancelled ▾

Оновити

Рисунок 4.29 – Кнопка зміни статусу

Реалізований модуль забезпечує контроль обробки замовлень і керування їхнім станом у системі [7].

4.2.6 Управління повідомленнями

Модуль управління повідомленнями забезпечує перегляд, фільтрацію та обробку звернень користувачів [4]. Реалізація виконана через контролер `Admin::FeedbacksController`.

Отримання списку повідомлень і фільтрація за статусом виконуються в методі `index`. Фрагмент реалізації наведено нижче, а інтерфейс списку повідомлень показано на рисунку 4.30.

```
def index
  @feedbacks = Feedback.order(created_at: :desc)
  if params[:status].present?
    @feedbacks = @feedbacks.where(status: params[:status])
  end
end
```

[← Повернутись назад](#)

Повідомлення користувачів

Всі

Очікують відповіді

Відповідь надіслано

Рисунок 4.30 – Список повідомлень із фільтрацією

					БР.ІІІ - 07.00.00.000 ПЗ	Арк.
						66
Змн.	Арк.	№ докум.	Підпис	Дата		

Для перегляду звернення використовується окрема сторінка адміністративної панелі, де відображаються контактні дані користувача та текст повідомлення. Інтерфейс перегляду показано на рисунку 4.31.

Рисунок 4.31 – Перегляд повідомлення користувача

Обробка повідомлення виконується через форму відповіді зі зміною статусу звернення. Фрагмент реалізації наведено нижче, а інтерфейс надсилання відповіді показано на рисунку 4.32.

```
<%= form_with model: [:admin, @feedback], method: :patch do |f| %>
  <%= f.text_area :response %>
  <%= f.select :status,
    ["new", "in_progress", "resolved"] %>
  <%= f.submit "Надіслати" %>
<% end %>
```

Рисунок 4.32 – Надсилання відповіді на повідомлення

					БР.ІІІ - 07.00.00.000 ПЗ	Арк.
						67
Змн.	Арк.	№ докум.	Підпис	Дата		

Реалізований модуль забезпечує обробку звернень користувачів і керування їхнім статусом у системі [7].

4.3 Реалізація бізнес-логіки системи

Бізнес-логіка визначає, як саме обробляються дані і виконуються операції в системі. У застосунку вона реалізована через контролери та моделі і включає обробку запитів, перевірку даних, управління станами об'єктів, роботу з кошиком, створення замовлень та авторизацію [4].

4.3.1 Реалізація логіки роботи кошика

Логіка роботи кошика визначає збереження й обробку вибраних товарів у системі. Дані кошика зберігаються через сесії, що дозволяє підтримувати стан кошика для авторизованих і неавторизованих користувачів [7].

Після авторизації користувача виконується об'єднання гостьового кошика з персональним кошиком користувача. Фрагмент реалізації наведено нижче.

```
def merge_guest_cart(user_id)
  return unless session[:cart]&.[]("guest")
  user_key = user_id.to_s
  session[:cart][user_key] ||= {}
  session[:cart]["guest"].each do |product_id, quantity|
    current = session[:cart][user_key][product_id] || 0
    session[:cart][user_key][product_id] =
      current + quantity
  end
  session[:cart].delete("guest")
end
```

Метод merge_guest_cart переносить товари з гостьового кошика до кошика користувача після входу в систему.

					БР.ІІІ - 07.00.00.000 ІІЗ	Арк.
						68
Змн.	Арк.	№ докум.	Підпис	Дата		

Схему об'єднання кошиків показано на рисунку 4.33.

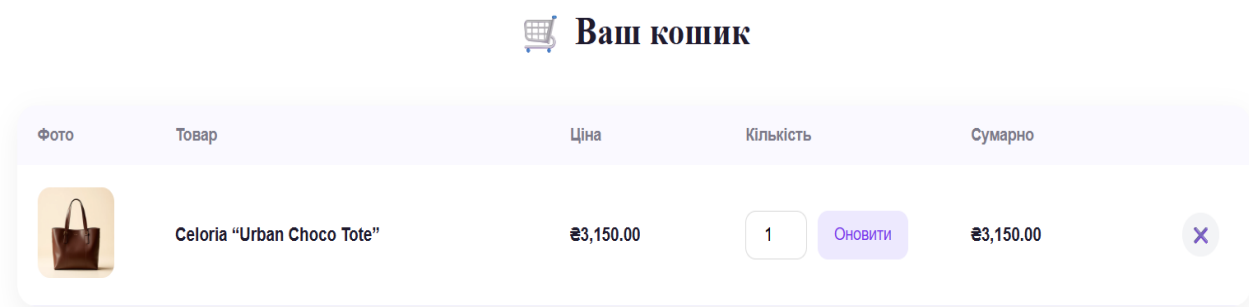


Рисунок 4.33 – Об'єднання гостьового кошика з кошиком користувача

Реалізована логіка забезпечує збереження товарів між різними станами користувача та підтримує коректну роботу кошика [11].

4.3.2 Реалізація процесу оформлення замовлення

Логіка оформлення замовлення визначає створення запису замовлення та обробку пов'язаних даних у системі [11]. Перед виконанням операції перевіряється наявність товарів у кошику.

Якщо користувач не авторизований, система перевіряє існування акаунта за вказаною електронною адресою або створює новий із подальшою авторизацією.

Формування замовлення включає підрахунок загальної вартості, створення записів для товарів та очищення кошика після успішного оформлення.

Реалізація процесу оформлення замовлення наведена у підрозділі 4.1.6 [10].

4.3.3 Управління статусами замовлення

Логіка управління статусами визначає допустимі переходи між станами замовлення та забезпечує коректність їх зміни [11].

Зміна статусів виконується з урахуванням бізнес-правил, що запобігають некоректним операціям і перевіряють допустимість переходів між станами.

Перевірка переходів реалізується на рівні контролера та моделі замовлення [7]. Реалізацію механізму зміни статусу наведено у підрозділі 4.2.5.

4.3.4 Авторизація та контроль доступу

Логіка авторизації визначає доступ до функціоналу системи залежно від ролі користувача [10]. Фрагмент реалізації наведено нижче.

```
def current_user
  @current_user ||= User.find_by(id: session[:user_id])
end

def admin?
  current_user&.admin?
end
```

Доступ до адміністративної частини системи надається лише користувачам із відповідними правами [12].

4.4 Реалізація аналітичних модулів

Аналітичний модуль забезпечує ефективну обробку даних системи та формування ключових показників. У застосунку аналітика впроваджена в адміністративній частині, що дозволяє отримувати такі показники, як дохід, кількість замовлень, середній чек та активність користувачів. Обробка даних здійснюється шляхом агрегування та групування записів у базі даних [4].

4.4.1 Обчислення аналітичних показників

Формування показників виконується в контролері Admin::AnalyticsController, де спочатку здійснюється фільтрація замовлень за обраним періодом (рисунок 4.34).

```
orders = Order.where(created_at:
start_date.beginning_of_day..end_date.end_of_day)

paid_orders = orders.where(status: ["paid", "shipped", "delivered"])
```

					БР.ІІІ - 07.00.00.000 ПЗ	Арк.
						70
Змн.	Арк.	№ докум.	Підпис	Дата		

← Назад

Аналітика

27 . 04 . 2026

27 . 04 . 2026

Фільтрувати

Рисунок 4.34 – Фільтрація замовлень за періодом

Основні показники розраховуються шляхом агрегування даних: визначається дохід, кількість замовлень та середній чек (рисунок 4.35).

```
@total_revenue = paid_orders.sum(:total)
```

```
@orders_count = orders.count
```

```
@average_order = paid_orders.average(:total)&.round(2) || 0
```

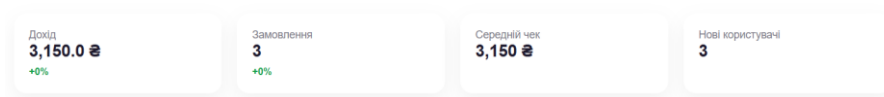


Рисунок 4.35 – Розрахунок аналітичних показників

Додатково визначається кількість нових користувачів та формуються списки популярних товарів і клієнтів [14].

4.4.2 Аналіз динаміки та показників зростання

Для аналізу динаміки виконується групування замовлень за датами, що використовується для побудови графіків [15]. Фрагмент наведено нижче.

```
@orders_by_day = paid_orders
```

```
.group("DATE(created_at)")
```

```
.order("DATE(created_at)")
```

```
.sum(:total)
```

```
@orders_count_by_day = orders
```

```
.group("DATE(created_at)")
```

```
.order("DATE(created_at)")
```

					БР.ІІІ - 07.00.00.000 ПЗ	Арк.
						71
Змн.	Арк.	№ докум.	Підпис	Дата		

.count

Реалізацію групування та формування графіків наведено на рисунку 4.36.



Рисунок 4.36 – Формування даних для графіків

Додатково здійснюється порівняння поточних показників із попереднім періодом, що дозволяє визначити тенденції зміни основних метрик [1].

4.4.3 Візуалізація аналітичних даних

Аналітичні дані представлені в адміністративному інтерфейсі у вигляді графіків і діаграм за допомогою бібліотеки Chart.js.

У системі реалізовано лінійний графік доходу, стовпчикову діаграму кількості замовлень і кругову діаграму розподілу замовлень за статусами.

Таким чином, аналітичний модуль забезпечує візуалізацію ключових показників системи [15].

4.5 Висновки по розділу:

Реалізація веб-застосунку охоплює користувацьку частину, адміністративну панель, бізнес-логіку та аналітичні модулі системи. Реалізований функціонал забезпечує роботу з товарами, кошиком, замовленнями, користувачами та повідомленнями.

Використання Ruby on Rails і шаблону MVC спрощує підтримку системи та подальший розвиток функціоналу веб-застосунку.

					БР.ІІІ - 07.00.00.000 ПЗ	Арк.
						72
Змн.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 5. ТЕСТУВАННЯ ВЕБ-ЗАСТОСУНКУ

Тестування є важливим етапом розробки веб-застосунку, оскільки дозволяє перевірити коректність роботи функціоналу, стабільність системи та відповідність реалізованих можливостей встановленим вимогам [4]. Під час тестування перевіряється робота користувацької та адміністративної частин застосунку, а також коректність виконання основних операцій.

Для перевірки веб-застосунку використовуються автоматизовані та функціональні методи тестування, які допомагають виявити можливі помилки та оцінити стабільність роботи системи [16].

5.1 Тестування користувацької частини веб-застосунку

Тестування веб-застосунку дає можливість оцінити правильність роботи основного функціоналу та взаємодії користувача із системою. Основна увага приділяється користувацькому інтерфейсу, зокрема навігації, роботі з каталогом товарів, перегляду сторінок і взаємодії з кошиком [6]. Також перевіряється правильність виконання основних операцій користувача та стабільність роботи інтерфейсу [4].

5.1.1 Автоматизоване тестування інтерфейсу

Для перевірки користувацького інтерфейсу використовується автоматизоване E2E тестування за допомогою інструменту Puppeteer. Воно запускається в середовищі розгорнутого застосунку, що дозволяє перевірити роботу системи в умовах, максимально наближених до реального використання. Такий підхід дає змогу зрозуміти, чи інтерфейс працює коректно і чи відповідає очікуванням користувачів. Автоматизовані тести імітують дії користувача в браузері та охоплюють основні сценарії роботи із системою.

Зокрема, перевіряється:

— відкриття головної сторінки;

					БР.ІІІ - 07.00.00.000 ПЗ	Арк.
						73
Змн.	Арк.	№ докум.	Підпис	Дата		

- перехід між сторінками застосунку;
- перегляд каталогу товарів;
- перехід на сторінку товару;
- наявність товарів у каталозі;
- робота з кошиком.

Тестові сценарії подані як набір перевірок, згрупованих за окремими функціональними частинами.

Фрагмент реалізації автоматизованого тестування наведено нижче.

```
describe('Робота з товарами', () => {
  test('Каталог містить товари', async () => {
    await page.goto(`${BASE_URL}/products`, {
      waitUntil: 'networkidle2'
    });
    const links = await page.$$('a[href*="/products/"]');
    expect(links.length).toBeGreaterThan(0);
  });
  test('Відкриття сторінки товару', async () => {
    await page.goto(`${BASE_URL}/products`);
    const links = await page.$$('a[href*="/products/"]');
    if (links.length > 0) {
      await links[0].click();
      await page.waitForNavigation();
      expect(page.url()).toMatch(/products\/\d+/);
    } else {
      throw new Error('Немає товарів для тесту');
    }
  });
});
```

					БР.ІІІ - 07.00.00.000 ПЗ	Арк.
						74
Змн.	Арк.	№ докум.	Підпис	Дата		

Під час тестування автоматично відкриваються сторінки, виконуються дії користувача і перевіряються результати, що дозволяє оцінити коректність інтерфейсу та роботи системи [16].

5.1.2 Перевірка роботи каталогу та кошика

Під час тестування користувацької частини веб-застосунку також перевірялась робота каталогу товарів і кошика [6]. Основна увага приділялась правильному відображенню товарів, роботі фільтрації, сортування та зміні кількості товарів у кошику.

Додатково перевірялась правильність обчислення загальної суми замовлення та збереження стану кошика між переходами між сторінками [11]. У процесі тестування критичних помилок у роботі цих компонентів виявлено не було.

5.2 Тестування адміністративної частини

Тестування адміністративної частини проводиться для перевірки правильності роботи функцій управління даними та контролю доступу до системи [12]. Основна увага приділяється роботі з товарами, користувачами, замовленнями та повідомленнями, а також дотриманню бізнес-логіки під час виконання адміністративних операцій.

5.2.1 Функціональне тестування адміністративної частини

Для оцінки адміністративної частини веб-застосунку було проведено функціональне тестування, яке включає перевірку основних функцій системи без застосування автоматизованих інструментів [4].

У рамках тестування здійснено перевірку:

- авторизації адміністратора та доступу до адміністративної панелі;
- управління товарами (створення, редагування, видалення);

					БР.ІІІ - 07.00.00.000 ПЗ	Арк.
						75
Змн.	Арк.	№ докум.	Підпис	Дата		

- управління користувачами (перегляд та зміна ролей);
- обробки замовлень та зміни їх статусів;
- роботи з повідомленнями користувачів;
- дотримання обмежень бізнес-логіки під час зміни статусів замовлення.

Результати тестування представлені у таблиці 5.1.

Таблиця 5.1

Результати функціонального тестування адміністративної панелі

№	Модуль	Сценарій	Дія	Очікуваний результат	Фактичний результат
1	Авторизація	Вхід в адмін-панель	Введення коректних даних	Успішний вхід в систему	Вхід виконано
2	Товари	Додавання товару	Заповнення форми та збереження	Створення нового товару	Виконано
3	Товари	Редагування товару	Зміна характеристик	Оновлення даних	Виконано
4	Товари	Видалення товару	Підтвердження видалення	Товар видалено	Виконано
5	Користувачі	Перегляд списку користувачів	Відкриття сторінки зі списком	Список відображається	Виконано
6	Користувачі	Редагування ролі користувача	Зміна ролі користувача	Вибір нової ролі та збереження	Виконано
7	Замовлення	Зміна статусу замовлення	Вибір допустимого статусу	Статус змінено	Виконано
8	Замовлення	Некоректна зміна статусу	Спроба обрати заборонений статус	Відображення помилки	Виконано
9	Повідомлення	Перегляд списку повідомлень користувачів	Відкриття звернення	Відображення даних	Виконано
10	Повідомлення	Відповідь на повідомлення користувача	Надсилання відповіді	Відповідь надіслано	Виконано

Проведене функціональне тестування показало правильну роботу основних модулів адміністративної панелі та підтвердило коректність виконання основних

операцій. Отримані результати свідчать про стабільну роботу функцій управління даними та взаємодії між окремими компонентами системи.

5.2.2 Перевірка бізнес-логіки та обмежень доступу

Під час тестування окремо перевірялась правильність реалізації бізнес-логіки системи та механізмів контролю доступу [12]. Особлива увага приділялась зміні статусів замовлень, перевірці допустимих переходів між станами та обмеженню доступу до адміністративної частини застосунку [11].

Результати перевірки показали, що система правильно обробляє недопустимі дії користувачів і дотримується встановлених правил роботи бізнес-логіки.

5.3 Результати тестування та оцінка роботи системи

Після виконання тестових сценаріїв було проведено аналіз отриманих результатів та оцінку стабільності роботи веб-застосунку [4]. Основна увага приділялась правильному виконанню операцій, роботі інтерфейсу та взаємодії між окремими компонентами системи.

5.3.1 Аналіз результатів тестування

Під час тестування було встановлено, що веб-застосунок загалом правильно реалізує основні сценарії взаємодії користувача із системою. Автоматизовані перевірки показали, що інтерфейс працює стабільно, навігація не викликає проблем, а основні операції виконуються як очікується [16]. Також було перевірено правильність переходів між сторінками та відображення даних у різних частинах застосунку.

Критичних помилок під час виконання тестових сценаріїв виявлено не було. Результати тестування наведені на рисунку 5.1.

					БР.ІІІ - 07.00.00.000 ПЗ	Арк.
						77
Змн.	Арк.	№ докум.	Підпис	Дата		

```

PS C:\Users\Honor\Desktop\tests> npx jest
console.log
Кнопки не знайдені, тест пропущено

    at Object.log (e2e.test.js:108:17)

PASS ./e2e.test.js (21.258 s)
  E2E тестування користувацького інтерфейсу Celoria Shop
    Завантаження сторінок
      ✓ Головна сторінка відкривається (3049 ms)
      ✓ Каталог відкривається (2337 ms)
      ✓ Кошик відкривається (609 ms)
    Навігація користувача
      ✓ Перехід з головної у каталог (3424 ms)
    Робота з товарами
      ✓ Каталог містить товари (2926 ms)
      ✓ Відкриття сторінки товару (3569 ms)
    Робота з кошиком
      ✓ Додавання товару у кошик (2308 ms)
      ✓ Перехід до кошика після дій (363 ms)

Test Suites: 1 passed, 1 total
Tests:      8 passed, 8 total
Snapshots:  0 total
Time:       21.372 s, estimated 24 s
Ran all test suites.

```

Рисунок 5.1 – Результати автоматизованого тестування

Функціональне тестування адміністративної частини також показало коректну роботу механізмів управління даними та дотримання бізнес-логіки [11]. Отримані результати свідчать, що система працює стабільно і реалізований функціонал відповідає встановленим вимогам.

5.4 Висновки по розділу:

Проведене тестування показало, що веб-застосунок правильно реалізує основні функціональні можливості та забезпечує стабільну взаємодію користувача із системою. Автоматизоване та функціональне тестування підтвердили правильну роботу користувацької й адміністративної частин застосунку.

Отримані результати свідчать про стабільну роботу системи та відповідність реалізованого функціоналу встановленим вимогам.

					БР.ІІІ - 07.00.00.000 ПЗ	Арк.
						78
Змн.	Арк.	№ докум.	Підпис	Дата		

ВИСНОВОК

У процесі виконання дипломної роботи були визначені основні принципи функціонування систем електронної комерції та сформульовані вимоги до веб-застосунку інтернет-магазину. Аналіз сучасних рішень дозволив врахувати їхні переваги та обмеження при проектуванні системи.

Обґрунтовано вибір архітектури та технологій розробки, які забезпечують ефективність, структурованість і масштабованість застосунку. Використання архітектури MVC дозволяє розділити обробку даних, бізнес-логіку та інтерфейс користувача, що спрощує супровід і подальший розвиток системи.

Реалізовано веб-застосунок, що забезпечує повний цикл взаємодії користувача із системою — від перегляду товарів до оформлення замовлення. Інтерфейс організовано з акцентом на зручність використання, логічну навігацію та швидкий доступ до основного функціоналу.

Адміністративна частина забезпечує ефективне управління товарами, користувачами, замовленнями та повідомленнями. Реалізовані механізми контролю доступу та перевірки дій користувачів сприяють безпечній роботі системи та запобігають некоректним операціям.

Бізнес-логіка системи гарантує коректну обробку даних, узгодженість роботи компонентів і контроль виконання операцій. Реалізовані перевірки та обмеження дозволяють уникнути помилок під час роботи з даними та забезпечують стабільність функціонування системи.

Аналітичні модулі дозволяють отримувати, обробляти та візуалізувати ключові показники діяльності системи, що спрощує аналіз ефективності її роботи та підтримує прийняття обґрунтованих рішень щодо подальшого розвитку.

Результати тестування підтверджують працездатність, стабільність та відповідність застосунку встановленим вимогам. Отримані результати можуть бути використані для подальшого розвитку та практичного застосування системи

					БР.ІІІ - 07.00.00.000 ІІЗ	Арк.
						79
Змн.	Арк.	№ докум.	Підпис	Дата		

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Глушков В. М. Інформаційні технології та системи в економіці. - Київ : КНЕУ, 2021. - 312 с.
2. Ковальчук О. І. Електронна комерція: стан і перспективи розвитку в Україні // Економічний вісник. - № 3 (2023). - С. 45–53.
3. Марченко І. П. Стан розвитку онлайн-торгівлі на ринку косметичних засобів України // Маркетинг і менеджмент інновацій. - № 2 (2022). - С. 85–94.
4. Sommerville I. Software Engineering. - 11th ed. - Pearson Education, 2020. - 792 p.
5. Fowler M. Patterns of Enterprise Application Architecture. - Addison-Wesley, 2002. - 533 p.
6. Krug S. Don't Make Me Think. - New Riders, 2014. - 216 p.
7. Hartl M. Ruby on Rails Tutorial. - Addison-Wesley, 2022. - 816 p.
8. Ruby on Rails Guides. Getting Started with Rails. - [Електронний ресурс]. - Режим доступу: https://guides.rubyonrails.org/getting_started.html - Дата звернення: 25.03.2026.
9. Ruby on Rails Guides. Action Controller Overview. - [Електронний ресурс]. - Режим доступу: https://guides.rubyonrails.org/action_controller_overview.html - Дата звернення: 30.03.2026.
10. Ruby on Rails Guides. Securing Rails Applications. - [Електронний ресурс]. - Режим доступу: <https://guides.rubyonrails.org/security.html> - Дата звернення: 14.04.2026.
11. Ruby on Rails API Documentation. ActiveRecord Transactions. - [Електронний ресурс]. - Режим доступу: <https://api.rubyonrails.org/classes/ActiveRecord/Transactions/ClassMethods.html> - Дата звернення: 15.04.2026.
12. OWASP Foundation. OWASP Cheat Sheet Series. - [Електронний ресурс]. - Режим доступу: <https://cheatsheetseries.owasp.org/> - Дата звернення: 16.04.2026.

					БР.ІІІ - 07.00.00.000 ПЗ	Арк.
						80
Змн.	Арк.	№ докум.	Підпис	Дата		

13. W3C. Web Design and Applications Standards. - [Електронний ресурс]. - Режим доступу: <https://www.w3.org/standards/webdesign/> - Дата звернення: 20.03.2026.

14. PostgreSQL Global Development Group. PostgreSQL Documentation. - [Електронний ресурс]. - Режим доступу: <https://www.postgresql.org/docs/> - Дата звернення: 24.03.2026.

15. Chart.js Documentation. - [Електронний ресурс]. - Режим доступу: <https://www.chartjs.org/docs/> - Дата звернення: 14.04.2026.

16. Puppeteer Documentation. - [Електронний ресурс]. - Режим доступу: <https://pptr.dev/> - Дата звернення: 14.04.2026.

17. DigitalOcean. Ruby on Rails Tutorials and Deployment Guides. - [Електронний ресурс]. - Режим доступу: <https://www.digitalocean.com/community/tags/ruby-on-rails> - Дата звернення: 30.03.2026.

18. Render. Deployment Platform Documentation. - [Електронний ресурс]. - Режим доступу: <https://render.com/docs> - Дата звернення: 23.04.2026.

19. Gamma E., Helm R., Johnson R., Vlissides J. Design Patterns: Elements of Reusable Object-Oriented Software. - Addison-Wesley, 1994. - 395 p.

20. Pressman R. S., Maxim B. R. Software Engineering: A Practitioner's Approach. - 9th ed. - McGraw-Hill Education, 2019. - 880 p.

21. Martin R. C. Clean Code: A Handbook of Agile Software Craftsmanship. - Prentice Hall, 2008. - 464 p.

22. Flanagan D. JavaScript: The Definitive Guide. - 7th ed. - O'Reilly Media, 2020. - 704 p.

23. Duckett J. HTML and CSS: Design and Build Websites. - Wiley, 2011. - 490 p.

24. Mozilla Developer Network. HTML: HyperText Markup Language. - [Електронний ресурс]. - Режим доступу: <https://developer.mozilla.org/en-US/docs/Web/HTML> - Дата звернення: 12.04.2026.

					БР.ІІІ - 07.00.00.000 ІІЗ	Арк.
						81
Змн.	Арк.	№ докум.	Підпис	Дата		

25. Mozilla Developer Network. CSS: Cascading Style Sheets. - [Електронний ресурс]. - Режим доступу: <https://developer.mozilla.org/en-US/docs/Web/CSS> - Дата звернення: 12.04.2026.

26. Mozilla Developer Network. JavaScript Guide. - [Електронний ресурс]. - Режим доступу: <https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide> - Дата звернення: 12.04.2026.

27. Nielsen J. Usability Engineering. - Morgan Kaufmann, 1994. - 362 p.

28. Google Developers. Web Performance Optimization. - [Електронний ресурс]. - Режим доступу: <https://developers.google.com/web/fundamentals/performance> - Дата звернення: 18.04.2026.

29. OWASP Foundation. Web Security Testing Guide. - [Електронний ресурс]. - Режим доступу: <https://owasp.org/www-project-web-security-testing-guide/> - Дата звернення: 18.04.2026.

30. GitHub Docs. GitHub Documentation. - [Електронний ресурс]. - Режим доступу: <https://docs.github.com/> - Дата звернення: 20.04.2026.

31. Ruby Documentation. Ruby Programming Language Documentation. - [Електронний ресурс]. - Режим доступу: <https://www.ruby-lang.org/en/documentation/> - Дата звернення: 22.04.2026.

32. Ruby on Rails API Documentation. Active Record Basics. - [Електронний ресурс]. - Режим доступу: <https://api.rubyonrails.org/classes/ActiveRecord/Base.html> - Дата звернення: 21.04.2026.

33. RubyGems Guides. RubyGems Documentation. - [Електронний ресурс]. - Режим доступу: <https://guides.rubygems.org/> - Дата звернення: 21.04.2026.

34. Selenium Documentation. - [Електронний ресурс]. - Режим доступу: <https://www.selenium.dev/documentation/> - Дата звернення: 22.04.2026.

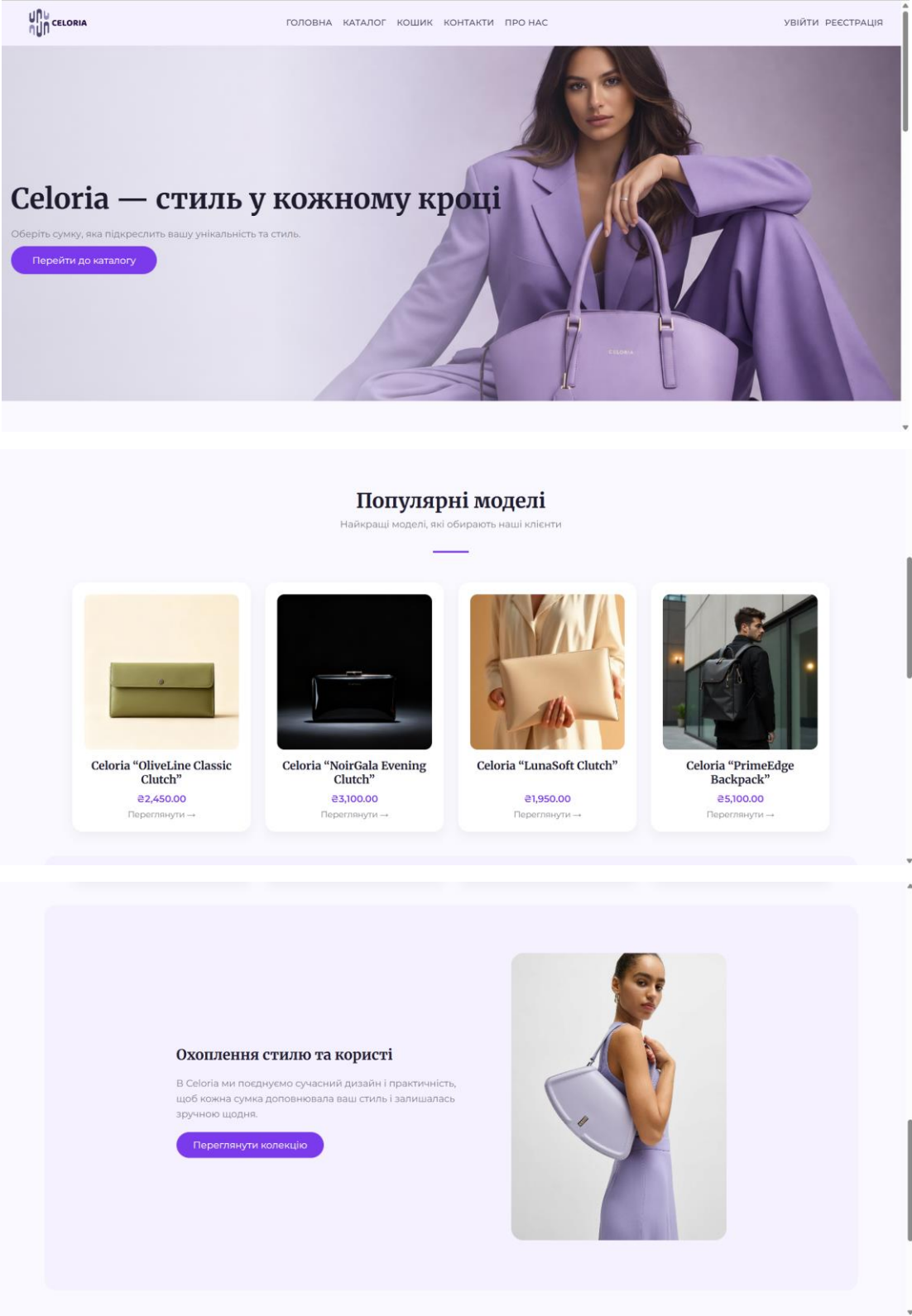
35. Klykaylo V. Репозиторій проєкту «Інтернет-магазин Celoria». - [Електронний ресурс]. - Режим доступу: https://github.com/klykaylova/Celoria_shop - Дата звернення: 23.04.2

					БР.ІІІ - 07.00.00.000 ІІЗ	Арк.
						82
Змн.	Арк.	№ докум.	Підпис	Дата		

ДОДАТКИ

ДОДАТОК А

Інтерфейс веб-застосунку:



Додаток А1. Головна сторінка.

Ваш кошик

Кошик порожній 🛒

[Перейти до каталогу](#)

Додаток А4. Сторінка «Кошик».

 CELORIA

ГОЛОВНАКАТАЛОГКОШИККОНТАКТИПРО НАС

УВІЙТИРЕЄСТРАЦІЯ

 **Контакти**

Ми будемо раді почути ваші відгуки, запитання або пропозиції.

Ваше ім'я

Email

Повідомлення

[Надіслати](#)

Celoria
Стиль у кожному кроці



Інформація
Головна
Про нас
Контакти

Каталог
Сумки
Клатчі
Рюкзак

Контакти
+380 XX XXX XX XX
Івано-Франківськ

© 2026 Celoria. Всі права захищені.

Додаток А5. Сторінка «Контакти».

Про нас

Celoria — це простір сучасного стилю, де кожна сумка створена з думкою про комфорт, витонченість та особистість кожної жінки.

Ми працюємо з ідеєю, що аксесуари не просто доповнюють образ — вони підкреслюють характер. Саме тому ми створюємо моделі, які поєднують стиль, естетику та практичність.



Наша історія

Бренд Celoria народився з бажання створити щось більше, ніж просто аксесуари. Ми почали з невеликої колекції шоперів ручної роботи, але з часом зрозуміли, що наші клієнти прагнуть більшого — дизайну, уваги до деталей, високої якості та відчуття унікальності.

Сьогодні Celoria — це сучасний український бренд, який поєднує моду, функціональність та естетику. Ми створюємо сумки, що надихають щодня.

Наші цінності



Стиль

Мінімалізм, трендовість, впізнаваний дизайн Celoria.



Практичність

Наші моделі створені для щоденного використання — легкі та зручні.



Якість

Ми використовуємо тільки перевірені матеріали, що служать роками.



Турбота

Ми дбаємо про сервіс, упаковку та позитивний досвід кожного клієнта.

Чому клієнти обирають нас?



Універсальний стиль

Підходить до будь-якого образу — від офісного до кешуал.



Ретельне виготовлення

Кожна модель проходить ручну перевірку якості.



Швидка доставка

Надсилаємо замовлення Новою Поштою та Укрпоштою.



Підтримка 24/7

Ми на зв'язку у будь-який час — допомагаємо та консультуємо.

Celoria

Стиль у кожному кроці



Інформація

Головна
Про нас
Контакти

Каталог

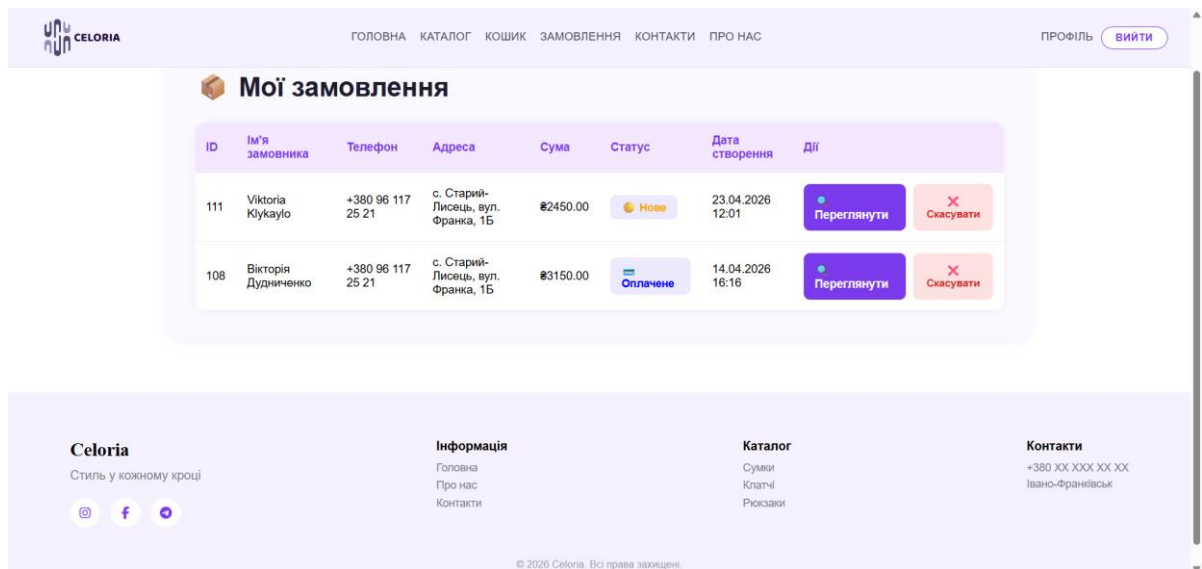
Сумки
Клатчі
Рюкзаки

Контакти

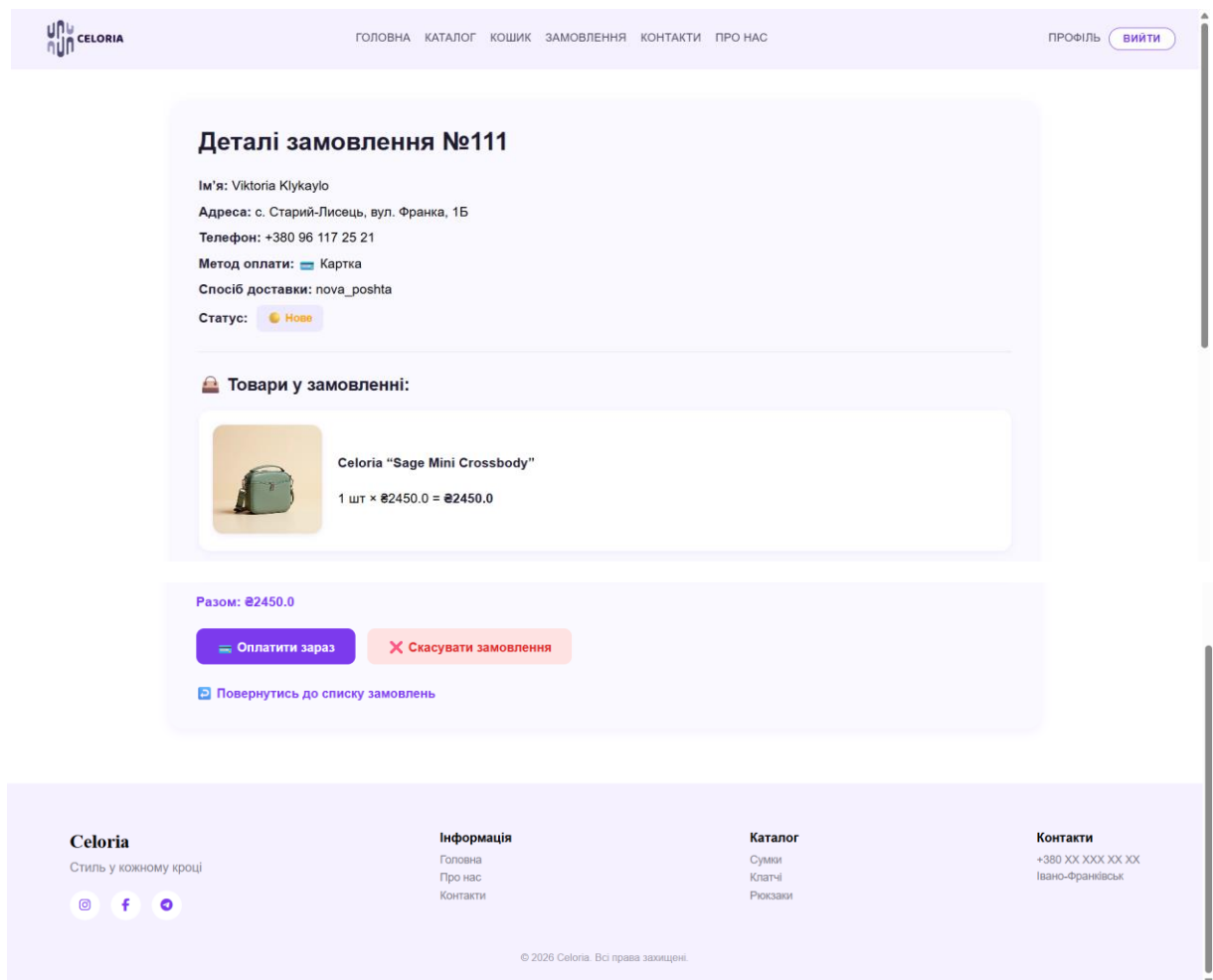
+380 XX XXX XX XX
Івано-Франківськ

Разом: ₹2450.0

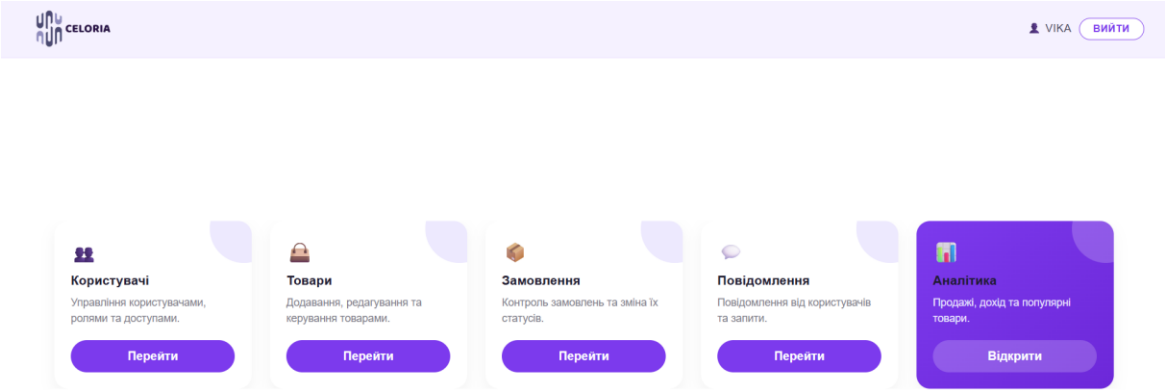
Додаток А9. Сторінка оформлення замовлення.



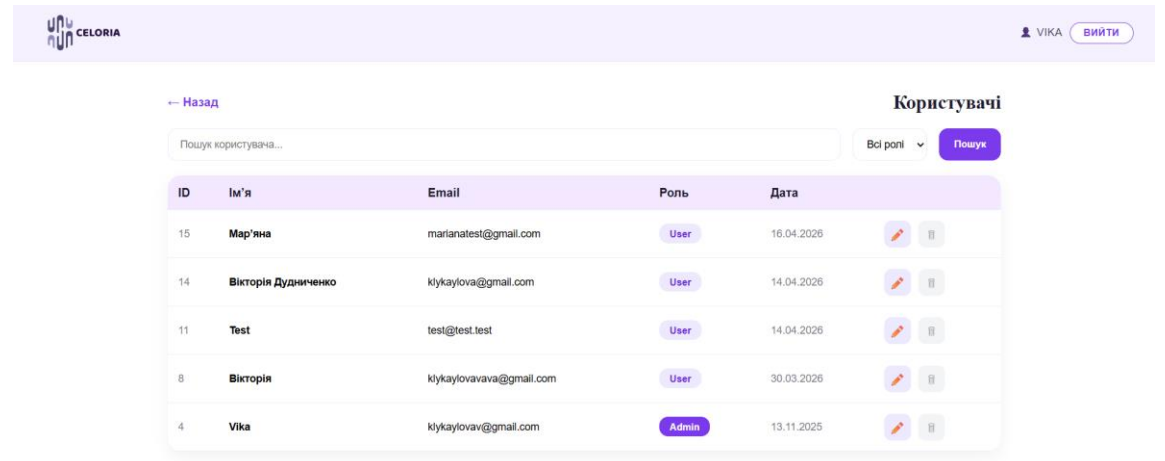
Додаток A10. Сторінка «Мої замовлення».



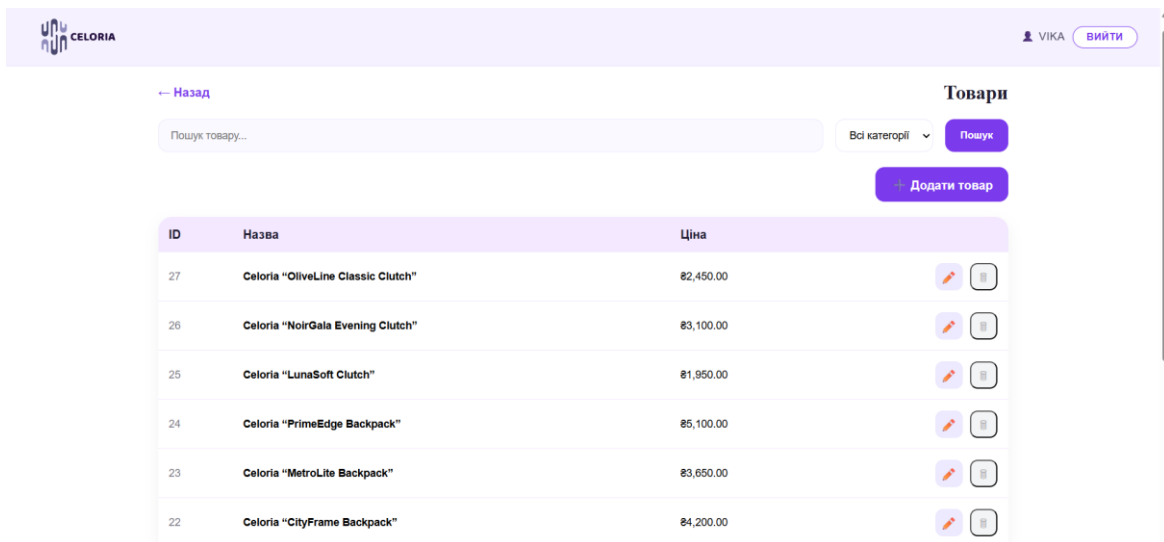
Додаток A11. Сторінка перегляду замовлення.



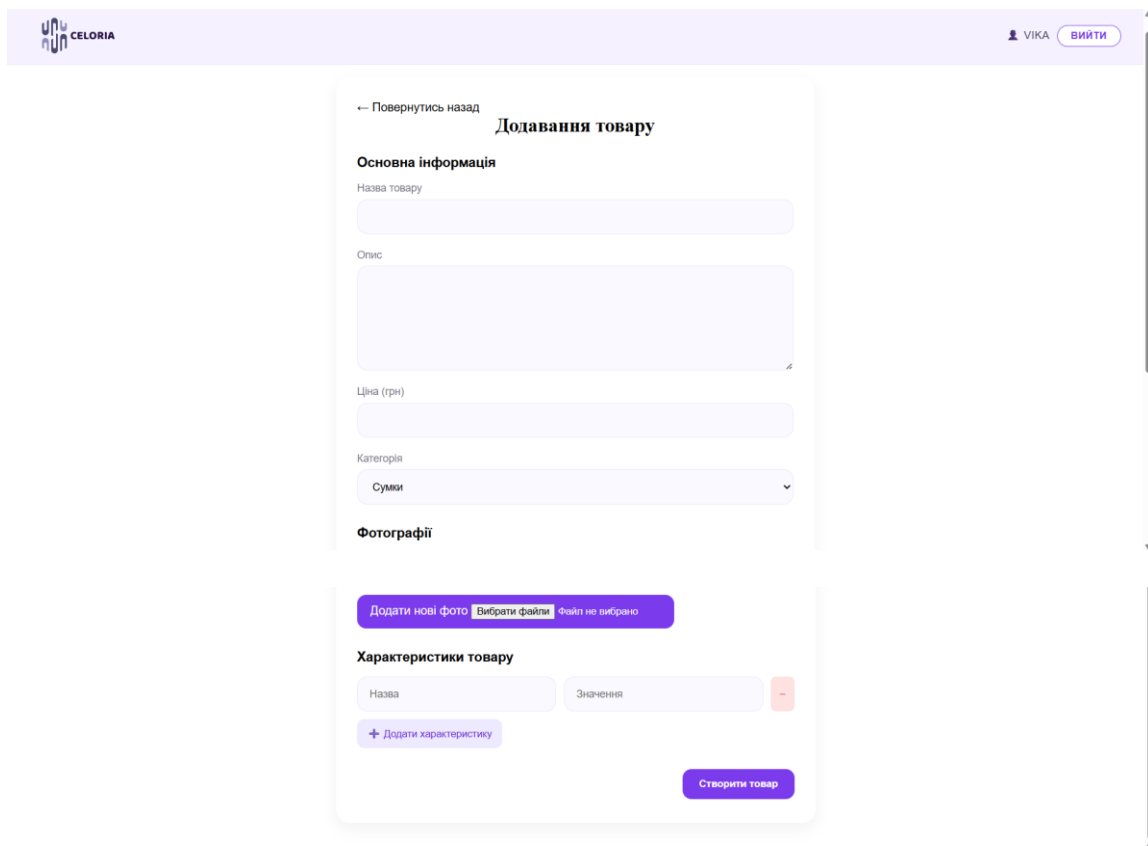
Додаток А12. Сторінка адміністративної панелі.



Додаток А13. Сторінка «Користувачі».



Додаток А14. Сторінка «Товари».



Додаток А15. Сторінка «Додавання товару».

[← Повернутись назад](#)

Редагування товару

Основна інформація

Назва товару

Celoria "OliveLine Classic Clutch"

Опис

OliveLine – універсальний клатч у трендовому оливковому відтінку. Підходить для щоденного використання: у місто, на роботу, в кафе. Лаконічний дизайн і базовий колір роблять його must-have аксесуаром.

Ціна (грн)

2450,0

Категорія

Клатчі

Фотографії

[Додати нові фото](#) [Вибрати файли](#) Файл не вибрано

Характеристики товару

Матеріал	матова екошкіра Smooth Green	–
Колір	Soft Olive (м'який оливковий)	–
Форма	прямокутна	–

Застібка	магнітна	–
Розмір	28 x 18 см	–

[+ Додати характеристику](#)[Зберегти зміни](#)

Додаток A16. Сторінка «Редагування товару».

[← Назад](#)

Замовлення

Всі статуси

ID	Користувач	Сума	Статус	Дата		
111	Вікторія Дудиченко klykaylova@gmail.com	₴2,450.00	New	23.04.2026		
108	Вікторія Дудиченко klykaylova@gmail.com	₴3,150.00	Paid	14.04.2026		

Додаток A17. Сторінка «Замовлення».

[← Повернутись назад](#)

Замовлення №111

Ім'я клієнта: Viktoriia Klykaylo
Телефон: +380 96 117 25 21
Адреса: с. Старий-Лисець, вул. Франка, 15
Сума замовлення: ₴2,450.00
Спосіб оплати: Картка
Поточний статус: New

Товари у замовленні:

Назва	Кількість	Ціна	Сума
Celoria "Sage Mini Crossbody"	1 шт.	₴2,450.00	₴2,450.00

Змінити статус

Cancelled Оновити

Додаток A18. Сторінка перегляду замовлення.

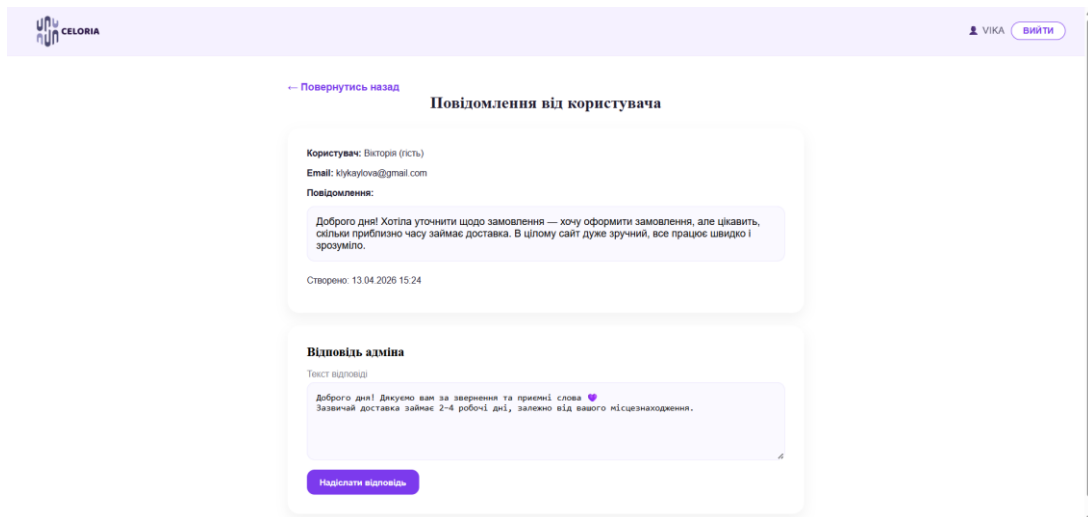
[← Повернутись назад](#)

Повідомлення користувачів

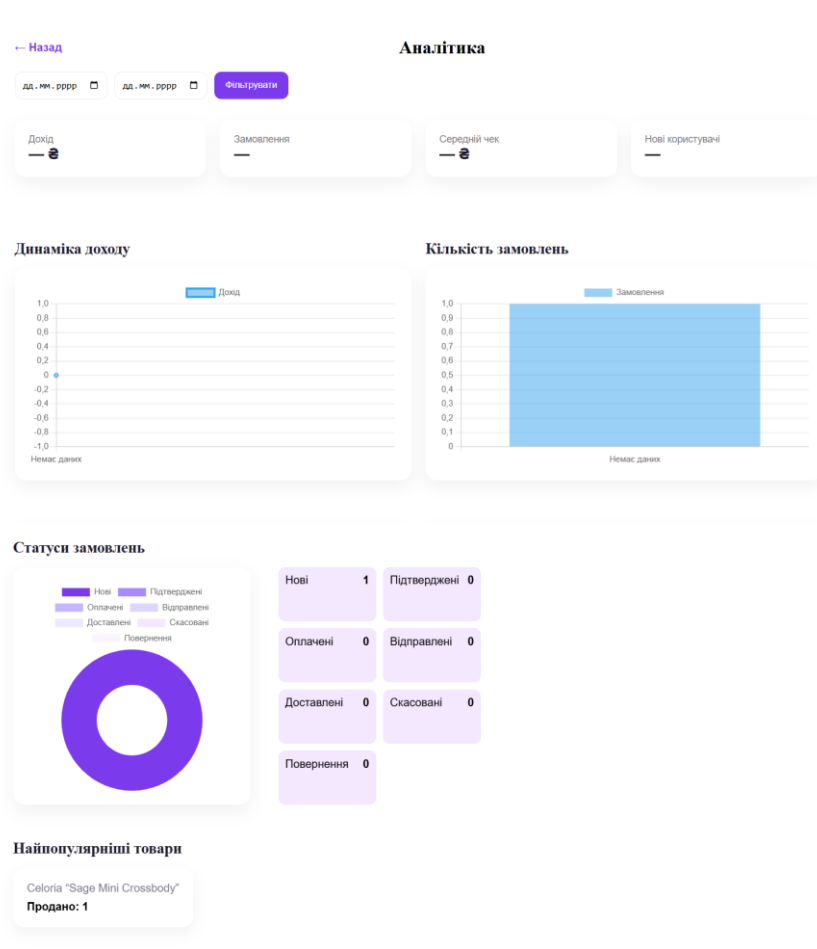
Всі Очікують відповіді Відповідь надіслано

ID	Користувач	Повідомлення	Статус	Дії
26	Вікторія (гість)	Доброго дня! Хотіла уточнити щодо замовлення — ...	Відповідь надіслано	Переглянути Видалити
25	Вікторія klykaylovavava@gmail.com	Доброго дня! Хочу поділитися враженням від поку...	Відповідь надіслано	Переглянути Видалити

Додаток A19. Демонстрація сторінки «Повідомлення».



Додаток А20. Демонстрація сторінки перегляду повідомлення.



А21. Сторінка «Аналітика».

ДОДАТОК Б

Фрагменти коду для здійснення автоматизованого тестування:

1. Ініціалізація:

```
const puppeteer = require('puppeteer');
```

```
jest.setTimeout(60000);
```

```
describe('E2E тестування користувацького інтерфейсу Celoria Shop', () => {
```

```
  let browser;
```

```
  let page;
```

```
  const BASE_URL = 'https://celoria-shop.onrender.com';
```

```
  beforeAll(async () => {
```

```
    browser = await puppeteer.launch({
```

```
      headless: false,
```

```
      slowMo: 50
```

```
    });
```

```
    page = await browser.newPage();
```

```
  });
```

```
  afterAll(async () => {
```

```
    await browser.close();
```

```
  });
```

2. Перевірка завантаження сторінок:

```
describe('Завантаження сторінок', () => {
```

```
test('Головна сторінка відкривається', async () => {  
  await page.goto(`${BASE_URL}/`, {  
    waitUntil: 'networkidle2',  
    timeout: 60000  
  });
```

```
  await page.waitForSelector('body');
```

```
  const content = await page.content();  
  expect(content.length).toBeGreaterThan(1000);  
});
```

```
test('Каталог відкривається', async () => {  
  await page.goto(`${BASE_URL}/products`, {  
    waitUntil: 'networkidle2'  
  });
```

```
  expect(page.url()).toContain('/products');  
});
```

```
test('Кошик відкривається', async () => {  
  await page.goto(`${BASE_URL}/cart`);  
  
  expect(page.url()).toContain('/cart');  
});
```

```
});
```

3. Перевірка навігації користувача:

```
describe('Навігація користувача', () => {

  test('Перехід з головної у каталог', async () => {

    await page.goto(`${BASE_URL}/`);

    await page.waitForSelector('a[href="/products"]', { timeout: 10000 });
    await page.click('a[href="/products"]');

    await page.waitForNavigation();

    expect(page.url()).toContain('/products');
  });

});
```

4. Перевірка роботи з товарами:

```
describe('Робота з товарами', () => {

  test('Каталог містить товари', async () => {

    await page.goto(`${BASE_URL}/products`, {
      waitUntil: 'networkidle2'
    });

    const links = await page.$$('a[href*="/products/"]');
    expect(links.length).toBeGreaterThan(0);
  });

  test('Відкриття сторінки товару', async () => {

    await page.goto(`${BASE_URL}/products`);
```

```

const links = await page.$$('a[href*="/products/"]');

if (links.length > 0) {
  await links[0].click();
  await page.waitForNavigation();

  expect(page.url()).toMatch(/products\/\d+/);
} else {
  throw new Error('Немає товарів для тесту');
}
});

});

```

5. Перевірка роботи кошика:

```

describe('Робота з кошиком', () => {

  test('Додавання товару у кошик', async () => {
    await page.goto(`${BASE_URL}/products`);

    const buttons = await page.$$('button');

    if (buttons.length > 0) {
      await buttons[0].click();

      const content = await page.content();
      expect(content).toBeTruthy();
    } else {

```

```
    console.log('Кнопки не знайдені, тест пропущено');  
  }  
});
```

```
test('Перехід до кошика після дій', async () => {  
  await page.goto(`${BASE_URL}/cart`);
```

```
  expect(page.url()).toContain('/cart');  
});
```

```
});
```

```
});
```

ДОДАТОК В

Посилання на репозиторій GitHub:

https://github.com/klykaylova/Celoria_shop

БІБЛІОГРАФІЧНА ДОВІДКА

Тема бакалаврської роботи: «Реалізація Full Stack застосунку електронної комерції з аналітичними модулями».

Обсяг пояснювальної записки: 79 аркушів

Дата закінчення бакалаврської роботи «28» травня 2026 р.

Підпис студента _____