

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

ПОЯСНЮВАЛЬНА ЗАПИСКА

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

на тему:

«Інтернет-магазин цифрової техніки»

Виконав студент групи ІПЗ-19-2
ГРІШИН Ярослав Олегович

Керівник роботи:
КРАВЧЕНКО Світлана Миколаївна

Рецензент:
МАРЧУК Галина Вікторівна

Житомир – 2023

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

«ЗАТВЕРДЖУЮ»

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

ЗАВДАННЯ
на кваліфікаційну роботу

Здобувач вищої освіти: **Грішин Ярослав Олегович**

Керівник роботи: **Кравченко Світлана Миколаївна**

Тема роботи: **«Інтернет-магазин цифрової техніки»**,

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

Вихідні дані для роботи: **Об'єктом дослідження є інтернет-магазин цифрової техніки, з фокусом на продажу телефонів, ноутбуків та інших електронних пристроїв. Предметом дослідження є використання веб-орієнтованих технологій для інтернет-магазину цифрової техніки.**

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

Розділ	Консультант	Завдання видав	Завдання прийняв
1	Кравченко С.М.	14.02.2023р.	14.02.2023р.
2	Кравченко С.М.	15.04.2023р.	15.04.2023р.
3	Кравченко С.М.	01.05.2023р.	01.05.2023р.

Календарний план

№ з/п	Назва етапів випускної кваліфікаційної роботи	Термін виконання етапів роботи	Примітка
1	Постановка задачі	2 березня 2023 – 12 березня 2023	Виконано
2	Пошук, огляд та аналіз аналогічних розробок	12 березня 2023 – 19 березня 2023	Виконано
3	Формулювання технічного завдання та опрацювання літературних джерел	19 березня 2023 – 28 березня 2023	Виконано
4	Проектування структури системи	28 березня 2023 – 5 квітня 2023	Виконано
5	Написання програмного коду	5 квітня 2023 – 23 травня 2023	Виконано
6	Тестування роботи додатку	23 травня 2023 – 30 травня 2023	Виконано
7	Попередній захист	5 травня 2023	Виконано

АНОТАЦІЯ

Бакалаврська робота складається з інтернет-магазину цифрової техніки та пояснювальної записки. Пояснювальна записка до випускної роботи містить 62 сторінки, 34 ілюстрацій та 12 таблиць.

Метою роботи є розробка сайту інтернет-магазину цифрової техніки з використанням HTML, CSS, NEXT.JS, а також розробка системи адміністрування з використанням NEXT.JS з використанням хмарної бази даних MongoDB Atlas.

У роботі визначено вимоги користувача та обґрунтовано створення концептуальної моделі. Також проаналізовано аналоги програмних продуктів та обрано засоби реалізації, розроблено структуру бази даних, обрано архітектуру та розроблено структуру системи. Також описано програмну реалізацію веб-сайту та наведено результати тестування системи.

КЛЮЧОВІ СЛОВА: ІНТЕРНЕТ-МАГАЗИН, БАЗА ДАНИХ, ВЕБ-САЙТ, ПРОДУКТИ, АДМІНІСТРУВАННЯ, NEXTJS.

					ІПЗ.КР.Б–121-23–ПЗ			
Змн.	Арк.	№ докум.	Підпис	Дата	Інтернет-магазин цифрової техніки Пояснювальна записка	Літ.	Арк.	Аркушів
Розроб.		Я. Грішин					4	
Керівник		С.Кравченко				Житомирська політехніка група ІПЗ-19-2		
Н. Контр.		О. Тичина						
Зав. Каф.		А. Морозов						

ABSTRACT

The bachelor's thesis consists of an online store for digital technology and an explanatory note. The explanatory note for the graduation project contains 62 pages, 34 illustrations, and 12 tables.

The aim of this work is to develop a website for an online store of digital technology using HTML, CSS, NEXT.JS, and also to develop an administration system using NEXT.JS with the cloud database MongoDB Atlas.

The work defines user requirements and justifies the creation of a conceptual model. It also analyzes analogs of software products and selects implementation tools, develops the structure of the database, chooses the architecture, and designs the system structure. Additionally, the software implementation of the website is described, and the results of system testing are provided.

KEYWORDS: ONLINE STORE, DATABASE, WEBSITE, PRODUCTS, ADMINISTRATION, NEXTJS.

ЗМІСТ

ВСТУП.....	7
РОЗДІЛ 1. АНАЛІЗ НАПРЯМКІВ ВИКОРИСТАННЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ДЛЯ ІНТЕРНЕТ-МАГАЗИНУ.....	10
1.1 Постановка задачі.....	10
1.2 Аналіз аналогів програмного продукту.....	12
1.3 Вибір архітектури системи інтернет-магазину цифрової техніки.....	17
1.4 Обґрунтування вибору інструментальних засобів та вимоги до апаратного забезпечення.....	18
Висновки до першого розділу.....	20
РОЗДІЛ 2. ПРОЕКТУВАННЯ ІНТЕРНЕТ-МАГАЗИНУ ЦИФРОВОЇ ТЕХНІКИ....	21
2.1. Визначення варіантів використання та структури системи	21
2.2. Розробка бази даних системи.....	25
2.3. Проектування та реалізація алгоритмів роботи системи.....	31
2.4. Реалізація додатку інтернет-магазину	33
Висновки до другого розділу.....	42
РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З ІНТЕРНЕТ-МАГАЗИНОМ ЦИФРОВОЇ ТЕХНІКИ.....	43
3.1. Порядок встановлення та налаштування системи.....	43
3.2. Структура інтерфейсу та порядок роботи інтернет-магазином цифрової техніки.....	45
3.3. Тестування роботи програмного продукту.....	56
Висновки до третього розділу.....	57
ВИСНОВКИ.....	59
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ.....	61
ДОДАТКИ.....	62

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

CRUD – create, read, update, delete (створити, прочитати, оновити, видалити)

БД – База даних

ПЗ – Програмне забезпечення

MDB – MongoDB

СКБД – система керування базами даних

JS – JavaScript

API – application programming interface

ВСТУП

Актуальність теми. Тема інтернет-магазинів цифрової техніки, таких як телефони, ноутбуки та інші електронні пристрої, є надзвичайно актуальною і в даний час. Розвиток інтернету, зростання популярності онлайн-покупок і широкий доступ до цифрових технологій призвели до значного зростання попиту на цифрові товари через інтернет.

Інтернет-магазини цифрової техніки надають споживачам можливість зручно і безпечно придбати різноманітні цифрові пристрої з будь-якого місця, де є доступ до Інтернету. Вони пропонують широкий асортимент товарів різних брендів і моделей, а також забезпечують зручність вибору і порівняння товарів, огляди і рейтинги, інформацію про технічні характеристики і ціни.

Особливо під час пандемії COVID-19 та війни, коли багато магазинів фізично закриті або мають обмежені години роботи, інтернет-магазини стали ще більш популярними. Багато людей віддають перевагу покупкам онлайн, щоб уникнути особистого контакту і забезпечити безпеку свого здоров'я.

Також важливо зазначити, що цифрові товари, такі як телефони і ноутбуки, є необхідними для багатьох аспектів нашого життя, включаючи комунікацію, роботу, навчання і розваги. Зростання числа людей, які користуються цифровими технологіями, створює постійний попит на цифрові товари і, відповідно, наявність інтернет-магазинів, які їх продають.

Узагалі, інтернет-магазини цифрової техніки є актуальними та популярними серед споживачів, оскільки вони надають зручність, широкий вибір і доступні ціни. Очікується, що ця тенденція буде продовжуватися і розвиватися в майбутньому.

Метою випускної роботи є розробка інтернет-магазину цифрової техніки, з фокусом на продаж телефонів, ноутбуків та інших схожих електронних пристроїв. Наша робота буде спрямована на створення функціонального та привабливого інтернет-магазину, який надасть споживачам можливість зручно придбати цифрову техніку через інтернет.

Основні завдання роботи включають в себе:

1. Вибір засобів реалізації системи: Вибір сучасних методів реалізації для створення інтернет-магазину.
2. Дизайн та веб-розробка: Створення привабливого та зручного дизайну для нашого інтернет-магазину.
3. Інтеграція платіжних систем: Забезпечення безпечної та зручної оплати для клієнтів шляхом інтеграції платіжних систем, таких як PayPal, Stripe або інших популярних методів оплати.
5. Управління товаром: Розробка системи управління товарами, що дозволить нам додавати, видаляти та оновлювати товари в інтернет-магазині.
8. Тестування та запуск: Проведення тестувань, виявлення помилок та усунення їх.

Ця робота буде включати детальний опис всіх кроків, зазначених вище, а також практичну реалізацію цього інтернет-магазину цифрової техніки.

Об'єктом дослідження є інтернет-магазин цифрової техніки, з фокусом на продажу телефонів, ноутбуків та інших електронних пристроїв.

Предметом дослідження є використання веб-орієнтованих технологій для інтернет-магазину цифрової техніки.

У роботі було застосовано методи веб-програмування для розробки функціональності та структури інтернет-магазину. Також було використано методи дизайну користувальницьких інтерфейсів для створення зручного та привабливого вигляду магазину.

Розроблена система інтернет-магазину може бути використана як практичний веб-сайт, який може бути адаптований для будь-якого існуючого інтернет-магазину. Крім того, маємо можливість розширити функціональність системи, додавши власні унікальні функції та особливості.

Пояснювальна записка до випускної роботи містить 62 сторінки, 34 ілюстрацій та 12 таблиць, список літературних джерел містить 14 найменувань.

РОЗДІЛ 1. АНАЛІЗ НАПРЯМКІВ ВИКОРИСТАННЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ДЛЯ ІНТЕРНЕТ-МАГАЗИНУ

1.1 Постановка задачі

Постановка задачі випускної роботи полягає в розробці інтернет-магазину цифрової техніки, з фокусом на продажу телефонів, ноутбуків та інших електронних пристроїв. Основні задачі, які ми поставили собі в рамках цієї роботи, включають:

1. Проектування системи:

- Визначити потреби і функціональні вимоги до інтернет-магазину.
- Створити структуру інформаційної системи, включаючи сторінки товарів, кошик, облікові записи користувачів та інші необхідні компоненти.
- Розробити логіку та алгоритми роботи системи

2. Розробка інтерфейсу:

- Створити привабливий дизайн інтерфейсу, враховуючи ергономіку та користувацький досвід.
- Забезпечити зручну навігацію та інтуїтивно зрозумілі елементи керування.
- Розробити адаптивний дизайн, що забезпечує коректне відображення інтерфейсу на різних пристроях і розмірах екранів.

3. Розробка бази даних:

- Створити структуру бази даних для зберігання інформації про товари, клієнтів, замовлення тощо.
- Реалізувати механізми доступу до даних, включаючи операції додавання, редагування та видалення записів.

4. Інтеграція платіжних систем:

- Розробити інтеграцію з платіжними системами, які відповідають потребам інтернет-магазину.

6. Тестування та вдосконалення:

- Провести ретельне тестування різних функціональних аспектів системи, включаючи функції пошуку, додавання товарів до корзини, оформлення замовлень тощо.

- Виявити та виправити помилки, а також забезпечити відповідність роботи системи заданим вимогам та очікуванням користувачів.

- Оптимізувати продуктивність системи, забезпечити швидкий час відгуку сторінок та оптимальне використання ресурсів сервера.

7. Аналітика та звітність:

- Реалізувати механізми для збору та аналізу даних про продажі, поведінку користувачів та ефективність маркетингових акцій.

На основі цього дослідження ми розробимо зручний та інтуїтивно зрозумілий інтерфейс магазину, що дозволить користувачам швидко знаходити потрібний їм товар та здійснювати покупку всього за кілька кроків. Забезпечимо можливість сортування товарів за різними параметрами, широкий вибір фільтрів і зручну систему пошуку. Ми також забезпечимо безпеку та конфіденційність персональних даних користувачів. Використаємо надійні методи шифрування та захисту інформації, щоб гарантувати, що дані користувачів залишаються конфіденційними та захищеними від несанкціонованого доступу. Інтеграція платіжної системи дозволить забезпечити зручність та безпеку оплати товарів у магазині. Ми розробимо механізми, що забезпечать інтеграцію з платіжною системою. Після розробки інтернет-магазину ми проведемо ретельне тестування, щоб переконатися у правильному функціонуванні всіх компонентів системи. Ми також виявимо та виправимо помилки, що можуть впливати на коректну роботу системи, та забезпечимо відповідність роботи системи заданим вимогам та очікуванням користувачів. В цілому, ми ставимо за мету створити інтернет-магазин, який буде забезпечувати зручну та безпечну купівлю цифрової техніки для наших клієнтів, забезпечуючи високу якість обслуговування та задоволення їх потреб.

1.2 Аналіз аналогів програмного продукту

Аналіз аналогів програмного продукту перед розробкою інтернет-магазину цифрової техніки є важливою складовою частиною процесу. Аналогічні програмні продукти можуть надати корисні вказівки, які допоможуть нам узгодити функціональність, покращити взаємодію з користувачем та забезпечити ефективну роботу вашого інтернет-магазину. Ось кілька кроків, які можна виконати для аналізу аналогів:

1. Переглянемо існуючі інтернет-магазини цифрової техніки: проведемо пошук і вивчимо популярні існуючі магазини цифрової техніки, щоб отримати уявлення про їх функціональність, дизайн та особливості. Проаналізуємо їхні можливості.

2. Визначимо ключові функції: Створимо список основних функцій, які має виконувати наш інтернет-магазин цифрової техніки. Розглянемо, як ці функції реалізовані в аналогічних програмних продуктах..

3. Оцінка інтерфейсу користувача: Проаналізуємо дизайн інтерфейсу користувача в аналогічних програмних продуктах. Розглянемо, як користувачі взаємодіють з магазином, наскільки зручний та інтуїтивно зрозумілий інтерфейс.

4. Зробимо висновки та складемо план: Після аналізу аналогів зберемо всі отримані дані та зробимо висновки. Визначимо, що ми хотіли б запозичити від аналогічних програмних продуктів та які унікальні функції ми хочемо розробити самостійно. Складемо план розробки з урахуванням виявлених переваг та недоліків аналогічних програмних продуктів.

На українському ринку існує кілька популярних інтернет-магазинів цифрової техніки, які можуть стати нам прикладами для аналізу з програмної розробки. Ось кілька з них, разом з їхніми плюсами і мінусами:

1. "Розетка": можна описати як інноваційну електронну платформу, що використовує передові технології для забезпечення зручного та ефективного досвіду покупок для клієнтів. Сайт Розетка має швидкий та надійний інтерфейс, де користувачі можуть швидко знайти та придбати товари різних [14].

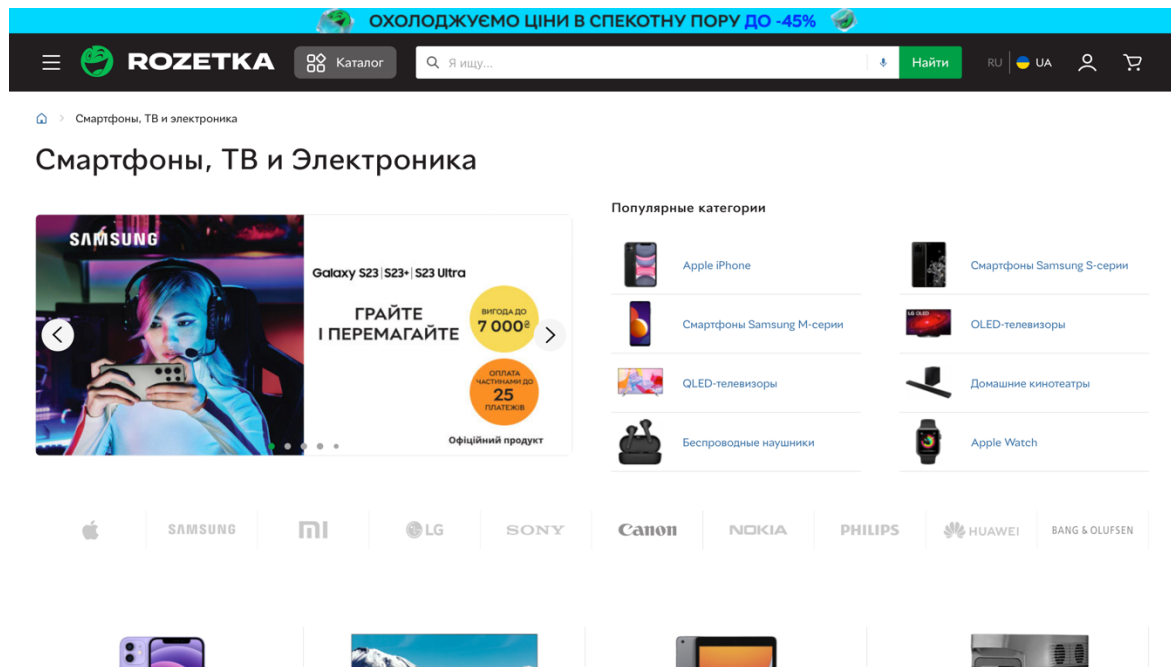


Рис.1.1. Головна сторінка "Розетка"

Плюси:

- Великий вибір товарів і різноманітні категорії.
- Зручний пошук товарів та фільтри для точного вибору.
- Зручний процес оформлення замовлення та оплати.
- Відгуки користувачів та рейтинги товарів допомагають у виборі.

Мінуси:

- Інтерфейс користувача може бути складним та перевантаженим.
- Деякі користувачі скаржаться на проблеми з доставкою та обробкою замовлень.
- Можуть виникати випадки неконсистентності даних, коли оновлення не відображаються одразу на всіх сторінках або між різними пристроями.

2. "Алло": - це онлайн-платформа, яка надає широкий вибір електроніки. Сайт Алло розроблений з використанням сучасних технологій та має зручний інтерфейс, що дозволяє зручно шукати, порівнювати та замовляти товари онлайн. Крім того, сайт пропонує різноманітні функції, систему рейтингу, можливість зберігання обраного товару [10].

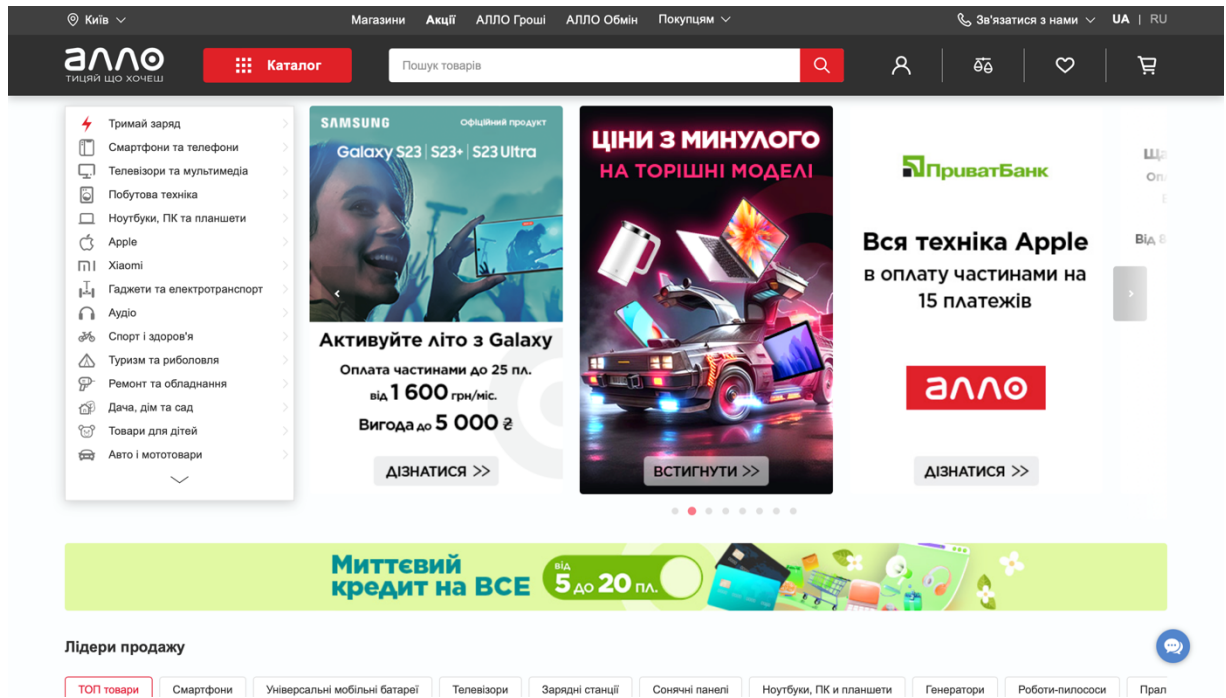


Рис.1.2. Головна сторінка "Алло"

Плюси:

- Зручний та простий інтерфейс користувача.
- Швидкий процес оформлення замовлення та оплати.
- Захист персональних даних та система оплати є надійними, що забезпечує безпеку покупок на сайті.
- Індивідуальні пропозиції та знижки для зареєстрованих користувачів.

Мінуси:

- Потенційні проблеми з завантаженням: Залежно від обсягу інформації та навантаженості серверів.
- Іноді можуть виникати помилки або технічні проблеми, які впливають на роботу сайту та користувачів.

3. "Цитрус" Сайт "Цитрус" має привабливий та інтуїтивно зрозумілий дизайн, що допомагає користувачам легко зорієнтуватися та знайти необхідний товар. Він володіє швидким завантаження сторінок і ефективними механізмами пошуку, що дозволяють клієнтам швидко здійснювати покупки без зайвого часу на очікування [5].

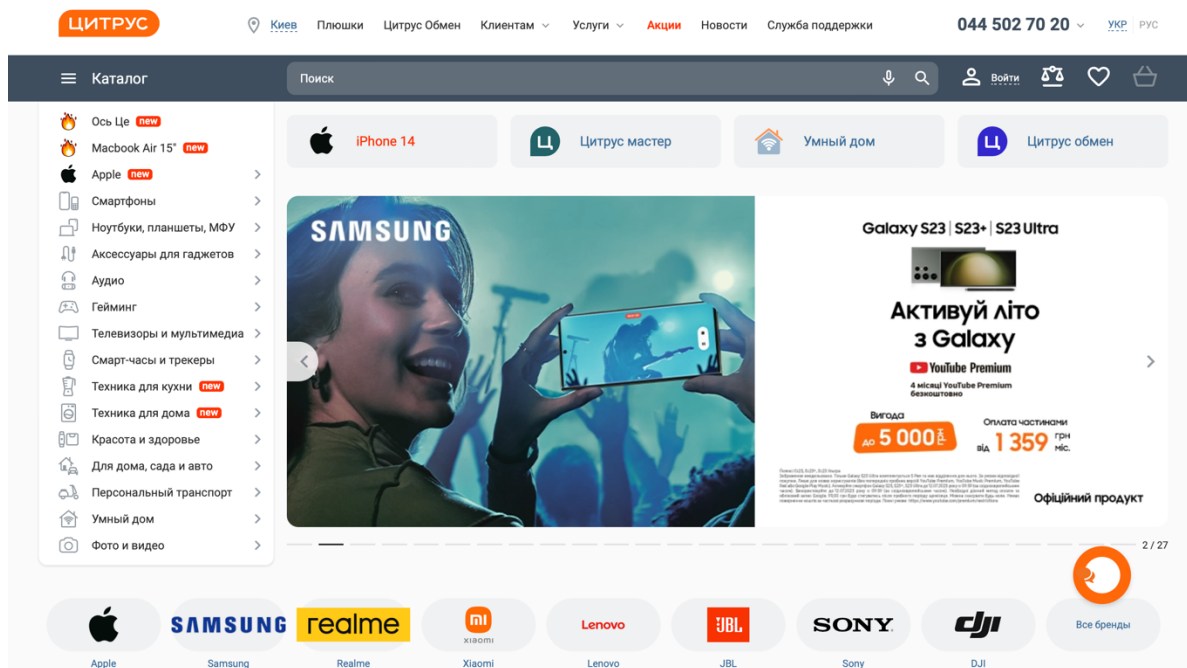


Рис.1.3. Головна сторінка "Цитрус"

Плюси:

- Сайт "Цитрус" вражає своєю швидкістю, що поліпшує користувацький досвід.
- Зручна система фільтрів для швидкого вибору товарів.
- Привабливий та зручний дизайн сайту.
- Захист даних та безпека: Сайт "Цитрус" забезпечує високий рівень захисту особистих даних та фінансової інформації клієнтів.

Мінуси:

- Недостатня інформація про деякі товари та їх характеристики.
- Іноколи можуть виникати технічні проблеми, такі як помилки завантаження сторінок або несправні посилання. Хоча такі проблеми можуть бути рідкісними, вони можуть тимчасово завадити.

Висновок, який потрібно зробити при розробці сайту на основі аналізу плюсів та мінусів вище згаданих інтернет-магазинів цифрової техніки, може бути наступним:

1. Функціональність:

- Потрібно забезпечити широкий вибір товарів та різноманітні категорії для задоволення потреб клієнтів.
- Важливим аспектом є зручний пошук товарів та наявність фільтрів, що допоможуть користувачам знайти потрібні їм товари швидко та ефективно.
- Процес оформлення замовлення та оплати повинен бути легким та зрозумілим для користувачів.

2. Інтерфейс користувача:

- Важливо створити простий та зручний інтерфейс, що сприятиме легкій навігації для користувачів.
- Звернути увагу на привабливий та естетичний дизайн, який буде привертати увагу та створювати позитивне враження.

3. Користувацький досвід:

- Забезпечити можливість відгуків користувачів та рейтингів товарів, що допоможе іншим покупцям при виборі товару.

4. Швидкість та надійність:

- Приділити увагу оптимізації сайту, щоб забезпечити швидку роботу та завантаження сторінок.

Враховуючи плюси та мінуси вище згаданих інтернет-магазинів, ми можемо створити свій власний програмний продукт, який буде враховувати недоліки конкурентів та вдосконалювати їх позитивні сторони, створюючи зручний, функціональний та приємний досвід користування для нашого магазину цифрової техніки.

1.3 Вибір архітектури системи інтернет-магазину цифрової техніки

Для реалізації веб-орієнтованої системи інтернет-магазину цифрової техніки нами було обрано NEXT.JS API Router. Розглянуті інші паттерни не забезпечують такої гнучкості щодо відображення даних.

NEXT.JS є фреймворком розробки веб-додатків, який базується на Node.js та React.js. Він надає можливість створення як статичних, так і динамічних веб-сайтів з використанням серверного та клієнтського рендеринга. API Router у NEXT.JS дозволяє створювати серверні API шляхи, які можуть бути використані для отримання або збереження даних.

Основні компоненти архітектури системи NEXT.JS API Router включають наступне:

1. Фронтенд (клієнтська частина): Використовуючи React.js, розробники можуть створювати компоненти, що відображають користувацький інтерфейс. NEXT.JS надає можливість рендерингу на стороні сервера та на стороні клієнта, що дозволяє використовувати різні підходи до відображення сторінок [9].

2. API шляхи: NEXT.JS дозволяє визначити API шляхи, використовуючи вбудовану функціональність API Router. Ці шляхи визначають, які дані можуть бути отримані або змінені на сервері. Розробники можуть створювати API шляхи, використовуючи декларативний підхід, де вони вказують URL-шаблони та функції-обробники для кожного шляху [3].

3. Обробники запитів: Кожен API шлях має пов'язану функцію-обробник, яка виконується при отриманні запиту на відповідний шлях. Ця функція може обробляти запити, виконувати бізнес-логіку, взаємодіяти з базою даних або іншими зовнішніми службами, і повертати результату вигляді відповіді.

4. Маршрутизація: NEXT.JS API Router надає можливість маршрутизувати запити на різні шляхи залежно від URL-шаблонів. Розробники можуть використовувати параметри шляху для передачі додаткових даних до функцій-обробників та динамічного створення API шляхів [3].

5. Валідація та автентифікація: NEXT.JS API Router підтримує можливості валідації та автентифікації запитів. Розробники можуть використовувати різні пакети або власні методи для перевірки достовірності користувачів, перевірки вхідних даних та забезпечення безпеки API [4].

6. Додаткові функціональні можливості: NEXT.JS API Router також надає ряд додаткових функціональних можливостей, таких як використання middleware для обробки запитів перед функціями-обробниками, підтримка WebSocket-з'єднань та інші.

Загальна архітектура системи NEXT.JS API Router базується на розділенні клієнтської та серверної частин, використанні API шляхів для взаємодії з даними, функціях-обробниках для обробки запитів та маршрутизації, а також додаткових можливостях для забезпечення безпеки та розширення функціональності.

1.4 Обґрунтування вибору інструментальних засобів та вимоги до апаратного забезпечення

Для створення інтернет-магазину доцільно застосувати: фреймворк NEXT.JS, CSS modules для стилізації компонентів, також NEXT.JS буде використаний для реалізації логіки роботи сайту на сервері, хмарна база даних MongoDB Atlas, графічний редактор Figma.

При виборі інструментальних засобів для розробки інтернет-магазину було враховано декілька ключових факторів.

1. NEXT.JS: NEXT.JS є популярним фреймворком реактивного веб-рендерингу, який забезпечує високу продуктивність та ефективну розробку. Він підтримує серверний рендеринг, статичний рендеринг та клієнтський рендеринг, що дозволяє досягти оптимальної швидкості завантаження сторінок. NEXT.JS також має зручну систему маршрутизації, що полегшує розробку та підтримку проекту[3].

2. CSS модулі: Використання CSS модулів дозволяє створювати локальні області стилів для кожного компонента. Це допомагає уникнути конфліктів стилів і забезпечує легку переносимість та повторне використання компонентів. CSS модулі також дозволяють зберігати стилі поряд з компонентами, що спрощує розробку та підтримку[8].

3. MongoDB Atlas: MongoDB - це нереляційна база даних, яка забезпечує гнучкість та масштабованість. Вона дозволяє зберігати дані у вигляді документів, що дозволяє швидко змінювати структуру даних та додавати нові поля. MongoDB також має гарну масштабованість, дозволяючи розширювати кластери та обробляти великі обсяги даних. MongoDB Atlas - це повністю керований хмарний сервіс бази даних MongoDB, наданий компанією MongoDB. Він пропонує надійне та масштабоване рішення для зберігання та управління даними для інтернет-магазину[11].

4. JetBrains WebStorm - це інтегроване середовище розробки (IDE) для веб-розробки, розроблене компанією JetBrains. Воно спеціалізується на наданні потужних інструментів для розробки веб-додатків на основі HTML, CSS та JavaScript. WebStorm пропонує засоби для підсвічування синтаксису, автодоповнення коду, рефакторингу, налагодження, автоматичної перевірки синтаксису та багато іншого. Його інтуїтивний інтерфейс дозволяє розробникам зосередитись на написанні якісного коду, спрощуючи процес розробки та забезпечуючи продуктивність. Загалом, WebStorm є потужним інструментом для розробки веб-додатків, який надає широкий набір функцій та засобів для полегшення процесу розробки та підвищення продуктивності розробників.

Обрані інструментальні засоби надають широкі можливості для ефективної розробки інтернет-магазину, забезпечуючи швидкість, гнучкість та зручність у використанні.

Висновки до першого розділу

Проведений аналіз в сфері інтернет-магазину цифрової техніки дозволив визначити основні напрямки автоматизації. Важливо реалізувати функціонал для збереження даних про товари та їх характеристики, а також про клієнтів та їх замовлення відповідно до категорій. Необхідно розробити механізми для швидкого пошуку та підбору товарів за параметрами, а також для ефективного управління замовленнями та інформацією про клієнтів.

Огляд існуючих аналогів показав, що основними функціями систем для інтернет-магазину цифрової техніки є: розміщення товарів, формування списків товарів, пошук товарів в Інтернеті та розміщення оголошень, наявність мережевої версії та веб-інтерфейсу. Також важливими функціями є відстеження замовлень після покупки та імпорт каталогу товарів для зручного використання програми. Також, для цього інтернет-магазину цифрової техніки було б доцільно реалізувати такі додаткові функції, як пошук товарів та фільтрація за певними параметрами, можливість друку звітів та ведення статистики роботи магазину.

В роботі обґрунтовано вибір архітектури для реалізації системи:

- NEXT.JS як front-end;
- CSS modules для стилізації компонентів;
- NEXT.JS як back-end;
- Хмарне середовище керування базами даних MongoDB Atlas;

Буде використано такі програмні продукти та інтегровані середовища розробки як: графічний редактор Figma, среда розробки Web Storm.

РОЗДІЛ 2. ПРОЕКТУВАННЯ СИСТЕМИ ІНТЕРНЕТ-МАГАЗИНУ ЦИФРОВОЇ ТЕХНІКИ

2.1. Визначення варіантів використання та структури системи

Варіанти використання та структура системи інтернет-магазину цифрової техніки можуть бути такими:

Функціональні вимоги:

1. Реєстрація та авторизація користувачів:

- Дозволити користувачам створювати облікові записи та виконувати вхід до системи[3].
- Забезпечити механізм підтвердження облікового запису через електронну пошту.

2. Пошук та перегляд товарів:

- Забезпечити можливість пошуку товарів.
- Відобразити інформацію про товар, включаючи назву, опис, зображення та відгуки.

3. Корзина та замовлення:

- Дозволити користувачам додавати товари до корзини та здійснювати оплату за допомогою різних платіжних методів.
- Забезпечити можливість перегляду

4. Управління адміністратором:

- Забезпечити адміністратору доступ до панелі управління для додавання, редагування та видалення товарів.
- Надати можливість керування категоріями товарів, керування замовленнями, відгуками користувачів та іншими аспектами системи.

5. Обробка платежів та доставка:

- Інтегрувати систему з провайдерами платіжних послуг для прийому платежів в режимі реального часу[12].

- Забезпечити можливість вибору різних методів доставки та розрахунку вартості доставки залежно від регіону та ваги товару.

6. Користувацький обліковий запис:

- Дозволити користувачам переглядати та редагувати свій профіль, адреси доставки, історію замовлень та збережені товари.

7. Відгуки та рейтинги:

- Дозволити користувачам залишати відгуки про товари.
- Забезпечити можливість перегляду та фільтрації відгуків, що допоможе користувачам при прийнятті рішення про покупку.

Це лише загальний огляд можливих варіантів використання та структури системи інтернет-магазину цифрової техніки. Фактична структура може варіюватись залежно від конкретних вимог, функціональності та потреб

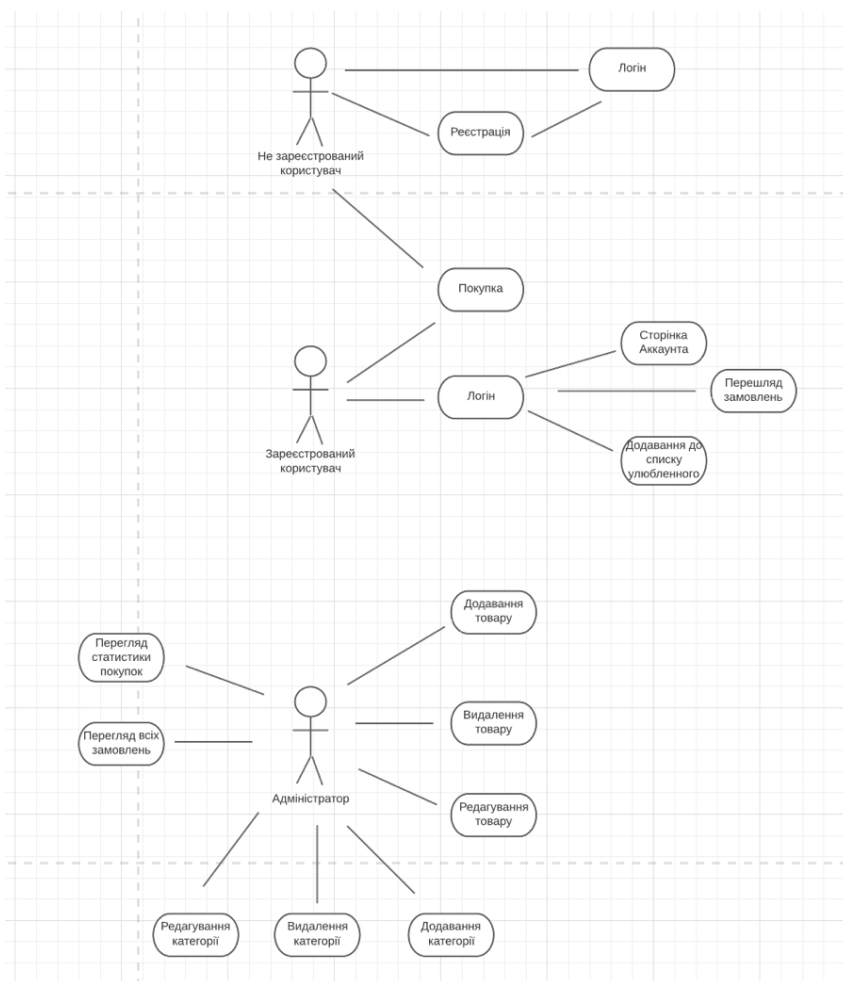


Рис. 2.1. Варіанти використання інтернет-магазину

Нефункціональні вимоги:

1. Навчання користувачів системи:

- Час, необхідний для ознайомлення з інструментами роботи інформаційної системи для звичайних користувачів, повинен становити 1 година, а для досвідчених - 30 хвилин.
- Система повинна реагувати на запити звичайних користувачів протягом 1 секунди, а на складніші запити - протягом 3 секунд.
- Інтерфейс користувача повинен бути інтуїтивно зрозумілим та не вимагати додаткової підготовки.
- Надійність системи.
- Доступність - час обслуговування системи не повинен перевищувати 10% загального часу роботи.
- Середній час безвідмовної роботи - 20 робочих днів.
- Максимальна норма помилок та дефектів у роботі системи - 1 помилка на 1000 запитів користувача.

2. Продуктивність:

- Система повинна підтримувати щонайменше 100 одночасно працюючих користувачів, які мають доступ до спільної бази даних.

3. Можливість експлуатації:

- Масштабованість - система повинна мати можливість збільшувати свою потужність та продуктивність разом зі зростанням числа користувачів, не негативно впливаючи на її функціонування.
- Автоматичне оновлення версій - оновлення версій повинно здійснюватися автоматично відповідно до вподобань користувачів та розширення переліку надання послуг.

Після проведення аналізу функціональних вимог було виявлено низку сутностей, які забезпечать реалізацію програмного комплексу системи. На рисунку 2.2 представлена діаграма класів для рівня API routes. У розділі 2.2

буде наведена модель бази даних системи, відповідно до якої буде організовано рівень моделі.

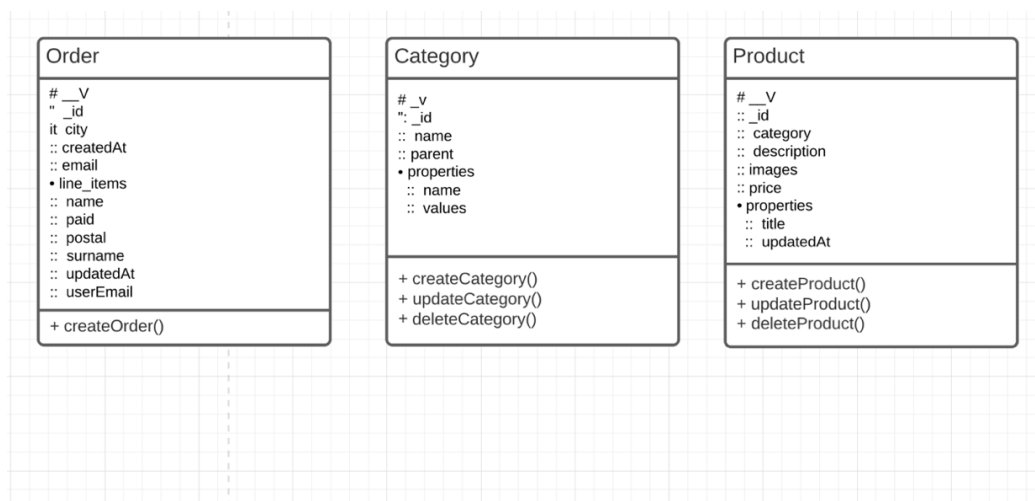


Рис. 2.2. Діаграма класів

Дана система інтернет-магазину реалізує функціонал дій над товарами, категоріями і замовленнями. Класи, які можна виділити на основі наданого опису, включають:

1. Order: Цей клас використовується для представлення та обробки інформації про замовлення, зроблені покупцями. У ньому можуть бути методи, такі як createOrder для створення замовлення в інтернет-магазині.

2. Category: Цей клас відповідає за роботу з категоріями на сайті. Він має методи, такі як createCategory для створення нової категорії на сайті, updateCategory для редагування існуючої категорії та deleteCategory для видалення обраної категорії з сайту.

3. Product: Цей клас відповідає за роботу з товарами на сайті. Він має методи, такі як createProduct для створення нового товару на сайті, updateProduct для редагування існуючого товару та deleteProduct для видалення обраного товару з сайту.

Отже, система забезпечує можливість взаємодії з товарами та категоріями, пошук товарів і збереження даних про замовлення. Для цього використовуються методи API, які доступні через маршрутизацію API для продуктів (product), категорій (category) та замовлень (order).

2.2. Розробка бази даних системи

Реалізована база даних MongoDB Atlas складається з 11 колекцій, які містять у собі всі дані для роботи програми. База даних має назву shop.

Таблиця 2.1

Інформація про колекції бази даних shop

Назва таблиці	Призначення
Accounts	Акаунти в системі (В одному акаунті може декілька users)
Admins	Адміністратори
Categories	Категорії
Details	Інформація про користувача
Orders	Замовлення
Products	Товари
Session	Стандартний клас NEXT auth
Users	Користувачі сайту
Wishlist	Список бажаного
Reviews	Відгуки до товару
Settings	Налаштування сайту

Структура бази даних наведена на рис. 2.3.

The screenshot displays the MongoDB Atlas interface for a database named 'test'. On the left, a sidebar lists the collections: accounts, admins, categories, details, orders, products, reviews, sessions, settings, users, and wishlists. The main panel shows a table of collection statistics. The table has columns for Collection Name, Documents, Logical Data Size, Avg Document Size, Storage Size, Indexes, Index Size, and Avg Index Size. The data is as follows:

Collection Name	Documents	Logical Data Size	Avg Document Size	Storage Size	Indexes	Index Size	Avg Index Size
accounts	3	4.94KB	1.65KB	44KB	1	36KB	36KB
admins	3	316B	106B	36KB	2	72KB	36KB
categories	6	815B	136B	36KB	1	36KB	36KB
details	3	534B	178B	36KB	1	36KB	36KB
orders	12	4.92KB	420B	36KB	1	36KB	36KB
products	9	4.63KB	527B	36KB	1	36KB	36KB
reviews	2	1.39KB	710B	36KB	1	36KB	36KB
sessions	6	684B	114B	36KB	1	36KB	36KB
settings	1	95B	95B	36KB	2	40KB	20KB
users	3	586B	196B	36KB	1	36KB	36KB
wishlists	3	294B	98B	24KB	1	24KB	24KB

Рис.2.3. Схема БД

Колекція "Accounts" це стандартна таблиця NEXT.JS auth. Вона використовується для авторизації користувачів через google:

Таблиця 2.2

Опис полів колекції "Accounts"

Назва	Тип даних	Опис поля
_id	ObjectID	Номер акаунту
provider	String	Сервіс для авторизації
providerAccountId	String	Id акаунту
userId	ObjectID	Id користувача

Колекція "Admins" використовується для зберігання і керування даними адміністраторів, які мають доступ до адміністративних функцій інтернет-магазину. Основна мета цієї таблиці - забезпечити контроль доступу та аутентифікацію адміністраторів.

Таблиця 2.3

Опис полів колекції "Admins"

Назва	Тип даних	Опис поля
_id	ObjectID	Номер
email	String	Email адміністратора

Колекція "Categories" використовується для зберігання інформації про категорії товарів. Вона дозволяє організувати товари в ієрархічну структуру категорій та надає зручний спосіб управління та організації товарів у магазині.

Таблиця 2.4

Опис полів колекції "Categories"

Назва	Тип даних	Опис поля
_id	ObjectID	Номер
name	String	Назва категорії
parent	ObjectID	Дочірня категорія
properties	Array	Властивості

Колекція "Details" представляє собою збірку даних, яка містить особисті дані користувачів, які вони надають самостійно. Ця колекція зберігає інформацію, яку користувачі вводять про себе, щоб забезпечити більш налаштований досвід використання сервісу.

Таблиця 2.5

Опис полів колекції " Details "

Назва	Тип даних	Опис поля
_id	ObjectID	Номер
userEmail	String	Email акаунту
name	String	Id акаунту
surname	String	Id користувача
email	String	Email користувача
city	String	Місто
postal	String	Номер пошти

Колекція "Orders" використовується для зберігання інформації про замовлення, зроблені клієнтами. Вона містить записи про кожне замовлення і включає такі поля:

Таблиця 2.6

Опис полів колекції " Orders "

Назва	Тип даних	Опис поля
_id	ObjectID	Номер
line_items	array	Містить дані про товари
name	String	Ім'я покупця
surname	String	Прізвище покупця
email	String	Email покупця
city	String	Місто куди відправляти
postal	String	Номер пошти

Колекція "Products" використовується для зберігання та управління інформацією про товари, які доступні для продажу в магазині. Основна мета цієї таблиці - забезпечити зручний доступ до даних про товари для розміщення їх на веб-сторінках магазину та керування продуктами.

Таблиця 2.7

Опис полів колекції "Products"

Назва	Тип даних	Опис поля
_id	ObjectID	Номер
title	String	Назва товару
description	String	Опис товару
price	Int32	Ціна
images	Array	Фото
category	String	Категорія товару
properties	String	Властивість товару

Колекція "Users" використовується для зберігання інформації про користувачів системи. Ця таблиця має рядки та стовпці, що дозволяють зберігати різні атрибути та дані, пов'язані з користувачами.

Таблиця 2.8

Опис полів колекції " Users "

Назва	Тип даних	Опис поля
_id	ObjectID	Номер
name	String	Ім'я акаунту
email	String	Email акаунту
image	String	Фото

Колекція "Wishlist" використовується для збереження інформації про бажані товари клієнтів. Ця таблиця дозволяє клієнтам додавати товари, які їм сподобалися, до свого списку бажань.

Таблиця 2.9

Опис полів колекції "Wishlist "

Назва	Тип даних	Опис поля
_id	ObjectID	Номер
userEmail	String	Email акаунту який додав
product	String	Товар в списку бажань

Зберігання відгуків у колекції "Reviews" дозволяє відображати та керувати відгуками клієнтів, забезпечувати іншим користувачам доступ до думок і рейтингів, а також використовувати цю інформацію для аналізу задоволення клієнтів та покращення якості товарів та послуг.

Таблиця 2.10

Опис полів колекції "Reviews "

Назва	Тип даних	Опис поля
_id	ObjectID	Номер
title	String	Заголовок
text	String	Текст відгуку
stars	Int32	Оцінка
product	ObjectID	Ідентифікатор товару
createdAt	Date	Дата і час створення відгуку
updatedAt	Date	Дата і час останнього оновлення відгуку

Колекція "Settings" використовується для зберігання різноманітних налаштувань та конфігураційних параметрів, які впливають на функціонування самого магазину. Колекція "Settings" дозволяє адміністраторам змінювати різні параметри без потреби внесення змін в сам код магазину. Це надає гнучкість та можливість налаштування магазину в залежності від потреб та змінюваних вимог. Ця колекція дозволяє зберігати та змінювати налаштування, що використовуються для налаштування різних аспектів веб-додатку.

Таблиця 2.11

Опис полів колекції " Settings "

Назва	Тип даних	Опис поля
_id	ObjectID	Номер
name	String	Назва налаштування
value	String	Id товару

Колекція "Sessions" використовується для збереження інформації про сесії користувачів під час їх взаємодії з магазином. Сесія представляє собою період часу, протягом якого користувач знаходиться на сайті або в додатку і має активну взаємодію.

Таблиця 2.12

Опис полів колекції " Sessions"

Назва	Тип даних	Опис поля
_id	ObjectID	Номер
sessionToken	String	Токен
userId	ObjectID	Ідентифікатор користувача
expires	Date	Дата і час завершення сесії

Отже, існує база даних, яка відповідає всім вимогам і забезпечує збереження та колективний доступ до інформації інтернет-магазину. Ця база даних має 11 колекцій, серед яких основними є "Wishlist", "Users", "Products", "Orders", "Details", "Categories", "Admins", "Sessions", "Settings", "Reviews", та "Accounts". Кожна колекція виконує важливу роль у збереженні та організації певних аспектів магазину. Загалом, ця база даних надає повну функціональність для здійснення операцій покупки, керування товарами та категоріями, обробки замовлень та збереження інформації про користувачів.

2.3. Проектування та реалізація алгоритмів роботи системи

Основними модулями системи є модуль оформлення замовлення користувачем та зберігання даних про замовлення у базі даних.

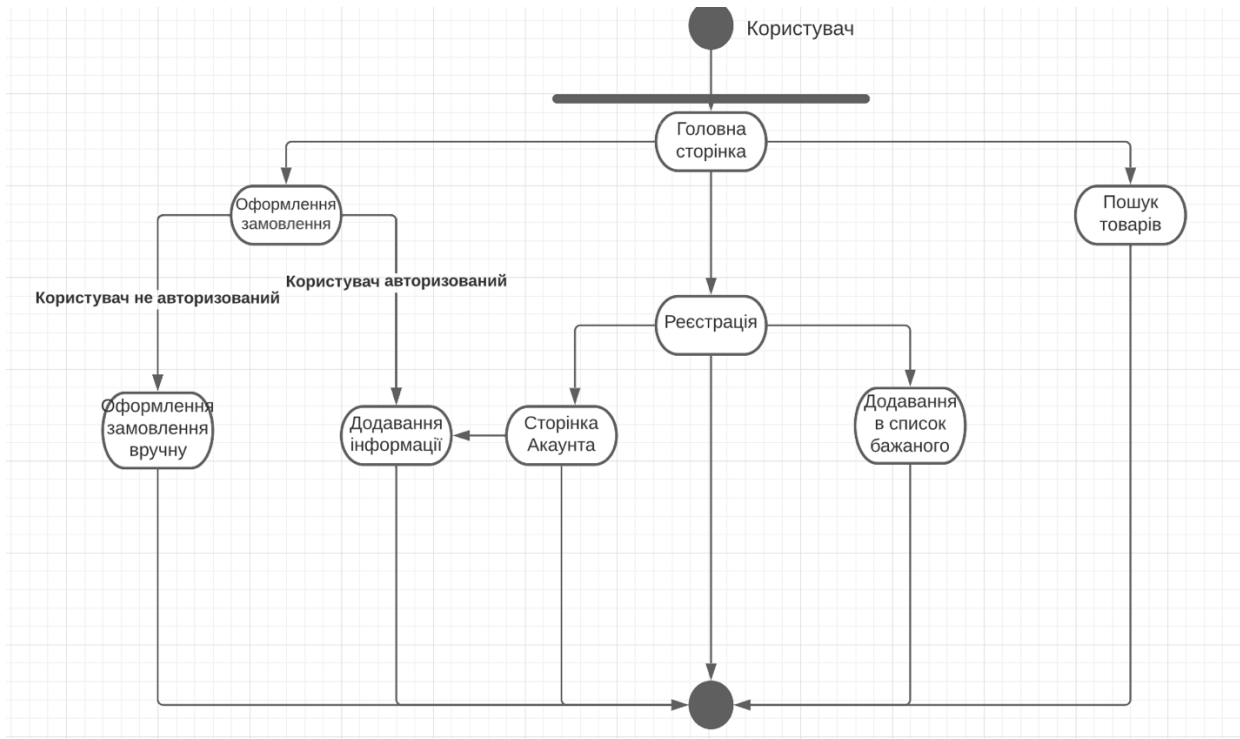


Рис. 2.4. Діаграма активності (роль Користувач)

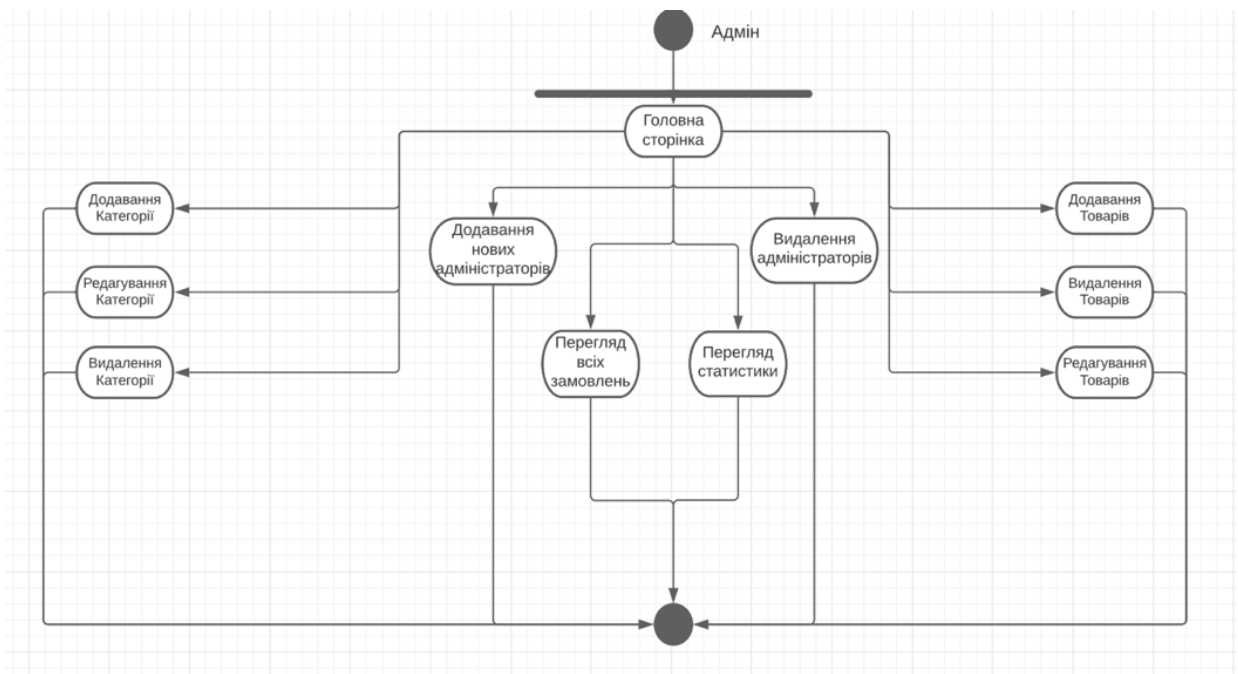


Рис. 2.5. Діаграма активності (роль Адмін)

Опис діаграм зображених на рисунках 2.4 та 2.5:

У разі авторизації адміністратора:

- Після успішної авторизації адміністратор отримує спеціальний доступ до адміністративної панелі, де він має повний контроль над функціями та налаштуваннями системи.

- Адміністратор може створювати нові записи, видаляти, редагувати та переглядати існуючі записи в клієнтській базі даних. Також він може керувати правами доступу інших користувачів, встановлювати обмеження і налаштовувати рівень привілеїв.

- Адміністратор, маючи доступ до бази даних замовлень, має можливість здійснювати різні дії над замовленнями з метою їх керування та обробки. Основні операції, які адміністратор може виконувати, включають: перегляд та видалення замовлень.

У разі авторизації звичайного користувача:

- Після успішної авторизації користувач буде перенаправлений на головну сторінку магазину, де він зможе отримати загальну інформацію про продукти та акції.

- Користувач може переглядати список товарів з детальним описом, зображеннями та цінами. Він може обирати та додавати певні товари до свого списку бажань для подальшого оформлення замовлення або для зручності перегляду.

- Крім того, користувач має можливість налаштовувати свій особистий профіль, де він може вказати адресу доставки, змінити особисті дані, зберегти вибрані налаштування та переглянути історію своїх замовлень.

Таким чином, процес авторизації визначає рівень доступу та функціональні можливості, які стають доступними користувачеві в системі інтернет-магазину. Залежно від ролі користувача, якою він авторизується, система надає відповідний набір прав і функцій.

2.4 Реалізація додатку інтернет-магазину.

1. Реалізація підключення до БД. Була створена функція `mongooseConnect` для забезпечення з'єднання з базою даних та налаштування параметрів з'єднання. Для цього функція використовує параметри, які знаходяться в конфігураційному файлі `.env`.

2. Реалізація реєстрації. Для реєстрації користувачів у системі була реалізована функція `authOptions()` з використанням бібліотеки `NextAuth`. Ця бібліотека надає зручний та гнучкий спосіб додавання різних стратегій аутентифікації, таких як соціальні мережі, пароль, одноразові коди та інші. У нашому випадку було додано можливість реєстрації через обліковий запис Google. Нижче наведено приклад коду:

```
import NextAuth from 'next-auth'
import GoogleProvider from 'next-auth/providers/google'
import {MongoDBAdapter} from "@next-auth/mongodb-adapter";
import clientPromise from "@lib/mongodb";

export const authOptions = {
  providers: [
    // OAuth authentication providers..
    GoogleProvider({
      clientId: process.env.GOOGLE_FRONT_ID,
      clientSecret: process.env.GOOGLE_FRONT_SECRET
    }),
  ],
  adapter: MongoDBAdapter(clientPromise),
};

export default NextAuth(authOptions);
```

3. Реалізація з'єднання. Для виведення списку товарів додатку було використано функцію `getServerSideProps()`. Ця функція є функцією серверної сторони веб-фреймворку `Next.js`, яка використовується для отримання даних перед рендерингом компоненту на сервері. В цьому коді ми викликаємо функцію `mongooseConnect()` для забезпечення з'єднання з базою даних. Потім, за допомогою методу `find()`, отримуємо всі категорії з колекції `Product`. Результат запиту зберігається у змінній `products`. Цей код дозволить нам

отримати список категорій з бази даних та передати його як властивість `products` в компонент для подальшого використання.

```
export async function getServerSideProps(context) {
  await mongooseConnect()
  const {id} = context.query;
  const product = await Product.findById(id);
  return {
    props:{
      product: JSON.parse(JSON.stringify(product)),
    }
  }
}
```

Також, аналогічним методом показаним вище, було реалізовано і виведення списку категорій. Отримані дані, такі як категорія, підкатегорії та продукти, вони будуть використані для рендерингу відповідних компонентів на сервері.

```
export async function getServerSideProps(context){
  const category = await Category.findById(context.query.id);
  const subCategories = await Category.find({parent:category._id});
  const catIds = [category._id, ...subCategories.map(cat => cat._id)];
  const products = await Product.find({category:catIds});
  return {
    props:{
      category: JSON.parse(JSON.stringify(category)),
      subCategories: JSON.parse(JSON.stringify(subCategories)),
      products: JSON.parse(JSON.stringify(products))
    }
  }
}
```

4. Бібліотека `Axios`. Також, для роботи з базою даних ми часто використовуємо бібліотеку `Axios`. Для прикладу можемо розглянути отримання даних для відображення списку відгуків на сторінці. Інші запити до бази даних мають аналогічний вигляд, цей використовується як приклад.

```
function loadReviews() {
  setLoading(true);
  axios.get('/api/reviews?product='+product._id).then(res => {
    setReviews(res.data);
    setStars(res.data);
    setLoading(false);
  });
}
```

Ця функція `loadReviews()` використовується для завантаження відгуків про певний товар з сервера. Вона має наступну структуру:

1. `axios.get('/api/reviews?product='+product._id)`: Здійснює GET-запит на сервер за допомогою бібліотеки Axios. URL-адреса `/api/reviews` вказує на ендпоінт на сервері, який відповідає за отримання відгуків. Запит включає параметр `product`, який містить ідентифікатор продукту (`product._id`), щоб отримати відгуки, пов'язані з цим продуктом.

2. `setReviews(res.data), setStars(res.data), setLoading(false)`: Встановлює стан на основі даних, отриманих від сервера. Це оновлює список відгуків для відображення.

Ця функція використовується для асинхронного завантаження відгуків з сервера та оновлення стану компонента, щоб відобразити ці відгуки користувачу.

5. Реалізація API функцій. Серверні функції також мають приблизно однаковий вигляд. Проаналізуємо функцію для отримання всіх відгуків до товару.

```
import {mongooseConnect} from "@lib/mongoose";
import {Review} from "@models/Review";

export default async function handle(req, res) {
  await mongooseConnect();

  if (req.method === 'POST') {
    const {title, text, stars, product} = req.body;
    res.json(await Review.create({title, text, stars, product}));
  }
  if (req.method === 'GET') {
    const {product} = req.query;
    res.json(await Review.find({product}, null, {sort: {createdAt: -1}}));
  }
}
```

Основна мета цього коду - підключення до бази даних MongoDB за допомогою Mongoose і обробка запитів POST і GET.

1. У перших двох рядках імпортуються потрібні модулі: `mongooseConnect` для підключення до бази даних і `Review` для роботи з моделлю відгуків.

2. Функція `handle` експортується як `default` і є асинхронною. Вона приймає параметри `req` і `res`, які представляють об'єкти запиту і відповіді сервера відповідно.

3. У блоку `if (req.method === 'POST')` витягуються дані з тіла запиту (`req.body`). Вони використовуються для створення нового об'єкту відгуку (`Review`) за допомогою `'Review.create'`. Результат цього створення повертається у відповідь в форматі JSON за допомогою `res.json`.

4. У блоку `if (req.method === 'GET')` витягується параметр `product` з об'єкту запиту (`req.query`). Він використовується для пошуку відгуків, що належать до даного продукту за допомогою `Review.find`. Результат цього пошуку повертається у відповідь в форматі JSON за допомогою `res.json`. Параметр `null` вказує, що всі поля повинні бути повернуті, а `{sort:{createdAt:-1}}` вказує сортування за полем `createdAt` в порядку спадання.

Цей код є частиною серверної логіки для обробки запитів POST і GET пов'язаних з відгуками на продукти. Він використовує `Mongoose` для підключення до бази даних і взаємодії з моделлю відгуків.

6. Реалізація оплати за допомогою `Stripe`. Для реалізації оплати на сайті було проаналізовано декілька сервісів, але ми вибрали `Stripe` за його зручний тестовий режим. В майбутньому, за потреби, можливо замінити на іншу платіжну систему.

```
const checkoutSession = await stripe.checkout.sessions.create({
  line_items,
  mode: 'payment',
  customer_email: email,
  success_url: process.env.PUBLIC_URL + '/cart?success=1',
  cancel_url: process.env.PUBLIC_URL + '/cart?cancel=1',
  metadata: {orderId: orderDoc.id.toString()},
});
res.json({
  url: checkoutSession.url,
});
```

У даному фрагменті коду створюється сесія оплати за допомогою сервісу `Stripe`. Цей фрагмент коду є частиною більшої системи, яка реалізує функціональність інтернет-магазину та обробку платежів.

Функція `stripe.checkout.sessions.create()` приймає кілька параметрів, включаючи `line_items` (список товарів для оплати), `mode` (режим оплати), `customer_email` (електронна пошта клієнта), `success_url` (URL-адреса для успішного повернення після оплати), `cancel_url` (URL-адреса для відмови від оплати) та `metadata` (додаткові метадані).

Після створення сесії оплати, відповідь відправляється назад клієнту за допомогою `res.json()`, яка містить об'єкт з полем `url`, що містить URL-адресу оплати. Цей URL-адрес ми використовуємо для перенаправлення клієнта на сторінку оплати.

Даний фрагмент коду є важливою частиною системи обробки платежів в інтернет-магазині і допомагає забезпечити безпечну та ефективну оплату товарів клієнтами. Використання сервісу Stripe дозволяє легко і безпечно обробляти платежі, забезпечуючи зручність та надійність для клієнтів.

7. Реалізація функції `upload`. Для загрузки зображень було використано сервіс Amazon AWS.

```
const form = new multiparty.Form();
const client = new S3Client({
  region: 'eu-north-1',
  credentials: {
    accessKeyId: process.env.S3_ACCESS_KEY,
    secretAccessKey: process.env.S3_SECRET_ACCESS_KEY,
  },
});
const links = [];
for (const file of files.file){
  const ext = file.originalFilename.split('.').pop()
  const newFilename = Date.now() + '.' + ext;
  await client.send(new PutObjectCommand({
    Bucket: bucketName,
    Key: newFilename,
    Body: fs.readFileSync(file.path),
    ACL: 'public-read',
    ContentType: mime.lookup(file.path),
  }));
  const link = `https://${bucketName}.s3.amazonaws.com/${newFilename}`;
  links.push(link)
```

Цей код представляє серверний шаблон для обробки запитів, пов'язаних з завантаженням файлів на Amazon S3. Він використовує бібліотеку `multipart` для обробки форм даних, що містять файли. Після обробки форми, він встановлює з'єднання з Amazon S3 та виконує наступні дії для кожного завантаженого файлу:

1. Генерує нове унікальне ім'я для файлу, використовуючи поточний час.
2. Відправляє файл на Amazon S3 з використанням встановлених облікових даних та інших параметрів, таких як контейнер, ключ, тип вмісту та права доступу.
3. Зберігає посилання на завантажений файл у масиві посилань.
4. По завершенні обробки всіх файлів, сервер повертає відповідь з масивом посилань на завантажені файли.

Цей код також встановлює конфігурацію API, яка вимикає вбудований розбірник `bodyParser` для обробки даних форми.

8. Реалізація пошуку товарів. Також на сайті був реалізований пошук товарів. Для цього було розроблено функцію зазначену знизу.

```
function searchProducts(search) {  
  axios.get('/api/products?search='+encodeURIComponent(search)).then(response => {  
    setProducts(response.data);  
    setLoading(false);  
  })  
}
```

Ця функція `searchProducts(search)` виконує пошук продуктів на сервері на основі заданого рядка пошуку.

У тілі функції використовується HTTP GET-запит до `/api/products`, де параметр пошуку `search` додається до URL-адреси у відповідному форматі за допомогою `encodeURIComponent(search)`. Після успішного виконання запиту, дані відповіді з сервера зберігаються у стані компонента за допомогою `setProducts(response.data)`. Крім того, змінна `loading` встановлюється на значення `false`, що означає завершення процесу завантаження.

Ця функція дозволяє виконувати пошук продуктів на сервері з використанням введеного рядка пошуку і оновлювати список продуктів у стані компонента.

9. Реалізація кошику. Дані з кошику зберігаються в локальному сховищі. Це допомагає зробити замовлення навіть неавторизованим користувачам.

```
export function CartContextProvider({children}) {
  const ls = typeof window !== "undefined" ? window.localStorage : null;
  const [cartProducts, setCartProducts] = useState([]);

  useEffect(() => {
    if (cartProducts?.length > 0){
      ls?.setItem('cart', JSON.stringify(cartProducts))
    }
  }, [cartProducts]);
  useEffect(() => {
    if (ls && ls.getItem('cart')) {
      setCartProducts(JSON.parse(ls.getItem('cart')));
    }
  }, [])

  function addProduct(productId) {
    setCartProducts(prev => [...prev, productId]);
  }
  function removeProduct(productId) {
    setCartProducts(prev => {
      const pos = prev.indexOf(productId);
      if (pos !== -1){
        return prev.filter((value, index) => index !== pos);
      }
      return prev;
    })
  }
  function clearCart() {
    setCartProducts([]);
  }
  return (
    <CartContext.Provider value={{cartProducts, setCartProducts, addProduct,
removeProduct, clearCart}}>
      {children}
    </CartContext.Provider>
  );
}
```

Ця функція CartContextProvider є компонентом, який використовується для створення контексту кошика (cart) та управління станом кошика.

У компоненті використовується локальне сховище (localStorage) для зберігання даних кошика між сесіями.

У стані компонента використовується стан cartProducts, який зберігає масив ідентифікаторів товарів у кошику.

За допомогою ефекту useEffect, компонент слідує за змінами у cartProducts. Коли cartProducts змінюється, відбувається збереження даних кошика у локальне сховище за допомогою ls?.setItem('cart', JSON.stringify(cartProducts)).

Також за допомогою другого ефекту useEffect, компонент завантажує дані кошика з локального сховища при монтажі компонента. Якщо дані кошика знайдено у локальному сховищі (ls.getItem('cart')), вони встановлюються у стан cartProducts за допомогою setCartProducts(JSON.parse(ls.getItem('cart'))).

Компонент надає функції 'addProduct', 'removeProduct' та 'clearCart', які дозволяють додавати, видаляти та очищати продукти у кошику відповідно.

Вертається компонент CartContext.Provider, який надає значення контексту кошика та обгортає дочірні компоненти ({children}).

10. Реалізація фільтрів товарів. На сторінці вибраної категорії також є відповідні фільтри реалізовані за допомогою функції зазначеної знизу.

```
function handleFilterChange(filterName, filterValue) {
  setFilterValues(prev => {
    const newValue = prev.map(prop => ({
      name: prop.name,
      value: prop.name === filterName ? filterValue : prop.value,
    }));
    return newValue;
  });
  setChanges(true);
}
```

У вищезазначеному коді ми маємо функцію handleFilterChange, яка відповідає за обробку змін в значеннях фільтрів. Ця функція приймає назву фільтра (filterName) і нове значення фільтра (filterValue).

У тілі функції використовується функція `setFilterValues` для оновлення стану `filterValues`. За допомогою методу `map` ми проходимо по кожному елементу в `prev` (попередньому значенні `filterValues`) і створюємо новий об'єкт, який має ту саму назву, що й попередній об'єкт, але нове значення, якщо назва збігається з `filterName`. Це дозволяє замінити значення фільтра на нове значення. Потім ми присвоюємо новий масив об'єктів фільтрів змінній `newValue` і встановлюємо його як новий стан `filterValues`.

Ця функція дозволяє реагувати зміни в значеннях фільтрів і оновлювати їх у стані компонента. Вона є важливою складовою частиною обробки фільтрів і може бути використана для оновлення відображення даних або виконання інших дій на основі нових значень фільтрів.

11. Анімація для даних. Ми маємо компонент `Spinner`, який відповідає за відображення спінера завантаження.

```
import {PulseLoader} from "react-spinners";
import styles from "./Spinner.module.css";

export default function Spinner() {
  return(
    <div className={styles.spinnerWrapper}>
      <PulseLoader speedMultiplier={3} color={'#555'}/>
    </div>
  )
}
```

Компонент `Spinner` має просту структуру. Він повертає елемент `div`, якому присвоєно клас `spinnerWrapper` зі стилів (визначених у файлі `Spinner.module.css`). Усередині цього елемента розміщений компонент `PulseLoader`, якому передається декілька властивостей: `speedMultiplier` (множник швидкості анімації) та `color` (колір спінера).

Цей компонент використовуватися для відображення спінера завантаження під час очікування асинхронних операцій або завантаження даних. Він додає динаміку та візуальну зворотну зв'язок для користувача, вказуючи на процес завантаження даних.

12. Моделі БД. На прикладі моделі "Wishlist" розглянемо визначення схеми та моделі колекції у базі даних.

```
mongoose.Promise = global.Promise;

const WishlistSchema = new Schema(
  {
    userEmail: {type: String, required: true},
    product: {type: Schema.Types.ObjectId, ref: Product},
  }
);
export const Wishlist = mongoose.models.Wishlist || mongoose.model('Wishlist',
WishlistSchema);
```

Схема включає два поля: "userEmail" та "product". Поле "userEmail" вказує на електронну пошту користувача і є обов'язковим. Поле "product" відноситься до моделі "Product" за допомогою посилання на його ідентифікатор.

Цей код допомагає нам працювати з колекцією "Wishlist" у базі даних, забезпечуючи доступ до полів і зв'язків, що дозволяє зберігати і отримувати дані про бажані товари користувачів.

Висновки до другого розділу

Інтернет-магазин цифрової техніки реалізує ряд функціональностей, таких як додавання товарів, категорій і збереження даних про замовлення. Ця функціональність реалізована через API-шляхи (api routes), які відповідають за обробку запитів для продуктів, категорій, облікових записів користувачів та кошика. В розділі реалізована база даних у відповідності до вимог та забезпечує збереження та колективний доступ до інформації інтернет-магазину. База даних цього магазину складається з 11 колекцій, які забезпечують збереження різних типів даних, необхідних для роботи магазину.

Останній пункт описує деталі реалізованого коду додатку. В цьому описі основна увага приділена функціям, які відповідають за збереження даних у колекції товарів і категорій, а також функціям, які відповідають за підключення до бази даних, основні операції з нею та також реалізація оплати через сервіс Stripe.

РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З ІНТЕРНЕТ-МАГАЗИНОМ ЦИФРОВОЇ ТЕХНІКИ

3.1. Порядок встановлення та налаштування параметрів системи

Інтернет-магазин цифрової техніки є веб-сайтом, який розроблено для використання у різних браузерях. Щоб користуватись цим сайтом, необхідно мати телефон, планшет або ноутбук з наступними мінімальними вимогами:

- Операційна система: Windows 8 і вище, macOS 10.11 і вище.
- Процесор: 1,3-3,8 ГГц.
- Android версія 6.0 і вище.
- 2 ГБ оперативної пам'яті (ОЗУ).
- Будь-який веб-браузер.

Розроблений додаток не потребує спеціалізованого обладнання, додаткових налаштувань або інструментів розгортання, крім стандартних, що використовуються для більшості сучасних веб-сайтів на будь-якій операційній системі. Це забезпечує зручність використання сайту і поліпшує досвід користувачів.

Продовжуючи з теми взаємодії компонентів системи, переходимо до наступного етапу - розробки діаграми розгортання, що описує архітектуру програмного комплексу. Діаграма розгортання містить зображення вузлів виконання програмних компонентів реального часу, процесів і об'єктів.

Діаграма розгортання використовується для відображення загальної конфігурації і топології розподіленої програмної системи і містить інформацію про розташування компонентів на окремих вузлах системи. Крім того, діаграма розгортання показує фізичні з'єднання - шляхи передачі інформації між апаратними пристроями, що використовуються для реалізації системи. Діаграма розгортання призначена для візуалізації елементів і компонентів програми, які існують тільки під час її виконання. На діаграмі розгортання відображаються лише компоненти програми, які є виконуваними файлами або

динамічними бібліотеками. Компоненти, які не використовуються під час виконання, не зображуються на цій діаграмі. Наприклад, компоненти з початковими текстами програм можуть бути присутніми лише на діаграмі компонентів, але не на діаграмі розгортання.

Діаграма розгортання включає графічні зображення процесорів, пристроїв, процесів та їх зв'язків. Вона відображає всі особливості реалізації системи в цілому, відрізняючись від діаграм логічного представлення. Діаграма розгортання є останнім етапом специфікації моделі ООП для конкретної програмної системи. Результат побудованої схеми показано на рисунку 3.1.

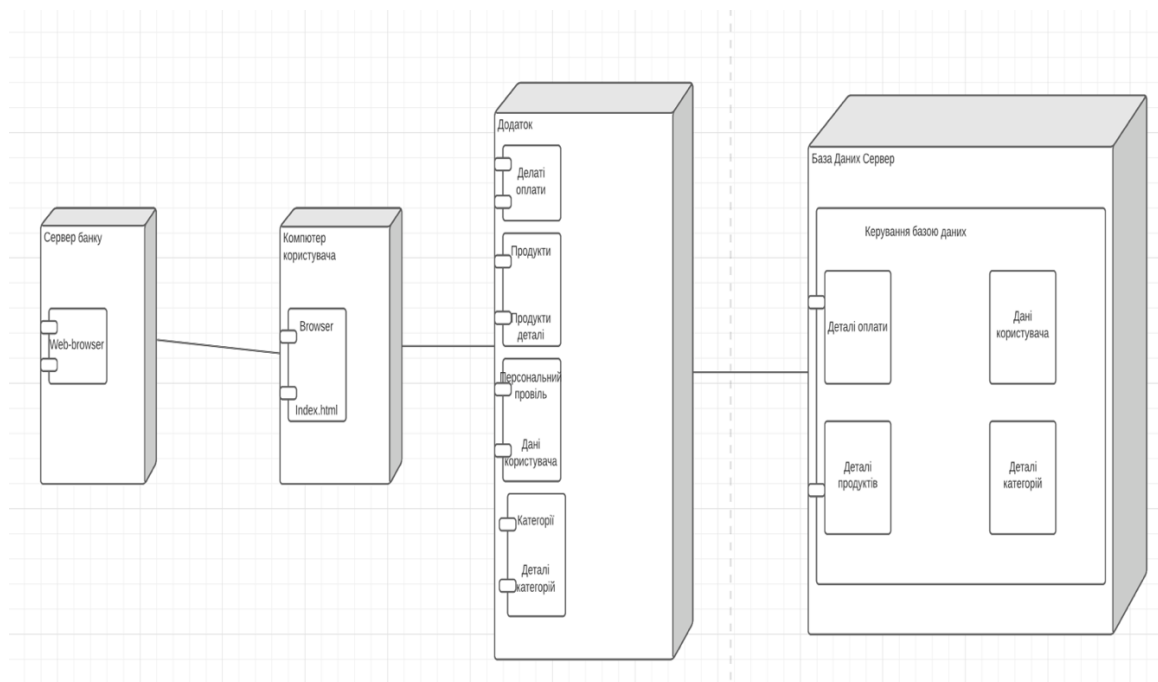


Рис. 3.1. Діаграма розгортання

Діаграма розгортання відображає ці вузли та їх взаємозв'язки, надаючи загальну конфігурацію і топологію системи. Вона допомагає візуалізувати як фізичну, так і логічну структуру системи, допомагає розуміти, як компоненти взаємодіють між собою і зовнішніми елементами. На діаграмі можна побачити наступні елементи:

1. - Вузол (node) - Вузол (node) є центральним елементом системи, який забезпечує обробку запитів від клієнтів і надає їм доступ до веб-сторінок та

інших ресурсів. Він містить програмне забезпечення для обробки HTTP-запитів, веб-сайтів та додатків.

1. - Вузол "Сервер банку" відповідає за обробку запитів, пов'язаних з оплатою, що надходять від користувачів. Цей сервер забезпечує безпеку та надійність операцій з фінансовими даними.
2. - Вузол "Користувач" включає обчислювальні машини, такі як телефони, планшети або ноутбуки, які використовуються користувачами для доступу до системи. Вони дозволяють користувачам переглядати і взаємодіяти з веб-інтерфейсом інтернет-магазину.

2 - Вузол "Додаток" - це програмний компонент, який виконується на вузлі і надає функціональні можливості додатку. Він містить виконувані файли або динамічні бібліотеки, які забезпечують роботу додатку і його інтерфейс.

3 - Вузол "База даних Сервер" представляє фізичну або віртуальну систему, яка зберігає і керує базою даних інтернет-магазину. Цей вузол містить необхідне програмне забезпечення та обладнання, включаючи сервер бази даних, системи управління базами даних (СУБД) та засоби забезпечення безпеки. Він забезпечує надійне зберігання та обробку даних, які використовуються в інтернет-магазині.

3.2. Структура інтерфейсу та порядок роботи з інтернет-магазином цифрової техніки

Коли користувач відкриває стартову сторінку, він може вибрати будь-який пункт з доступного меню, що надається. Це дає йому можливість обрати певну функціональність, яка його цікавить або відповідає його потребам. Вибір пункту меню може включати такі опції, як перегляд продуктів, додавання їх до кошика, перегляд історії замовлень, зміна налаштувань акаунту тощо. Завдяки широкому спектру доступних пунктів меню, користувач може зручно навігувати і взаємодіяти з функціями та можливостями, які пропонує сторінка.

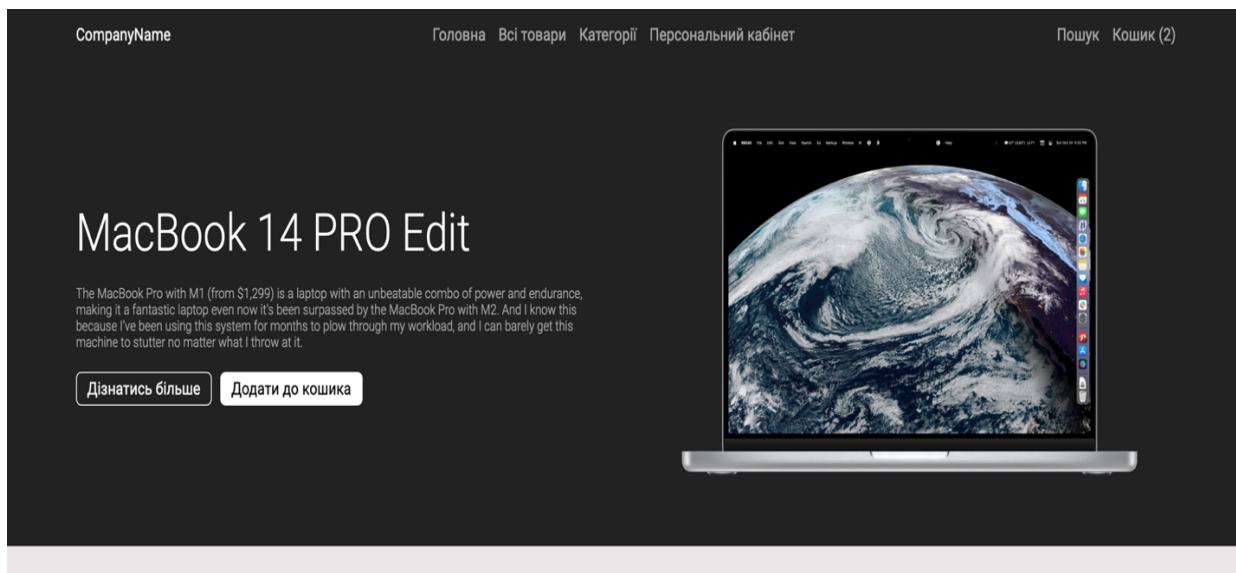


Рис.3.2. Головна сторінка

На стартовій сторінці також відображені нові надходження, що допомагає користувачеві ознайомитися з останніми трендами та актуальними моделями.

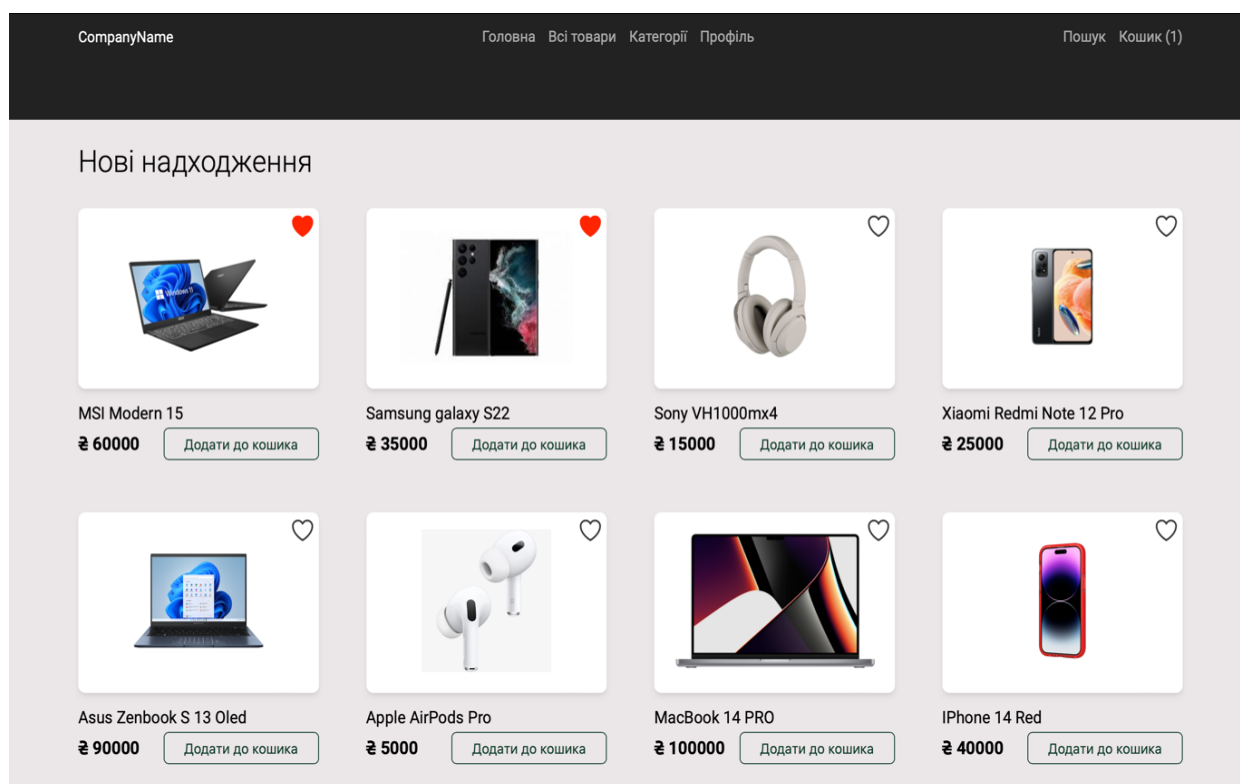


Рис.3.3. Секція нових надходжень на головній сторінці

На сторінці категорій ми можемо побачити вибір товарів, розподілених за різними категоріями, щоб задовольнити потреби і побажання користувачів.

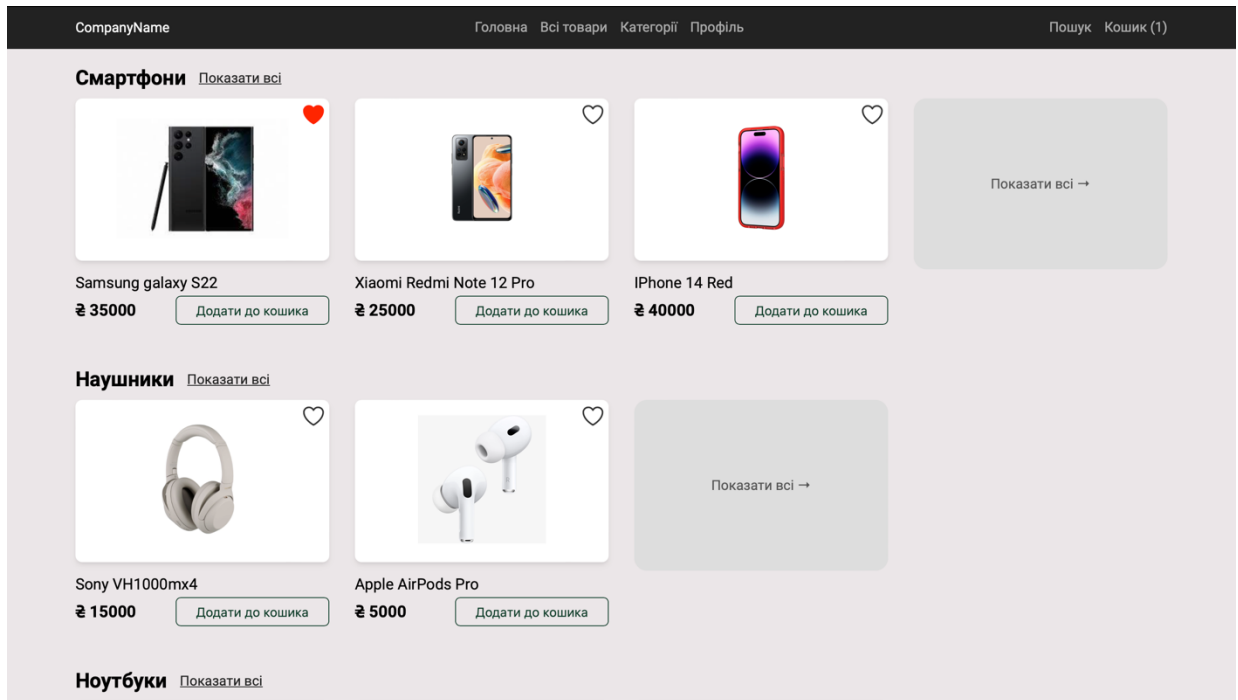


Рис.3.4. Сторінка всіх категорій

Кожна категорія містить асортимент товарів від різних виробників та брендів. Завдяки зручному інтерфейсу, ми можемо швидко переглядати товари в категоріях, використовуючи різні фільтри, які допоможуть знайти саме те, що ми шукаємо.

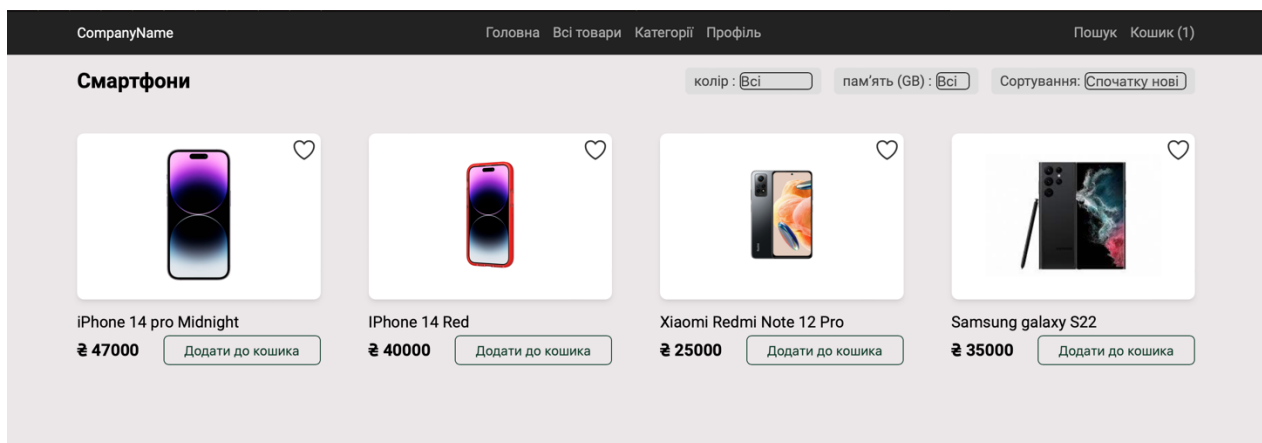


Рис.3.5. Сторінка вибраної категорії

В профілі можливо заповнити дані для замовлень, щоб не повторювати їх написання в кошику при кожному замовленні. Також ми можемо бачити список бажань.

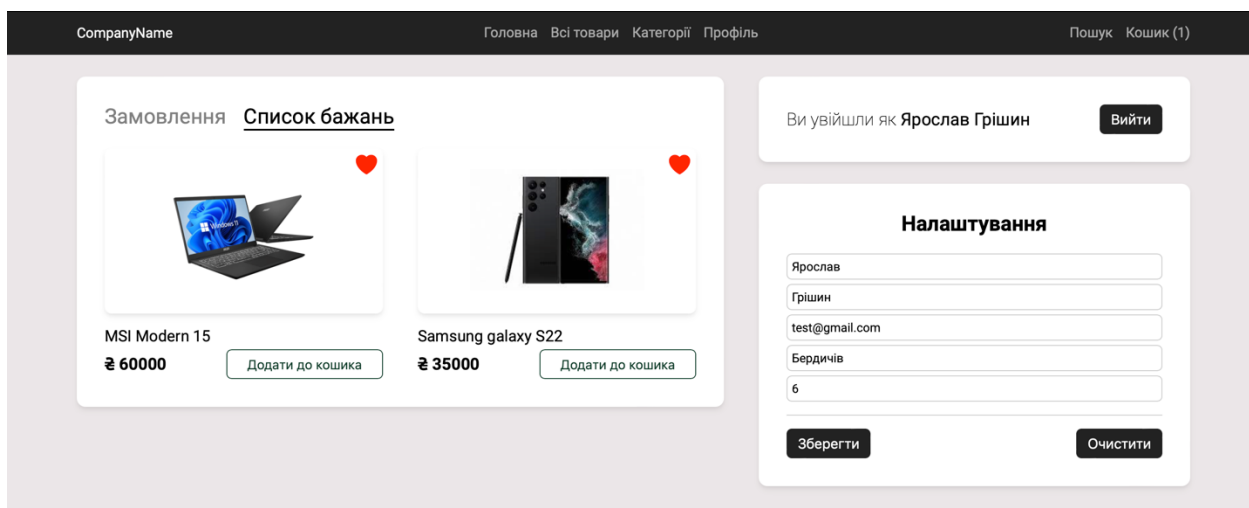


Рис.3.6. Сторінка профілю

Коли ми додаємо товар до кошика, він зберігається там і відображається на сторінці кошика. Ми можемо переглянути всі додані товари, переконатися, що вони відповідають нашим потребам і кількостям, а також змінити їх, якщо потрібно. Також ми можемо бачити оновлену інформацію про замовлення.

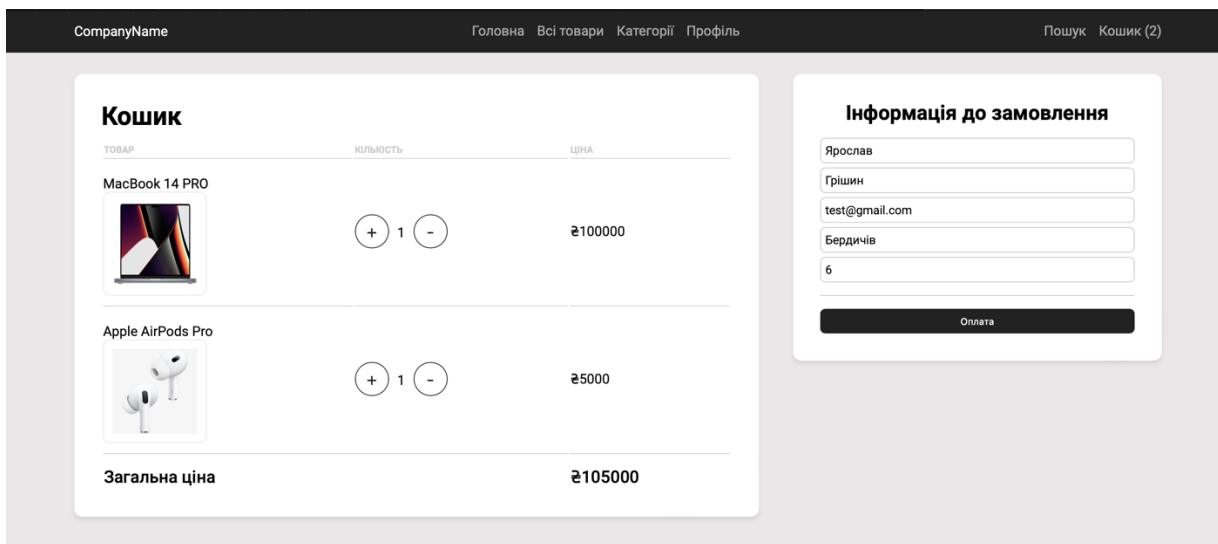


Рис.3.7. Сторінка кошику

Натиснувши на кнопку оплатити нас перенаправити на платіжну систему Stripe. Вибір пав саме на цю систему через зручний тестовий режим. При бажанні платіжну систему можливо замінити на іншу.

test-ecommerce TEST MODE

Заплатить test-ecommerce

105 000,00 €

MacBook 14 PRO 100 000,00 €

Apple AirPods Pro 5 000,00 €

На платформе stripe | Условия Конфиденциальность

Оплатить картой

Эл. почта test@gmail.com

Данные карты

4242 4242 4242 4242

12 / 23 123

Имя и фамилия, указанные на карте

Ярослав

Страна или регион

Украина

☐ Надежно сохранить мои данные для оформления платежа одним щелчком

Выполняйте оплату быстрее в test-ecommerce и везде, где принимают Link.

Оплатить

Рис.3.8. Введення тестових даних для оплати

При успішній оплаті ми побачимо ось таке повідомлення.

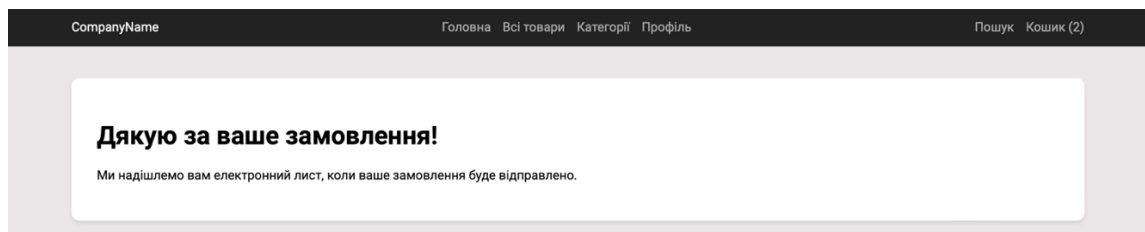


Рис.3.9. Оплата успішна

В самому профілі користувача ми можемо побачити саме замовлення.

Company Name Головна Всі товари Категорії Профіль Пошук Кошик (2)

Дякую за ваше замовлення!

Ми надіслали вам електронний лист, коли ваше замовлення буде відправлено.

Замовлення Список бажань

14.06.2023, 21:33:19
Test test
test
Нова Пошта № - 123
1 x Sony WH1000mx4

20.06.2023, 09:30:02
Ярослав Грішин
Бердичів
Нова Пошта № - 6
1 x MacBook 14 PRO
1 x Apple AirPods Pro

Ви увійшли як **Ярослав Грішин** Вийти

Налаштування

Ярослав

Грішин

test@gmail.com

Бердичів

6

Зберегти Очистити

Рис.3.10. Меню замовлень в профілі користувача

Також є система пошуку товарів. При введенні назви товару сторінка підгружає їх з бази даних без підзагрузок на самому сайті.

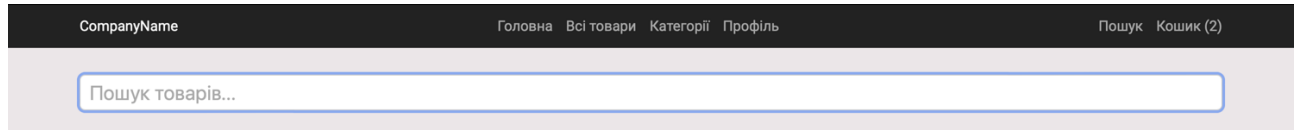


Рис.3.11. Сторінка пошуку

Ось ми бачимо результат пошуку за запитом, що ввів користувач.

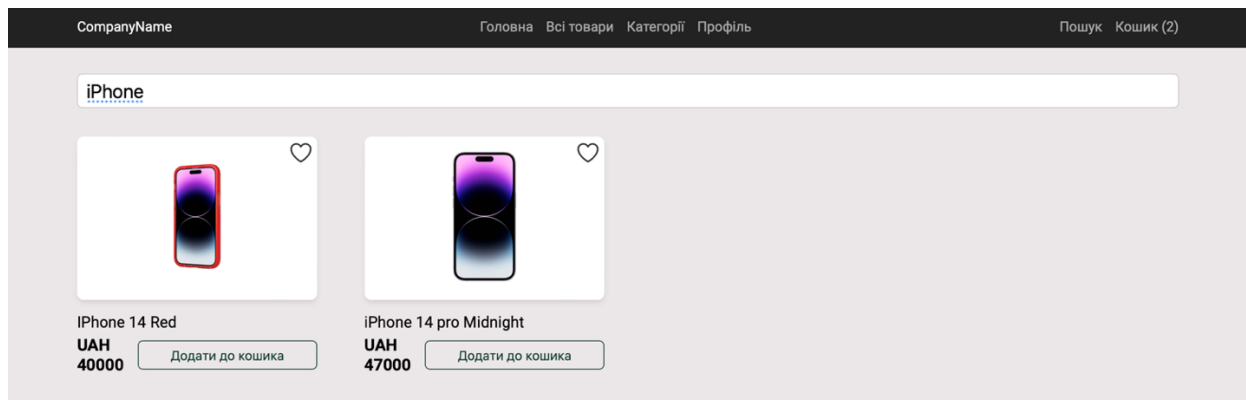


Рис.3.12. Результат пошуку

Також реалізована сторінка товару, яка в свою чергу розділена на дві секції: перша – фото та опис, друга – секція рейтингу та відгуків до товару, де авторизовані користувачі можуть додати свій відгук до вибраного товару.



Рис.3.13. Фото та опис товару

Ваш відгук про товар

☆☆☆☆☆

Заголовок

Ваш відгук

Залишити відгук

Усі відгуки

☆☆☆☆☆ 19.06.2023, 22:44

Text title x2

Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled it to make a type specimen book. It has survived not only five centuries, but also the leap into electronic typesetting, remaining essentially unchanged. It was popularised in the 1960s with the release of Letraset sheets containing Lorem Ipsum passages, and more recently with desktop publishing software like Aldus PageMaker including versions of Lorem Ipsum.

★★★★☆ 19.06.2023, 22:39

Text Title

Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled it to make a type specimen book. It has survived not only five centuries, but also the leap into electronic typesetting, remaining essentially unchanged. It was popularised in the 1960s with the release of Letraset sheets containing Lorem Ipsum passages, and more recently with desktop publishing software like Aldus PageMaker including versions of Lorem Ipsum.

Рис.3.14. Відгуки до вибраного товару

Оскільки ми авторизовані, то можемо додати відгук до цього товару.

Ваш відгук про товар

★★★★☆

Чудово!

Але є проблеми з доставкою!

Залишити відгук

Рис.3.15. Додавання нового відгуку

Ваш відгук про товар

☆☆☆☆☆

Заголовок

Ваш відгук

Залишити відгук

Усі відгуки

★★★★☆ 20.06.2023, 09:52

Чудово!

Але є проблеми з доставкою!

☆☆☆☆☆ 19.06.2023, 22:44

Text title x2

Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled it to make a type specimen book. It has survived not only five centuries, but also the leap into electronic typesetting, remaining essentially unchanged. It was popularised in the 1960s with the release of Letraset sheets containing Lorem Ipsum passages, and more recently with desktop publishing software like Aldus PageMaker including versions of Lorem Ipsum.

★★★★☆ 19.06.2023, 22:39

Text Title

Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled it to make a type specimen book. It has survived not only five centuries, but also the leap into electronic typesetting, remaining essentially unchanged. It was popularised in the 1960s with the release of Letraset sheets containing Lorem Ipsum passages, and more recently with desktop publishing software like Aldus PageMaker including versions of Lorem Ipsum.

Рис.3.16. Відгук додано

В цьому веб-додатку також є і панель адміністратора з усіма основними функціями. Панель адміністратора реалізована як окремий проект, який підключається до спільної бази даних що і сам інтернет-магазин.

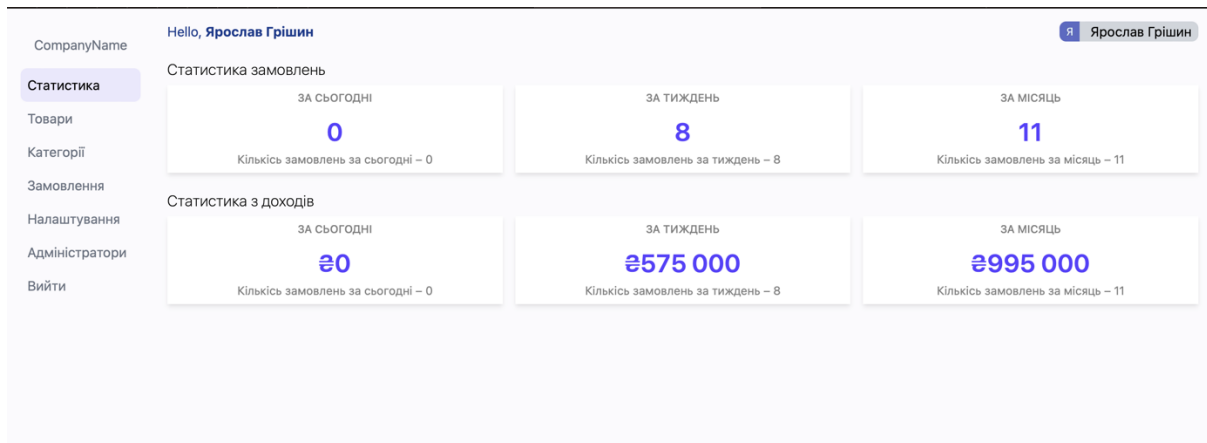


Рис.3.17. Головна сторінка зі статистикою в панелі адміністратора

Ми можемо виконувати всі зазначені дії над наявними товарами або створити новий. Для прикладу розглянемо створення нового.

CompanyName	Додати новий товар	
Статистика	НАЗВА ТОВАРУ	
Товари	iPhone 14 pro Midnight	Редагувати Видалити
Категорії	iPhone 14 Red	Редагувати Видалити
Замовлення	MacBook 14 PRO	Редагувати Видалити
Налаштування	Apple AirPods Pro	Редагувати Видалити
Адміністратори	Asus Zenbook S 13 Oled	Редагувати Видалити
Вийти	Xiaomi Redmi Note 12 Pro	Редагувати Видалити
	Sony VH1000mx4	Редагувати Видалити
	Samsung galaxy S22	Редагувати Видалити
	MSI Modern 15	Редагувати Видалити

Рис.3.18. Список товарів

При створенні нового товару можемо додати категорії, а також властивості цих категорій.

Company Name: Новий товар

Статистика

Товари

Категорії

Замовлення

Налаштування

Адміністратори

Вийти

Назва товару: Новий Товар

Категорія: Смартфони

Колір: чорний

Пам'ять (GB): 64

Фотографії товару

Додати

Опис товару: Новий Товар

Ціна: 2000

Зберегти

Рис.3.19. Додавання нового товару

Новий Товар

Редагувати

Видалити

Рис.3.20. Перевірка в панелі адміністратора

Також новий товар зі всіма вказаними даними був автоматично доданий на головному сайті.

Новий Товар

€ 2000

Додати до кошика

Рис.3.21. Перевірка на сайті

Сторінка категорій з функціями додавання нової категорії, видалення та редагування вже існуючих. Є можливість додати нові властивості вже створеним категоріям.

Company Name

Категорії

Головна панель

Створити нову категорію

Товари

Назва категорії

Без батьківської категорії

Категорії

Властивості

Додати нову властивість

Замовлення

Зберегти

Налаштування

Адміністратори

НАЗВА КАТЕГОРІЇ

БАТЬКІВСЬКА КАТЕГОРІЯ

Вийти

Смартфони		Редагувати	Видалити
iPhone	Смартфони	Редагувати	Видалити
Samsung	Смартфони	Редагувати	Видалити
Xiaomi	Смартфони	Редагувати	Видалити
Наушники		Редагувати	Видалити
Ноутбуки		Редагувати	Видалити

Рис.3.22. Сторінка категорій

Адміністратор має можливість переглянути всі замовлення зроблені клієнтами сайту. Та вже може зв'язатися з клієнтом та відправляти товари за їх вказаними адресами.

Company Name	Замовлення		
Статистика	ДАТА	ОДЕРЖУВАЧ	ТОВАР
Товари	20.06.2023, 09:30:02	Ярослав Грішин test@gmail.com	MacBook 14 PRO x 1
Категорії		Мікро - Бердичів Нова Пошта # - 6	Apple AirPods Pro x 1
Замовлення	18.06.2023, 16:30:54	Yaroslav Hrishyn yaroslav@gmail.com	MacBook 14 PRO x 1
Налаштування		Мікро - Berdychiv Нова Пошта # - 5	Apple AirPods Pro x 1
Адміністратори	18.06.2023, 16:29:45	test1 test	Samsung galaxy S22 x 2
Вийти		test	
		Мікро - test	Apple AirPods Pro x 1

Рис.3.23. Сторінка з всіма замовленнями

Також ми можемо переглянути та додати до системи нових адміністраторів. До панелі адміністратора можливо зайти тільки по цим електронним поштам.

CompanyName

Адміністратори

Статистика

Додати нового адміністратора

google email

Зберегти

Товари

Категорії

Замовлення

Налаштування

Адміністратори

Вийти

GOOGLE EMAIL АДМІНІСТРАТОРА

ipz192_gyao@student.ztu.edu.ua	16.06.2023	Видалити
the.raks.za@gmail.com	16.06.2023	Видалити
for.steam.za@gmail.com	17.06.2023	Видалити

Рис.3.24. Сторінка з адміністраторами

CompanyName

Адміністратори

Головна панель

Додати нового адміністратора

google email

Зберегти

Товари

Категорії

Замовлення

Налаштування

Адміністратори

Вийти

GOOGLE EMAIL АДМІНІСТРАТОРА

for.steam.za@gmail.com		
test@gmail.com		

✓

Адміністратора створено

ОК

Рис.3.25. Додавання нового адміністратора

На сторінці налаштувань можливо змінити рекомендований товар, який є на головній сторінці інтернет-магазину.

CompanyName

Налаштування

Статистика

Рекомендований товар (На головній сторінці)

MacBook 14 PRO

Зберегти

Товари

Категорії

Замовлення

Налаштування

Адміністратори

Вийти

Рис.3.26. Сторінка налаштувань

3.3.Тестування роботи програмного продукту

В процесі розробки та впровадження програмного продукту успішне тестування відіграє важливу роль. Цей етап спрямований на виявлення помилок, перевірку функціональності та оцінку загальної якості та ефективності продукту. У цьому розділі будуть розглянуті тест-кейси для основних функцій та методів системи, а також надані результати загального тестування функціоналу.

Тестування процесу додавання товару в кошик

Передумови: користувач авторизований на веб-сайті магазину та товар доступний для покупки;

Кроки: відкрити сторінку з товару, натиснути на кнопку "Додати в кошик";

Очікуваний результат: товар успішно додається в кошик, кількість товарів в кошику збільшується на одиницю, виводиться підтвердження про додавання товару в кошик;

Фактичний результат: товар успішно додається в кошик, кількість товарів в кошику збільшується на одиницю, виводиться підтвердження про додавання товару в кошик.

Тестування процесу оформлення замовлення

Передумови: товари додані в кошик;

Кроки: відкрити сторінку кошика, заповнити обов'язкові поля форми замовлення, натиснути на кнопку "Оформити замовлення";

Очікуваний результат: замовлення успішно оформлене, користувач перенаправляється на сторінку з підтвердженням замовлення, виводиться підтвердження про успішне оформлення замовлення;

Фактичний результат: замовлення успішно оформлене, користувач перенаправляється на сторінку з підтвердженням замовлення, виводиться підтвердження про успішне оформлення замовлення.

Тестування функції пошуку товарів

Передумови: Користувач на веб-сайті магазину;

Кроки: відкрити сторінку пошуку товарів, ввести ключове слово або фразу для пошуку;

Очікуваний результат: результати пошуку, що відповідають введеному запиту, які відповідають критеріям пошуку, відображаються коректно;

Фактичний результат: товари, які відповідають критеріям пошуку, відображаються коректно.

Тестування функції додавання відгуку до товару

Передумови: користувач авторизований на веб-сайті магазину, сторінка товару, до якого буде додано відгук, доступна;

Кроки: відкрити сторінку товару, до якого потрібно додати відгук, заповнення форми додавання відгуку, натиснути на кнопку "Залишити відгук":

Очікуваний результат: після натискання на кнопку "Залишити відгук" відгук з'являється на сторінці товару;

Фактичний результат: після натискання на кнопку "Залишити відгук" відгук з'являється на сторінці товару.

Було також здійснено тестування усього функціоналу системи. Під час даного тестування помилок та чи недоліків в роботі системи не було виявлено. Додаток повністю готовий до користування. Однак, незважаючи на успішне тестування, важливо продовжувати вдосконалювати та розширювати функціонал системи для забезпечення найвищої якості та задоволення користувачів.

Висновки до третього розділу

В третьому розділі дослідження ми розглянули інтерфейс та порядок роботи з інтернет-магазином цифрової техніки. Ми дослідили порядок встановлення та налаштування параметрів системи. Було показано, що

правильне налаштування системи є важливим етапом у забезпеченні оптимальної роботи магазину та задоволення потреб користувачів.

Ми розглянули структуру інтерфейсу і показали основні елементи, які присутні в інтернет-магазині цифрової техніки. Ми показали, що правильна організація інтерфейсу є ключовим аспектом, що допомагає користувачам легко зорієнтуватися в магазині, знаходити потрібні товари та здійснювати покупки без зайвих перешкод.

У цьому розділі ми провели детальне дослідження процесу налаштування системи, структури інтерфейсу та тестування роботи програмного комплексу інтернет-магазину цифрової техніки. Наші результати демонструють, що правильний порядок дій та належне налаштування грають важливу роль у створенні функціонального та зручного магазину для користувачів. Забезпечення ефективної роботи програмного комплексу через тестування дозволяє запобігати можливих помилок та недоліків, що можуть вплинути на користувачів та навіть призвести до втрати бізнесу. Всі помилки було виправлено, на даний час сайт працює коректно та виконує всі поставлені задачі.

ВИСНОВКИ

Під час виконання випускної роботи бакалавра було спроектовано та реалізовано інтернет-магазин цифрової техніки для цього було вирішено наступні завдання:

- Проведено аналіз аналогічних розробок Алло, Розетка, Цитрус та визначено, що основним функціоналом є: розміщення товарів, пошук товарів за параметрами, додавання товарів до кошика, оформлення замовлення, оплата товарів онлайн, відстеження статусу замовлення та доставка товарів. Однак доцільно реалізувати для даного інтернет-магазину ще наступні функції: збереження списку улюблених товарів, можливість залишати відгуки про товари, персоналізований підбір товарів на основі попередніх покупок.

- Для реалізації інтернет-магазину цифрової техніки було вибрано наступні засоби: Next.js: Використання Next.js дозволило створити швидкий та ефективний веб-додаток з покращеним користувацьким досвідом.CSS Modules: Використання CSS Modules допомогло забезпечити чистий та структурований код CSS для стилізації інтерфейсу магазину. MongoDB Atlas: Використання MongoDB Atlas дозволило забезпечити надійне зберігання даних про товари, замовлення та користувачів магазину.

- Було проаналізовано функціональні та нефункціональні вимоги до інтернет-магазину цифрової техніки була побудована діаграма варіантів використання для визначення основних функцій та можливостей системи. На основі цієї діаграми була розроблена діаграма класів, яка допомагає проаналізувати структуру системи та взаємозв'язки між класами.

- Під час проектування інтернет-магазину цифрової техніки була створена база даних у MongoDB, яка забезпечує збереження і колективний доступ до інформації, необхідної для функціонування магазину. База даних складається з 11 колекцій, які відображають різні аспекти системи та зберігають відповідні дані.

- В проекті було реалізовано інтерфейс системи, який включає два основних представлення: адміністратора та користувача. Ці представлення надають різні можливості та функціонал для відповідних ролей. Адміністраторський інтерфейс дозволяє адміністраторам керувати та налаштовувати систему. Вони можуть змінювати зовнішній вигляд системи, налаштовувати права доступу користувачів, переглядати статистику роботи системи та здійснювати інші адміністративні дії. Користувацький інтерфейс призначений для звичайних користувачів. Він надає їм можливість переглядати товари, здійснювати покупки, додавати товари до кошика, залишати відгуки та взаємодіяти з різними функціями системи, спеціально розробленими для користувачів.

Було проведено тестування інтернет-магазину, включаючи функціональне, адаптивне, кроссбраузерне, безпеки та продуктивність. Всі виявлені помилки були виправлені, і на даний момент сайт працює без проблем і успішно виконує всі поставлені завдання.

Результатом розробки є програмний комплекс, який готовий до використання в промислових умовах. Цей комплекс може бути успішно використаний для продажу товарів через інтернет, забезпечуючи надійну та ефективну платформу для електронної комерції.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Vice R. Mastering React / R. Vice, A. Horton., 2016. – 260 с. – (Packt Publishing)
2. Eric Elliott. Programming JavaScript Applications — O'Reilly Media, 254 p
3. Next.js документація [Електронний ресурс] – Режим доступу до ресурсу: <https://nextjs.org/docs>
4. Блог Next.js [Електронний ресурс] – Режим доступу до ресурсу: <https://nextjs.org/blog/>
5. Інтернет-магазин 'Цитрус' [Електронний ресурс] – Режим доступу до ресурсу: <https://ctr.com.ua>
6. Документація MongoDB [Електронний ресурс] – Режим доступу до ресурсу: <https://docs.mongodb.com/>
7. Документація Mongoose [Електронний ресурс] – Режим доступу до ресурсу: <https://mongoosejs.com/docs/>
8. Офіційна документація CSS Modules для керування стилями в Next.js [Електронний ресурс] – Режим доступу до ресурсу: <https://nextjs.org/docs/basic-features/built-in-css-support#css-modules>
9. Офіційна документація React [Електронний ресурс] – Режим доступу до ресурсу: <https://reactjs.org/docs/>
10. Інтернет-магазин 'Алло' [Електронний ресурс] – Режим доступу до ресурсу: <https://allo.ua>
11. Офіційна документація MongoDB Atlas [Електронний ресурс] – Режим доступу до ресурсу: <https://docs.atlas.mongodb.com/>
12. Офіційна документація Stripe [Електронний ресурс] – Режим доступу до ресурсу: <https://stripe.com/docs>
13. Офіційна документація Next.js API Routes [Електронний ресурс] – Режим доступу до ресурсу: <https://nextjs.org/docs/api-routes/introduction>
14. Інтернет-магазин 'Розетка' [Електронний ресурс] – Режим доступу до ресурсу: <https://rozetka.com.ua>

ДОДАТКИ

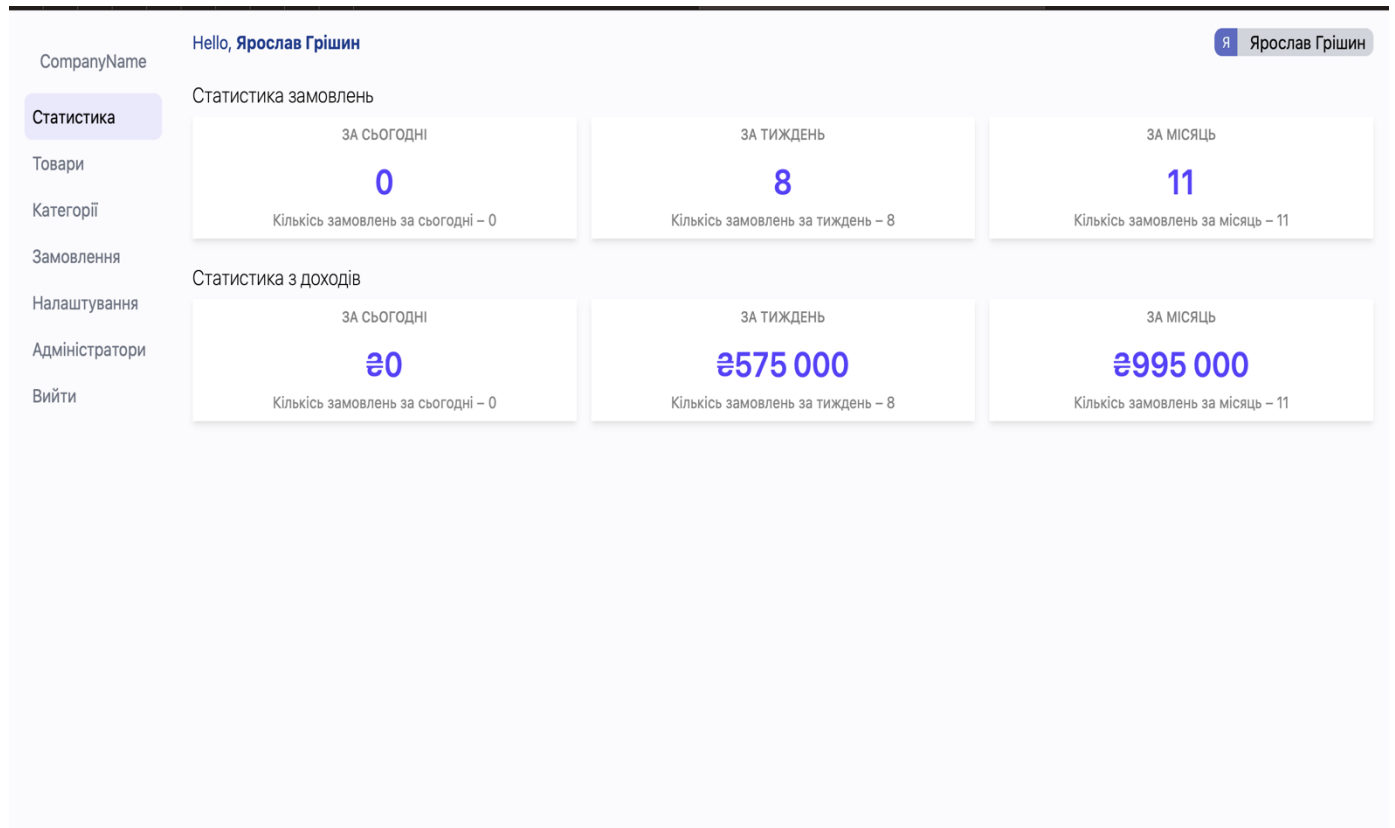


Рисунок А.1 – Сторінка зі статистикою.

/api/products.js

```
import {mongooseConnect} from "@lib/mongoose";
import {isAdminRequest} from "@pages/api/auth/[...nextauth]";
import {Admin} from "@models/Admin";

export default async function handler(req, res) {
  await mongooseConnect();
  await isAdminRequest(req, res);

  if (req.method === 'POST') {
    const {email} = req.body;
    res.json(await Admin.create({email}));
  }
  if (req.method === 'GET') {
    res.json(await Admin.find())
  }
}
```

/api/categories.js

```
import {Category} from "@models/Category";
import {mongooseConnect} from "@lib/mongoose";
import {isAdminRequest} from "@pages/api/auth/[...nextauth]";

export default async function handle(req, res){
  const {method} = req;
  await mongooseConnect();
  await isAdminRequest(req,res);

  if (method === 'POST'){
    const {name,parentCategory,properties} = req.body;
    const categoryDoc = await Category.create({
      name,
      parent:parentCategory || undefined,
      properties:properties
    });
    res.json(categoryDoc)
  }

  if (method === 'GET') {
    res.json(await Category.find().populate('parent'));
  }

  if (method === 'PUT') {
    const {name,parentCategory,properties,_id} = req.body;
    const categoryDoc = await Category.updateOne({_id},{name,parent:parentCategory || undefined,properties});
  }
}
```

```

    res.json(categoryDoc)
  }

  if (method === 'DELETE') {
    const {_id} = req.query;
    await Category.deleteOne({_id:req.query.id})
    res.json('ok')
  }
}

```

/api/orders.js

```

import {mongooseConnect} from "@/lib/mongoose";
import {Order} from "@/models/Order";

export default async function handler(req, res){
  await mongooseConnect()
  res.json(await Order.find().sort({createdAt: -1}));
}

```

/api/upload.js

```

import multiparty from "multiparty";
import {PutObjectCommand, S3Client} from '@aws-sdk/client-s3'
import fs from 'fs';
import mime from 'mime-types'
import mongoose from "mongoose";
import {mongooseConnect} from "@/lib/mongoose";
import {isAdminRequest} from "@/pages/api/auth/[...nextauth]";
const bucketName = 'pcesdr-ecommerce'
export default async function handle(req,res) {
  await mongooseConnect();
  await isAdminRequest(req,res);

  const form = new multiparty.Form();
  const {fields, files} = await new Promise((resolve, reject) => {
    form.parse(req, async (err, fields, files) => {
      if (err) reject(err);
      resolve({fields, files});
    });
  });
  console.log('length:',files.file.length);
}

```

```

const client = new S3Client({
  region: 'eu-north-1',
  credentials: {
    accessKeyId: process.env.S3_ACCESS_KEY,
    secretAccessKey: process.env.S3_SECRET_ACCESS_KEY,
  },
});

const links = [];
for (const file of files.file){
  const ext = file.originalFilename.split('.').pop()
  const newFilename = Date.now() + '.' + ext;
  await client.send(new PutObjectCommand({
    Bucket: bucketName,
    Key: newFilename,
    Body: fs.readFileSync(file.path),
    ACL: 'public-read',
    ContentType: mime.lookup(file.path),
  }));
  const link = `https://${bucketName}.s3.amazonaws.com/${newFilename}`;
  links.push(link)
}
return res.json({links});
};

export const config = {
  api: {bodyParser: false}
};

```

/api/[...nextauth].js

```

import NextAuth, {getServerSession} from 'next-auth'
import GoogleProvider from 'next-auth/providers/google'
import {MongoDBAdapter} from "@next-auth/mongodb-adapter";
import clientPromise from "@/lib/mongodb";

const adminEmails = ['for.steam.za@gmail.com']
export const authOptions = {
  providers: [
    // OAuth authentication providers..
    GoogleProvider({
      clientId: process.env.GOOGLE_ID,
      clientSecret: process.env.GOOGLE_SECRET

```

```

    }},

    ],
    secret: process.env.SECRET,
    adapter: MongoDBAdapter(clientPromise),
    callbacks: {
      session: ({session,token,user}) => {
        if (adminEmails.includes(session?.user?.email)){
          return session;
        } else {
          return false;
        }
      },
    },
  },
};

export default NextAuth(authOptions);

export async function isAdminRequest(req, res) {
  const session = await getServerSession(req,res,authOptions);
  if (!adminEmails.includes(session?.user?.email)) {
    res.status(401);
    res.end();
    throw 'not admin'
  }
}

```

/model/Category.js

```

import mongoose, {Schema} from "mongoose";

mongoose.Promise = global.Promise;

const categorySchema = new Schema(
  {
    name: {type: String, required: true},
    parent: {type:mongoose.Types.ObjectId, ref:'Category'},
    properties: [{type:Object,}]
  },
);

export const Category = mongoose.models.Category || mongoose.model('Category', categorySchema);

```

/model/Order.js

```
import mongoose, {Schema} from "mongoose";

mongoose.Promise = global.Promise;

const OrderSchema = new Schema(
  {
    line_items:Object,
    name:String,
    surname:String,
    email:String,
    city:String,
    postal:String,
    paid:Boolean
  }, {
    timestamps:true,
  }
);
export const Order = mongoose.models.Order || mongoose.model('Order', OrderSchema);
```

/model/Product.js

```
import mongoose, {Schema} from "mongoose";

mongoose.Promise = global.Promise;

const productSchema = new Schema(
  {
    title: {type: String, required: true},
    description: String,
    price: {type: Number, required: true},
    images: [{type: String}],
    category: {type: mongoose.Types.ObjectId, ref:'Category'},
    properties: {type:Object},

  },{
    timestamps: true
  }
);
```

```
);  
export const Product = mongoose.models.Product || mongoose.model('Product', productSchema);
```

/model/Admin.js

```
import mongoose, {Schema} from "mongoose";  
  
mongoose.Promise = global.Promise;  
  
const adminSchema = new Schema(  
  {  
    email: {type: String, required: true, unique:true},  
  },  
);  
export const Admin = mongoose.models?.Admin || mongoose.model('Admin', adminSchema);
```

mongoose.js

```
import mongoose from "mongoose";  
  
export function mongooseConnect() {  
  
  const uri = process.env.MONGODB_URI;  
  
  if (mongoose.connection.readyState === 1){  
    return mongoose.connection.asPromise();  
  }  
  else {  
    return mongoose.connect(uri);  
  }  
}
```